Haptic Feedback in Room-Scale VR

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Philipp Erler
Matrikelnummer 01426424

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Dipl.-Ing. Markus Schütz

Wien, 18. Juli 2017

Philipp Erler   Michael Wimmer

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Haptic Feedback in Room-Scale VR

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Philipp Erler
Registration Number 01426424

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Dipl.-Ing. Markus Schütz

Vienna, 18th July, 2017

Philipp Erler	Michael Wimmer

Erklärung zur Verfassung der Arbeit

Philipp Erler
Lassallestraße 30/22, 1020 Wien

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

Wien, 18. Juli 2017

Philipp Erler

Acknowledgements

This work would not have been possible without the support of many great people. At least, it would have taken much more time.

First of all, I want to thank my parents, Karin and Uwe Erler, for supporting me during my entire studies. Thanks to them, I could concentrate on my academic career. Next, I thank Lisa Fellinger for proofreading all my texts and improving my English. I thank Stefan Zaufl for pointing out fundamental flaws in my plans, and I thank him for our nice and productive conservations.

From the TU Wien, I thank Michael Wimmer for the smooth execution of this thesis, Markus Schütz for his good ideas and his constructive feedback, and Annette Mossel for sharing her experience with user studies.

I thank the staff of the VRVis in general and the Semantic Modelling and Acquisition group in particular. Most importantly, I thank Andreas Walch, Attila Szabo, Bernhard Rainer, Georg Haaser, Harald Steinlechner, Michal Domanski, Michael Schwärzler, and Stefan Maierhofer for their good ideas, as well as their help with F# and Aardvark.

Last but not least, I thank all 26 testers for their time, sweat, and feedback.

Kurzfassung

Virtuelle Realität (VR) wird im Moment zu einem verbreiteten Medium. Aktuelle Systeme wie die HTC Vive bieten ein genaues Tracking des HMD und der Controller. Dies ermöglicht extrem immersive Interaktionen mit der virtuellen Umgebung. Diese Interaktionen können durch Feedback weiter verbessert werden. Zum Beispiel kann ein Controller vibrieren, wenn er in der Nähe eines aufnehmbaren Balles ist.

Wir haben eine Nutzerstudie durchgeführt, um Folgendes zu analysieren:

1. Das Aufnehmen und Werfen mit Controllern in einem einfachen Basketball-Spiel.
2. Den Einfluss von haptischen und optischen Feedback auf Leistung, Präsenz, Benutzerfreundlichkeit und Arbeitsbelastung.
3. Die Vorteile von Punktwolken-Bearbeitung in VR gegenüber Punktwolken-Bearbeitung am Computer.

Durch die Entwicklung den Punktwolken Editors für VR sind einige neue Verfahren entstanden. Unsere neue Betrachtungsmethode, die beidhändige Zwick-Geste, erweitert die Handgriff-Metapher, um Punktwolken in VR effizient zu bewegen, drehen und skalieren. Unsere neue Renderingtechnik nutzt den Geometry Shader, um dünn besetzte Punktwolken schnell zu zeichnen. Selektionvolumen am Controller sind unsere neue Methode, um Punkte in der Punktwolke effizient zu selektieren. Die sich ergebende Selektion kann in Echtzeit dargestellt werden.

Die Ergebnisse der Nutzerstudie zeigen, dass:

1. das Aufnehmen mit einem Controllerknopf intuitiv ist, das Werfen aber nicht. Den Knopf loszulassen ist eine schlechte Metapher für das Loslassen eines virtuellen Objekts, um es zu werfen.
2. das Hinzufügen von haptischen, optischen oder beiden Feedbacktypen zum Aufnehmen die Leistung der Nutzer und die Präsenz verbessert. Allerdings sind nur Teilbereiche wie Genauigkeit und Vorhersagbarkeit signifikant verbessert. Das Feedback hat fast keinen Einfluss auf Benutzerfreundlichkeit und Arbeitsbelastung.
3. das Editieren von Punktwolken in VR mit der beidhändigen Zwick-Geste und den Selektionsvolumen ist signifikant besser als am Computer mit Orbitkamera und Lasso-Selektionen.

Abstract

Virtual reality (VR) is now becoming a mainstream medium. Current systems like the HTC Vive offer accurate tracking of the HMD and controllers, which allows for highly immersive interactions with the virtual environment. The interactions can be further enhanced by adding feedback. As an example, a controller can vibrate when it is close to a grabbable ball.

As such interactions are not exhaustingly researched, we conducted a user study. Specifically, we examine:

1. grabbing and throwing with controllers in a simple basketball game.
2. the influence of haptic and optical feedback on performance, presence, task load, and usability.
3. the advantages of VR over desktop for point-cloud editing.

Several new techniques emerged from the point-cloud editor for VR. The bi-manual pinch gesture, which extends the handlebar metaphor, is a novel viewing method used to translate, rotate, and scale the point-cloud. Our new rendering technique uses the geometry shader to draw sparse point clouds quickly. The selection volumes at the controllers are our new technique to efficiently select points in point clouds. The resulting selection is visualized in real time.

The results of the user study show that:

1. grabbing with a controller button is intuitive but throwing is not. Releasing a button is a bad metaphor for releasing a grabbed virtual object in order to throw it.
2. any feedback is better than none. Adding haptic, optical, or both feedback types to the grabbing improves the user performance and presence. However, only sub-scores like accuracy and predictability are significantly improved. Usability and task load are mostly unaffected by feedback.
3. the point-cloud editing is significantly better in VR with the bi-manual pinch gesture and selection volumes than on the desktop with the orbiting camera and lasso selections.

Contents

CHAPTER 1

Introduction

After being around for 60 years, virtual reality finally evolved from a hyped research area to a rather common medium. The current room-scale VR systems, like the HTC Vive [HTC16], are now affordable for consumers and allow for highly immersive virtual worlds with a strong feeling of presence. Natural interactions are easy to learn and therefore improve the usability of VR applications. The interactions can be further enhanced with feedback, for example, vibration of a controller when it is close to a grabbable ball. This improves the efficiency of the interaction and can strengthen the presence. Viewing and editing point clouds in VR are a new field of research, which may benefit from more natural interaction methods.

1.1 Problem Statement

It is important to have intuitive and efficient methods of interaction with the virtual world. The choice of interaction techniques can greatly improve the feeling of immersion and presence. There are many interaction techniques, most of them have been known for more than 10 years. The most important one is the virtual hand [WJ88], which maps the position and rotation of the real controller directly onto its virtual counterpart. It is used to select and manipulate objects in VR.

One interaction that occurs frequently is *grabbing* an object with the virtual hand. There is a variety of input methods for this: real hands with gesture recognition [LS16], finger tracking with data gloves [SZ94], and hand-held controllers [FHKH06]. However, grabbing and *throwing* physically simulated objects, e.g. balls, with hand-held controllers is a combination that is not sufficiently researched. Thanks to the many current VR systems like the HTC Vive [HTC16] and Oculus Rift [Ocu16], such hand-held controllers have become the most common input methods. Therefore, our first research question is “whether grabbing and throwing virtual balls with hand-held controllers is intuitive”.

Often, interaction and movement techniques can be further improved with different kinds of *feedback*. As an example, simple vibrations of a hand-held controller are enough to convey a sense of touch and make it easier for the player to grab virtual objects. Another example is optical feedback. A highlight on an object shows that it can be grabbed. However, the effect of such feedback has not been adequately quantified with current VR systems. Therefore, our second research question is "whether feedback has a statistically significant influence on grabbing".

Some techniques are better suited for certain tasks than others. Interacting with rigid bodies is well researched, while interactions with point clouds have not yet been studied sufficiently. One reason is that algorithms to render large point clouds efficiently were proposed quite recently [RL00]. Rendering dense point clouds requires high computation power, especially in VR. The necessary graphics hardware to efficiently display millions of points 90 times per second for both eyes only became available in the last few years. Additionally, the VR systems before the HTC Vive and Oculus Rift were experimental and certainly not fit for regular use. Simply put, VR and point clouds were challenging research areas on their own. Now, their basic problems are solved and it is possible to combine them, but it is unclear whether this is beneficial. Therefore, our third research question is "whether point-cloud editing is better in VR than on the desktop".

1.2 Contributions

To answer the three research questions posed above, we conducted a large user study. Specifically, the goal of the user study was to examine:

1. grabbing and throwing with controllers in a simple basketball game.
2. the influence of haptic and optical feedback on performance, presence, task load, and usability.
3. the advantages of VR over desktop for point-cloud editing.

For the user study, we implemented three VR applications, which led to further contributions:

1. An efficient and easy-to-understand software architecture.
2. A new method to render sparse point clouds using the geometry shader. Compared with the technique by Schütz and Wimmer [SW15], the rendering time of our new method is only about a tenth but still produces a similar visual quality.
3. A new method using selection volumes at controllers to efficiently select points of a point cloud in VR.
4. A combination of selection volumes with the instant selection visualization by Rainer [Rai16] to enable point-cloud selections in real time.
5. A new viewing method for point clouds in VR, the bi-manual pinch gesture, which extends the handlebar metaphor [SGH$^+$12].

Chapter 2 describes the state-of-the-art in VR technology, interaction and movement techniques, presence, feedback, and point clouds. The design decisions for the three applications created for this thesis are explained in Chapter 3, and Chapter 4 gives details of the implementation. Chapter 5 describes how the user study was conducted, and Chapter 6 presents its results. Finally, Chapter 7 concludes the thesis and proposes possible future work.

CHAPTER 2

Related Work

Virtual reality brings computer science and humans together. Therefore, it is a very broad field incorporating knowledge from the realms of mechanics, electronics, physiology, psychology, and especially computer science. The focus of this thesis is on human-machine interaction in VR. Therefore, already known interaction and movement techniques as well as an analysis of the presence are described in detail.

2.1 VR Technology

This section is dedicated to VR hardware. It gives a brief overview of some current VR systems with their input and output devices. Additionally, some groundbreaking concepts and devices are described.

In 1965, Sutherland described a theoretical 'ultimate display' that would provide input to all senses [Sut65]:

> The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked.

A few years later, he developed one of the first head mounted displays, called 'The Sword of Damocles' [Sut68]. It featured one miniature cathode ray tube for each eye, and it also used mechanical or ultrasonic head tracking to control the view in the virtual environment. Afterwards, VR has found applications in flight simulators [Mos93], education [DSL96], and phobia treatment [GPHC$^+$02].

Between 1990 and 2000, many HMDs for consumers and professionals were developed. One example is the Forte VFX1 [Min98] [Hel98], which was released in 1995 for 695$. It had an LCD with 263x230 pixels for each eye and could track the head orientation. Another example is the Sony Glasstron PLM-S700 [Son98], which was announced in 1998 for 298,000 yen, around 2,500$. It featured an LCD with 800x600 pixels per eye and optional see-through ability but no head tracking at all.

As an alternative to HMDs, the CAVE was introduced in 1992 [CNSD$^+$92]. It uses beamers projecting stereo images onto the sides of a room. The user simply needs to wear light-weight anaglyph or shutter glasses and has a complete field of view. The disadvantages of CAVEs over HMDs include a higher computation effort, more required space, and higher costs.

The current generation of VR devices began with the Kickstarter campaign of the Oculus Rift [Ocu16]. Two major improvements of the Rift are precise, low-latency head tracking with 6 Degrees of Freedom (DoF) and low-persistence OLEDs with high resolution [Ocu14]. Other companies jumped on the newly sparked hype and started their own products, for example the HTC Vive [HTC16] and PlayStation VR [Son17]. These three VR systems are compared in Table 2.1.

Property	Oculus Rift Touch	HTC Vive	PlayStation VR
Price	750$ (+2nd camera)	800$	400$
Display	OLED	OLED	OLED
Resolution	2160 x 1200	2160 x 1200	1920 x 1080
Refresh rate	90Hz	90Hz	120 Hz
Field of view	110°	110°	100°
Tracking area	1.5m x 3.4m (5.1m^2)	3.5m x 3.5m (12.25m^2)	3m x 1.9m (5.7m^2)

Table 2.1: The Oculus Rift, HTC Vive, and PlayStation VR are compared. The displays are similar, but the HTC Vive has a much larger tracking area. The data is taken from the manufacturers and Digital Trends [HTC17a] [Dig16] [Dig17] [Son16].

The displays are quite similar. However, the HTC Vive has the largest tracking area and is the most expensive of the listed products. The Rift and Vive require a computer with a strong graphics card, which is about 1000$ [Log17]. The PlayStation VR requires only a PlayStation 4, which costs roughly 350$ plus PlayStation Move controllers for another 100$. So, a good VR experience can still be expensive, at least for the average consumer, but enthusiasts and companies can easily obtain a decent VR system. VR has not yet arrived at the mass market, but maybe one game could achieve this like Pokemon Go did for AR in 2016 [Nia16] – hopefully in a more sophisticated way and with more persistence.

Data collection of the VR platforms is becoming more and more important. Among other data like location information, Oculus collects "Information about the games, content, or other apps installed on your device[...]" and "Information about your physical movements and dimensions when you use a virtual reality headset" [Ocu17]. Both Valve with SteamVR [Val17b] and Sony with PlayStationVR on the other side, only collect personal billing information. Therefore, the data collection of Oculus can be a deal breaker, especially for professional uses.

Many current VR systems aim for seated or standing experiences and have only a small tracking volume. This is perfect, e.g. for sitting in a cockpit and flying a plane. These applications are especially prone to cyber sickness, which is caused through perceiving visual movement without feeling the expected forces [Dra98]. There is no general solution for this cause of cyber sickness. Mapping real walking to virtual locomotion provides some relief but creates other problems, for example how to deal with a limited tracking area. In experimental set-ups, users walked in place [TDS99] or were strapped to devices to keep them on the spot [IYT07]. There are treadmill-like input devices, which suffer from slow and loud mechanics [Iwa99]. Then again, there are also systems like the Virtualizer [Cyb13], where users walk almost naturally on a slippery plate. Most games and experiences for the HTC Vive pursue another approach: to adapt the virtual space to the limited real space. Users can specify a free space within the tracking space, which is the so-called chaperone. When the users are too close to its bounds, a transparent grid is displayed as a wall in the virtual environment.

In “VR Funhouse” by NVIDIA [NVI16], “Space Pirate Trainer” by I-Illusions [I-I16], and “The Brookhaven Experiment” by Phosphor Games Studios [Pho16], the players can move in a limited virtual space and are teleported between the levels. In “Tiltbrush” by Google [Goo16], users can move their ’paintings’ faster than themselves. Therefore, they just have no reason to walk. When the accessible virtual space is much larger than the real space, the users usually teleport to a new location by pointing with a controller, as they do in “The Lab” by Valve [Val16].

A variety of different input methods for grabbing in VR is described. The first one was the ‘bat’ introduced in 1988 by Ware [WJ88]. It is a space mouse with six DoF and a button. Hand-held controllers also have six DoF but can be mapped directly to their virtual representation, which makes interactions with them more natural [FHKH06]. Hands and gestures can be tracked, as well, for example with Leap Motion [LS16]. Tracking fingers is also possible, e.g. with data gloves [SZ94].

2.2 Interaction Techniques

This section describes many interaction techniques for VR, most of which have been proposed in the 1990s. The referenced work deals with questions like “How can users select and manipulate objects efficiently?” or “Which technique is the best to interact with small, fast, or overlapping objects?”.

The classical virtual hand implements a linear mapping from physical to virtual space [WJ88]. The Go-Go interaction technique enhances the virtual hand with non-linear extension of the virtual arm beginning at about 2/3 of the maximal physical arm extension [PBWI96]. The HOMER interaction technique combines ray casting with teleporting the virtual hand to the selected object in order to simplify rotations around the object axes [BH97]. Donaldson and Whiting developed another extension to the virtual hand interaction: They extend the hand grab volume along the bisector of head to hand and wrist direction [DW16]. Worlds in miniature [SCP95] is an interaction metaphor proposed by Stoakley et al., which gives users a miniature of the scene to interact with. For example, they can pick up an avatar representation and put it down somewhere else to teleport.

Cutler et al. proposed multiple one-handed and two-handed interaction methods for workbenches [CFH97]. As an example, they introduced the pinch gesture for one hand to select a single object and move it, and they modified the pinch gesture for both hands to enable scaling as well. Song et al. condense the idea of the two-handed pinch gesture to a handlebar metaphor, which can be used with the hand tracking of the Kinect [SGH$^+$12].

The well-known 2D user interfaces (menus) from desktop applications can be used in VR as well. Rather than pointing with a mouse, the users point with a wand at the menu elements [SRH05]. However, Northway states that simple 2D-menus should be avoided, especially floating head-locked ones because they break the presence [Nor16a]. Instead, they should be part of the virtual environment, for example as objects to interact with. One example is “Fantastic Contraption” [Nor16b], where the players can build vehicles, which move around obstacles and into target areas. The parts they need to build those objects can be taken from the cat-shaped toolbox shown in Figure 2.1. Changing the scene, for example to a level where players can select and load previous games, is done by putting on a virtual helmet. A motion like pulling an arrow from a quiver on your back can be used as a shortcut. In “Fantastic Contraption” for example, the players can pull the last used part from their back and do not need to walk back to the cat. These two interactions, to change the level with a helmet and to pull a copy of the last used object from your back, may become standards like Ctrl + Z to undo and Ctrl + Y to redo in desktop applications.

2.3 Movement Techniques

This section is about movement of users in a virtual environment. Different techniques have been proposed for different applications. Some cause less simulator sickness, others cause less disorientation, and others are more accurate. Some techniques let the users walk naturally in a limited physical space and use inaccuracies in human perception to extend the accessible virtual space.

Flying techniques require a target direction and a way to change the velocity. Teleportation changes the user’s location instantaneously to e.g. a ray-cast hit point. The users can act as input devices by tracking their physical walking or with the leaning technique. Bowman et al. compares these movement techniques and many others [BKH97]. They find that pointing directed steering is more comfortable but slightly less accurate and harder to learn than gaze directed steering.

Redirected walking scales the amount of virtual rotation and movement compared to the physical rotation and movement. This allows the users to explore virtual spaces much larger than the tracked space [RKW01]. This technique is hardly noticeable when applied within certain limits [SBJ$^+$10]. The flexible spaces approach by Vasylevska et al. exploits change blindness by using a dynamic layout of the virtual rooms and corridors to fit more virtual space into the physical space [VKBS13]. The change blindness can be increased with twisted corridors. The overlap of rooms can then be more than 60% and still be unnoticed [VK15].

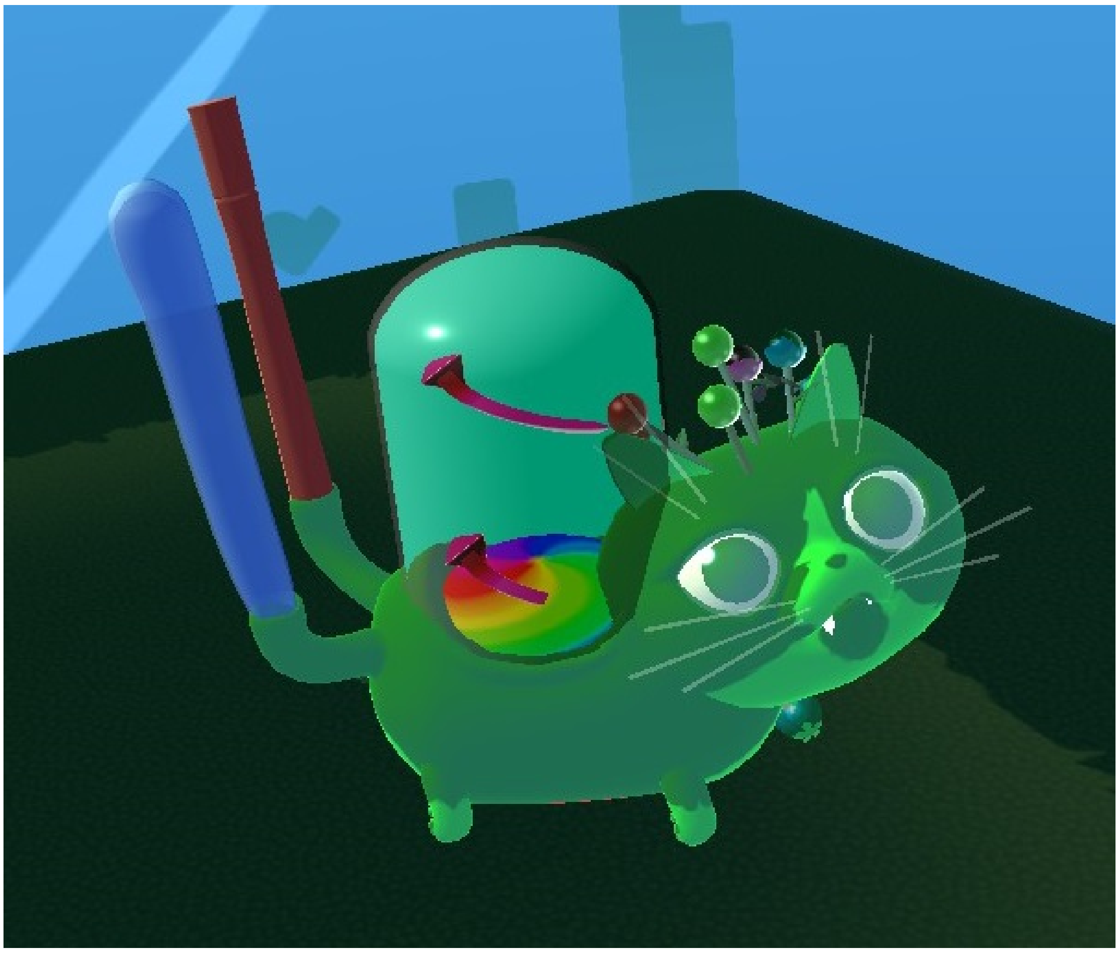


Figure 2.1: A cat as a toolbox in “Fantastic Contraption”. The balloons can be taken to build a vehicle. The needles can be taken to pop used balloons. The balloons and needles on the cat regrow.

XinReality is a Wiki for AR and VR. It contains a list of established and experimental movement techniques [Xin16]. One interesting technique is to freeze the game when the user approaches the tracking area bounds, rotate the scene with the user, and unfreeze the game [Rei16]. This technique displays the remaining space in front of the user and suggests a turn direction to prevent cord tangling. WalkAbout by Tekton Games is similar, but here the users freeze the game with a controller button [Tek16].

2.4 Presence

This section describes the feeling of presence in VR. It can be measured through questionnaires or indirectly through physiological reactions. Certain factors of a VR experience can improve the player’s presence. However, it can be broken easily by minor errors. For simplicity, social presence is ignored here.

Presence can be defined in different ways. One popular definition is the "feeling of being there" in the virtual environment [SVdM99]. A good, immersive virtual environment features a strong presence. One basic principle is that presence comes from continuity and not from complexity [Fos15]. Therefore, less is more, especially when considering limited development resources. With a deep presence, players automatically follow the rules of the virtual world. Hackett et al. state that objects should be consistent. For example, if one object can be interacted with, all similar objects should be as well. Otherwise, the expectations of the users are broken what results in less presence [HHH$^+$16].

The depth of presence can be measured through questionnaires, for example the Presence Questionnaire (PQ) by Witmer et al. [WS98], by Slater et al. [Sla99], and Schubert [Sch03]. The igroup presence questionnaire [igr16] has some items from the mentioned questionnaires plus two more. With its 14 items, it is rather short. The PQ used by the UQO Cyberpsychology Lab is based on the one by Witmer, but it is shorter with 24 items [UQO13]. It is controversial whether arousal, measured through e.g. heart rate and skin conductivity, is linked to presence or not [DKFD00].

Slater et al. found that dynamic shadows improve the presence [SUC95]. Steed et al. found that an avatar might have an important influence on presence and embodiment, but they could not get statistically significant results [SFL$^+$16]. Usoh et al. compared natural walking, walking-in-place, and flying [UAW$^+$99]. Flying results in significantly worse presence than walking-in-place and natural walking. Natural walking creates a better presence than walking-in-place. They also found that people who play games more often feel less present.

Some developers of current VR experiences give insights in their practical work, for example at the Virtual Reality Developers Conference, Vision VR/AR Summit, Oculus Connect, or Steam Dev Days. Padget and Hamm recommended some ways to improve presence at the Vision VR/AR Summit 2016 [PH16]. From their experience in real-world applications, they found that an avatar helps improving the presence. However, there is an uncanny valley in avatar quality, which means that a less realistic avatar can be better. The avatar should match the expectations of the users. Therefore, developers should either provide different realistic hand models with various sizes and skin tones or simply provide stylized hands, gloves, or tools. Since current VR projects are rather small compared to AAA games, the latter solution is more popular. In the case of the Oculus Rift and HTC Vive, showing models of the controllers in VR is fine for many applications. Huebner and Johansson suggest replacing the avatar's hand with the grabbed object [HJ16]. Padget and Hamm recommend avoiding arms or even full-body avatar because pose estimation is very difficult and inaccurate with the Rift and Vive. A failing estimation can easily break the presence [PH16].

2.5 Feedback

Feedback, in this context, means that a system sends information to the user, instead of only receiving inputs. Among others, the feedback can change the users' performance and feeling of presence. One and the same information can be transmitted to different senses. Typical feedback types are: optical, haptic, and auditory. There are also experimental devices for olfactory and gustatory output. Optical output is usually the main source of information in VR experiences, but auditory and haptic outputs can be very important. The most common forms of haptic feedback are vibrations (vibration alarm), forces (force-feedback joystick), and tactile feedback (braille display).

The experience of Hacket et al. from “Tilt Brush” shows what happens when feedback is entirely missing. A completely black or white environment makes the users feel lost. Therefore, they recommend to have at least a ground and sky-box [HHH$^+$16]. Donaldson and Whiting recommend to fade to white and play an acoustic signal in order to reduce the simulator sickness caused through teleportation [DW16]. They also suggest fading out and in when users expect acceleration.

The research of force feedback input started in 1971 with Project GROPE [BJOYBJK90], around the same as the first HMD. Its aim was and is to let chemists feel the forces of molecules. Okamura et al. proposed a model for the macroscopic effects of microscopic surface texture. To be more exact, they describe the vibrations caused by tapping, stroking, and puncturing various materials [ODH98] and how vibrating devices can display feedback based on this model [OCD01]. Vuskovic et al. describe an advanced haptic feedback model for cutting through tissues in a surgeon simulation with a scalpel-like input device [VKSR00]. Some data gloves can produce vibrations for individual fingers, and they enable finger and sometimes arm tracking. For example, Manus-VR [Man16] and Cyberglove Systems [Cyb17] produce such gloves. Other gloves are like an exoskeleton and can apply strong forces to the fingers [Bur99]. Some gloves focus on tactile feedback [SZ94].

Sallnäs et al. show that the force feedback produced by a pencil-like input device can significantly improve task performance and presence [SRGS00]. The Vive controllers can give haptic feedback in the form of vibrations. Padget and Hamm [PH16] claim that the vibration feedback of current VR systems has a huge effect on presence but it looks like it has not yet been quantified.

Hoffman et al. found that using real objects in a virtual environment improves the realism [HHS$^+$98]. This is done by tracking real objects and mapping their position and orientation onto a virtual object with a similar appearance. Such objects are called props. For example, a real fire extinguisher can make a fire fighter simulation in VR much more realistic. HTC offers additional trackers for the Vive [HTC17b], which can be attached to an extinguisher to create a prop. An industrial robot attached to props can produce realistic force feedback [Joh16].

Padget and Hamm recommend using vibration feedback when picking up objects or moving through static geometry in order to include the haptic sense for a better performance and presence [PH16]. Donaldson and Whiting recommend vibrating the controller when it is within an object [DW16]. Also, they suggest highlighting objects that can be grabbed.

For audio, Padget and Hamm suggest adding multiple sound sources for ambient sounds to the scene at different locations. This is a simple way to make the ambient sound change with the head movement [PH16]. Podkosova et al. propose a sound model for real-time use, which can simulate sound reflections [PUK16]. With this model, users are guided around walls significantly better than with the baseline model, which only simulates distances with attenuation and obstacles with a low-pass filter.

2.6 Point Clouds

Laser scans aim to capture buildings with sub-millimeter accuracy and entire landscapes with centimeter accuracy. The results are large point clouds, often with billions of points. Rendering such massive amounts of points requires special techniques like level-of-detail (LoD) handling.

The basic idea of the point-cloud rendering algorithms is that the points are not seen as objects without volume, but they are assumed to be samples of a 2D-surface. The algorithms aim to reconstruct this surface with an approximation. The onscreen size of this approximation is often determined through the LoD level of the rendered points.

QSplat [RL00] was the first system that could handle such amounts of data. It uses a bounding-sphere hierarchy, in which a bounding sphere represents the average of its contained points. It traverses the hierarchy downwards until the covered area on the screen becomes too small. The last traversed bounding sphere is seen as a surface sample and therefore rendered as a square or circle primitive (splat). With this approach, it can render the necessary points efficiently for different levels of detail, and it guarantees a hole-free reconstruction of continuous surfaces.

Sequential point trees [DVS03] are a data structure that gives the computational effort of rendering point clouds completely to the GPU. Sequential processing on the GPU replaces the hierarchical rendering traversal. This requires the entire point cloud to be loaded into the graphics card memory.

Layered Point Clouds (LPC) [GM04] is an approach that stores a subsample of the children nodes' points in their parent node. Its hierarchy is traversed until a suitable number of points for the current LoD is reached. The slow upload of the subsamples to the GPU is done only once. Afterwards, the points can be rendered in parallel by the GPU.

XSplat [PSL05] introduced a block-based sequential multiresolution point hierarchy with an efficient LoD-block paging mechanism and dynamic mapping into video-cache. This makes for an efficient out-of-core system. However, its internal structure creates some memory and GPU overhead.

Instant points [WS06] is a system that can render unprocessed point clouds with only little preprocessing. It is based on sequential point trees but reduces the memory consumption through less overhead and omitting e.g. normals. The system uses two octrees, one for visibility calculations and the other one for the subsamples. Like in LPC, the instant points store sub-samples in the nodes, but the sub-sampling is defined by the octree. This means that the traversal depth determines the size of the splats.

The octree structure by Wand et al. [WBB$^+$07] distributes the points in leaf nodes and simplified multiresolution representations in inner nodes. The depth of the octree is defined by a maximal number of points per leaf node.

Surfels [PZVBG00] are new primitives for rendering points with normals. They are oriented disks or ellipses and have the attributes like depth, texture color, and normal. They require pre-processing but can be drawn quickly and with high quality.

Surface splatting [ZPVBG01] is a rendering technique that uses filtering for projected points. The points contributing to a pixel are convolved with an elliptical, weighted average filter in screen space. With this, points farther away from a pixel have less influence on its color. Also, a depth threshold for points ensures that only points belonging the same surface are filtered.

Botsch et al. [BHZK05] proposed an approximation of the filtering in surface splatting. The filtering is split and calculated in multiple render passes, which allows them to render their point clouds on the GPU while the original surface splatting was done with a software renderer.

Scheiblauer [SP11] takes a different approach. Surfels and other previous techniques for point rendering require normals. However, most point clouds do not provide this information. Therefore, Scheiblauer suggested drawing points as screen-aligned circles. The rest of the rendering pipeline works like surface splatting but with a Gaussian weighting.

Auto Splats [PJW12] is another approach to deal with the missing normals. It calculates the normals in screen space without any pre-processing involved. This allows for higher image quality with surface splatting than with screen-aligned circles.

Schütz and Wimmer [SW15] present a technique that uses a fast interpolation to fill in holes in point clouds without occluding details. The points are rendered as flat squares in the beginning. Then, the fragment shader assigns a depth value in a way that a sphere, cone, or paraboloid is created. In the end, the normal depth test creates sharp boundaries. The final result is similar to a Voronoi diagram.

Wand et al. [WBB$^+$07] propose a method for editing point clouds, which can be adding, deleting, and tagging points, for example. This makes it necessary to select the points before such an operation. Wand et al. show how this can be done when storing the points in leaf nodes and the LoD information in the inner nodes of an octree. This has one major drawback: points that are currently not loaded are ignored in the selection. Another approach, modifiable nested octree [SW11], uses an additional octree for the selection, which eliminates this drawback.

CHAPTER 3

System Design

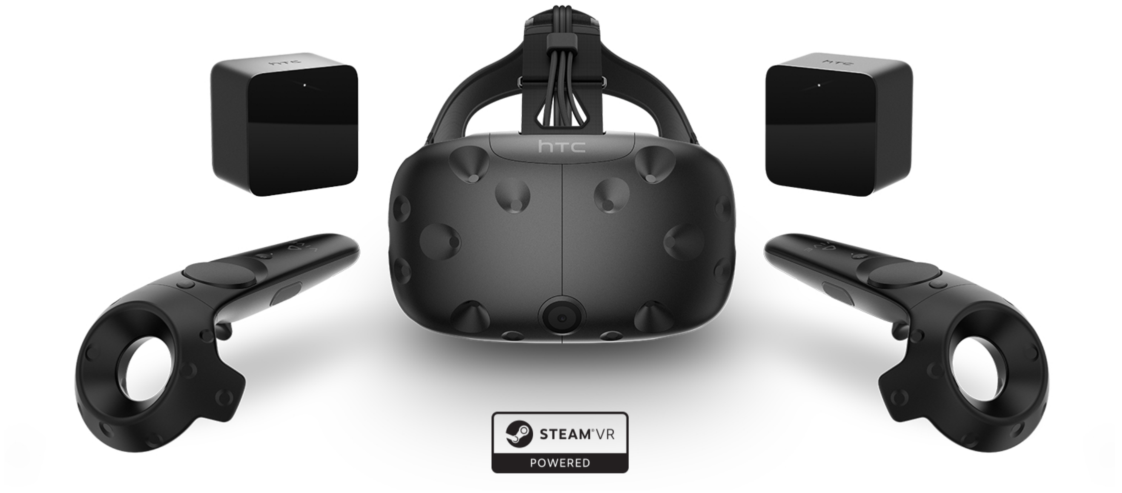


Figure 3.1: The HTC Vive comes with one HMD, two lighthouse base stations, and two controllers. Image by BagoGames [Bag16].

This chapter describes the applications created for this thesis and the involved design decisions. The applications are a simple basketball game, the grabbing test, and point-cloud editing tools for desktop and VR. The basketball game is a VR game in which the players throw a ball into a basket. In the grabbing test, the users grab one of four balls to score. Different feedback types

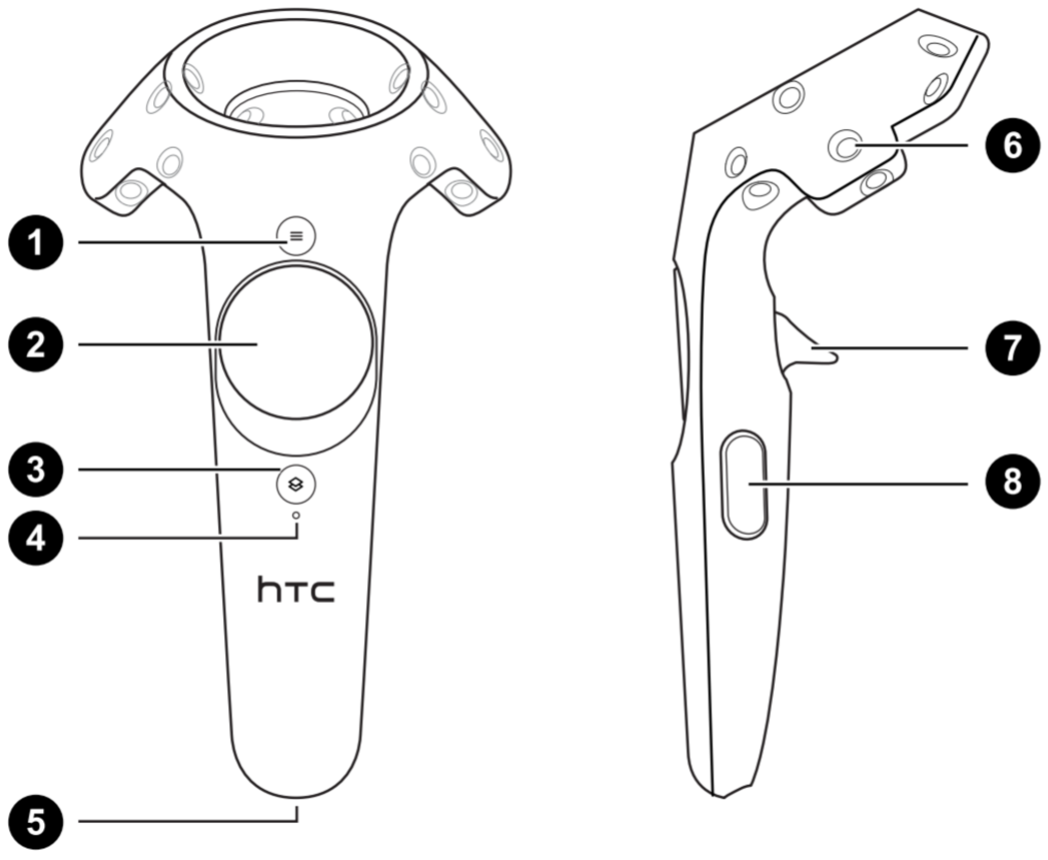


Figure 3.2: The HTC Vive controller: (1) menu button, (2) trackpad, (3) system button, (4) status light, (5) USB charging adapter, (6) tracking sensor, (7) trigger, (8) grip button. Imagen taken from Vive PRE User Guide [HTC17a].

support the users in their attempts to grab one of the balls. In the point-cloud editor, the users can delete points from the laser scan of a building.

The applications are developed and run on a PC with Microsoft Windows 10, HTC Vive, NVIDIA GeForce GTX 1070, 16 GB RAM, and an Intel i5-4690K. Figure 3.1 shows the HTC Vive, and Figure 3.2 illustrates the buttons of its controllers. The applications of this thesis use the trigger for grabbing and the trackpad for advanced interactions.

3.1 Basketball Game

The basketball game demonstrates the level of human-machine interactions that can be achieved with the current technology and minimal development resources. The goal is to create a simple yet fun game that creates a strong feeling of presence. Additionally, it can be used as a basic testing suite for future research in interaction and movement techniques. It is a simple but complete

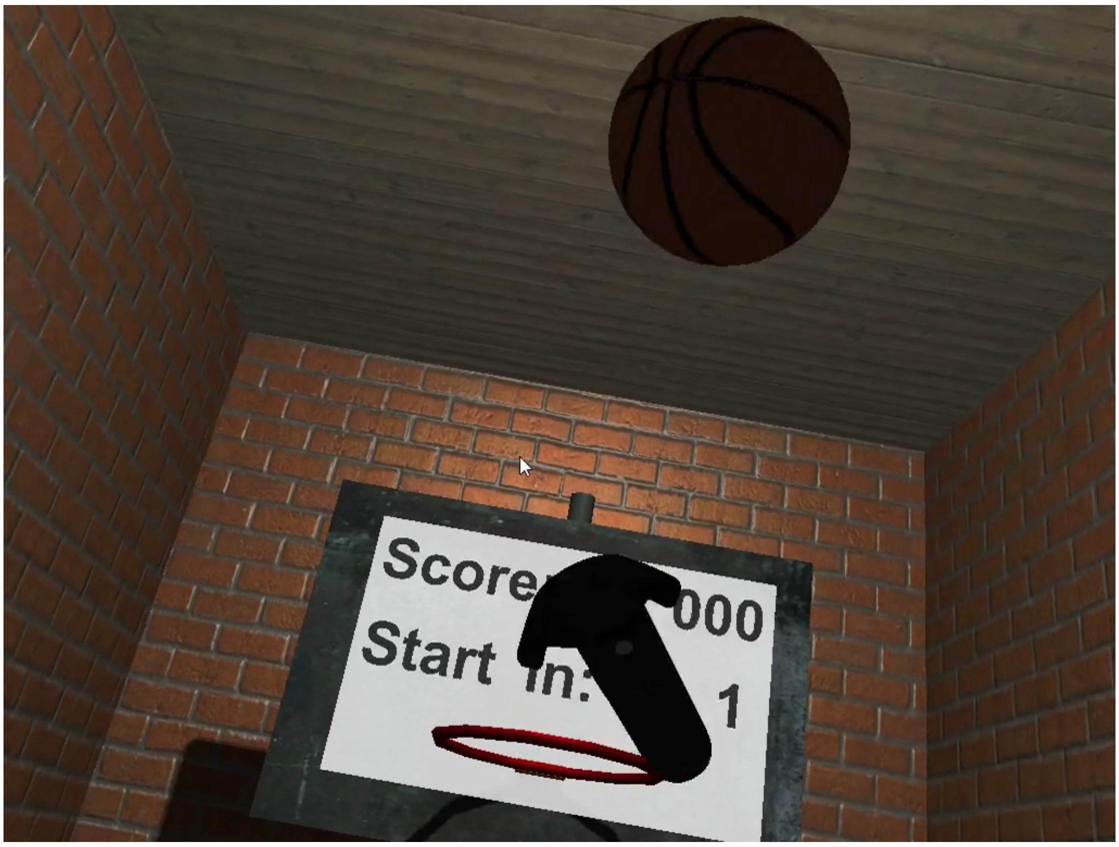


Figure 3.3: Throwing in the basketball game.

game containing a higher-level goal. Figure 3.3 displays a user throwing the ball towards the basket.

As the name suggests, the basketball game is about throwing a ball as often as possible into a basket. In this simple version, only one player throws a single ball again and again into one basket. There is a warm-up phase that lasts until the player has scored three times. Five rounds follow, each 60 seconds long. The basket behaves differently in each of these rounds:

1. the basket stands still in the center of the goal room.
2. the basket jumps to a random position in the goal room after each goal. All players get the same positions in the same order to ensure comparable scores.
3. the basket moves constantly from the left to the right end of the goal room based on a sinusoid function.
4. the basket moves in a circle through the goal room.
5. the basket moves fast in a circle through the goal room.

The basketball game can also be started in a short mode with a command-line parameter. Then, it contains only round 1 and 4. This is useful for a short introduction.

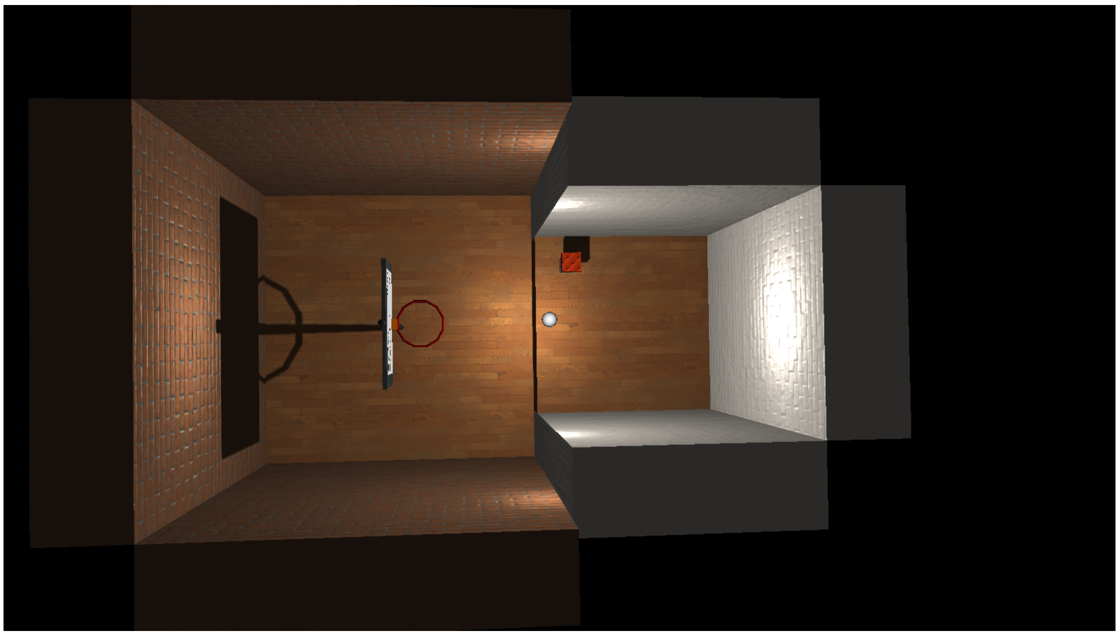


Figure 3.4: The layout of the rooms in the basketball game. The smaller room is the tracking room where the player stands, grabs the ball, and throws it. The basket is in the bigger room, the goal room.

The user stands in a room of 2.9m x 2.9m, which is about the size of the Vive's tracking area. Its height is 4.2m, which should be enough to not cause claustrophobia, but still let its contained light source illuminate everything nicely. In the user's room, the basketball lies on a pedestal and can be grabbed by users of all sizes. The tracking room is enclosed by three walls and the goal room, which is 4.5m x 4.5m wide and 5.5m high. The walls also give a visual hint of the physical space without breaking the presence. The goal room is set roughly one meter below the user's room to discourage the user from entering. This height difference is just one meter because the edge should not occlude anything, and it should not cause fear of heights. The goal room contains the basket and a score board. In an average game, the player has to throw the ball about 4m far and 0.5m above the head. Figure 3.4 shows the rooms from above.

A goal counts only when the ball enters the basket from above and falls down to at least the level of the ring. This is adapted from the official basketball rules [FIB17]. Since the goal room is not accessible for the players, they cannot take the ball back. Therefore, the ball is re-spawned at the pedestal one second after it scored or three seconds after it hit the ground of the goal room. It is highlighted in red to signal the imminent re-spawn.

The used interaction technique is the virtual hand. This means that the position and orientation of the real HMD and controllers are directly applied to their virtual representations, the camera and controller models. With this simple direct mapping, the users need to learn very little before they can start playing. Only a single button of the controller is needed. The users can grab the ball by keeping the trigger pulled and moving the controller near the ball. The ball then moves as

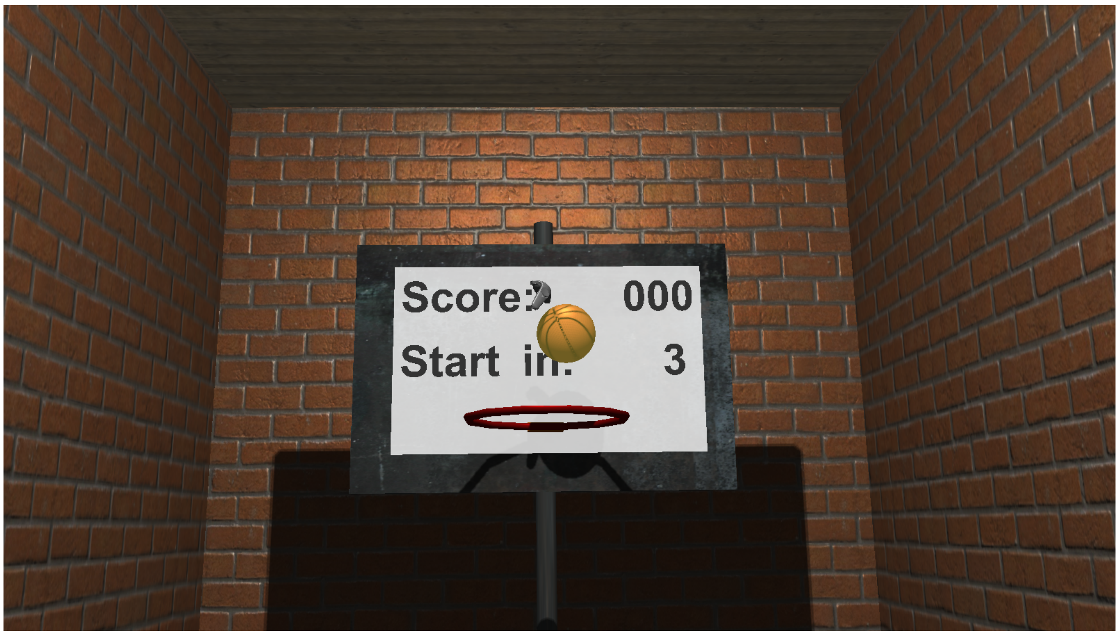


Figure 3.5: The Go-Go technique makes it too easy for users to score or is unintuitive for throwing.

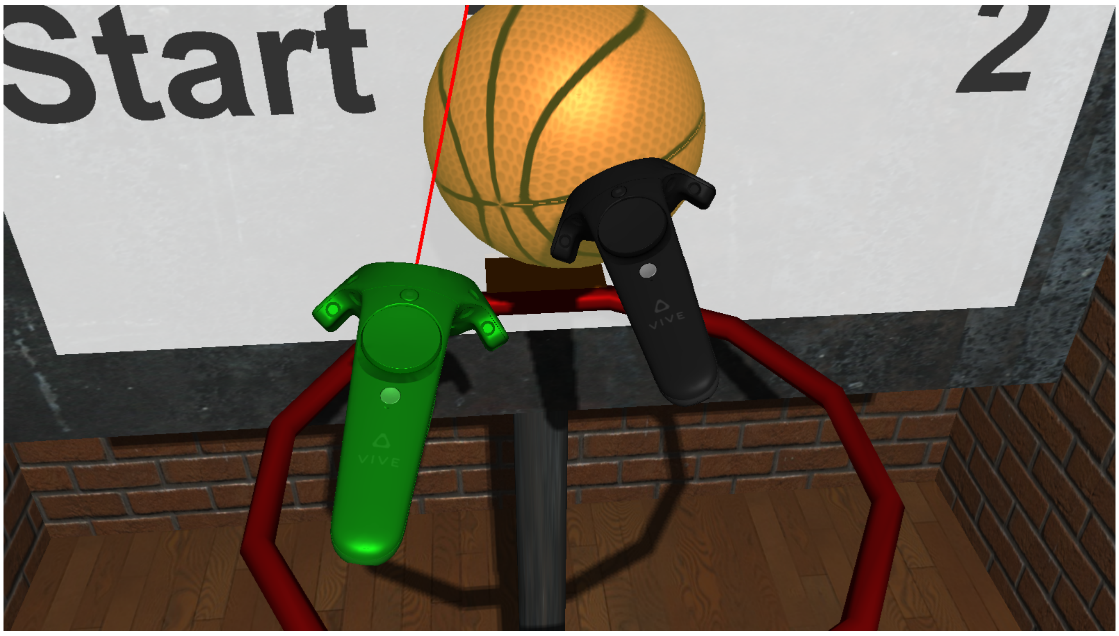


Figure 3.6: The flying makes it too easy for users to score, and it can also cause cyber sickness.

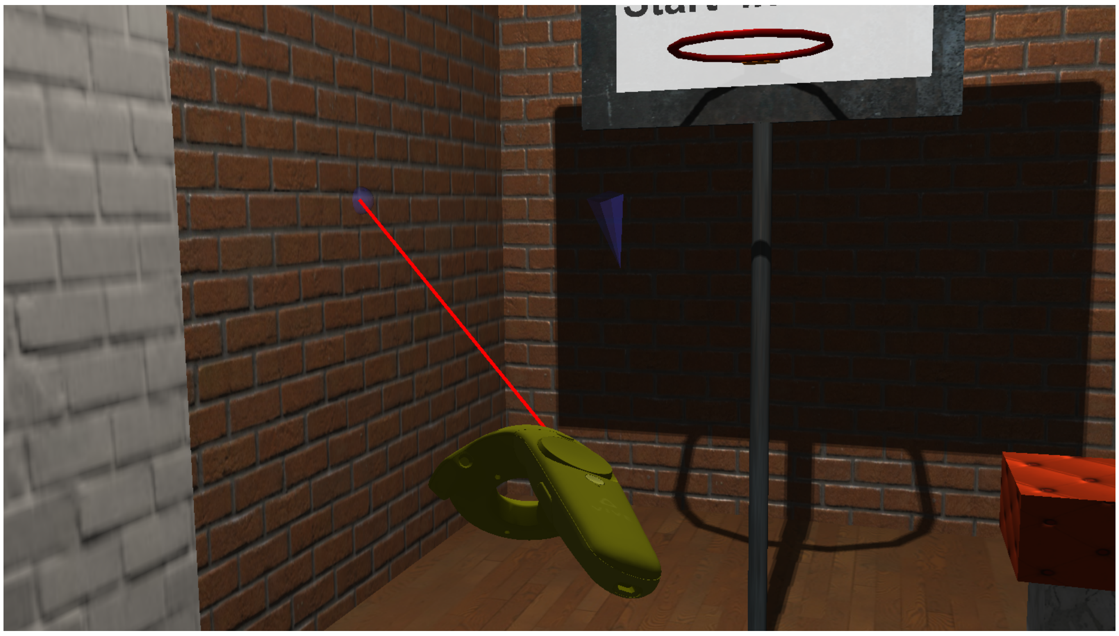


Figure 3.7: The blue cone represents the HMD after the teleportation. When using a ray-cast hit point as the target position, users cannot leave such a closed in-door scene.

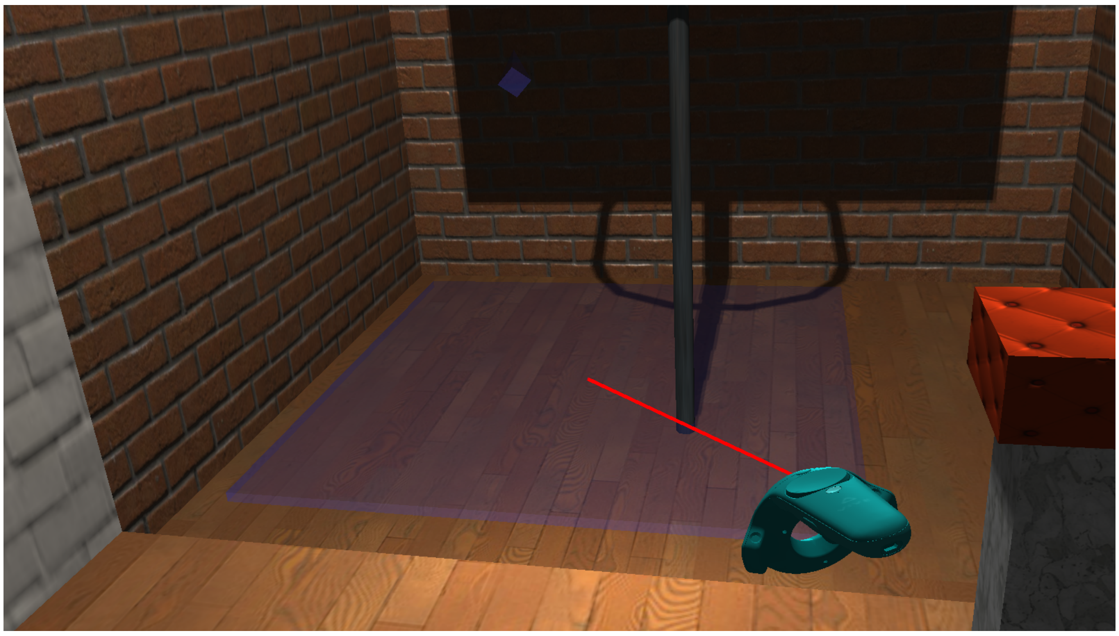


Figure 3.8: The blue field represents the tracking area after the teleportation. This makes it easier for users to make plans but can decrease the presence.

if attached to the controller. When the trigger is released, the ball is detached and moves with the velocity of the controller. This way, the trigger state encodes the grabbing intention of the players.

Some other interaction and movement techniques have been tried, mostly out of curiosity and to test the software architecture. As an example, the Go-Go technique (exponential extension of the virtual arm) made the basketball game too simple, because users could simply let the ball fall into the basket. Depending on the settings, the extension could also be short enough to not break the game but still make the throwing less intuitive. The Go-Go technique is shown in Figure 3.5. Another experiment was flying by pointing with the controller in a direction and pressing the trigger. Looking down while flying up is very nauseating, even for a developer who is used to VR. Flying in the basketball game does not make sense. It breaks the game since they can just fly to the basket and drop the ball, or the players are confined within the tracking room. Figure 3.6 depicts a user who has flown to the basket. Teleportation moves the current view to the hit point of a ray cast. A cone above the hit point represents the position of the HMD after the teleport. Allowing teleports to vertical walls and the ceiling can be nauseating but also very fun, especially when changing the virtual gravity accordingly. In closed indoor scenes like in the basketball game, the players cannot leave the scene through this kind of teleportation. Figure 3.7 displays the HMD position after the teleportation. Displaying not only the HMD but also the tracking area position enables 'strategic' teleporting but might be bad for the presence. Therefore, users can teleport just far enough to reach their target but are reminded of the physical space limits all the time. Figure 3.8 shows the representation of the tracking area and HMD after the teleportation.

For a deep presence, an experience should serve as many different senses as possible. With the HTC Vive, the possible senses are optical, auditory, and haptic. The graphics are as good as possible in the limited development time. High-resolution textures and real-time shadows are a must, as well as maintaining 90 frames per second. Therefore, complex shading and effects have to be omitted. There should be some representation for the controllers. Some other games use hand or glove models. Using such representations can cause problems when users have to work with hands of a strange color or size [PH16]. Many applications, like the basketball game, use the controller models provided by Steam VR. The model quality is good and can be further enhanced with a texture, occlusion, and specular mapping.

For the auditory output, sound sources in 3D space are playing sound samples. Being the most important object in the scene, the ball must have a fitting bounce sound. This sound is played at the contact point when the ball collides with any other object. The pitch is changed randomly by up to $\pm 40\%$ to create more varying sounds. Additional sound feedback is given in the form of a siren located at the basket. which is triggered when the player scores. Also, a pop sound is played at the pedestal when the ball re-spawns. There are two sources of ambient sound located outside the rooms. The ambient sounds are rather quiet and placed in roughly opposite directions from the tracking room. This improves the player's orientation, as suggested by Padget and Hamm [PH16]. Surround sound would be simple to implement. Unfortunately, the Vive's HMD has only one port for stereo sound.

Haptic feedback is given by letting the controllers vibrate when they overlap with other objects. The vibrations are stronger when the relative velocity is higher. A short and strong vibration is triggered when a controller hits a virtual object, where the vibration strength depends on the impact strength. They also vibrate when the user can grab the ball. One experiment was to let both controllers vibrate with a certain pattern, e.g. when the player scores. This pattern can be, for example, a sinusoid or saw function. However, hallway testing showed that this feedback for scoring is considered unintuitive and annoying. Therefore, it was removed in the end. Vibration patterns for the other haptic feedback sources were removed as well, because they lower the perceived maximal vibration strength, which is already too low.

The controllers have physics colliders to allow for dribbling without grabbing. Additionally, there is a sphere collider for the head on the HMD. This prevents balls from flying through the head, which is the main source of clipping as the users usually avoid sticking their heads into walls. The head collider also allows them to shoot the ball with their head. The comments of users who tried it indicate that this increases the presence a lot.

3.2 Grabbing Test

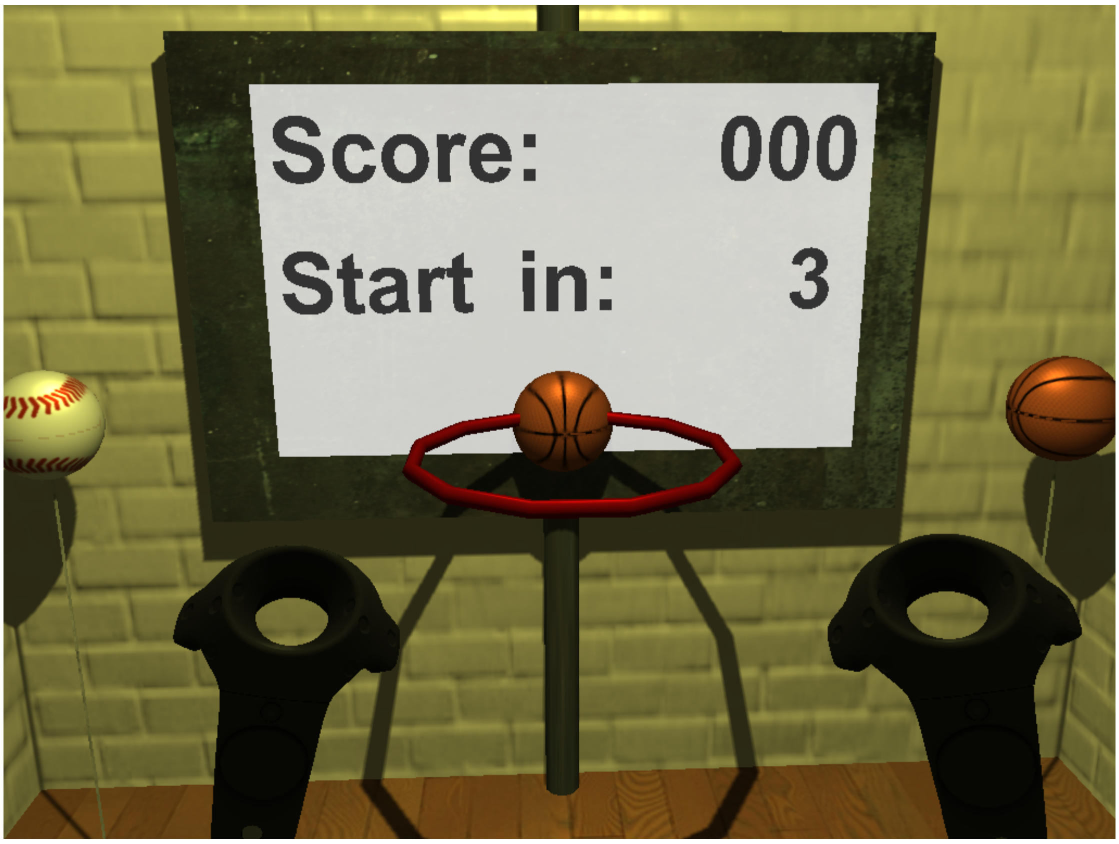


Figure 3.9: In the grabbing test application, there are four different balls, one in each corner of the room. The ball at the basket shows, which one has to be thrown into the basket next.

While the basketball game is suited for a first analysis of the quality of human-machine interactions, we found that the focus is too much on throwing to allow for an analysis of feedback for grabbing. Therefore, another application is needed. This application is called grabbing test and shares the concept of the warm-up phase and round structure, the sound, vibrations, and physics with the basketball game.

In the grabbing test, the basket is easily accessible in the tracking room and the goal room is removed. Four balls lie on top of thin rods in each corner of the room. The next ball to be thrown into the basket is displayed on the basket, as illustrated in Figure 3.9. After each goal, the target ball changes and all balls are reset. To ensure comparable results, the same seed for the random number generator is used for every tester. This way, the basket always displays the required balls in the same order. The scoring is simpler than in the basketball game. The ball must only touch the volume in the ring without being grabbed. This is because the users in the hallway tests sometimes released the ball only below the ring.

The grabbing is more difficult in the grabbing test. In the basketball game, the users can pull the trigger first and then move the controller to the ball in order to grab it. In the grabbing test, they must first move the controller to the ball and then pull the trigger. Different kinds of feedback indicate whether the controller is close enough to the ball. If the user fails to grab the ball at the right time, the controller may hit the ball and push it from the rod instead. The ball will then roll and bounce on the ground causing the player to lose time while chasing the ball. The feedback types can be set with a command-line parameter when starting the application:

1. No feedback at all.
2. Haptic feedback, which is the same vibration model as in the basketball game, described in Section 3.1.
3. Optical feedback, which is highlighting like in the basketball game, described in Section 3.1
4. Both haptic and optical feedback.

One session consists of a warm-up phase until the user has scored three times, followed by four rounds of 30 seconds each with increasing difficulty. Increasing difficulty means that the grabbing volume around the controller shrinks after each round. In the beginning, it is scaled to 150%, which shrinks down to 105% in the last round. In Figure 3.10, the grabbing volume is shown as a transparent blue version of the controller model. The scaling origin is placed so that the distance between the surfaces of the controller and the selection volume is almost the same at any point.

The point-light source is yellow during the warm-up phase and white during the test. This makes it possible for the users to notice when the game is over without looking at the remaining time. However, the yellow light was obviously not striking enough. Many testers, especially the most motivated and concentrated ones, did not notice it. Also, there was an experiment with a short green flash when a new round begins. Some testers were confused and thought the test was already over. Therefore, the green flash was removed.

3.3 Point-Cloud Editing

Point clouds contain information, usually colors, sampled at different points in space. Going one step further, people may want to add, modify, tag, and delete certain points manually. This

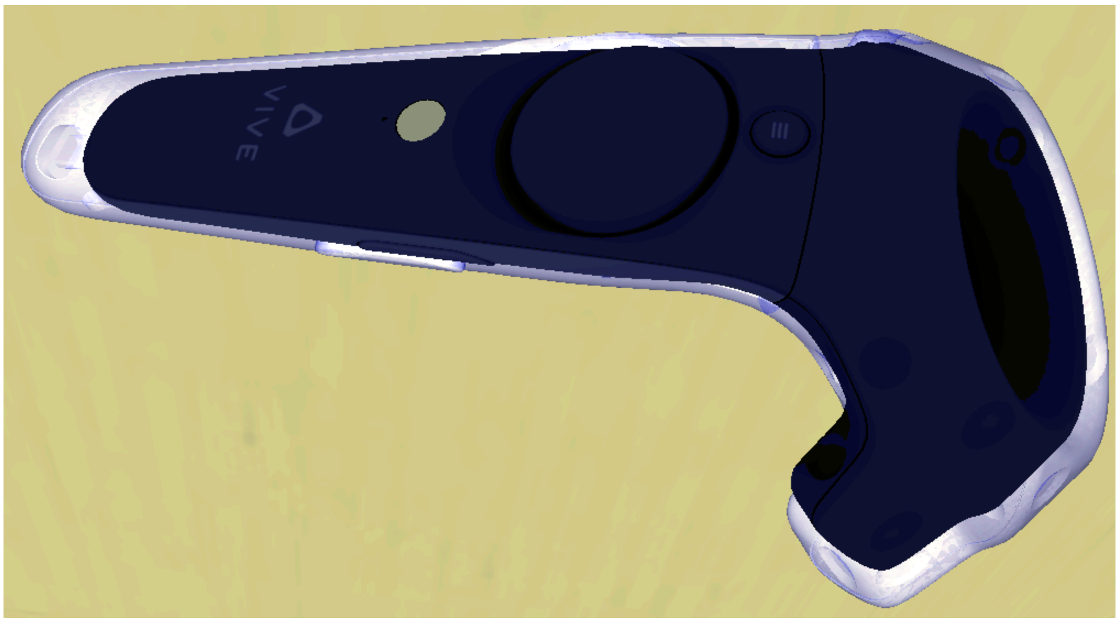


Figure 3.10: The transparent blue model is a visualization of the grabbing volume. The balls can be grabbed when they intersect with this volume. If the user fails to do so, the ’real’ black controller can push the balls away.

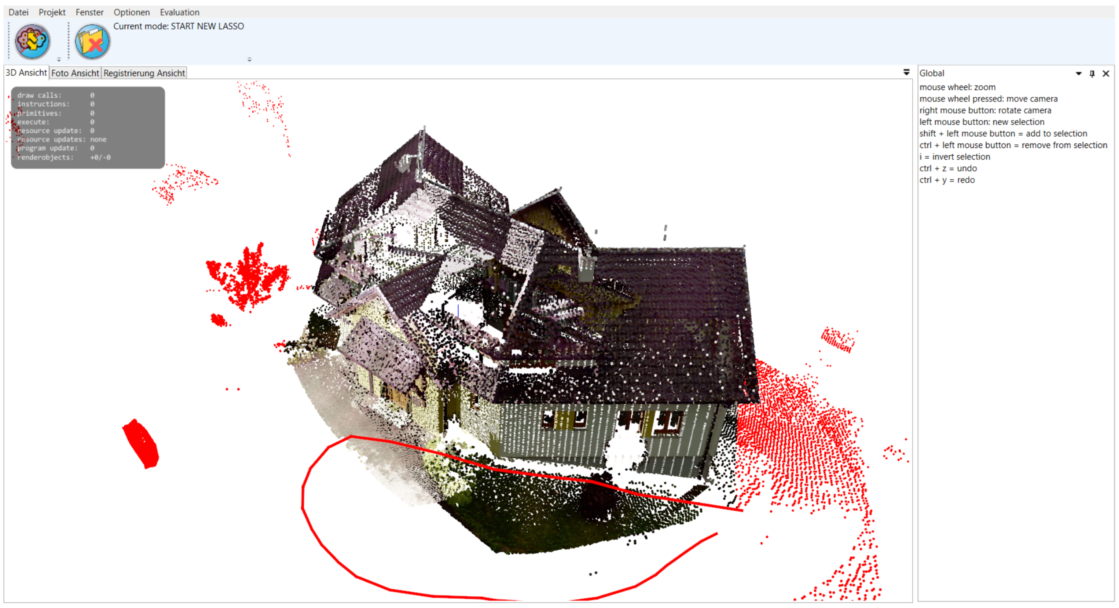
section explains how this editing can be done efficiently in VR. In the case we study in this thesis, the goal is to delete outliers and vegetation from the laser scan of a building. A user study is performed to find the differences in efficiency and usability between the desktop and VR versions of a point-cloud editing tool. The users were asked to select all points which do not belong to the house. Their selections are compared to a ground truth. The selections and the ground truth base their coordinates on the most common reference system in Austria, which uses Gauß-Krüger coordinates [Krü12]. This allows the comparison to work with different scans of the same object but only one ground-truth selection. The comparison decides whether a point is: correctly selected, wrongly selected, correctly non-selected, or wrongly non-selected.

Both, the desktop and VR versions, use the Aardvark rendering engine with its data structures and rendering algorithms for point clouds. The point clouds can contain billions of points, which is too much data to fit into the main and graphics memory at once. Therefore, the applications need to be able to handle out-of-core data, more on this in Section 4.5. Reading and changing the point-cloud data must be done as fast as possible. However, the data cannot be changed in real time, even with current SSDs.

3.3.1 Desktop

The point-cloud editing application for desktops is a trimmed version of the Virtual Geodetic Mapper (VGM) developed at the VRVis [Rot16], which is shown in Figure 3.11.

The controls are as specified by a contractor of the VRVis, presumably to keep everything the



Current mode: START NEW LASSO

```
draw calls:        0
instructions:      0
primitives:        0
execute:           0
resource update:   0
resource updates:  none
program update:    0
renderobjects:     +0/-0
```

mouse wheel: zoom
mouse wheel pressed: move camera
right mouse button: rotate camera
left mouse button: new selection
shift + left mouse button = add to selection
ctrl + left mouse button = remove from selection
i = invert selection
ctrl + z = undo
ctrl + y = redo

Figure 3.11: The point cloud is rendered in the central part of the window. The controls are described on the right. The menu allows users to load point clouds and start the comparison. The red points are already selected. The points enclosed by the red line will be selected, too.

way the users are used to from other applications. It implements an orbiting camera like in Potree [Sch15] and CloudCompare [GM15]. The orbiting center can be moved by pressing the mouse wheel and moving the mouse. The viewing direction, or to be more exact, the spherical coordinates of the camera can be changed by pressing the right mouse button and moving the mouse. A new selection is made by pressing the left mouse button and drawing a curve on the point cloud. The start and end points are connected when the button is released. This lasso forms a polygon. The points that are inside the polygon on the screen plane, whether visible or occluded, are selected. The winding number algorithm is used to check whether they are inside or not [AM95].

Selections can be unified by keeping the shift-key pressed while releasing the left mouse button. When the ctrl-key is pressed instead, the new selection is subtracted from the previous selections. Pressing the i-key inverts the current selection. These interactions encode Boolean OR, AND, and NOT operations. The full version of the VGM provides additional operations, such as XOR, which are left out for simplicity. User actions can be undone and redone with ctrl+z and ctrl+y.

3.3.2 Virtual Reality

Point-cloud editing in VR is a novel technology. Therefore, the interactions used in this application are one main contribution of this thesis. Figure 3.12 shows the point cloud while a user selects points. Haptic feedback is used to support selecting single points.

As with the balls in the grabbing test, the point cloud can be grabbed with the trigger. The point

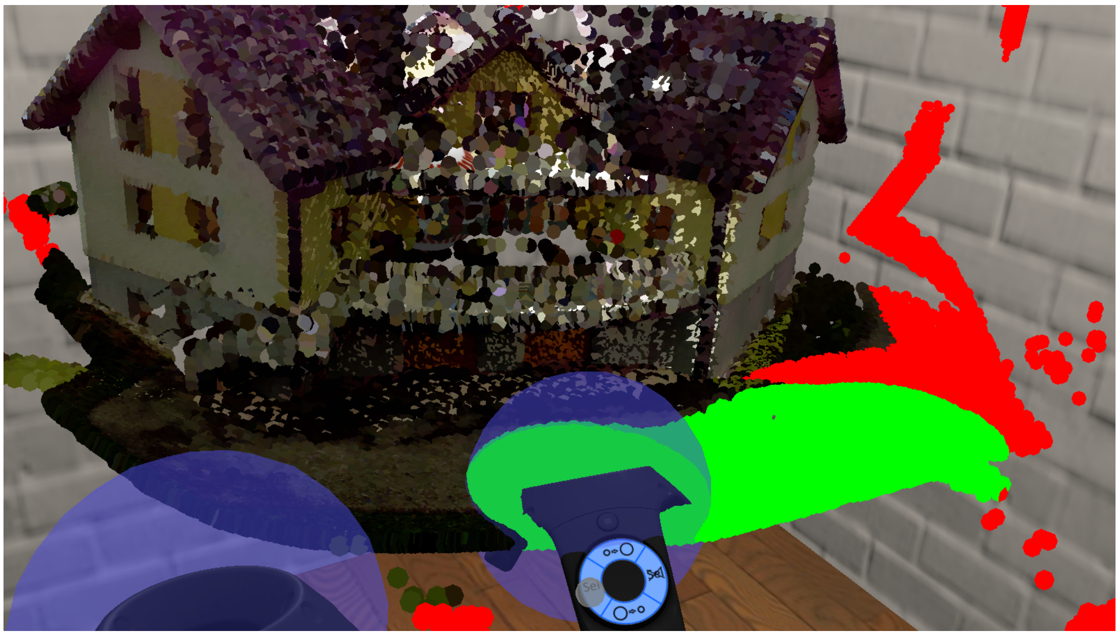


Figure 3.12: The user has already selected the red points. The green points are selected as soon as the trackpad is released.

cloud is then attached to the controller, which means that it is rotated and translated with the controller. When the point cloud is grabbed with both controllers, it can be scaled by moving the controllers together or apart. This is much like the pinch gesture known from touch-screen interactions, e.g., on smartphones. An important part of this gesture is that the controllers stay in their relative place within the point cloud, no matter how it is scaled, translated, or rotated.

Early hallway testing showed that some users seem not to understand the movement of single points when they scale the entire point cloud. There are three factors probably causing this confusion:

1. The scaling moves some points closer and the others farther.
2. The scaling is usually accompanied by a translation.
3. The points are rendered as disks. Due to occlusions, their size cannot be compared easily.

However, when the points become occluded by the walls, their movement becomes obvious. Therefore, the point-cloud editing is done in the same virtual room as the grabbing test. Another reason to introduce a virtual room is that the users would feel lost in a completely white or black environment [HHH$^+$16].

At first, the point cloud sat on top of a pedestal to offer a fixed point of interest. However, some testers found the pedestal annoying because it blocked their sight. The point of interest seemed unimportant, so the pedestal was removed, leaving only an empty room and the point cloud. The point cloud is reasonably scaled to give the users a quick overview before they start to interact.

A transparent blue sphere is attached to the center of the ring of both controllers. This sphere acts as a selection volume. Points inside this volume can be interacted with. Selecting points is done by pressing the trackpad, moving the controller through the point cloud and releasing the trackpad. The green instant selection visualization disappears, and the red selection appears as soon as the data is updated.

Points are selected by pressing the left part of the trackpad. When pressing the right part, selected points can be un-selected. The upper and lower parts can be pressed to scale the selection volume between diameters of about one centimeter to one meter. Users tend to move their thumbs too close to the edge. Then, the controller loses the thumb position and assumes the middle, which could cause a rather random behavior. Because of this, the center of the trackpad is ignored. The selected points can be turned invisible by pressing the app-menu button. This also starts the comparison of the user's selection with the ground truth.

The strength of the haptic feedback changes with the number of points in the selection volume. The difference between zero and one point is the most important, and everything beyond three points does not need to be felt very accurately. Therefore, a limited growth function is used. Hallway testing showed that the controllers are vibrating almost the entire time. This becomes quite annoying after a while. Also, the constant vibrations mess with the controllers' tracking by causing the inertial sensors to drift. Therefore, the vibration strength was limited at one fourth of the maximal strength.

CHAPTER 4

Implementation

The source code of the three applications for this thesis is available on Github: `https://github.com/ErlerPhilipp/VR_DA`. They are written in F# [F#05], a functional-first programming language on top of the Common Language Infrastructure (CLI) [Int12]. Because of the CLI, F# can use all libraries that are available for C#, such as Bullet Physics [Rea17] and OpenVR [Val15b]. However, these object-oriented libraries clash with the functional paradigm. Fortunately, F# also supports object-oriented programming.

4.1 Software Architecture

The software architecture is explained on the example of the basketball game. The grabbing test application is very similar. The point-cloud editor does not need physics, and is therefore mostly a sub set of the basketball game.

The most basic type is the scene object, for example balls and walls. It stores information about the physical properties, geometry, appearance, rotation, and translation of the object.

The applications consist of three main parts: logical scene, physics scene, and graphics scene. Each scene has its own representation of the virtual world, according to its responsibilities. The current logical scene containing the game state is passed to the physics and graphics scenes, which update their representation. This structure may seem unnecessarily redundant, but it creates manageable views of the whole scene. Additionally, it could be parallelized easily. The structure is illustrated in Figure 4.1.

The logical scene contains the game state with the game objects, sound sources, lights, and settings for the physics simulation. An updated game state is the result of a function taking the old game state and an event. As an example, it can receive a button press event from a controller. Then, it checks whether a ball is near the controller. If there is one, the ball is grabbed by returning the new game state with a changed object type.

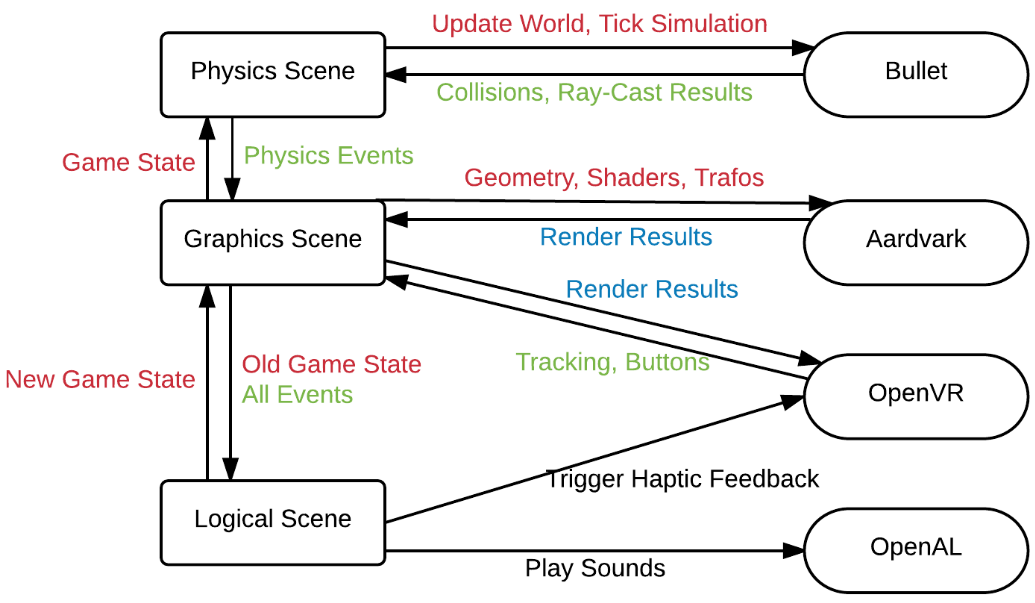


Figure 4.1: The applications consist of three main parts: graphical, logical, and physical scene. Information about the game state (red) are passed between the scene views. This information, e.g. position of objects, is also given to Bullet Physics and Aardvark. The rendering results of Aardvark are submitted to OpenVR to be displayed on the HMD. The old game state is updated with events from the physics simulation, tracking or other inputs. The logical scene also triggers haptic feedback and plays sounds.

The physics scene takes care of the physics simulation with its properties like gravity. It contains physics objects such as spherical rigid bodies for the balls and mesh colliders for the basket. The physics scene also manages ray casts and collision events.

The graphics scene not only handles the rendering, but it is also the interface between the hardware and the application. It manages the main loop, where it receives inputs, for example tracking data and button states. The inputs are converted to events and result in an updated logical scene. The physics simulation is also updated by the graphics scene. Its events then update the logical scene. The graphics scene creates additional events like the start and end of a frame for clean-up, as well as a timer event for time-dependent processes.

4.2 Rendering

Aardvark is an open-source rendering engine developed by the VRVis research center [VRV17]. It updates the rendering only when the relevant inputs change [HSMT15], which reduces unnecessary calculations. It has a functional API for F#.

Aardvark uses FShade [Haa14] for shader programming. The shaders can then be written with the F#-syntax, which is converted to GLSL-syntax and then compiled. Also, partial shader programs can be combined to use the entire shading pipeline. As an example, one partial shader program makes a subdivision sphere at a given position, another calculates the UV-coordinates and normals, the next one transforms the sphere to normalized device coordinates, yet another shader adds texturing, and a final shader implements Phong-shading [Pho75]. Partial shader programs, like the texturing and lighting, can be shared with other non-spherical objects.

The rendering results of Aardvark are submitted to the HMD via OpenVR. OpenVR is a collection of header files, which define interfaces to the VR hardware. OpenVR is implemented by SteamVR supporting the Oculus Rift and HTC Vive. SteamVR also manages the so-called chaperone, a grid representing the boundaries of a pre-defined space. Users should remove all obstacles from this space so that they can move safely. Simply put, the chaperone prevents the players from running into real objects. OpenVR is on the hardware side and provides more basic functions. For example, programs can call OpenVR functions to obtain the tracking results of the HMD and the controllers. OpenVR also provides a stencil mesh [Val15a], which can be applied in order to prevent rendering to invisible pixels. Pixels can be invisible because the rendered image is distorted to revert the effect of the lenses.

4.3 Haptic Feedback

Haptic feedback has exactly one function in the C# wrapper of OpenVR [Val17a]:

```
public void TriggerHapticPulse(
   uint unControllerDeviceIndex,
   uint unAxisId,
   char usDurationMicroSec
);
```

All the information from the official documentation is [Lud15]:

1. `uint unControllerDeviceIndex` - The tracked device index of the controller to trigger a haptic pulse.
2. `uint unAxisId` - The ID of the axis to trigger a haptic pulse.
3. `char usDurationMicroSec` - The duration of the desired haptic pulse in microseconds.
4. Trigger a single haptic pulse on a controller.
   Note: After this call the application may not trigger another haptic pulse on this controller and axis combination for 5ms.

`uint unAxisId` is always zero when using the HTC Vive.

`char usDurationMicroSec` should not be a 'char' but a 'ushort' [M-C16]. Both types are 16 Bits wide in C#, but 'char' has some hidden interpretation attached, which causes the vibration to be very weak. This bug has been known since May 2016 and still exists one year later. This duration is probably the time in which the vibration unit receives electricity. This would mean that a short duration delivers not sufficient energy to overcome the inertia, and a long duration causes the vibration unit to turn back and weaken its previous impulse. Simply put, the duration is actually the vibration strength with meaningful values between 0 and 2000. Sometimes, when the function is called frequently with strongly varying durations, the vibration stops for about a second. This might be a bug in the controllers.

When keeping the update rate at 90 FPS, it suffices to call this function in every frame. Being closer to the 5ms cool-down does not increase the vibration strength. Therefore, no extra thread is needed for the vibration control.

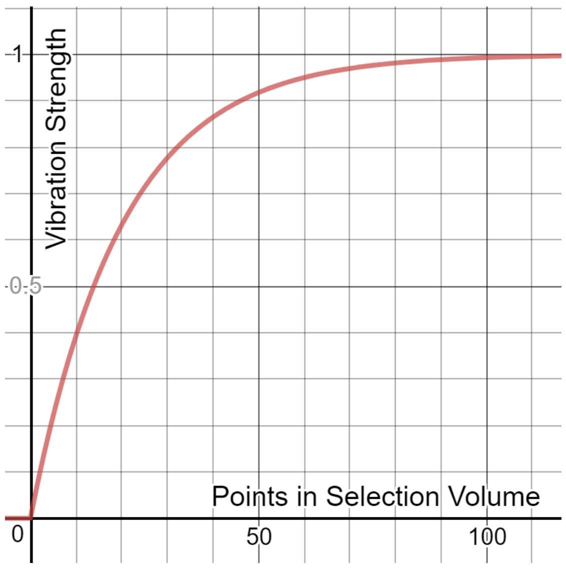


Figure 4.2: The number of points in the selection volume is converted to vibration strength with a limited growth function: $strength(numPoints) = clamp(1 - e^{-0.05*numPoints}, 0, 1)$.

In the basketball game and grabbing test, there are multiple ’channels’ for vibrations:

1. Vibrate shortly and strongly when the controller hits another object.
2. Vibrate constantly when the controller overlaps with another object. The vibration strength depends on the relative velocity.
3. Vibrate with a pattern when the player scores. This was removed because some users found it distracting and annoying.

A list contains the vibration events, which store their channel, duration in seconds, and strength. The events for overlaps are removed and re-added in each frame after all other channels. So, the vibration always represents the controller’s overlap, but only if no other vibration source fires.

In the point-cloud editor, the overlap vibration depends on the number of points within the selection volume. The point cloud is searched until 100 of such points are found. The number of points is converted to a vibration strength via a clamped limited growth function as shown in Figure 4.2. The idea behind the function is that the difference between zero and one points is the most interesting and should have a large influence on the vibration. The difference between one and two points is already much less important. Due to the Level-of-Detail (LoD) system, one visible point can be multiple times in the data. Therefore, the function is not too steep.

The vibrations can mess with the inertial sensors of the controllers. This makes the virtual controllers drift away from their actual position, especially when the line-of-sight to both cameras is blocked. This is one reason, why the vibrations in the point-cloud editor are relatively weak. Another reason is that haptic feedback is perceived rather subconsciously, and some people find it annoying. See Section 6 for more information.

4.4 Physics

The applications use Bullet Physics, an open-source physics engine by Erwin Coumans [Rea17]. Among others, it features rigid and soft-body dynamics, constraints, ray casts, and destructible objects. The rigid-body simulation is used for the ball movement and collision detection. The ray casts are used for the teleportation.

Bullet allows various combinations of settings and flags enabling different behaviors. The useful kinds of physics objects are: static object, kinematic object, rigid body, and ghost. All of them need a collider to define their shape. This can be a geometric body like a sphere, or it can be a mesh. For performance reasons, collider meshes should be much simpler than meshes for rendering. Typical collider meshes contain only a few hundred vertices. They must be concave, water tight, and consist only of triangles.

Static objects are objects that should never move, for example walls. They do not react to collisions, as if they had infinite mass. Kinematic objects do not react to collisions either, but they can be moved by the game logic. The game controllers are kinematic objects updated with their tracking position. The velocity of kinematic objects is calculated from the time step and the distance of their jumps. In contrast, the velocity of static objects is assumed to be zero. Rigid bodies are physically simulated including inertia, friction, and restitution. Continuous collision detection (CCD) makes sure that they do not jump from one frame to the other through flat colliders. It tests collisions not only with the ball itself but also with a cylinder around its trajectory. Rigid bodies can be set asleep when no moving body is near them, which accelerates the simulation. Ghosts do not react to collisions, just like static objects. The difference is that objects colliding with ghosts also ignore the collision. Some kinds of ghosts store the collision and penetration data. This can be used for trigger volumes, which check if a ball flies through the basket, for example.

Not all physics objects should collide with each other. Therefore, collision groups can be defined. As an example, the trigger volume of the basket does not need to check collisions with the walls or controllers. This also applies to ray casts, which search the first hit of a ray with certain objects. This can be used to find teleportation targets. With the right collision groups, the users can teleport to walls but not to controllers.

Rigid bodies penetrating other non-ghost objects are pushed outwards. Due to the discrete time steps, this happens during practically every collision. The push force and collision forces are added, which adds energy. Simply put, a perfectly bouncing ball flies higher after each hit with the ground. To prevent this, the split-impulse technique needs to be activated, which introduces some instability for stacked objects [Cou08].

Constraints restrict the translation and / or rotation of physics objects. For example, a stick can be attached to a wall with a ball-socket constraint. This lets the stick rotate around the socket but keeps the distance constant. However, 'constant' is only a theoretical optimal case because solving the constraints can be inaccurate or even unstable. This is a bigger issue when a light object is attached to a heavy object, or when a chain of objects is created. This possible instability is the reason why the controllers are kinematic objects, instead of rigid bodies attached to the controllers' positions like in the NVIDIA Funhouse.

4.5 Out-of-Core Point-Cloud Rendering

Point clouds often contain billions of points. For the rendering, each point has 3D-coordinates, typically with one 4-byte float per axis. Usually, they also have an RGB-color assigned, which is 15 bytes (3 * 4B + 3 * 1B = 15B) per point. 1 billion (15 GB) points are already enough to make them fit only into the memory of very few non-consumer graphics cards, such as the current NVIDIA Quadro [NVI17]. Also, the main memory of many consumer PCs would not suffice to contain all the data at once. Therefore, so-called out-of-core data structures are required. This means that the data is loaded continuously from the hard drive to the graphics memory, replacing other data that is not needed anymore.

Aardvark contains such out-of-core data structures. However, making all data accessible for rendering is not enough. No current graphics card can render billions of points in real time. Therefore, an efficient LoD structure is needed. Aardvark implements an octree structure similar to nested octrees [WS06]: The leaf nodes contain the original points, and the other nodes contain a random sub-set of their children's points. This way, the LoD system does not require a lot of pre-processing but contains redundant data.

The octree itself is always in main memory. However, the contents of its nodes are only loaded if necessary. When a node's contained points are accessed, a cache is searched. If the points are not cached, they are loaded from the hard drive into the cache. A separate octree stores additional LoD information, e.g. the bounding box of each octree node.

The traversal itself dependents strongly on its purpose and can be tuned in numerous ways. The point-cloud editing tool in VR uses the same traversal for rendering as the VGM. The traversal depth depends on the size of the octree node on the near plane. The arrays of points of all traversed nodes are collected in a hash set. This hash set of arrays contains the positions and colors of all points that are to be rendered. The arrays are then split equally and converted to Vertex Buffer Objects (VBO) in parallel. These VBOs are cached for a while.

The comparison of the user selection with the ground truth requires a point-in-sphere test for each point with each selection volume (see Section 4.7) in each operation. With a point cloud of just 2 million points and a short operation with only 20 spheres, this comparison already takes many minutes. Therefore, a rather small point cloud is used for this thesis.

A small point cloud has the disadvantage that the points can be sparse, which means that the rendering can leave holes between the points. These holes can be filled in, typically with view-aligned primitives like squares or circles. However, when such primitives are too large, they occlude each other and hide details. Also, when the camera is turned just a little, the occlusion can jump. With the jitter of the HMD tracking, this causes a constant flickering in VR.

The high-quality interpolation technique by Schütz and Wimmer [SW15], which is implemented in the VGM, solves this problem by turning the flat primitives into spheres, cones, or paraboloids. This is done by assigning depth values in the fragment shader, which creates a pixel-accurate shape. However, setting the depth in the fragment shader makes the normal early z-test impossible, which results in more load for the fragment shader and therefore slower rendering.

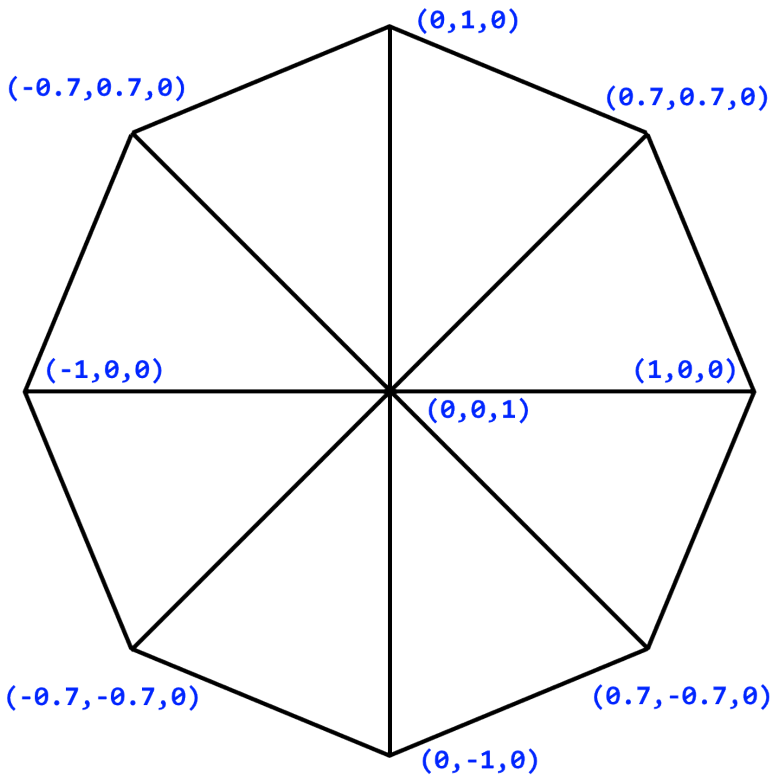


Figure 4.3: The points in the VR point-cloud editing tool are rendered as imposter cones. Such a cone consists of 8 triangles in a triangle fan with an elevated center vertex. This cone is simple, fast, has an almost circular outline, and enables large points without much flickering from overlap.

Due to the extreme overdraw in point clouds, the early z-test would give a considerable speed-up. Therefore, we propose a new point-cloud rendering method that moves shape generation from the fragment to the geometry shader. To keep the shader fast, the generated geometry should be as simple as possible while maintaining a good visual quality. Cones are the best choice because they are more “pointed” than spheres and paraboloids and therefore preserve more details. Additionally, their tip makes an approximation of cones with triangles easier than an approximation of spheres and paraboloids. The cones here are basically octagons made out of a triangle fan with eight pieces. Only the center vertex is moved closer to the camera. Figure 4.3 shows the cone as it is produced by the geometry shader at a point's position.

The old and new rendering techniques are compared in Section 6.4.5.

4.6 Bi-Manual Pinch Gesture

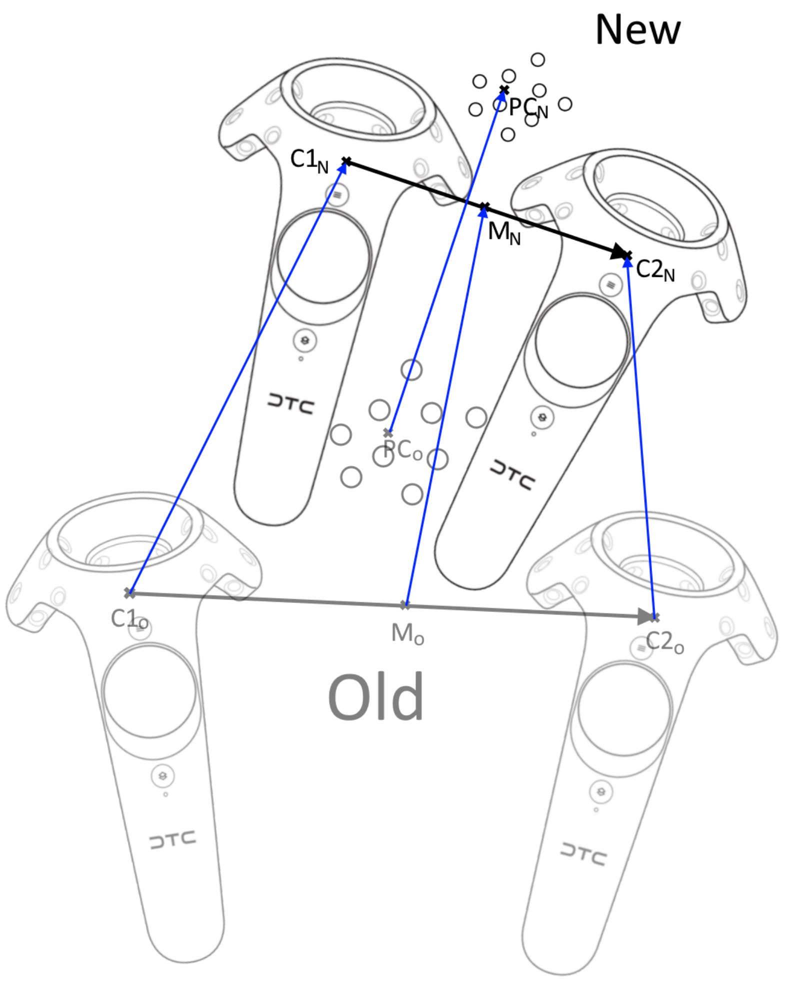


Figure 4.4: The point cloud is transformed depending on the changing controller transformations. This creates a viewing method like the pinch gesture known from touchscreens.

There are two basic approaches for viewing point clouds: let the user move through the point cloud or move the point cloud around the user. The metaphor used here is the bi-manual pinch gesture, which implements the second approach. It is similar to the pinch gesture used with touchscreens but with both hands in 3D instead of fingers in 2D. Because most people know the 2D version from their smartphones, it easy to learn. The 3d-version is already implemented in some experimental VR programs, but it is not yet described in detail in the literature. The handlebar metaphor by Song et al. [SGH+12] is very similar but works with rigid bodies instead of point clouds.

The basic idea of the bi-manual pinch gesture is to keep the translation, rotation, and scaling linear. Then, the relative positions of the controllers within the point cloud stay the same after the interaction. This viewing technique worked well in the closed in-door scene. However, it may not work as well in out-door scenes. The design decisions are described in Section 3.3.2 and the results in Section 6.4.5. Figure 4.4 illustrates the transformation of the point cloud during the pinch gesture.

$\boldsymbol{C1_O}$, $\boldsymbol{C2_O}$, $\boldsymbol{C1_N}$, and $\boldsymbol{C2_N}$ are 4x4 transformation matrices of both controllers with their previous and current transformations. The matrices contain a rotation, scale, and translation. $\boldsymbol{PC_O}$ is the old model matrix of the point cloud, and $\boldsymbol{PC_N}$ is the new one.

The new scale of the point cloud is defined by the distance between the controllers' old and new transformations. The uniform scaling is the new distance divided by the old distance. The point cloud moves with the geometrical center of both controllers. The point cloud is rotated with the axis through both controllers. A matrix that rotates the old axis onto the new axis is applied to the point cloud. This rotation step contains a superfluous DoF: the rotation around the axis. Controlling this DoF in a meaningful way is future work. The three partial transformations transform the point cloud to its new state. The center point of both controllers acts as a pivot point for the combined transformation. Algorithm 4.1 shows the involved calculations in detail. Note that Aardvark applies transformation matrices from left to right.

Algorithm 4.1: Pinch Gesture

```
Function calcPinchInfo($\boldsymbol{C1}$, $\boldsymbol{C2}$)
    $\boldsymbol{c1} = \boldsymbol{C1} \cdot (0,0,0,1)^T$
    $\boldsymbol{c2} = \boldsymbol{C2} \cdot (0,0,0,1)^T$
    $\boldsymbol{m} = (\boldsymbol{c1} + \boldsymbol{c2})/2$
    $\boldsymbol{c12} = \boldsymbol{c2} - \boldsymbol{c1}$
    return $\boldsymbol{m}, \boldsymbol{c12}$
Function calcNewPointCloudTrafo($\boldsymbol{C1_O}, \boldsymbol{C2_O}, \boldsymbol{C1_N}, \boldsymbol{C2_N}, \boldsymbol{PC_O}$)
    $\boldsymbol{m_O}, \boldsymbol{c12_O}$ = calcPinchInfo($\boldsymbol{C1_O}, \boldsymbol{C2_O}$)
    $\boldsymbol{m_N}, \boldsymbol{c12_N}$ = calcPinchInfo($\boldsymbol{C1_N}, \boldsymbol{C2_N}$)
    $\boldsymbol{S}$ = Matrix4D.Scale($|\boldsymbol{c12_N}|/|\boldsymbol{c12_O}|$)
    $\boldsymbol{R}$ = Matrix4D.RotateInto($\boldsymbol{c12_O}/|\boldsymbol{c12_O}|, \boldsymbol{c12_N}/|\boldsymbol{c12_N}|$)
    $\boldsymbol{T}$ = Matrix4D.Translation($\boldsymbol{m_N} - \boldsymbol{m_O}$)
    $\boldsymbol{Pivot} = Matrix4D.Translation(\boldsymbol{m_N})$
    $\boldsymbol{PC_N} = \boldsymbol{PC_O} \cdot \boldsymbol{Pivot}^{-1} \cdot \boldsymbol{S} \cdot \boldsymbol{R} \cdot \boldsymbol{T} \cdot \boldsymbol{Pivot}$
    return $\boldsymbol{PC_N}$
```

4.7 Real-Time Point Selection Visualization

For any user interaction, immediate feedback is mandatory. This applies to selections in point clouds as well. However, changing the data according to the selection in any way still takes too long, even with modern SSDs. Consider a point cloud in which each point has a single bit storing whether it is selected. Assuming a low update rate of 20 FPS for the interactions and a modern SSD with a theoretical write speed of about 2 GB/s [Sam17], 800 million points could be selected or unselected at once. This applies only if no other reading or writing happens and if the file system is neglected. However, different states might be required, the update rate should be at least 90 FPS in VR, and at least one read traversal is done in each frame for the rendering. Therefore, just a few million points can be selected or unselected per frame in practice. The LoD-structure of Aardvark's point clouds and its caches must be updated, as well. Then, not even one hundred thousand points can be selected at once. All in all, the duration until the result of only a single selection is shown can be several seconds, which is much too long for real-time use. Therefore, a fast visualization of the selection gives the feedback until the data is updated.

The instant selection visualization we build upon was originally developed by Rainer in a student project for the VRVis [Rai16]. It works like shadow volumes [Cro77]: the stencil buffer is used to efficiently categorize a pixel's state as inside or outside a selection volume. This selection visualization supports multiple operations to combine volumes, such as union, subtract, and XOR. However, only union is used for the VR point clouds. The necessary render passes for Rainer's selection visualization are:

1. Render selectable geometry, e.g. point cloud
2. Render selection volumes to stencil buffer with depth test enabled
3. Color all pixels with non-zero stencil value
4. Render non-selectable geometry
5. Render non-selectable transparent geometry

In render pass 1, the point cloud is rendered as described in Section 4.5.

Render pass 2 requires more explanation. The selection volume is a geometric object that represents the controller's selection volume along its trajectory during an operation. Figure 4.5 illustrates this selection volume path. On the data side, the path is an array that contains the selection volume scale and position relative to the point cloud in discrete time steps. More exactly, when OpenVR delivers a new tracking result for the controller, the program checks if there already is an element in the path near the new controller position. If there is none, the new position and scale are added to the path. The necessary distance between path elements is a trade-off between performance and accuracy. 1 mm was accurate enough during the user study. Alternatively, path segments could be rendered as cylinders with hemispheres at the ends, which would allow longer segments. The selection volume path is rendered to only the stencil buffer with the default depth test enabled.

In render pass 3, only the visible pixels within the selection volume are colored. This is done by rendering a full-screen quadrangle filled with a selection color. The stencil test must be set to succeed at one or greater. Afterwards, the stencil buffer can be reset to zero.

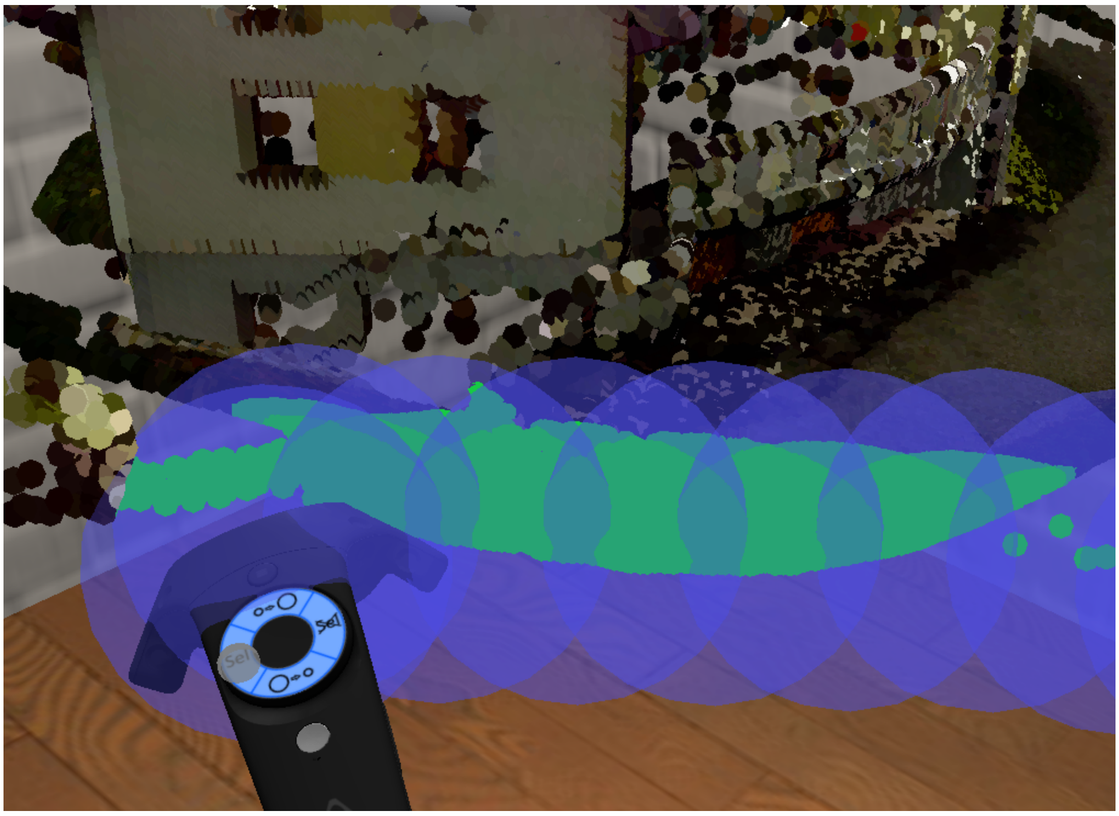


Figure 4.5: The blue spheres represent the controller's selection volume during an operation. Together, they form the selection volume of the entire operation. The points inside this selection volume are colored green by the instant selection visualization. When the operation is finished and the data updated, the selected points become red.

Render pass 4 is dedicated to the normal geometry like the controllers and room walls. This is done after rendering the point cloud with its selection. Otherwise, the controllers or walls would also appear selected.

Render pass 5 finally takes care of normal transparent geometry like the selection volume attached to the controller. Also, the thumb representation on the trackpad is a transparent sphere.

The only visible difference between the instant and the data selection is, apart from the different colors, that it is possible for points to be only partly covered by the instant selection visualization. When the center of a point is covered, it will be selected in the actual data. This inaccuracy becomes less important the smaller the points are rendered.

CHAPTER 5

User Study Design

In this chapter, we describe the main contribution of this thesis, a user study to analyze

1. grabbing and throwing with controllers in a simple basketball game.
2. the influence of haptic and optical feedback on performance, presence, task load, and usability.
3. the advantages of VR over desktop for point-cloud editing.

Such a user study is necessary to see how real people use our systems. Also, we can perform a statistical analysis in order to check if differences between groups are significant or just random. One group consists of all results with haptic feedback enabled, for example.

The differences in objective measures like scores and error rates are compared as well as the differences in subjective measures like usability, task load, and presence. The objective measures are saved in log files, and the subjective measures are taken by means of questionnaires after each test. Each questionnaire has a field for optional comments. Verbal comments and apparent problems were noted for later analysis.

Performing the user study required three iterations: hallway tests, pre-tests, and the actual user study. For the hallway tests, random colleagues were asked to give quick feedback to features and experiments. This was done with many features but most extensively when developing the VR point-cloud interactions. Testing the user study with colleagues and friends helped to optimize the tests. One tester found that the IGroup presence questionnaire [igr16] contains redundant questions. For example, item 4 “I did not feel present in the virtual space” and item 6 “I felt present in the virtual space” are just inverted. The answers might not be mutually exclusive but still confused the testers. The actual user study was performed with unbiased testers from different backgrounds. The appointments were made per e-mail or Doodle MeetMe page [Doo17], where availability was shown with a linked calendar. The users are described in detail in Section 6.

The typical process of one session is: the basketball game first, the grabbing test second, and the point-cloud editing third. After each program, the users fill out the corresponding questionnaires.

The entire session took around 1.5 to 2 hours. The basketball part required roughly 15 minutes, the grabbing test 30, and the point clouds 45.

The testers play the short version of the basketball game with only two rounds. Afterwards, the testers fill in the user information, simulator sickness, task load, usability, and presence questionnaires. Apart from the grabbing and throwing analysis, the basketball game also serves as a 'calibration' phase for the testers. In this phase, the testers could adapt to the Vive. Also, the questions were explained, which is important because the questionnaires were in English but not a single tester was an English native speaker. This introduction was meant to decrease the amount of erroneous answers and with it the variance within the test groups.

Next comes the grabbing test with four rounds, one for each feedback type: none, haptic, optical, and both. The order of the different feedback types is randomized to take learning and fatigue into account. The randomization is done by choosing one of the 24 permutations of the four feedback types. For each feedback type, the tester did the grabbing test and then answered the presence, usability, and task load questionnaires. The simulator sickness questionnaire is answered only once after all four grabbing tests.

Finally, the point-cloud editing is tested. Some testers did the desktop version first and VR second, others did it the other way around. After the desktop version, the tester filled in the task load and usability questionnaires, and after the VR version, the simulator sickness, task load, usability, and presence questionnaires.

The questionnaires were online forms made with Google Forms [Goo17]. Other providers were considered, but only Google Forms offers users to export the answers for free. Export, especially as a comma-separated file, was necessary to perform the analysis.

5.1 User Information

To gather more context information about the testers, they were asked to provide their:

1. name, which also serves as a key to associate all questionnaire responses.
2. e-mail address to share the anonymized results afterwards.
3. gender and age to see if the tester sample is representative for certain groups.
4. usage of computers, digital games, augmented reality, and virtual reality. This helps to estimate their experience and affinity for technology. Possible answers are: never, rarely (about once a year), sometimes (about once a month), regularly (about every week), and often (almost every day).

5.2 Simulator Sickness

The questionnaire for simulator sickness is the one by Bouchard [BRR07], which is based on the one by Kennedy [KLBL93]. The testers are to select how much – none, slight, moderate, or severe – each symptom is affecting them. The symptoms are:

1. general discomfort

2. fatigue
3. headache
4. eye strain
5. difficulty focusing
6. salivation increasing
7. sweating
8. nausea
9. difficulty concentrating
10. « fullness of the head »
11. blurred vision
12. dizziness with eyes open
13. dizziness with eyes closed
14. vertigo (experienced as loss of orientation with respect to vertical upright)
15. stomach awareness (indicates a feeling of discomfort which is just short of nausea)
16. burping

The categorical answers are converted to a linear numerical scale: none -> 0.0, slight -> 0.33, moderate -> 0.66, and severe -> 1.0. This allows testing the significance with ANalysis Of VAriance (ANOVA).

5.3 Presence

The used questionnaire for presence is the one by the UQO Cyberpsychology Lab [UQO13], which is based on the one by Witmer [WS98]. It consists of 24 items to be answered on a 7-point Likert scale. The Likert scale is a linear scale, typically with 5 or 7 segments. For example, "How much were you able to control events?" has possible answers between "Not at all (0)" and "Completely (6)". The items of the presence questionnaire are:

1. How much were you able to control events?
2. How responsive was the environment to actions that you initiated (or performed)?
3. How natural did your interactions with the environment seem?
4. How much did the visual aspects of the environment involve you?
5. How natural was the mechanism which controlled movement through the environment?
6. How compelling was your sense of objects moving through space?
7. How much did your experiences in the virtual environment seem consistent with your real world experiences?
8. Were you able to anticipate what would happen next in response to the actions that you performed?
9. How completely were you able to actively survey or search the environment using vision?
10. How compelling was your sense of moving around inside the virtual environment?
11. How closely were you able to examine objects?
12. How well could you examine objects from multiple viewpoints?
13. How involved were you in the virtual environment experience?
14. How much delay did you experience between your actions and expected outcomes?

15. How quickly did you adjust to the virtual environment experience?
16. How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?
17. How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?
18. How much did the control devices interfere with the performance of assigned tasks or with other activities?
19. How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?
20. How much did the auditory aspects of the environment involve you?
21. How well could you identify sounds?
22. How well could you localize sounds?
23. How well could you actively survey or search the virtual environment using touch?
24. How well could you move or manipulate objects in the virtual environment?

The total score is calculated by averaging all items. Items 14, 17, and 18 must be reversed before. The following grouping is described by the authors of the questionnaire:

1. Realism: Items 3 + 4 + 5 + 6 + 7 + 10 + 13
2. Possibility to act: Items 1 + 2 + 8 + 9
3. Quality of interface: Items (all reversed) 14 + 17 + 18
4. Possibility to examine: Items 11 + 12 + 19
5. Self-evaluation of performance: Items 15 + 16
6. Sounds: Items 20 + 21 + 22
7. Haptic: Items 23 + 24

5.4 Usability

The usability is measured with the System Usability Scale (SUS) by Brooke [Bro96]. It consists of 10 items to be answered on a 5-point Likert scale between “strongly disagree” and “strongly agree”. The items are:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

In the context of this thesis, “system” refers to the software only. The testers are not expected to think about setting up the Vive. The individual scores are summed and multiplied by 2.5.

Although the total score is then in a range of 0-100, it should not be interpreted as percentage. Instead, it should be considered for a percentile ranking. A score of 68 or higher is above average.

5.5 Task Load

The used questionnaire is the NASA Task Load Index by Hart and Staveland [HS88]. It consists of 6 items to be answered on a 5-point Likert scale. The version used in this thesis is the raw TLX [Har06], which omits the weighting. All items except for item 4 are inversed. The items are:

1. Mental Demand - How mentally demanding was the task?
2. Physical Demand - How physically demanding was the task?
3. Temporal Demand - How hurried or rushed was the pace of the task?
4. Performance - How successful were you in accomplishing what you were asked to do?
5. Effort - How hard did you have to work to accomplish your level of performance?
6. Frustration - How insecure, discouraged, irritated, stressed, and annoyed were you?

5.6 Objective Scores

Each of the three application stores certain objective user scores in text files. The basketball game only stores how often the player has thrown the ball into the basket.

The grabbing test logs:

1. when the program started.
2. which feedback type was enabled.
3. when a new round started.
4. for each round:
   a) when the user scored.
   b) how often the user tried to grab a ball.
   c) how often the user successfully grabbed a ball.

In the point-cloud editing tools, the selections of the testers are compared to a manually created ground-truth selection. The correctly selected, wrongly selected, correctly non-selected, or wrongly non-selected points are counted. Also, the start-up time of the program and the time when the comparison is started are noted. Since the application and test were explained to the testers by reference to the running application, the comparison time minus the start-up time is only approximately the working time.

The results of this user study are presented in Chapter 6.

CHAPTER 6

Results

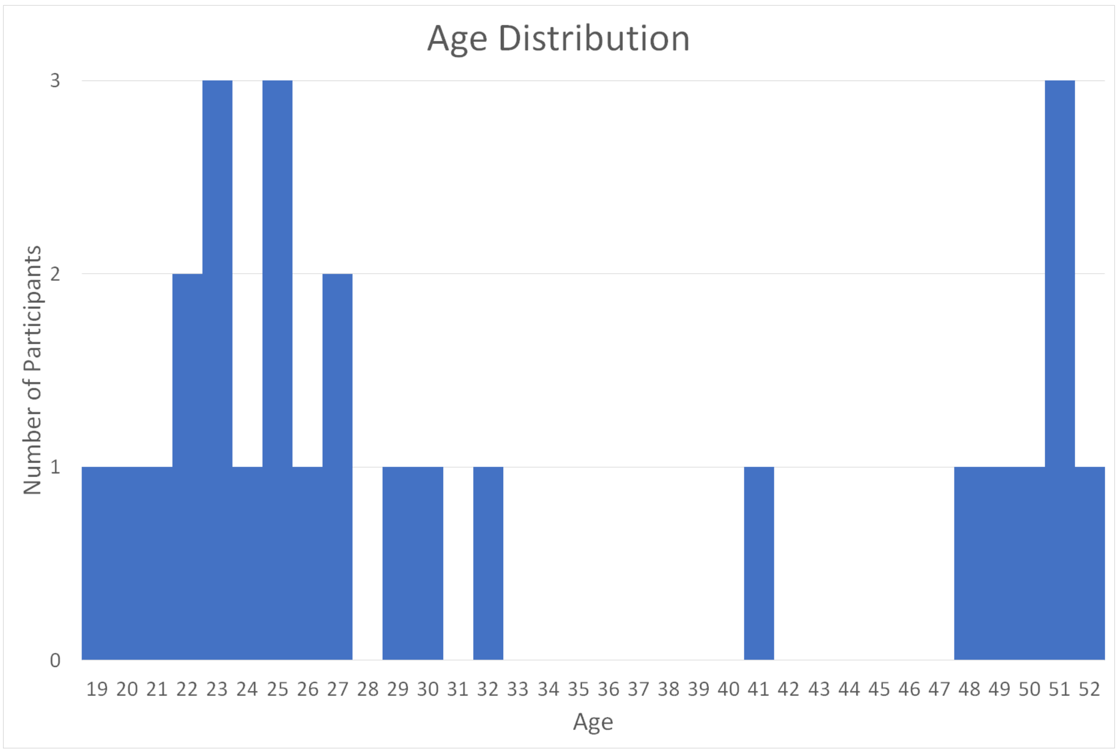


Figure 6.1: This age distribution chart shows that many testers were between 20 and 30 years old, and several testers were around 50.

26 testers were recruited from colleagues, friends, sport club mates, and others. The information about the user study was spread by asking them personally, in social networks, and with posters. Most testers have an academic background, and all of them were fit and healthy. The user study

was carried out over a period of two months, allowing word-of-mouth and recommendations to bring more testers. Trying out VR was enough motivation since no compensation has been paid.

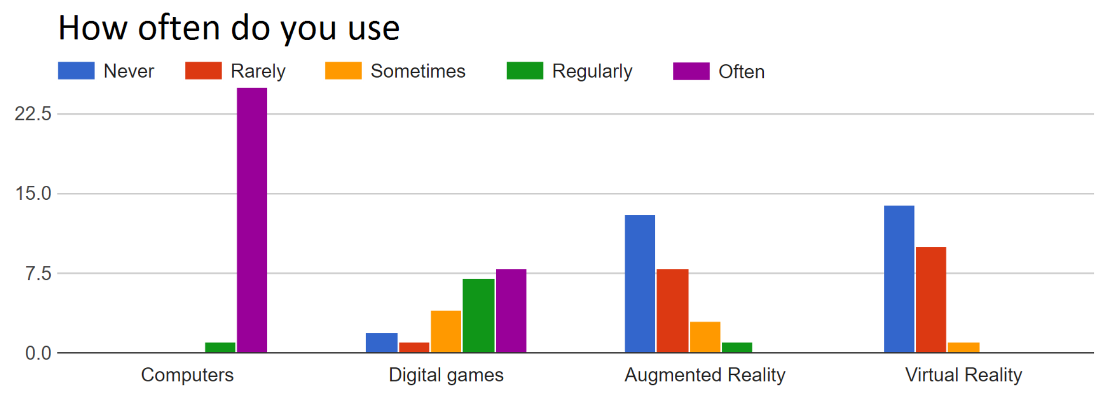


Figure 6.2: This chart shows how often the testers use computers, digital games, augmented reality, and virtual reality.

Of the 26 participants of the user study, 16 were male and 10 female. Many testers were between 20 and 30 years old, and several testers were around 50, as displayed in Figure 6.1. Figure 6.2 shows how often the testers use computers, digital games, augmented reality, and virtual reality. Almost every tester uses computers on a daily basis. About half the testers play digital games at least every week. One half never had contact to augmented reality, most of the other half played or had played Pokemon Go [Nia16] to some extent. The usage of VR is similar. About one half has never tried VR, the others tried it once at least.

The significance is tested with one-way ANOVA [Fis18]. The results are expressed in the following pattern: Group 1 (M = 0.3, SD = 0.1) is significantly [F(1, 100) = 64.7, p = 0.001] better than Group 2 (M = 0.1, SD = 0.13). M is the arithmetic mean. Higher means are better, except for the durations. The SD is the standard deviation. An SD near zero means that random errors like the weather during the tests had only a minor influence on the results. [F(1, 100) = 64.7, p = 0.001] describes the significance calculated with ANOVA. F(1, 100) means that the DoF between the groups is 1 (number of groups - 1), and the DoF within the groups is 100 (participants per group - 1 summed). The 64.7 is the so-called F-ratio, which is the mean squares between the groups divided by the mean squares within the groups. The critical value (not shown) is taken from a table based on the DoF. Since the table is always the same and the DoF are known, writing the critical value would be redundant. The F-ratio must be at least equal to the critical value. Otherwise, the analyzed effect is not significant for the tested probability level. The p = 0.001 is the probability that the effect is caused by random errors. The significance levels are p < 0.001, p < 0.01, and p < 0.05. 5% probability or more are considered insignificant.

For the analysis, a small program was written, which assembles the questionnaire responses to sessions and performs the analysis on different data series. All questionnaire scores are normalized from the Likert scale to a 0-1 range, where 0.0 is bad and 1.0 is good.

6.1 Simulator Sickness

Simulator sickness was not a problem in this user study, even though the air conditioning and ventilation did not work properly. Only one tester wanted slightly longer breaks and another reported nausea and headache for a few hours after the user study. Both testers are around 50 years old.

Basketball (M = 0.12, SD = 0.21) causes slightly less simulator sickness than the grabbing test (M = 0.13, SD = 0.19). Point-cloud editing (M = 0.08, SD = 0.14) creates the least simulator sickness. However, the differences are not statistically significant. Every bit of optical flow can increase the simulator sickness. Turning the head creates optical flow over the entire field-of-view. This is in accordance with the basketball and grabbing test generating more simulator sickness.

6.2 Basketball Game

The results of the basketball game allow for an analysis of grabbing and throwing. Also, it gives insights into what is important for a physics-based VR game.

The scores are between 3 and 23 (M = 11.38, SD = 5.26), which is a wide range with a high variance. This indicates that some players could get along much better with the throwing than others.

6.2.1 Presence

The average overall presence score (M = 0.79, SD = 0.09) is assembled from its components: realism (M = 0.81, SD = 0.09), possibility to act (M = 0.79, SD = 0.10), quality of interface (M = 0.85, SD = 0.13), possibility to examine (M = 0.84, SD = 0.11), self-evaluation of performance (M = 0.83, SD = 0.18), sounds (M = 0.69, SD = 0.29), and haptic (M = 0.71, SD = 0.21).

When analyzing each single question of the presence questionnaire, one finds that:

1. Q: How much were you able to control events?
   A: The players' control of events (M = 0.68, SD = 0.24) was surely affected by the flawed throwing.
2. Q: How responsive was the environment to actions that you initiated (or performed)?
   A: Responsiveness was no big issue (M = 0.81, SD = 0.12).
3. Q: How natural did your interactions with the environment seem?
   A: The interactions felt not too natural (M = 0.69, SD = 0.21), probably caused by the throwing.
4. Q: How much did the visual aspects of the environment involve you?
   A: The visual aspects involved surprisingly well (M = 0.76, SD = 0.20), considering that rather basic graphics with simple geometry and only normal mapping were used. Adding higher details to the geometry and some effects like bloom should improve this factor even more.

5. Q: How natural was the mechanism which controlled movement through the environment?
   A: The movement mechanism was natural walking and therefore quite good (M = 0.87, SD = 0.16).
6. Q: How compelling was your sense of objects moving through space?
   A: The objects moved rather compellingly (M = 0.78, SD = 0.17). Some users found that the ball flies as if it had lost some air, but most users liked it. The issue is to model air resistance with one factor for the linear damping. Linear damping is an exponential decrease of the velocity over time.
7. Q: How much did your experiences in the virtual environment seem consistent with your real world experiences?
   A: The VR experience seemed rather consistent with the real world (M = 0.74, SD = 0.17).
8. Q: Were you able to anticipate what would happen next in response to the actions that you performed?
   A: The players could anticipate the response to their actions quite well (M = 0.79, SD = 0.17). The reasons are probably that the game is very simple and the physics are assessable.
9. Q: How completely were you able to actively survey or search the environment using vision?
   A: The users could completely survey the environment with vision (M = 0.90, SD = 0.14).
10. Q: How compelling was your sense of moving around inside the virtual environment?
    A: The users found their own movement compelling (M = 0.91, SD = 0.13). This benefits from the 90 FPS and low persistence display of the Vive.
11. Q: How closely were you able to examine objects?
    A: The users could examine the objects very closely (M = 0.88, SD = 0.11). This is unexpectedly good, but it can be improved even more, for example with displacement mapping.
12. Q: How well could you examine objects from multiple viewpoints?
    A: The testers could examine objects from multiple viewpoints (M = 0.86, SD = 0.14).
13. Q: How involved were you in the virtual environment experience?
    A: The players felt very involved in the virtual environment (M = 0.93, SD = 0.11).
14. Q: How much delay did you experience between your actions and expected outcomes?
    A: Delay was no big problem (M = 0.86, SD = 0.18). Sometimes, the game froze after a score for about one second. This hints that the logging with writing on the hard drive is not as asynchronous as it should be.
15. Q: How quickly did you adjust to the virtual environment experience?
    A: Most players adapted very quickly to the virtual environment (M = 0.85, SD = 0.25). This is surely supported by using natural movement and the virtual hand.
16. Q: How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?
    A: Most players felt proficient in moving and interacting in the end (M = 0.81, SD = 0.17). Improving the grabbing should increase this score even further.
17. Q: How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?
    A: The players were mostly happy with the visual display quality (M = 0.88, SD = 0.13).

The resolution is still too low for a display as close to the eyes as this. The effect is made worse by the strong Fresnel lenses. Still, this does not disturb the fun. After all, gamers thinking of good games usually think of old, ugly games.

18. Q: How much did the control devices interfere with the performance of assigned tasks or with other activities?
    A: The control devices did not hinder the players much (M = 0.81, SD = 0.22). The most annoying issue was the cable. Regardless whether it was lying or hanging, some players kept ensnaring themselves.
19. Q: How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?
    A: The players could concentrate on the task instead of the mechanisms for the most part (M = 0.77, SD = 0.23). Some players needed to concentrate on releasing the trigger at the right time.
20. Q: How much did the auditory aspects of the environment involve you?
    A: The auditory aspects did not involve the players all too well (M = 0.61, SD = 0.33). When asking the testers which sounds they noticed, all heard the ball bounce sound, and most of them noticed the score horn. However, only some testers perceived the re-spawn popping sound consciously and very few even noticed the ambient sound. The problem is probably not the sound volume. Instead, it seems like audio is perceived rather subconsciously. Maybe, many quieter sources for the ambient sound are better than only two louder ones. Also, there was a bug causing very loud bounce sounds on some collisions.
21. Q: How well could you identify sounds?
    A: The players could easily identify the sounds (M = 0.78, SD = 0.34), at least the sounds they noticed.
22. Q: How well could you localize sounds?
    A: The players had trouble to localize the sounds (M = 0.67, SD = 0.36). The main reason is probably that the Vive has only one port for stereo sound. A surround sound headset would surely improve the localization. Also, a more advanced sound model, such as the one by Podkosova [PUK16] might help.
23. Q: How well could you actively survey or search the virtual environment using touch?
    A: Surveying the environment with touch did not work very well (M = 0.60, SD = 0.35). The analysis of haptics proved difficult because the vibrations are felt rather subconsciously. When asking the testers about the haptic feedback, some asked if there were any. The average score of 60% is still surprisingly good, considering that the vibration does not encode the texture of objects, and it encodes only roughly the shape and velocity. Real force feedback and tactile feedback, for example with props, would largely improve the haptics scores.
24. Q: How well could you move or manipulate objects in the virtual environment?
    A: Even with the flawed haptic feedback, the players could move objects very well (M = 0.82, SD = 0.18).

6.2.2 Task Load

The task load questionnaire shows that the basketball game creates a somewhat high task load (M = 0.62, SD = 0.15). It is mentally demanding (M = 0.77, SD = 0.20), mostly caused by adapting to VR. It is somewhat physically demanding as well (M = 0.62, SD = 0.24). Most players started sweating slightly, and one player hurt his elbow a bit when throwing. The task felt quite rushed (M = 0.56, SD = 0.29) because of the countdown. The players did not feel very successful (M = 0.55, SD = 0.27), which is mostly caused by the difficult throwing. Accordingly, they had to work quite hard for their performance (M = 0.46, SD = 0.19).

6.2.3 Usability

The game's usability is quite high (M = 0.83, SD = 0.14). Still, the players would rather not play the game frequently (M = 0.61, SD = 0.33). This is mostly caused by the throwing and the lack of content in the short version. The game is not unnecessarily complex (M = 0.92, SD = 0.13). It is quite easy to use (M = 0.82, SD = 0.27), and only few players would need a technical person (M = 0.80, SD = 0.34). The system seemed well integrated (M = 0.82, SD = 0.26). The game is not very inconsistent (M = 0.89, SD = 0.21). The players think that most people would learn to use the system very quickly (M = 0.83, SD = 0.21). Despite the throwing, most players found the game not very cumbersome to use (M = 0.89, SD = 0.21) and felt very confident (M = 0.83, SD = 0.23). The players did not need to learn a lot (M = 0.90, SD = 0.14).

6.2.4 Summary

To summarize the experience with the basketball game, a single programmer and a few months are enough to create a highly involving game with a deep presence. Actually, most of the time was spent on adding a proper VR and physics support to the rendering engine. However, creating a game that motivates for more than three minutes requires more content and an artist for better models.

Although the grabbing was no problem for the players, it can still be improved. As an example, the grabbing trigger volume could be around the balls instead of the controllers. This would ensure that the grabbing volume has the same distance to the physics collider at any point. Otherwise, the grabbing volume could be an extruded controller mesh, which has the disadvantage that the distance cannot be changed at runtime. Also, the grabbed object could move to an optimal position at the controller to make the throwing afterwards more consistent.

When objects are grabbed, there are basically two possibilities to manipulate them: with physics enabled or disabled. Enabled physics means that the objects are connected to the hand via physical constraints. This lets the objects collide with other objects while moving, which makes the overall experience more natural. The virtual forces, e.g. friction, acting on the grabbed object and the virtual hand could be used for force feedback. On the other hand, the constraints can make the physics simulation unstable, which then leads to jumps. This is even more risky when the hand representation is also attached via a constraint to the actual hand. Without physics, the grabbed objects and hand can penetrate other objects without any collisions. This prevents the instabilities but feels like a ghost hand. The problems with enabled physics mainly come from one source: the hand representation in VR has basically infinite strength, except if force feedback limits the movement of the real hands. A soft-body simulation could help to lessen the impact of this infinite force. A virtual soft basketball would deform when squeezed between two hands. In contrast, a virtual rigid basketball would just jump uncontrollably. Combining finger tracking with a soft-body simulation should make decent ball games in VR possible. However, these ideas require validation.

The user study revealed that releasing the ball willingly does not feel like throwing a basketball. This is because the real ball is too large to be held by one hand before throwing. Instead, it simply lies on the hand, which requires no releasing. Some players also tried to do the real basketball shoot with its fast wrist rotation in VR. At the end of the motion, the controller is moved downwards. When the virtual ball is released in this very moment, the ball flies downwards, instead of upwards as expected. Therefore, a natural throwing interaction for basketball requires a different grabbing system. Maybe, a more realistic hand representation would help. The grab could attach the ball to an open palm of the hand with a physics constraint. When the hand is palm-up, the constraint could be removed and the ball stays on the hand just through gravity and collisions. The throwing would then be completely natural. Finger tracking with data gloves and a soft-body simulation for the balls might also work. However, this solution is prone to instabilities of the physics simulation and this needs to be verified in future work.

6.3 Grabbing Test

The grabbing test is used to examine the influence of different feedback types on performance, presence, usability, and task load. The feedback types are: haptic feedback, optical feedback, both haptic and optical, and no feedback. The derived hypotheses are:

H1 Comparing haptic, optical, both, and no feedback shows significantly different
(a) performance, (b) presence, (c) usability, (d) task load.

H2 Haptic feedback compared to no feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

H3 Optical feedback compared to no feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

H4 Both feedback compared to no feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

H5 Haptic feedback compared to optical feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

H6 Both feedback compared to haptic feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

H7 Both feedback compared to optical feedback shows significantly improved
(a) performance, (b) presence, (c) usability, (d) task load.

The means and standard deviations of the performance can be seen in Table A.1, the presence scores in Table A.2, the task load scores in Table A.3, and the usability scores in Table A.4 in Appendix A.

6.3.1 Haptic vs. Optical vs. Both vs. No Feedback

H1a - Comparing haptic, optical, both, and no feedback shows significantly different performance - is rejected.

Figure 6.3 illustrates the scores in each round, and Figure 6.4 shows the average accuracy of grabbing. The differences between the overall scores with no feedback (M = 21.15, SD = 7.25), haptic feedback (M = 23.35, SD = 6.79), optical feedback (M = 24.96, SD = 7.37), and both feedback types (M = 24.54, SD = 7.53) are not significant.

H1b - Comparing haptic, optical, both, and no feedback shows significantly different presence - is partly confirmed.

Figure 6.5 displays all presence scores from the questionnaires. The overall presence with no feedback (M = 0.76, SD = 0.09), haptic feedback (M = 0.81, SD = 0.14), optical feedback (M = 0.81, SD = 0.11), and both feedback types (M = 0.83, SD = 0.10) is not significantly different.

The possibility to act with no feedback (M = 0.74, SD = 0.12), haptic feedback (M = 0.79, SD = 0.18), optical feedback (M = 0.82, SD = 0.17), and both feedback types (M = 0.84, SD = 0.14) is not significantly different. Still, the answers to item 8 “Were you able to anticipate what would happen next in response to the actions that you performed?” with no feedback (M = 0.61, SD = 0.22), haptic feedback (M = 0.74, SD = 0.23), optical feedback (M = 0.77, SD = 0.24), and both feedback types (M = 0.79, SD = 0.19) are significantly different [F(3, 100) = 3.54, p = 0.05].

The group of questions on haptics with no feedback (M = 0.48, SD = 0.19), haptic feedback (M = 0.77, SD = 0.17), optical feedback (M = 0.60, SD = 0.27), and both feedbacks (M = 0.76, SD = 0.21) is answered significantly different [F(3, 100) = 10.55, p = 0.001].

1. The answers to item 23 with no feedback (M = 0.37, SD = 0.41), haptic feedback (M = 0.81, SD = 0.17), optical feedback (M = 0.45, SD = 0.43), and both feedbacks (M = 0.74, SD = 0.29) “How well could you actively survey or search the virtual environment using touch?” are significantly different [F(3, 100) = 10.40, p = 0.001].
2. The answers to item 24 “How well could you move or manipulate objects in the virtual environment?” with no feedback (M = 0.59, SD = 0.19), haptic feedback (M = 0.72, SD = 0.21), optical feedback (M = 0.75, SD = 0.25), and both feedbacks (M = 0.77, SD = 0.20) are significantly different [F(3, 100) = 3.62, p = 0.05].

H1c - Comparing haptic, optical, both, and no feedback shows significantly different usability - is rejected.

The differences of the usability and its items are not significant.

H1d - Comparing haptic, optical, both, and no feedback shows significantly different task load - is rejected.

The differences of the task load and its items are not significant.

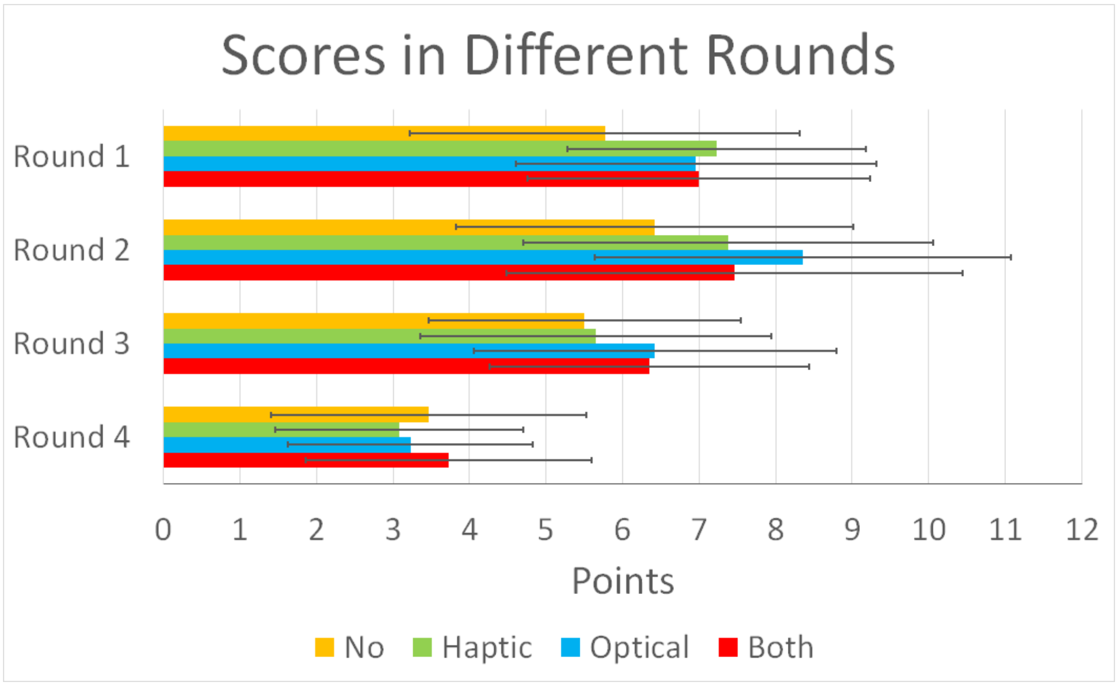


Figure 6.3: This graph displays the average scores in each round for all feedback types.

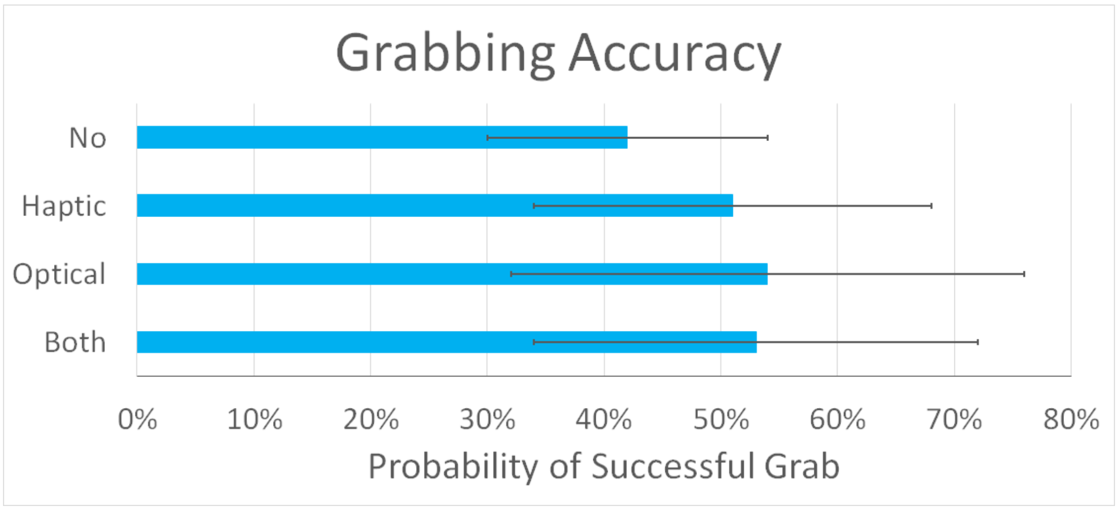


Figure 6.4: This graph displays the average success rate of grabbing for all feedback types.

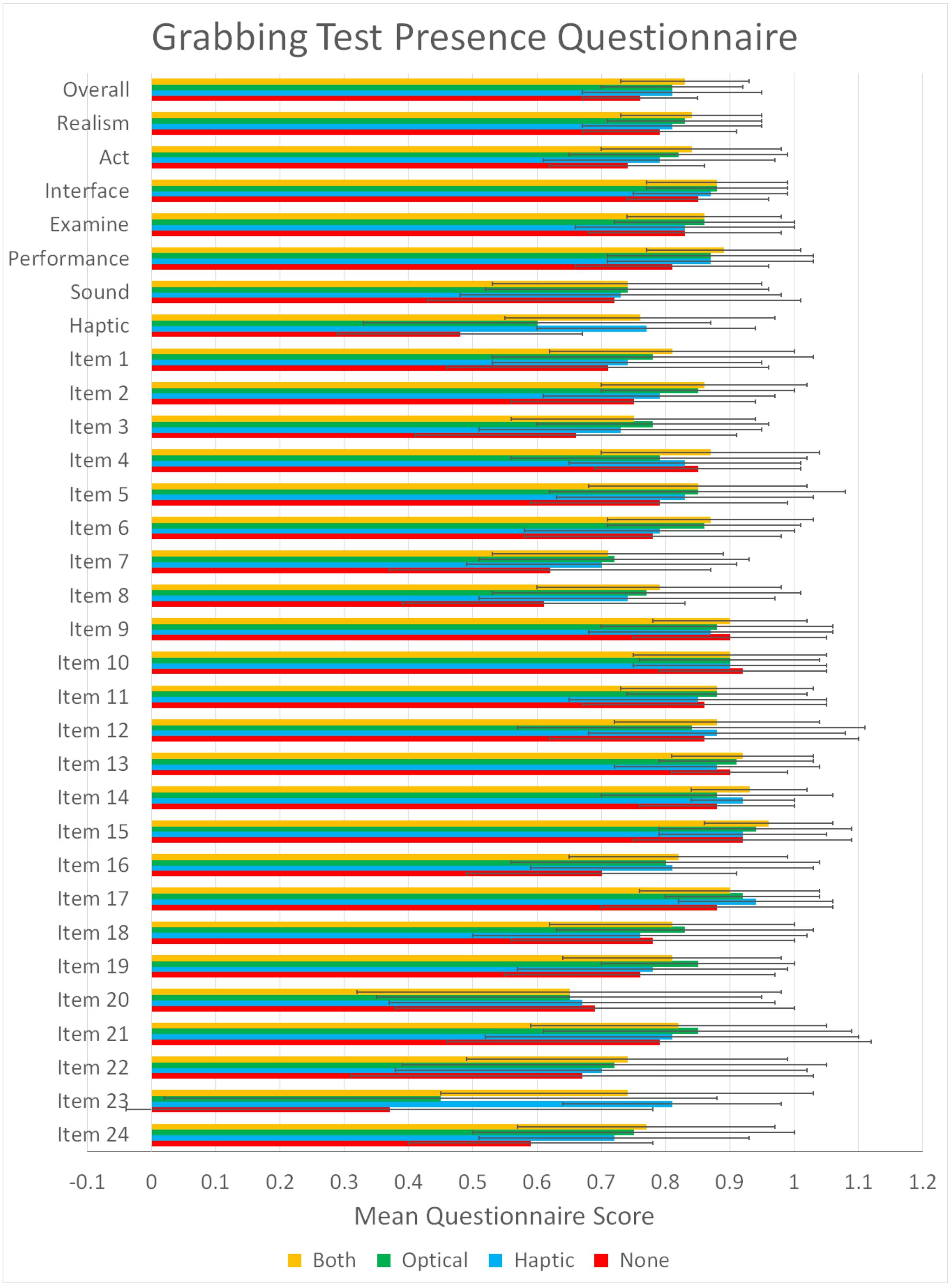


Figure 6.5: This graph displays the average presence and its sub-scores reported with the presence questionnaire.

6.3.2 Haptic vs. No Feedback

H2a - Haptic feedback compared to no feedback shows significantly improved performance - is partly confirmed.

The final scores with haptic feedback (M = 23.35, SD = 6.79) are not significantly better than with no feedback (M = 21.15, SD = 7.25). However, the performance in the first round without feedback (M = 5.77, SD = 2.55) is significantly lower [F(1, 50) = 5.20, p = 0.05] than in the first round with haptic feedback (M = 7.23, SD = 1.95). The grabbing success rates with no feedback (M = 0.42, SD = 0.12) are significantly lower [F(1, 50) = 5.08, p = 0.05] than with haptic feedback (M = 0.51, SD = 0.17).

H2b - Haptic feedback compared to no feedback shows significantly improved presence - is partly confirmed.

The overall presence with no feedback (M = 0.74, SD = 0.09) is not significantly lower than with haptic feedback (M = 0.81, SD = 0.14).

The possibility to act with no feedback (M = 0.74, SD = 0.12) is not significantly lower than with haptic feedback (M = 0.79, SD = 0.18). Still, the answers to item 8 "Were you able to anticipate what would happen next in response to the actions that you performed?" with no feedback (M = 0.61, SD = 0.22) are significantly worse [F(1, 50) = 3.54, p = 0.05] than with haptic feedback (M = 0.74, SD = 0.23).

The group of questions on haptics with no feedback (M = 0.48, SD = 0.19) is answered significantly worse [F(1, 50) = 31.77, p = 0.001] than with haptic feedback (M = 0.77, SD = 0.17).

1. Naturally, the answers to item 23 "How well could you actively survey or search the virtual environment using touch?" with no feedback (M = 0.37, SD = 0.41) are significantly worse [F(1, 50) = 25.51, p = 0.001] than with haptic feedback (M = 0.81, SD = 0.17).
2. The answers to item 24 "How well could you move or manipulate objects in the virtual environment?" with no feedback (M = 0.59, SD = 0.19) are significantly worse [F(1, 50) = 3.62, p = 0.05] than with haptic feedback (M = 0.72, SD = 0.21).

H2c - Haptic feedback compared to no feedback shows significantly improved usability - is rejected.

The differences of the usability and its items are not significant.

H2d - Haptic feedback compared to no feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.3 Optical vs. No Feedback

H3a - Optical feedback compared to no feedback shows significantly improved performance - is partly confirmed.

The final scores with optical feedback (M = 24.96, SD = 7.37) are not significantly better than with no feedback (M = 21.15, SD = 7.25). However, the performance in the second round with no feedback (M = 6.42, SD = 2.60) is significantly lower [F(1, 50) = 6.53, p = 0.05] than the second round with optical feedback (M = 8.35, SD = 2.72). The grabbing success rates with no feedback (M = 0.42, SD = 0.12) are significantly lower [F(1, 50) = 5.81, p = 0.05] than with optical feedback (M = 0.54, SD = 0.22).

H3b - Optical feedback compared to no feedback shows significantly improved presence - is partly confirmed.

The overall presence with no feedback (M = 0.76, SD = 0.09) is not significantly lower than with optical feedback (M = 0.81, SD = 0.11).

The possibility to act with no feedback (M = 0.74, SD = 0.12) is not significantly lower than with optical feedback (M = 0.82, SD = 0.17). However,

1. The answers to item 2 “How responsive was the environment to actions that you initiated (or performed)?” with no feedback (M = 0.75, SD = 0.19) are significantly worse [F(1, 50) = 3.37, p = 0.05] than with optical feedback (M = 0.85, SD = 0.15).
2. The answers to item 8 “Were you able to anticipate what would happen next in response to the actions that you performed?” with no feedback (M = 0.61, SD = 0.22) are significantly worse [F(1, 50) = 6.13, p = 0.05] than with optical feedback (M = 0.77, SD = 0.24).

The group of questions on haptics with no feedback (M = 0.48, SD = 0.19) is not answered significantly worse than optical feedback (M = 0.60, SD = 0.19). However, the answers to item 24 “How well could you move or manipulate objects in the virtual environment?” with no feedback (M = 0.59, SD = 0.19) are significantly worse [F(1, 50) = 6.61, p = 0.05] than with optical feedback (M = 0.75, SD = 0.25).

H3c - Optical feedback compared to no feedback shows significantly improved usability - is rejected.

The differences of the usability and its items are not significant.

H3d - Optical feedback compared to no feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.4 No vs. Both Feedback

H4a - Both feedback compared to no feedback shows significantly improved performance - is partly confirmed.

The final scores with both feedback types (M = 24.54, SD = 7.53) are not significantly better than with no feedback (M = 21.15, SD = 7.25). However, the grabbing success rates with no feedback (M = 0.42, SD = 0.12) are significantly lower [F(1, 50) = 6.35, p = 0.05] than with both feedback types (M = 0.53, SD = 0.19).

H4b - Both feedback compared to no feedback shows significantly improved presence - is confirmed.

The overall presence with no feedback (M = 0.76, SD = 0.09) is significantly lower [F(1, 50) = 6.77, p = 0.05] than with both feedback types (M = 0.83, SD = 0.10).

The possibility to act with no feedback (M = 0.74, SD = 0.12) is significantly lower [F(1, 50) = 7.36, p = 0.01] than with both feedback types (M = 0.84, SD = 0.14).

1. The answers to item 2 "How responsive was the environment to actions that you initiated (or performed)?" with no feedback (M = 0.75, SD = 0.186) are significantly worse [F(1, 50) = 4.80, p = 0.05] than with both feedback types (M = 0.859, SD = 0.165).
2. The answers to item 8 "Were you able to anticipate what would happen next in response to the actions that you performed?" with no feedback (M = 0.61, SD = 0.22) are significantly worse [F(1, 50) = 10.30, p = 0.01] than with both feedback types (M = 0.79, SD = 0.19).

The self-evaluation of performance with no feedback (M = 0.81, SD = 0.15) is not significantly lower than with both feedback types (M = 0.89, SD = 0.12). However, the answers to item 16 "How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?" with no feedback (M = 0.70, SD = 0.21) are significantly worse [F(1, 50) = 5.11, p = 0.05] than with both feedback types (M = 0.82, SD = 0.17).

The group of questions on haptics with no feedback (M = 0.48, SD = 0.19) is answered significantly worse [F(1, 50) = 24.02, p = 0.001] than with both feedback types (M = 0.76, SD = 0.21).

1. Naturally, the answers to item 23 "How well could you actively survey or search the virtual environment using touch?" with no feedback (M = 0.37, SD = 0.41) are significantly worse [F(1, 50) = 14.15, p = 0.001] than with both feedback types (M = 0.74, SD = 0.29).
2. The answers to item 24 "How well could you move or manipulate objects in the virtual environment?" with no feedback (M = 0.59, SD = 0.19) are significantly worse [F(1, 50) = 10.99, p = 0.05] than with both feedback types (M = 0.77, SD = 0.20).

H4c - Both feedback compared to no feedback shows significantly improved usability - is partly confirmed.

The overall usability with both feedback types (M = 0.84, SD = 0.11) is not significantly better than with no feedback (M = 0.77, SD = 0.16). However, the answers to item 3 "I thought the system was easy to use" with no feedback (M = 0.76, SD = 0.25) are significantly worse [F(1, 50) = 7.06, p = 0.05] than with both feedback types (M = 0.91, SD = 0.14).

H4d - Both feedback compared to no feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.5 Haptic vs. Optical Feedback

H5a - Haptic feedback compared to optical feedback shows significantly improved performance - is rejected.

The differences of the scores are not significant.

H5b - Haptic feedback compared to optical feedback shows significantly improved presence - is partly confirmed.

The overall presence with haptic feedback (M = 0.81, SD = 0.14) is not significantly higher than with optical feedback (M = 0.81, SD = 0.11).

Nevertheless, the group of questions on haptics with optical feedback (M = 0.60, SD = 0.27) is answered significantly worse [F(1, 50) = 6.99, p = 0.05] than with haptic feedback (M = 0.77, SD = 0.17). Naturally, the answers to item 23 "How well could you actively survey or search the virtual environment using touch?" with optical feedback (M = 0.45, SD = 0.43) are significantly worse [F(1, 50) = 15.92, p = 0.001] than with haptic feedback (M = 0.81, SD = 0.17).

H5c - Haptic feedback compared to optical feedback shows significantly improved usability - is rejected.

The differences of the usability and its items are not significant.

H5d - Haptic feedback compared to optical feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.6 Both vs. Haptic Feedback

H6a - Both feedback compared to haptic feedback shows significantly improved performance - is rejected.

The differences of the scores are not significant.

H6b - Both feedback compared to haptic feedback shows significantly improved presence - is rejected.

The differences of the presence and its items are not significant.

H6c - Both feedback compared to haptic feedback shows significantly improved usability - is rejected.

The differences of the usability and its items are not significant.

H6d - Both feedback compared to haptic feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.7 Both vs. Optical Feedback

H7a - Both feedback compared to optical feedback shows significantly improved performance - is rejected.

The differences of the scores are not significant.

H7b - Both feedback compared to optical feedback shows significantly improved presence - is partly confirmed.

The overall presence with both feedback types (M = 0.83, SD = 0.10) is not significantly higher than with optical feedback (M = 0.81, SD = 0.11).

Again, the group of questions on haptics with optical feedback (M = 0.60, SD = 0.27) is answered significantly worse [F(1, 50) = 5.37, p = 0.05] than with both feedback types (M = 0.76, SD = 0.21). The answers to item 23 "How well could you actively survey or search the virtual environment using touch?" with optical feedback (M = 0.45, SD = 0.43) are significantly worse [F(1, 50) = 8.20, p = 0.001] than with both feedback types (M = 0.74, SD = 0.29).

H7c - Both feedback compared to optical feedback shows significantly improved usability - is rejected.

The differences of the usability and its items are not significant.

H7d - Both feedback compared to optical feedback shows significantly improved task load - is rejected.

The differences of the task load and its items are not significant.

6.3.8 Summary and Analysis

Despite the competitive setting and the chocolate promised for good scores, the testers' motivation and patience was diverse. This affected their performance, which caused higher variances within the results and with it less significant statistics.

The testers used different strategies. For example, some testers used both controllers simultaneously. This did not necessarily improve their scores. Still, it had an impact on their grabbing success rate, mostly because the ball can only be held in one controller at once. If they fail to grab the ball, it is often squeezed between both controllers leading to very high forces. These forces then make the ball dash away and bounce wildly through the room. This makes the physics less predictive and makes the testers chase the ball longer.

Another strategy is grab spamming, which means that the users press and release the trigger repeatedly and fast. Simply put, this replaces accuracy with luck. The spamming did not necessarily increase their scores because they did not always notice when they successfully grabbed the ball and released it accidentally. Such testers complained more about the grabbing being unresponsive or even broken. The connection might also be the other way around. The testers failed in grabbing, which made the grabbing feel unresponsive and broken, which in turn made them start spamming.

A simpler test could probably produce more significant results. For example, the testers could only use one controller to take the different strategies into account. Also, the test could be simplified by doing only the actual grabbing, so without throwing. Trigger spamming could be resolved by only allowing the grabbing every few seconds. Also, the questionnaires could be re-formulated so that good results are always on the right.

Performance

The influence of the different feedback types on the final scores is not significant. The average scores suggest that any feedback leads to better performance than no feedback at all. Haptic feedback has the highest scores in the first round. This is not significant but hints haptic feedback might be more intuitive than optical feedback. Optical feedback alone results in the highest overall scores, even higher than both haptic and optical feedback. This indicates that combining different feedback types does not necessarily improve the performance.

The influence of feedback is significant in only two cases: no feedback vs. haptic feedback in round 1 and no feedback vs. optical feedback in round 2. The scores of each round are shown in Figure 6.3. Haptic feedback is better than no feedback in the first two rounds, while optical feedback is better than no feedback in the first three rounds. This indicates that haptic feedback is perceived with more delay than optical feedback and helps more in the easier rounds.

The fourth round is almost the same for each feedback type. The distance between the controller and its grabbing volume approaches the tracking accuracy of the Vive. Therefore, the scores in the fourth round are dominated by random effects, such as jitter. However, these effects are not significant and require further analysis.

The second round has the best scores of all rounds. This implies that the learning effect dominates from round one to two. Afterwards, the increasing difficulty dominates.

Presence

The overall presence scores of both feedback types are significantly better than no feedback. All other comparisons are not significant. However, the average presence scores suggest that haptic feedback creates a deeper presence than optical feedback. This implies that adding feedback for another sense might be better than adding feedback for a sense that is already used.

The ability to act with both feedback types is reported significantly higher than without any feedback. This comes mostly from two questions: “How responsive was the environment to actions that you initiated (or performed)?” and “Were you able to anticipate what would happen next in response to the actions that you performed?”. Also, the self-evaluation of performance plays a role, especially through the question “How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?”. The influence of one or both feedback types compared with no feedback is significant. However, when comparing one feedback type with the other or with both, the influence is not significant. This means that, in the grabbing test, no feedback is especially bad.

Not surprisingly, the haptics scores are significantly better with vibrations enabled than without. The question “How well could you actively survey or search the virtual environment using touch?” with only haptic feedback is answered better than with both feedback types, although not significantly so. This suggests that the haptic feedback is superseded by optical feedback to some degree.

Not in a single aspect is Haptic feedback alone significantly different than both feedback types together. The same applies to optical feedback, except for the haptics scores of the presence questionnaire. Therefore, combining both feedback types does not necessarily increase presence.

Usability

Overall usability with no feedback is worse than with any other feedback type. However, the difference is not significant. The usability scores for haptic, optical, and both feedback types are almost the same. Only the answers to the question “I thought the system was easy to use” without feedback are significantly worse than with both feedback types.

Task Load

The feedback type has no significant influence on task load. The average scores of the overall questionnaire and all its items are almost the same with a high variance. Only the question “Performance - How successful were you in accomplishing what you were asked to do?” suggests that no feedback is the worst option. This is similar to the self-evaluation of performance in the presence questionnaire.

6.4 Point-Cloud Editing

Comparing point-cloud editing on the desktop with VR shows that the users are much more accurate and motivated in VR. The performance is measured by counting wrong points. The selected and non-selected points are compared with a manually created ground-truth selection. The task is to select all points that do not directly belong to the house in a laser scan. The performance, usability and task load of the VR and desktop versions are compared. The presence of the VR version is compared to the basketball game, because comparing it with a desktop program makes no sense. The hypotheses are:

- H1 Point-cloud editing in VR has a significantly higher user performance than on the desktop.
- H2 Point-cloud editing in VR has a significantly lower task load than on the desktop.
- H3 Point-cloud editing in VR has a significantly better usability than on the desktop.
- H4 Point-cloud editing in VR does not have a significantly worse presence than the basketball game.

6.4.1 H1 - Point-cloud editing in VR has a significantly higher user performance than on the desktop - is confirmed.

The users achieve significantly [F(1, 50) = 14.96, p = 0.001] fewer wrong points in VR (M = 859.35, SD = 781.49) than on the desktop (M = 2546.00, SD = 2035.25). The wrong points are a sum of wrongly selected points and wrongly non-selected points. The number of wrongly selected points is significantly [F(1, 50) = 6.89, p = 0.05] lower in VR (M = 259.77, SD = 310.30) than on the desktop (M = 1314.88, SD = 1985.78). Also, the number of wrongly non-selected points is significantly [F(1, 50) = 15.25, p = 0.001] lower in VR (M = 599.58, SD = 685.95) than

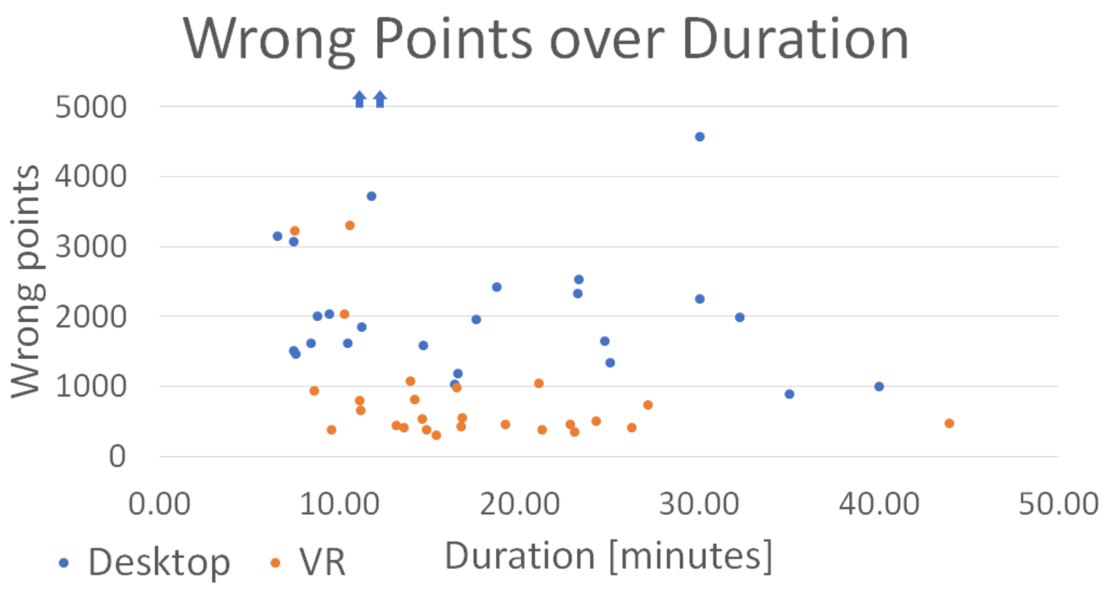


Figure 6.6: This scatter plot shows the working duration on the x-axis and the number of wrong points on the y-axis. The point-cloud editing in VR results in significantly fewer wrong points than on the desktop with similar durations.

on the desktop (M = 1231.12, SD = 428.12)s. Figure 6.6 shows a scatter plot of the number of wrong points with the working duration.

One reason for the considerable differences is that the testers could up-scale the point cloud in VR much further and therefore select much more accurately. In the desktop version, the camera has a minimal distance to the orbiting center of about 3 virtual meters. With moving the orbiting center, the users can in theory view some points very closely. However, this takes time and is very sensitive to minimal mouse movements. In VR, however, the users can move and scale the house from a bird's to a snail's perspective within seconds and with full control. This can leave several centimeters between single points, which makes the selection very easy. Another reason might be that just viewing the point cloud in VR gives more information, for example about edges.

An operation in the desktop version is to draw a lasso while an operation in the VR version is to press the trigger, move the controller through the points and release the trigger again. Although the operations are not completely comparable, their number is a hint for how often the view is suitable for selecting more points. The number of operations in VR is (M = 169.00, SD = 110.54) significantly higher [F(1, 50) = 26.76, p = 0.001] than on the desktop (M = 51.77, SD = 24.85). The main reason is that in VR, the view can be changed – either by moving the head or moving the point cloud – while selecting points. Therefore, the view in VR can always be suitable for selecting more points. In contrast, one can only select or change the view at a time in the desktop version.

The average duration in seconds in the desktop version (M = 1058.23, SD = 573.43) is not significantly longer than in the VR version (M = 1029.85, SD = 456.98). The duration on the desktop ranges from 6 to 40 minutes, while it ranges from 7 to 44 minutes in VR. This means that the users were able to get significantly better results in about the same time in VR.

6.4.2 H2 - Point-cloud editing in VR has a significantly lower task load than on the desktop - is confirmed.

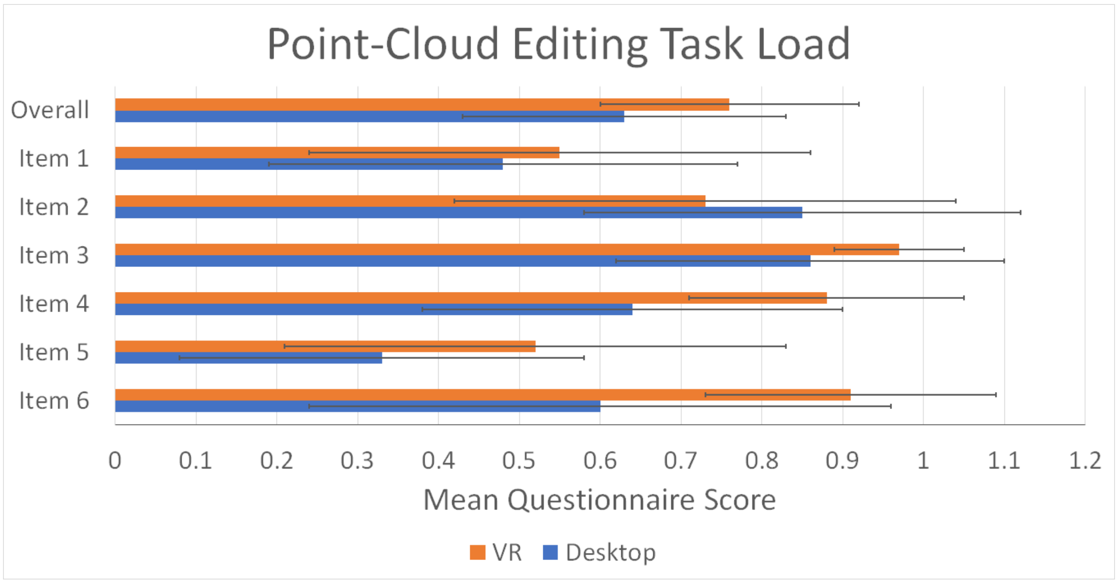


Figure 6.7: This graph displays the results of the task load questionnaire for point-cloud editing on the desktop and in VR.

Figure 6.7 illustrates the results of the task load questionnaire. Some questions are inversed. Therefore, high scores of up to 1.0 mean good results, while low scores of down to 0.0 mean bad results. The overall task load in VR (M = 0.76, SD = 0.16) is significantly [F(1, 50) = 6.91, p = 0.05] lower than on the desktop (M = 0.63, SD = 0.20).

1. Q: Mental Demand - How mentally demanding was the task?
   A: The mental demand in VR (M = 0.55, SD = 0.31) is not significantly lower than on the desktop (M = 0.48, SD = 0.29).
2. Q: Physical Demand - How physically demanding was the task?
   A: The physical demand in VR (M = 0.73, SD = 0.31) is slightly, but not significantly, higher than on the desktop (M = 0.85, SD = 0.27).
3. Q: Temporal Demand - How hurried or rushed was the pace of the task?
   A: Although both tasks had no time limit, the users felt significantly [F(1, 50) = 5.13, p = 0.05] more rushed in the desktop version (M = 0.86, SD = 0.24) than in the VR version (M = 0.97, SD = 0.08). One tester even compared the VR version to a Zen garden and suggested adding relaxing music.

4. Q: Performance - How successful were you in accomplishing what you were asked to do?
   A: The testers were not only more relaxed in VR, but they also felt significantly [F(1, 50) = 14.73, p = 0.001] more successful (M = 0.88, SD = 0.17) than on the desktop (M = 0.64, SD = 0.26).
5. Q: Effort - How hard did you have to work to accomplish your level of performance?
   A: The testers worked significantly [F(1, 50) = 5.88, p = 0.05] less hard in VR (M = 0.52, SD = 0.31) than on the desktop (M = 0.33, SD = 0.25).
6. Q: Frustration - How insecure, discouraged, irritated, stressed, and annoyed were you?
   A: The testers were significantly [F(1, 50) = 15.39, p = 0.001] more frustrated in the desktop version (M = 0.60, SD = 0.36) than in the VR version (M = 0.91, SD = 0.18).

6.4.3 H3 - Point-cloud editing in VR has a significantly better usability than on the desktop - is confirmed.

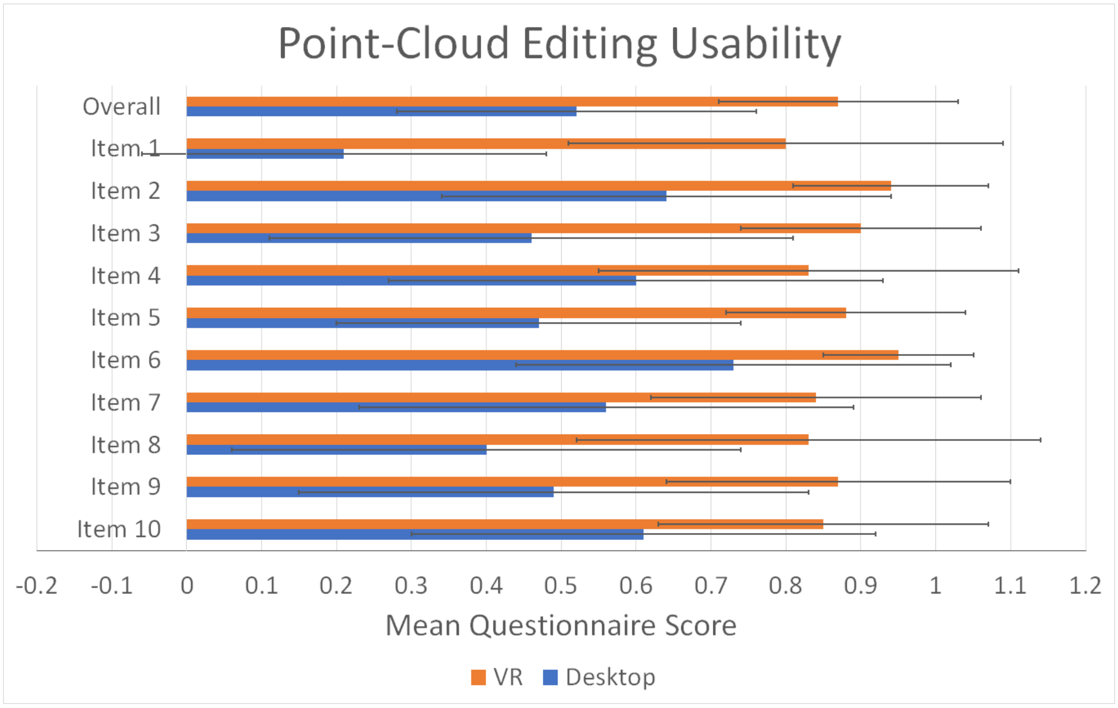


Figure 6.8: This graph displays the results of the usability questionnaire for point-cloud editing on the desktop and in VR.

Figure 6.7 illustrates the results of the usability questionnaire. The overall usability of the VR version (M = 0.87, SD = 0.16) is significantly [F(1, 50) = 38.20, p = 0.001] better than the usability of the desktop version (M = 0.52, SD = 0.24).

1. Q: I think that I would like to use this system frequently.
   A: The testers like to use the VR version (M = 0.80, SD = 0.29) significantly [F(1, 50) = 54.69, p = 0.001] better than the desktop version (M = 0.21, SD = 0.27).

2. Q: I found the system unnecessarily complex.
   A: The testers found the desktop version (M = 0.64, SD = 0.30) significantly [F(1, 50) = 20.55, p = 0.001] more complex than the VR version (M = 0.94, SD = 0.13).
3. Q: I thought the system was easy to use.
   A: The testers found the VR version (M = 0.90, SD = 0.16) significantly [F(1, 50) = 33.06, p = 0.001] easier to use than the desktop version (M = 0.46, SD = 0.35).
4. Q: I think that I would need the support of a technical person to be able to use this system.
   A: The testers also think that they would need significantly [F(1, 50) = 7.32, p = 0.01] less support for the VR version (M = 0.83, SD = 0.28) than for the desktop version (M = 0.60, SD = 0.33).
5. Q: I found the various functions in this system were well integrated.
   A: The testers found the functions in VR (M = 0.88, SD = 0.16) significantly [F(1, 50) = 41.21, p = 0.001] better integrated than on the desktop (M = 0.47, SD = 0.27).
6. Q: I thought there was too much inconsistency in this system.
   A: The testers found significantly [F(1, 50) = 13.43, p = 0.001] more inconsistency on the desktop (M = 0.73, SD = 0.29) than in VR (M = 0.95, SD = 0.10).
7. Q: I would imagine that most people would learn to use this system very quickly.
   A: The testers think that most people would learn the VR version (M = 0.84, SD = 0.22) significantly [F(1, 50) = 12.16, p = 0.01] faster than the desktop version (M = 0.56, SD = 0.33).
8. Q: I found the system very cumbersome to use.
   A: The testers found the desktop version (M = 0.40, SD = 0.34) significantly [F(1, 50) = 21.23, p = 0.001] more cumbersome than the VR version (M = 0.83, SD = 0.31).
9. Q: I felt very confident using the system.
   A: The testers were significantly [F(1, 50) = 21.07, p = 0.001] more confident in VR (M = 0.87, SD = 0.23) than on the desktop (M = 0.49, SD = 0.34).
10. Q: I needed to learn a lot of things before I could get going with this system.
    A: The testers needed to learn significantly [F(1, 50) = 9.93, p = 0.01] less for the VR version (M = 0.85, SD = 0.22) than for the desktop version (M = 0.61, SD = 0.31).

6.4.4 H4 - Point-cloud editing in VR does not have a significantly worse presence than the basketball game - is confirmed.

The focus in this task is on user performance, which does probably not depend on presence. Therefore, presence is only a secondary goal, but its scores are still surprisingly good compared to the basketball game, which has a focus on presence. The presence of the point-cloud editing in VR (M = 0.75, SD = 0.10) is not significantly worse than in the basketball game (M = 0.79, SD = 0.09).

The ability to act is rated significantly [F(1, 50) = 12.41, p = 0.001] better in the point-cloud editing (M = 0.90, SD = 0.11) than in the basketball game (M = 0.79, SD = 0.10). Surprisingly, the haptics in the point-cloud editing (M = 0.82, SD = 0.17) are also rated significantly [F(1, 50) = 4.66, p = 0.05] better than in the basketball game (M = 0.71, SD = 0.21).

Going into more detail, the testers reported that they were able to control the events in point-cloud editing (M = 0.93, SD = 0.12) significantly [F(1, 50) = 20.84, p = 0.001] better than in basketball (M = 0.68, SD = 0.24). Also, they found the point-cloud editing (M = 0.90, SD = 0.12) significantly [F(1, 50) = 6.06, p = 0.05] more responsive than the basketball game (M = 0.81, SD = 0.12). The testers found the interactions in the point-cloud editing (M = 0.90, SD = 0.12) more, but not significantly, natural than the basketball game (M = 0.81, SD = 0.12). This is surprising, too, considering that surely nobody has ever scaled a real house with his bare hands. They could anticipate what happened significantly [F(1, 50) = 5.02, p = 0.05] better in the point-cloud editing (M = 0.89, SD = 0.15) than in the basketball game (M = 0.79, SD = 0.17). The testers could examine the object significantly [F(1, 50) = 6.24, p = 0.05] closer in point-cloud editing (M = 0.95, SD = 0.09) than in the basketball game (M = 0.88, SD = 0.11). Also, they could examine the objects significantly [F(1, 50) = 7.00, p = 0.05] better from multiple viewpoints in point-cloud editing (M = 0.95, SD = 0.09) than in the basketball game (M = 0.86, SD = 0.14). The testers could not survey significantly better with touch in the point-cloud editing (M = 0.72, SD = 0.29) than in the basketball game (M = 0.60, SD = 0.35). However, they could manipulate the objects significantly [F(1, 50) = 5.56, p = 0.05] better in the point-cloud editing (M = 0.92, SD = 0.12) than in the basketball game (M = 0.82, SD = 0.18).

6.4.5 Summary and Analysis

The user study gives a first hint that point-cloud editing might be easier and more efficient in VR than on the desktop. The testers liked the VR version, and they disliked the desktop version. Especially viewing and selecting seem to be better in VR, which results here in better performance, usability and task load. The testers had significantly fewer wrong points in VR in about the same time as on the desktop.

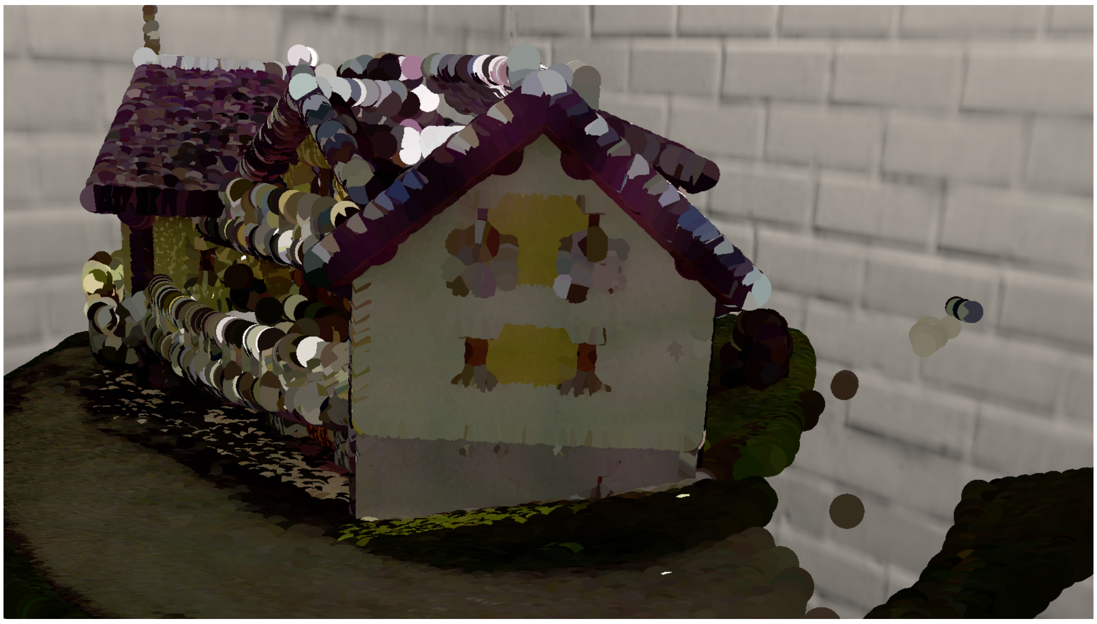
One tester stated that 15 minutes is the maximum time to work comfortably. All in all, fatigue was no big problem because the testers were relaxed, and most of them finished quickly enough. In theory, the system could also be used while sitting in a chair, which would make this even less of an issue.

Another tester found the controller buttons hard to hit. When more options are added, a different assignment of buttons is necessary. This could also help when the trackpad loses the thumb position at the edges. A better button assignment could be, for example, to use the trackpad to control a toolbox and switch between different options like select and un-select. The grip button could then be pressed to actually select the points.

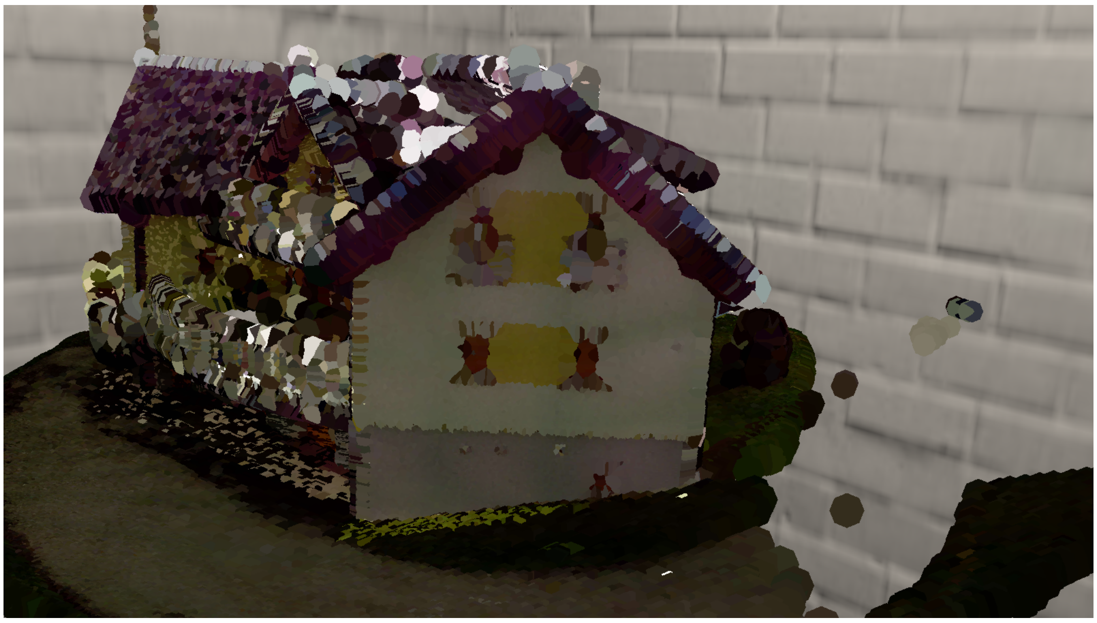
Point-Cloud Rendering

In order to reduce overlap and also show more details of a point cloud, the individual points are drawn as cones. While the old rendering technique created the cone geometry in the fragment shader, the new technique does it in the geometry shader.

The draw time of the old and new cones was measured 900 times, 3 seconds after the start-up. This was done three times in separate runs on the same NVIDIA GeForce GTX 1070. Figure 6.9 shows the same point cloud rendered with both techniques and their average draw time. The visual quality of the new cones is very similar to the old ones, which shows that the difference



(a) Old cones - 13.533 ms



(b) New cones - 1.858 ms

Figure 6.9: The slightly tilted grass of the house scan with about 700,000 points is shown from above. The old cones are view-aligned squares that are turned into pixel-accurate cones in the fragment shader, while the new cones are points turned into approximated, triangulated cones in the geometry shader. Both techniques produce similar visual quality, but the new version is much faster.

between pixel-accurate cones and small approximated cones is negligible. Also, the new cones need less than 2 ms draw time, while the old cones need more than 13 ms. To summarize these results, rendering a rather sparse point cloud with big points is faster using the geometry shader than setting the depth in the fragment shader, at least on the NVIDIA GeForce GTX 1070.

The LoD system for rendering points should be improved because it led to frequent popping when the house was scaled large. This is probably caused by the LoD being sensitive to the jitter of the HMD tracking. When the scale of the house is small, the LoD often made accidentally non-selected points obvious. This is caused by the strong contrast between selected and non-selected points. The LoD system renders too many points when the point cloud is scaled very small and the view is close. In future work, an overall budget for the rendered points should be introduced, similar to the point-in-selection-volume check. This budget could also be multiplied with a quality factor for different GPUs.

Bi-Manual Pinch Gesture

The viewing seems to be one major advantage of VR over desktop in our user study. This finding cannot be generalized yet because other viewing methods still need to be compared. For example, a fly-through method may work both in VR and on desktop.

The orbiting camera of the VGM has a minimal distance to the orbiting center. Therefore, the results may not be completely applicable to other desktop applications such as Potree [Sch15] and CloudCompare [GM15], which have an orbiting camera without a minimal distance. However, the three point-cloud viewers for desktop share the fundamental flaw that controlling the 6 DoF requires two different but not independent operations. To be more exact, the problems are:

1. Moving the orbiting center and changing the camera direction is partly redundant because both move the camera.
2. When moving the orbiting center, the possible directions depend on the camera direction.

With the bi-manual pinch gesture, however, all 6 DoF can be controlled with one operation. Only rotation around the axis between the controllers cannot be controlled accurately at the moment. If necessary, this rotation can be done very quickly with one hand after the scaling, which seems to suffice for most users. Also, future work may add control over this axis by incorporating the controllers' rotation.

Most testers could learn the bi-manual pinch gesture very quickly. However, some of them were confused in the beginning, probably because the scaling moves some points closer and some farther. The walls of the in-door scene seem to help, considering that no tester lost the point cloud. However, this is probably different in open scenes without walls.

Point-Cloud Selections

The selection technique seems to be the second major advantage of VR over desktop in our user study. The task to select vegetation in a house scan did not even use the full potential of the VR selection technique. Consider a scan of a multi-story building with the task to delete the furniture. The desktop version would require at least two operations, select the object from one direction and limit the range of the selection from another direction, for each piece of furniture. In the VR version, there would be no difference. One tester, an archeologist, said that he would have liked the software to remove cables from a mine scan.

The strategy that was used most often was to make a rough selection first and then make it more and more detailed. Most of the testers always looked onto the house scan from above, even when they had the controllers below the ground.

Some testers did not recognize the selection volume as a sphere. Instead, they thought it was a disk. This could be improved simply by using a better shading than transparent flat shading. However, the highlights of e.g. Phong shading could confuse the users. Also, different selection volumes would help. A cube or cone would be especially useful when selecting points in a corner.

The selection visualization could be improved by tinting the selected points instead of overwriting their colors. This could help with differentiating points made of different materials.

One tester scaled the house very small, so that the entire point-cloud was displayed with only a few visible points. She then mistook it for outliers and selected everything, overriding the previous selection. Therefore, an undo / redo system is necessary.

One tester suggested to activate the vibration only when currently selecting or un-selecting. Then, the already selected or un-selected points could be ignored for the vibration. On the other hand, the users would not get the feedback for the number of points inside the selection volume before they select. Therefore, the vibration would not display a predicting of the action, but instead, it would give feedback for the result. Some testers probably did not perceive the vibrations consciously, and one tester just found the vibration annoying.

The pinch gesture scaling and selection-volume scaling are somewhat redundant. Most testers appreciated it, anyway. The pinch gesture was used regularly, while the volume scaling was done only a few times in one session. Some testers used differently scaled selection volumes in each hand for rough and fine selections.

CHAPTER 7

Conclusion and Future Work

This thesis focuses on interactions in room-scale VR. We created a simple basketball game to analyze grabbing and throwing with controllers. Furthermore, we developed an application to examine the influence of haptic and optical feedback for grabbing. The third application created for this thesis, point-cloud editing in VR, allows users to view a point cloud as well as select and delete points of it.

We conducted a user study to answer our three research questions:

1. Is grabbing and throwing virtual balls with hand-held controllers intuitive?
2. Has feedback a statistically significant influence on grabbing?
3. Is editing point clouds better in VR than on desktop?

7.1 Grabbing and Throwing with Hand-Held Controllers

The first task for the participants of our user study was to play the basketball game. The testers grabbed the ball and threw it into the basket as often as they could in the given time.

The user study showed that grabbing with a controller button is intuitive but throwing is not. Releasing a button is a bad metaphor for releasing a grabbed virtual object in order to throw it. In future work, we could test whether a method with a focus on physics helps. The controller would then have a model of an open hand as its virtual representation and the ball is only attached to the hand via a physics constraint until the hand is palm-up. Then, the throwing could be completely natural. The greatest challenge of this method would probably be a stable collision handling.

7.2 The Influence of Haptic and Optical Feedback

After the basketball game, the testers did the grabbing test four times, with no, haptic, optical, and both feedback enabled in random order.

The study revealed that any feedback is better than none. Adding haptic, optical, or both feedback types to the grabbing improves the user performance and presence. However, only sub-scores like accuracy and predictability are significantly improved. Usability and task load are mostly unaffected by feedback. A future user study should focus even more on the grabbing by omitting the second controller and without throwing the balls into a basket. Then, one might find more significant differences between the feedback types.

7.3 Innovations for Point-Cloud Editing in VR

The development of the point-cloud editing tool in VR did not only serve as basis for the third task of the user study, but also led to three new technical contributions. The first innovation is the bi-manual pinch gesture, a viewing technique for translating, rotating, and scaling point clouds. Our testers could learn this viewing technique very quickly, and they were able to use it efficiently in most cases. This technique should be improved in future work by incorporating wrist rotation into the rotation around the axis between the controllers.

The second novelty is a fast rendering technique for sparse point clouds. We draw the points as not-so-small cones to fill in holes and keep the overlap small at the same time. This helps reduce the flickering caused by jitter of the HMD tracking. To speed-up the drawing, the cones are not created in the fragment shader like in previous techniques but in the geometry shader. In future work, we will test whether instance-drawing is even faster than the geometry shader.

The third new technique is a method for efficient point selections and the visualization of these selections in real time. Users can move a selection volume, a sphere in our user study, through the point cloud. The position of the selection volume is stored in discrete steps, which creates an array of positions. This array can be rendered to the stencil buffer, and with the correct stencil test, every point inside the trajectory of the selection volume can be colored as part of the selection. When the user finishes the selection operation, the array of sphere positions can be used for point-in-sphere checks to update the state of only the selected points.

7.4 The Advantages of Point-Cloud Editing in VR

The last task of the user study was to clear a point cloud in VR and on desktop. The testers removed everything that did not directly belong to the house in the laser scan, most importantly soil and vegetation.

The results of the user study indicate that the point-cloud editing in VR with our innovations is indeed better than on desktop with an orbiting camera and lasso selections. However, to say that point-cloud editing in VR is better in general requires more validation. For example, other viewing methods for VR and desktop need to be compared, most of all a fly-through camera. If VR convinces in these tests as well, new challenges and questions may arise. For example, when users clear point clouds for an hour or more, is the increased efficiency worth the possible simulator sickness and fatigue? Or can the work in VR even keep the users fit and healthy?

APPENDIX A

Grabbing Test Results

Feedback	None		Haptic		Optical		Both	
	M	SD	M	SD	M	SD	M	SD
Score All Rounds	21.15	7.25	23.35	6.79	24.96	7.37	24.54	7.53
Score Round 1	5.77	2.55	7.23	1.95	6.96	2.36	7.00	2.24
Score Round 2	6.42	2.60	7.38	2.68	8.35	2.72	7.46	2.98
Score Round 3	5.50	2.04	5.65	2.29	6.42	2.37	6.35	2.09
Score Round 4	3.46	2.06	3.08	1.62	3.23	1.60	3.73	1.87
Score Grabs	25.42	8.62	27.19	7.50	28.54	8.34	28.62	9.62
Score Grab Attempts	61.54	16.27	57.42	21.61	59.19	23.07	58.08	20.64
Score Success Rate	0.42	0.12	0.51	0.17	0.54	0.22	0.53	0.19

Table A.1: Means and standard deviations of the grabbing test score results.

Feedback	None		Haptic		Optical		Both	
	M	SD	M	SD	M	SD	M	SD
Presence Overall	0.76	0.09	0.81	0.14	0.81	0.11	0.83	0.10
Presence Realism	0.79	0.12	0.81	0.14	0.83	0.12	0.84	0.11
Presence Act	0.74	0.12	0.79	0.18	0.82	0.17	0.84	0.14
Presence Interface	0.85	0.11	0.87	0.12	0.88	0.11	0.88	0.11
Presence Examine	0.83	0.15	0.83	0.17	0.86	0.14	0.86	0.12
Presence Performance	0.81	0.15	0.87	0.16	0.87	0.16	0.89	0.12
Presence Sound	0.72	0.29	0.73	0.25	0.74	0.22	0.74	0.21
Presence Haptic	0.48	0.19	0.77	0.17	0.60	0.27	0.76	0.21
Presence Item 1	0.71	0.25	0.74	0.21	0.78	0.25	0.81	0.19
Presence Item 2	0.75	0.19	0.79	0.18	0.85	0.15	0.86	0.16
Presence Item 3	0.66	0.25	0.73	0.22	0.78	0.18	0.75	0.19
Presence Item 4	0.85	0.16	0.83	0.18	0.79	0.23	0.87	0.17
Presence Item 5	0.79	0.20	0.83	0.20	0.85	0.23	0.85	0.17
Presence Item 6	0.78	0.20	0.79	0.21	0.86	0.15	0.87	0.16
Presence Item 7	0.62	0.25	0.70	0.21	0.72	0.21	0.71	0.18
Presence Item 8	0.61	0.22	0.74	0.23	0.77	0.24	0.79	0.19
Presence Item 9	0.90	0.15	0.87	0.19	0.88	0.18	0.90	0.12
Presence Item 10	0.92	0.13	0.90	0.15	0.90	0.14	0.90	0.15
Presence Item 11	0.86	0.19	0.85	0.20	0.88	0.14	0.88	0.15
Presence Item 12	0.86	0.24	0.88	0.20	0.84	0.27	0.88	0.16
Presence Item 13	0.90	0.09	0.88	0.16	0.91	0.12	0.92	0.11
Presence Item 14	0.88	0.12	0.92	0.08	0.88	0.18	0.93	0.09
Presence Item 15	0.92	0.17	0.92	0.13	0.94	0.15	0.96	0.10
Presence Item 16	0.70	0.21	0.81	0.22	0.80	0.24	0.82	0.17
Presence Item 17	0.88	0.18	0.94	0.12	0.92	0.12	0.90	0.14
Presence Item 18	0.78	0.22	0.76	0.26	0.83	0.20	0.81	0.19
Presence Item 19	0.76	0.21	0.78	0.21	0.85	0.15	0.81	0.17
Presence Item 20	0.69	0.31	0.67	0.30	0.65	0.30	0.65	0.33
Presence Item 21	0.79	0.33	0.81	0.29	0.85	0.24	0.82	0.23
Presence Item 22	0.67	0.36	0.70	0.32	0.72	0.33	0.74	0.25
Presence Item 23	0.37	0.41	0.81	0.17	0.45	0.43	0.74	0.29
Presence Item 24	0.59	0.19	0.72	0.21	0.75	0.25	0.77	0.20

Table A.2: Means and standard deviations of the grabbing test presence results.

Feedback	None		Haptic		Optical		Both	
	M	SD	M	SD	M	SD	M	SD
Task Load Overall	0.55	0.20	0.58	0.17	0.62	0.20	0.61	0.22
Task Load Item 1	0.66	0.25	0.71	0.25	0.77	0.25	0.71	0.27
Task Load Item 2	0.60	0.28	0.56	0.23	0.61	0.26	0.62	0.27
Task Load Item 3	0.44	0.32	0.44	0.26	0.42	0.31	0.46	0.33
Task Load Item 4	0.57	0.28	0.69	0.25	0.72	0.27	0.68	0.26
Task Load Item 5	0.40	0.28	0.38	0.22	0.46	0.27	0.45	0.30
Task Load Item 6	0.64	0.33	0.70	0.29	0.75	0.32	0.74	0.31

Table A.3: Means and standard deviations of the grabbing test task load results.

Feedback	None		Haptic		Optical		Both	
	M	SD	M	SD	M	SD	M	SD
Usability Overall	0.77	0.16	0.81	0.16	0.82	0.16	0.84	0.11
Usability Item 1	0.56	0.31	0.62	0.31	0.63	0.30	0.63	0.29
Usability Item 2	0.88	0.21	0.89	0.14	0.91	0.14	0.93	0.11
Usability Item 3	0.76	0.25	0.84	0.26	0.87	0.22	0.91	0.14
Usability Item 4	0.81	0.29	0.81	0.27	0.78	0.29	0.84	0.26
Usability Item 5	0.80	0.21	0.86	0.21	0.88	0.17	0.86	0.16
Usability Item 6	0.85	0.23	0.87	0.23	0.88	0.21	0.88	0.17
Usability Item 7	0.77	0.24	0.82	0.23	0.79	0.27	0.84	0.21
Usability Item 8	0.73	0.28	0.79	0.27	0.77	0.28	0.84	0.25
Usability Item 9	0.68	0.27	0.80	0.22	0.82	0.24	0.77	0.18
Usability Item 10	0.87	0.25	0.85	0.23	0.87	0.21	0.89	0.19

Table A.4: Means and standard deviations of the grabbing test usability results.

Bibliography

[AM95] David G Alciatore and Rick Miranda. A winding number and point-in-polygon algorithm. *Glaxo Virtual Anatomy Project Research Report, Department of Mechanical Engineering, Colorado State University*, 1995.

[Bag16] BagoGames. HTC Vive Now Up For Pre-Order, 2016. Accessed: 2017-05-03. URL: `https://www.flickr.com/photos/bagogames/25845851080`, published under CC BY 2.0 (`https://creativecommons.org/licenses/by/2.0/`).

[BH97] Doug A Bowman and Larry F Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 35–ff. ACM, 1997.

[BHZK05] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, pages 17–141. IEEE, 2005.

[BJOYBJK90] Frederick P Brooks Jr, Ming Ouh-Young, James J Batter, and P Jerome Kilpatrick. Project gropehaptic displays for scientific visualization. In *ACM SIGGraph computer graphics*, volume 24, pages 177–185. ACM, 1990.

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

[Bro96] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189:4–7, 1996.

[BRR07] Stéphane Bouchard, Geneviève Robillard, and Patrice Renaud. Revising the factor structure of the simulator sickness questionnaire. *Annual review of cybertherapy and telemedicine*, 5:128–137, 2007.

[Bur99] Grigore C Burdea. Haptic feedback for virtual reality. In *Virtual reality and prototyping workshop*, volume 2, pages 17–29. Citeseer, 1999.

[CFH97] Lawrence D Cutler, Bernd Fröhlich, and Pat Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 107–114. ACM, 1997.

[CNSD+92] Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. The cave: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–73, 1992.

[Cou08] Erwin Coumans. Bouncing ball, 2008. Accessed: 2017-05-18. URL: `http://www.bulletphysics.org/Bullet/phpBB3/viewtopic.php?f=9&t=3003`.

[Cro77] Franklin C Crow. Shadow algorithms for computer graphics. In *Acm siggraph computer graphics*, volume 11, pages 242–248. ACM, 1977.

[Cyb13] Cyberith. Virtualizer, 2013. Accessed: 2017-04-19. URL: `http://cyberith.com/product/`.

[Cyb17] CyberGlove Systems. CyberTouch, 2017. Accessed: 2017-04-19. URL: `http://www.cyberglovesystems.com/cybertouch/`.

[Dig16] Digital Trends. Spec Comparison: Does the Rift's Touch update make it a true Vive competitor?, 2016. Accessed: 2017-04-19. URL: `http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/`.

[Dig17] Digital Trends. Spec Comparison: Can the plucky PlayStation VR upset HTC's Vive?, 2017. Accessed: 2017-04-25. URL: `http://www.digitaltrends.com/virtual-reality/playstation-vr-vs-htc-vive-spec-comparison/`.

[DKFD00] Cath Dillon, Edmund Keogh, Jonathan Freeman, and Jules Davidoff. Aroused and immersed: the psychophysiology of presence. In *Proceedings of 3rd International Workshop on Presence, Delft University of Technology, Delft, The Netherlands*, pages 27–28, 2000.

[Doo17] Doodle. MeetMe page, 2017. Accessed: 2017-05-03. URL: `http://doodle.com/meetme`.

[Dra98] Mark Heider Draper. *The adaptive effects of virtual interfaces: vestibulo-ocular reflex and simulator sickness*. PhD thesis, University of Washington, 1998.

[DSL96] Chris Dede, Marilyn C Salzman, and R Bowen Loftin. Sciencespace: Virtual realities for learning complex and abstract scientific concepts. In *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pages 246–252. IEEE, 1996.

[DVS03] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 657–662. ACM, 2003.

[DW16] Nick Donaldson and Nick Whiting. Going Off the Rails: The Making of 'Bullet Train', 2016. Accessed: 2017-04-19. URL: `http://www.gdcvault.com/play/1023648/Going-Off-the-Rails-The/`.

[F#05] F# Software Foundation. F#, 2005. Accessed: 2017-05-15. URL: `http://fsharp.org/`.

[FHKH06] Bernd Frohlich, Jan Hochstrate, Alexander Kulik, and Anke Huckauf. On 3d input devices. *IEEE Comput. Graph. Appl.*, 26(2):15–19, March 2006.

[FIB17] FIBA. 2014 Official Basketball Rules, 2017. Accessed: 2017-04-25. URL: `http://www.fiba.com/documents/2015/Official_Basketball_Rules_2014_Y.pdf`.

[Fis18] Ronald Fisher. The Correlation Between Relatives on the Supposition of Mendelian Inheritance. *Philosophical Transactions of the Royal Society of Edinburgh*, 1918.

[Fos15] John Foster. About VR: Designing for Believability, 2015. Accessed: 2017-04-19. URL: `http://www.gdcvault.com/play/1022815/About-VR-Designing-for/`.

[GM04] Enrico Gobbetti and Fabio Marton. Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics*, 28(6):815–826, 2004.

[GM15] Daniel Girardeau-Montaut. CloudCompare, 2015. Accessed: 2017-07-09. URL: `http://www.danielgm.net/cc/`.

[Goo16] Google. Tilt Brush, 2016. Accessed: 2017-04-19. URL: `https://www.tiltbrush.com/`.

[Goo17] Google. Forms, 2017. Accessed: 2017-05-03. URL: `https://www.google.com/forms/about/`.

[GPHC$^+$02] Azucena Garcia-Palacios, H Hoffman, Albert Carlin, TA u Furness, and Cristina Botella. Virtual reality in the treatment of spider phobia: a controlled study. *Behaviour research and therapy*, 40(9):983–993, 2002.

[Haa14] Georg Haaser. FShade, 2014. Accessed: 2017-05-15. URL: `https://github.com/krauthaufen/FShade`.

[Har06] Sandra G Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 904–908. Sage Publications Sage CA: Los Angeles, CA, 2006.

[Hel98] Doug Helmer. Forte Technologies Inc. VFX1 HEADGEAR Virtual Reality System, 1998. Accessed: 2017-04-25. URL: http://www.combatsim.com/archive/htm/htm_arc4/vfx1.htm.

[HHH+16] Patrick Hackett, Curtis Hickman, Tyler Hurd, Alex Schwartz, and Scott Stephan. A Year in Roomscale: Design Lessons from the HTC VIVE and Beyond. VRDC 2016 Talk, 2016. Accessed: 2017-04-19. URL: http://www.gdcvault.com/play/1023661/A-Year-in-Roomscale-Design/.

[HHS+98] Hunter G Hoffman, Ari Hollander, Konrad Schroder, Scott Rousseau, and Tom Furness. Physically touching and tasting virtual objects enhances the realism of virtual experiences. *Virtual Reality*, 3(4):226–234, 1998.

[HJ16] Kai Huebner and Jakob Johansson. Enabling Hands in Virtual Reality, 2016. Accessed: 2017-04-19. URL: http://www.gdcvault.com/play/1023642/Enabling-Hands-in-Virtual/.

[HS88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[HSMT15] Georg Haaser, Harald Steinlechner, Stefan Maierhofer, and Robert F. Tobler. An incremental rendering vm. In *Proceedings of the 7th Conference on High-Performance Graphics*, HPG '15, pages 51–60, New York, NY, USA, 2015. ACM.

[HTC16] HTC. Vive, 2016. Accessed: 2017-04-19. URL: https://www.vive.com/.

[HTC17a] HTC. Vive PRE User Guide, 2017. Accessed: 2017-05-02. URL: http://www.htc.com/managed-assets/shared/desktop/vive/Vive_PRE_User_Guide.pdf.

[HTC17b] HTC. Vive Tracker, 2017. Accessed: 2017-04-19. URL: https://www.vive.com/anz/vive-tracker/.

[I-I16] I-Illusions. Space Pirate Trainer, 2016. Accessed: 2017-04-19. URL: http://www.spacepiratetrainer.com/.

[igr16] igroup. Igroup Presence Questionnaire, 2016. Accessed: 2017-04-19. URL: http://www.igroup.org/pq/ipq/index.php.

[Int12] International Organization for Standardization. ISO/IEC 23271:2012, 2012. Accessed: 2017-05-15. URL: https://www.iso.org/standard/58046.html.

[Iwa99] Hiroo Iwata. Walking about virtual environments on an infinite floor. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 286–293. IEEE, 1999.

[IYT07] Hiroo Iwata, Hiroaki Yano, and Masaki Tomiyoshi. String walker. In *ACM SIGGRAPH 2007 emerging technologies*, page 20. ACM, 2007.

[Joh16] Leif Johnson. Watch How a $25,000 Robot Makes Virtual Reality Way Better, 2016. Accessed: 2017-05-30. URL: `https://motherboard.vice.com/en_us/article/VR-robot-feedback`.

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

[Krü12] Louis Krüger. Konforme Abbildung des Erdellipsoids in der Ebene. *Veröffentlichung des Königlich Preuszischen Geodätischen Instituts*, 1912.

[Log17] Logical Increments. Building a PC for Virtual Reality: Oculus Rift, HTC Vive, and VR Gaming, 2017. Accessed: 2017-04-25. URL: `http://www.logicalincrements.com/articles/vrguide`.

[LS16] Jonathan Lin and Jürgen P Schulze. Towards naturally grabbing and moving objects in vr. *Electronic Imaging*, 2016(4):1–6, 2016.

[Lud15] Joe Ludwig. IVRSystem::TriggerHapticPulse, 2015. Accessed: 2017-05-18. URL: `https://github.com/ValveSoftware/openvr/wiki/IVRSystem::TriggerHapticPulse`.

[M-C16] M-Ch. [C# wrapper] TriggerHapticPulse - type of argument unDurationMicroSec should be ushort not char., 2016. Accessed: 2017-05-18. URL: `https://github.com/ValveSoftware/openvr/issues/123`.

[Man16] Manus-VR. Manus-VR, 2016. Accessed: 2017-05-30. URL: `https://manus-vr.com/`.

[Min98] Mindflux. VFX1 HEADGEAR, 1998. Accessed: 2017-04-25. URL: `http://www.mindflux.com.au/products/iis/vfx1-2.html#specifications`.

[Mos93] Michael Moshell. Three views of virtual reality: virtual environments in the us military. *Computer*, 26(2):81–82, 1993.

[Nia16] Niantic. Pokemon Go, 2016. Accessed: 2017-04-19. URL: `http://www.pokemongo.com/`.

[Nor16a] Colin Northway. Menus Suck. VRDC 2016 Talk, 2016. Accessed: 2017-04-19. URL: `http://www.gdcvault.com/play/1023668/Menus/`.

[Nor16b] Northway Games. Fantastic Contraption, 2016. Accessed: 2017-04-19. URL: `http://fantasticcontraption.com/`.

[NVI16] NVIDIA. VR Funhouse, 2016. Accessed: 2017-04-19. URL: http://www.geforce.com/whats-new/tag/nvidia-vr-funhouse/.

[NVI17] NVIDIA. NVIDIA Quadro GP100, 2017. Accessed: 2017-05-22. URL: http://images.nvidia.com/content/pdf/quadro/data-sheets/302049-NV-DS-Quadro-Pascal-GP100-US-NV-27Feb17-HR.pdf.

[OCD01] Allison M Okamura, Mark R Cutkosky, and Jack T Dennerlein. Reality-based models for vibration feedback in virtual environments. *IEEE/ASME Transactions on Mechatronics*, 6(3):245–252, 2001.

[Ocu14] Oculus VR. Announcing the Oculus Rift Development Kit 2 (DK2), 2014. Accessed: 2017-04-25. URL: https://www3.oculus.com/en-us/blog/announcing-the-oculus-rift-development-kit-2-dk2/.

[Ocu16] Oculus. Rift, 2016. Accessed: 2017-04-19. URL: https://www.oculus.com/rift/.

[Ocu17] Oculus. Privacy Policy, 2017. Accessed: 2017-04-19. URL: https://www.oculus.com/legal/privacy-policy/.

[ODH98] Allison M Okamura, Jack T Dennerlein, and Robert D Howe. Vibration feedback models for virtual environments. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 674–679. IEEE, 1998.

[PBWI96] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM, 1996.

[PH16] Ben Padget and Vincent Hamm. Pillars of Presence: Amplifying VR Immersion. Vision VR/AR Summit 2016 Talk, 2016. Accessed: 2017-04-19. URL: https://www.youtube.com/watch?v=TdJf3X5tBvU.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[Pho16] Phosphor Games Studios. The Brookhaven Experiment, 2016. Accessed: 2017-04-19. URL: http://phosphorgames.com/blog/games/the-brookhaven-experiment/.

[PJW12] Reinhold Preiner, Stefan Jeschke, and Michael Wimmer. Auto splats: Dynamic point cloud visualization on the gpu. In *EGPGV*, pages 139–148, 2012.

[PSL05] Renato Pajarola, Miguel Sainz, and Roberto Lario. Xsplat: External memory multiresolution point visualization. In *Proceedings IASTED Invernational Conference on Visualization, Imaging and Image Processing*, pages 628–633, 2005.

[PUK16] Iana Podkosova, Michael Urbanek, and Hannes Kaufmann. A hybrid sound model for 3d audio games with real walking. In *Proceedings of the 29th International Conference on Computer Animation and Social Agents*, CASA '16, pages 189–192, New York, NY, USA, 2016. ACM.

[PZVBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000.

[Rai16] Bernhard Rainer. SelectionVisualization, 2016. Accessed: 2017-05-15. URL: `https://github.com/JimmyLaessig/SelectionVisualization`).

[Rea17] Real-Time Physics Simulation. Bullet Physics Library, 2017. Accessed: 2017-04-19. URL: `http://bulletphysics.org/`.

[Rei16] Sean Reilly. VR Locomotion Demo, 2016. Accessed: 2017-04-19. URL: `https://github.com/qwv4851/Movement`.

[RKW01] Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. Redirected walking. In *Proceedings of EUROGRAPHICS*, volume 9, pages 105–106. Citeseer, 2001.

[RL00] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352. ACM Press/Addison-Wesley Publishing Co., 2000.

[Rot16] Gerhard Franz Roth. VRVis: Was das neue K1-Zentrum für Visual Computing kann, 2016. Accessed: 2017-05-02. URL: `https://factorynet.at/a/vrvis-was-das-neue-k1-zentrum-fuer-visual-computing-kann`.

[Sam17] Samsung. Samsung SSD 960 EVO NVMe M.2 1TB, 2017. Accessed: 2017-07-08. URL: `http://www.samsung.com/us/computing/memory-storage/solid-state-drives/ssd-960-evo-m-2-1tb-mz-v6e1t0bw/`.

[SBJ+10] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. Estimation of detection thresholds for redirected walking techniques. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):17–27, 2010.

[Sch03] Thomas W. Schubert. The sense of presence in virtual environments: A three-component scale measuring spatial presence, involvement, and realness. *Zeitschrift für Medienpsychologie*, 15(2):69–71, 2003.

[Sch15] Markus Schütz. Potree 1.3, 2015. Accessed: 2017-07-09. URL: `http://www.potree.org/`.

[SCP95] Richard Stoakley, Matthew J Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press/Addison-Wesley Publishing Co., 1995.

[SFL$^+$16] Anthony Steed, Sebastian Frlston, Maria Murcia Lopez, Jason Drummond, Ye Pan, and David Swapp. An 'in the wild'experiment on presence and embodiment using consumer virtual reality equipment. *IEEE transactions on visualization and computer graphics*, 22(4):1406–1414, 2016.

[SGH$^+$12] Peng Song, Wooi Boon Goh, William Hutama, Chi-Wing Fu, and Xiaopei Liu. A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1297–1306, New York, NY, USA, 2012. ACM.

[Sla99] Mel Slater. Measuring presence: A response to the witmer and singer presence questionnaire. *Presence: Teleoperators and Virtual Environments*, 8(5):560–565, 1999.

[Son98] Sony. Sony Announces New Personal LCD MonitorPC Glasstron, 1998. Accessed: 2017-04-25. URL: `https://www.sony.net/SonyInfo/News/Press_Archive/199809/98-101/`.

[Son16] Sony. PlayStation VR FAQ, 2016. Accessed: 2017-04-25. URL: `https://blog.eu.playstation.com/2016/10/03/playstation-vr-the-ultimate-faq/`.

[Son17] Sony. PlayStation VR, 2017. Accessed: 2017-04-19. URL: `https://www.playstation.com/en-us/explore/playstation-vr/`.

[SP11] Claus Scheiblauer and Michael Pregesbauer. Consolidated visualization of enormous 3d scan point clouds with scanopy. In *Proceedings of the 16th International Conference on Cultural Heritage and New Technologies*, pages 242–247, 2011.

[SRGS00] Eva-Lotta Sallnäs, Kirsten Rassmus-Gröhn, and Calle Sjöström. Supporting presence in collaborative environments by haptic force feedback. *ACM Trans. Comput.-Hum. Interact.*, 7(4):461–476, December 2000.

[SRH05] Frank Steinicke, Timo Ropinski, and Klaus Hinrichs. Vr and laser-based interaction in virtual environments using a dual-purpose interaction metaphor. In *IEEE VR 2005 Workshop Proceedings on New Directions in 3D User Interfaces*, pages 61–64, 2005.

[SUC95] Mel Slater, Martin Usoh, and Yiorgos Chrysanthou. The influence of dynamic shadows on presence in immersive virtual environments. In *Selected Papers of the Eurographics Workshops on Virtual Environments '95*, VE '95, pages 8–21, London, UK, UK, 1995. Springer-Verlag.

[Sut65] Ivan E Sutherland. The ultimate display. *Multimedia: From Wagner to virtual reality*, 1965.

[Sut68] Ivan E Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM, 1968.

[SVdM99] Martijn J Schuemie and CAPG Van der Mast. Presence: Interacting in virtual reality. In *Proceedings, Twente Workshop on Language Technology*, volume 15, pages 213–217, 1999.

[SW11] Claus Scheiblauer and Michael Wimmer. Out-of-core selection and editing of huge point clouds. *Computers & Graphics*, 35(2):342–351, April 2011.

[SW15] Markus Schütz and Michael Wimmer. High-quality point based rendering using fast single pass interpolation. In *Proceedings of Digital Heritage 2015 Short Papers*, pages 369–372, September 2015.

[SZ94] David J Sturman and David Zeltzer. A survey of glove-based input. *IEEE Computer graphics and Applications*, 14(1):30–39, 1994.

[TDS99] James N Templeman, Patricia S Denbrook, and Linda E Sibert. Virtual locomotion: Walking in place through virtual environments. *Presence: teleoperators and virtual environments*, 8(6):598–617, 1999.

[Tek16] Tekton Games. WalkAbout, 2016. Accessed: 2017-04-19. URL: `http://tektongames.com/walkabout-presskit/`.

[UAW$^+$99] Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. Walking> walking-in-place> flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[UQO13] UQO Cyberpsychology Lab. Presence Questionnaire, 2013. Accessed: 2017-04-19. URL: `http://w3.uqo.ca/cyberpsy/docs/qaires/pres/PQ_va.pdf`.

[Val15a] Valve. IVRSystem::GetHiddenAreaMesh, 2015. Accessed: 2017-05-18. URL: `https://github.com/ValveSoftware/openvr/wiki/IVRSystem::GetHiddenAreaMesh`.

[Val15b] Valve. OpenVR, 2015. Accessed: 2017-05-15. URL: `https://github.com/ValveSoftware/openvr`.

[Val16] Valve. The Lab, 2016. Accessed: 2017-04-19. URL: `http://store.steampowered.com/app/450390/`.

[Val17a] Valve. openvr_api.cs, 2017. Accessed: 2017-05-18. URL: `https://github.com/ValveSoftware/openvr/blob/master/headers/openvr_api.cs`.

[Val17b] Valve. Privacy Agreement, 2017. Accessed: 2017-04-19. URL: `http://store.steampowered.com/privacy_agreement/`.

[VK15] Khrystyna Vasylevska and Hannes Kaufmann. Influence of path complexity on spatial overlap perception in virtual environments. In *Proceedings of the 25th International Conference on Artificial Reality and Telexistence and 20th Eurographics Symposium on Virtual Environments*, pages 159–166. Eurographics Association, 2015.

[VKBS13] Khrystyna Vasylevska, Hannes Kaufmann, Mark Bolas, and Evan A Suma. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pages 39–42. IEEE, 2013.

[VKSR00] V Vuskovic, M Kauer, Gábor Székely, and M Reidy. Realistic force feedback for virtual reality based diagnostic surgery simulators. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1592–1598. IEEE, 2000.

[VRV17] VRVis. Aardvark - An Advanced Rapid Development Visualization And Rendering Kernel, 2017. Accessed: 2017-04-19. URL: `https://www.vrvis.at/research/projects/aardvark/`.

[WBB$^+$07] Michael Wand, Alexander Berner, Martin Bokeloh, Arno Fleck, Mark Hoffmann, Philipp Jenke, Benjamin Maier, Dirk Staneker, and Andreas Schilling. Interactive editing of large point clouds. In *SPBG*, pages 37–45, 2007.

[WJ88] Colin Ware and Danny R Jessome. Using the bat: A six-dimensional mouse for object placement. *IEEE Computer Graphics and Applications*, 8(6):65–70, 1988.

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

[WS06] Michael Wimmer and Claus Scheiblauer. Instant points: Fast rendering of unprocessed point clouds. In *SPBG*, pages 129–136, 2006.

[Xin16] XinReality. Locomotion Methods, 2016. Accessed: 2017-04-19. URL: `https://xinreality.com/wiki/Locomotion_Methods`.

[ZPVBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378. ACM, 2001.