TU WIEN Informatics

Evaluation of Immersion in Externally Directed VR

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Silvester Farda, BSc
Matrikelnummer 00727611

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Dr. Horst Eidenberger, Ao.Univ.Prof.

Wien, 7. Juli 2021

Silvester Farda	Horst Eidenberger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Evaluation of Immersion in Externally Directed VR

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Silvester Farda, BSc
Registration Number 00727611

to the Faculty of Informatics

at the TU Wien

Advisor: Dr. Horst Eidenberger, Ao.Univ.Prof.

Vienna, 7th July, 2021

Silvester Farda	Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Silvester Farda, BSc

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, 7. Juli 2021

Silvester Farda

Acknowledgements

First, I would like to thank my supervisor, Prof. Dr. Host Eidenberger, who guided me through the entire process of creating this thesis. He showed an exceptional degree of patience and provided me not only with advice, but also with a lot of valuable feedback and input along the way.

I would also like to thank a number of people, without whom my study and thus ultimately this thesis, would not have been possible. First and foremost, I'd like to thank my whole family, including my parents, sister, grandmother, aunt and other relatives, for providing me with emotional and motivational support and were always there when I needed them. This is especially true for my mother, Margarete Farda, who did not only always encourage me to pursue my interests, but also helped me in any way she could and provided me with financial as well as social support. I would also like to mention my late grandmother, Margarethe Haas, who continuously supported me in any way possible. I would also like to include my father, Christoph Farda, who, sadly, already passed in 2007 but was a big inspiration and ultimately became an important motivator for me to study in the first place.

Also, a big thank you to my girlfriend, Karin Maier, who not only supported me morally and emotionally as well as in many other regards, but also stoically listened to me even during the darker times and stressful periods. She continuously provided me with valuable input regarding structural and linguistic questions concerning my thesis and also graciously helped me with hours of proofreading.

I would also like to thank all participants of my user study and the whole *Virtual Jump* and *Jump Cube* team, that not only provided the foundation for the simulation used in this project, but also helped me with a lot of personal tips and input along the way. Many of these team members later became colleges and some even friends, who gave me a lot of insight into their own thesis writing process.

There are, of course, also countless other people who supported me in one way or another and played an important role in the process of this thesis, my study and my personal development and life in general, to whom I want to express my sincerest gratitude.

Thank you all, from the bottom of my heart!

Kurzfassung

In dieser Diplomarbeit wird der Einfluss von Effekten, die von Personen außerhalb einer Virtual Reality Simulation ausgelöst werden, auf Immersion erforscht. Die Basis für die VR Simulation bildet das *Virtual Jump* Projekt [EM15], dessen System nicht nur als physische Grundlage für ein immersives VR Erlebnis dient, sondern auch als Software-Basis für die Grundlage der VR Simulation herangezogen wird. Zusätzlich zu dieser, im Rahmen meiner Diplomarbeit modifizierten und erweiterten Simulation, wurde auch eine zweite, vollständig eigenständige Applikation entworfen, die als Interaktionskonsole zum Einsatz kommt. Diese Anwendung kann von Personen außerhalb der Simulation bedient werden und erlaubt es, entweder mittels Mausklick oder via Handbewegungen die von einem *Leap Motion Controller* erfasst werden, Effekte innerhalb der virtuellen Umgebung auszulösen. Beide Applikationen laufen auf getrennten Geräten und kommunizieren mittels eines eigenen, Nachrichten-basierten Kommunikationsansatz, der Auslöse-Impulse über eine herkömmliche Ethernet Verbindung sendet.

Nach erfolgreicher Implementierung des Prototyps wurde eine User-Study durchgeführt. Aufgrund der *COVID-19* Pandemie, mussten einige Einschränkungen bei den Features der Applikationen in Kauf genommen werden, sowie der Testablauf modifiziert und die Gruppengröße reduziert werden.

Obwohl meine Daten aufgrund der kleinen Testgruppe nicht repräsentativ sind, gaben meine Teilnehmer an, dass extern ausgelöste Effekte ihre Immersion sehr wohl beeinflusst haben. Interessanterweise, scheint dieser Einfluss ein positiver statt, wie ursprünglich vermutet, ein negativer zu sein. Parallel dazu, beleuchtet meine User Study auch die Interaction der „Jumper“ und „Controller“ – also der beiden Rollen – und die Auswirkungen des systeminhärenten Macht- bzw. Kontrollgefälles zwischen den beiden Teilnehmern währen der Testsprünge.

Zusammenfassend lässt sich sagen, dass, obwohl die hier gesammelten Daten nicht repräsentativ sind, die von externen Personen ausgelösten Effekte, einen überwiegend positiven Einfluss hatten. Außerdem hat sich gezeigt, dass die Manifestation und das Ausmaß von Effekten durch das Kontroll- und Machtgefälle zwischen den beiden Rollen sehr individuell und im hohen Maß von den teilnehmenden Personen abhängig ist. Abschließend lässt sich sagen, dass beide Rollen Großteils positiv aufgenommen wurden und meine Kandidaten durchaus Spaß hatten, obwohl die Rolle des „Controllers“ wohl die beliebtere war.

Abstract

In this thesis, the impact of humanly controlled, externally triggered effects on immersion inside a virtual reality simulation is researched. The VR simulation is based on the *Virtual Jump* project [EM15] which not only serves as physical foundation for creating an immersive VR experience, but also provides the original VR content that this thesis builds upon. The software for the external interaction console is created as a completely separate application from the ground up. By using one of the interaction console applications the user can trigger actions within the VR simulation, either by using traditional input devices like a mouse or by performing hand movements that are captured with the *Leap Motion Controller*. Both the original *Virtual Jump* content, as well as the newly created interaction console application are based on the *Unity* game engine and can be run as stand-alone applications. The interaction between these two applications is realized with a custom, message-based communication approach, that transfer event triggers over a traditional Ethernet connection.

With a prototype implementation of the remote interaction console, as well as with a modified version of the original *Virtual Jump* content, a small-scale user study was carried out. In order to allow the necessary test jumps to be performed with a maximum of safety in mind, even in light of the *COVID-19* pandemic, some concessions to the software implementation as well as a variety of changes to the testing procedure and number of test subjects had to be made.

The majority of my participants, although not being representative due to sample size, claimed that externally triggered effects did in fact have an influence on immersion when being in a VR simulation. Surprisingly, these effects seem to be positive rather than negative. The user study also covers the inter-person interaction between jumpers and controllers, as well as power dynamics that could be observed during the test jumps.

To summarize, my user study, although not representative, showed that the degree of immersion of my test subjects in VR was mostly affected positively by humanly-controlled, externally triggered effects. It also revealed that the power dynamics between participants within the two roles highly depended on the individual users. Nevertheless, among my participants, both roles seem to have been enjoyed during the test jumps, with the role of the controller seeming to be the more desirable one.

Contents

CHAPTER 1

Introduction

1.1 Aim

The aim of this diploma thesis is to explore the influence of externally triggered effects on immersion inside a Virtual Reality (VR) simulation. For evaluating this, a specialized test setup was created. The VR simulation is based on the *Virtual Jump Simulator* [EM15] originally created in 2015, which allows a person to experience a virtual parachute jump over the city of Vienna. The original content created for this simulation was extended by including additional effects and implementing a custom trigger logic, which receives external commands from another program. This "Interaction Console" is a completely separate application that can be run on another computer that just has to be connected to the VR simulation host via Ethernet. The external user can interact with the console application by using a traditional button interface, either on a desktop, a smartphone/tablet or by pure hand movement that is captured with the *LEAP Motion* system. The commands triggered on that console then trigger effects such as explosions or scents inside the VR simulation. In this test setup, two people are required: one inside the VR simulation and one on the interaction console. The one inside VR can experience the original *Jump Cube* content as well as any potentially triggered effects. The person operating the external console can trigger effects within the simulation of the other one and is able to instantly see the reactions of the other "player".

With the setup described above, this work tries to answer the following questions:

1. Are there any effects of externally triggered stimuli on immersion in a VR simulation?
2. Do the triggered effects increase or decrease immersion?
3. Are these effects perceived substantially different from pre-programmed ones?
4. Do the externally triggered effects interfere with the content itself?

5. How do console operators respond when directly seeing the reactions of the person in VR?
6. Is there any kind of power dynamic forming between these unequal roles?
7. Which interface for the interaction console is favored?
8. Is a setup like this entertaining enough to display it at an exhibition or to have any commercial potential?

Although not all of these questions might be answerable conclusively in the scope of this work, exploring and evaluating user's reactions might at least provide some answers and more insight for possible larger scale studies in this field.

1.2 Motivation

Over the last decade, popularity of Virtual Reality has increased manyfold. One majorly contributing factor for this, is the increased availability and affordability of well-supported Head Mounted Displays (HDMs). When I first used a VR headset, I was instantly fascinated by the thought of stepping into another, completely virtual world. Over time I tried out a variety of headsets and devices, as well as VR contents on different platforms. Among my favorite experiences was the parachute jump simulator of the *Virtual Jump* project. One of the key benefits of this system is, that it not only relies on convincing images provided via high-tech VR goggles, but also allows the user to physically experience the jump as well. With this combination the immersion was tremendously improved and well-liked by many users. I was able to experience this first-hand, when operating the *Jump Cube* simulator at different exhibitions and congresses. On most occasions, people queued up quickly to take the simulated jump and even when we moved to registering people for specific time slots, the *Jump Cube* was almost always fully booked. At some point I started to wonder when people queued up and waited for participating in an immersive VR experience, why has VR not been more widely adopted in the mainstream market yet?

Aside from the most obvious reasons being high cost and the required space for a basic VR setup, another possible reason might be that today's VR is mainly designed as a single-player experience. Giving the huge popularity of multiplayer games, this might very well be a deal-breaker for a lot of players – especially casual ones. The reason for the sparse availability of VR multiplayer content might be, again costs and needed space aside, the challenges and problems that arise when trying to operate more than one VR headset side by side. Although it is possible with some systems, the VR experience is often greatly diminished. In the game *Star Trek™: Bridge Crew* by *Ubisoft*, which was released for Sony's *PlayStation VR* platform, two players can use their VR headsets respectively, but the simulation is limited to sitting only. The most likely reason I can think of, is the need for avoiding physical interference or collision when both players are moving simultaneously. After doing some research, I found that *Sony* itself released a

game called *The Playroom VR* for its platform, where they took a completely different approach to enable multiplayer gaming: only one player is wearing an HMD while the others are using more conventional input methods to interact with the game.

Since one of the main benefits of VR – at least for me – was the immersion or more precisely to completely step into a different world when putting on a VR headset, I immediately asked myself how this "mixed" approach might affect this crucial part of every VR experience. Placing effects or olfactory stimuli in a simulation is nothing out of the ordinary, but there is a fundamental difference between programmatically scripted effects and ones triggered by another human being. Manually triggered actions usually don't have any inherent pattern or logic to them and might therefore be perceived differently than scripted ones. This might very well make a notable difference to immersion and the overall VR experience as a whole. The question I want to answer is, how and to what extent external, human interaction within a running VR simulation influences immersion for the player. Another interesting point of observation, besides the influence on immersion, is the bidirectional human-human interaction which might even develop some kind of power- or group-dynamic during the simulation. I decided to test this by combining the well-established *Jump Cube* simulation with an external console application to implement a test setup for answering the questions at hand.

1.3 Methodology

My first step was to do research and look for scientific publications regarding the influence of external stimuli on immersion in a VR environment. Interestingly, this topic does seem to be hardly covered by scientific publications. There are some publications mentioning it briefly, but it does not seem to have been fully explored anywhere.

The literary research for publications about mixed VR in general did yield better results. There are a few papers covering the challenges of such scenarios, both from a scientific, as well as a commercial background. These publications cover a broad spectrum of applications ranging from asymmetric collaboration in VR [TNNL19] all the way to shared virtual reality gaming [LM16]. Although none these resources could answer my questions directly, they provide a solid foundation for my work.

Next, I designed a system which allowed me to test the impact of externally triggered, humanly-controlled effects on immersion. The *Virtual Jump* simulation provided a good basis for my work, both software wise and as a physical testing environment, since it has been tested extensively and proved to work reasonably reliable over the years. There was already scientific literature published using this system and for this specific combination of the *Jump Cube* and the *Virtual Jump* content, there is even already immersion-related evaluation data by my supervisor available to work with, should need be. The original *Virtual Jump* content was created using the *Unity Engine*. I am familiar with this commonly used framework since I already worked with it on many occasions during several courses within my master's program. I even created a simulation content for the *Jump Cube* for a previous project. For simplicity's sake and to achieve more

reproduceable results, I opted for using the original *Virtual Jump* content and simply adapt it to my needs for this thesis. The necessary changes and additions to the existing simulations did however, hold some challenges for me, which I had to master.

Beside the actual VR simulation, I also needed to implement a way of interacting with the simulation from the outside. For this, I created another standalone application using *Unity*, that serves as an interaction console. This console provides the user with two different types of interfaces to choose from: a traditional button-based UI, which can either be used on a desktop, as well as on any *Android* [and] device, and a hand-movement based system, which utilizes the *Leap Motion Controller* [Ult]. With either of these interfaces, a user can trigger effects inside the VR simulation running on another computer. The communication between the console and the VR simulation host was implemented via Ethernet and entirely message-based in order to avoid unnecessary overhead and make it as efficiently as possible.

After implementing all necessary changes to the *Virtual Jump* simulation and after the interaction console was created, I performed a small-scale user study. My goal was to evaluate the experiences of people in VR ("Jumpers") and at the same time of the people operating the external interaction console ("Controllers"), so I had to create two different survey forms for these groups, since they would need entirely different questions. Due to unexpected circumstances in 2020, the user-study had to be carried out as fast as possible since the *COVID-19* pandemic only allowed me to perform tests in a rapidly closing time window. Additionally, the user study was reduced to a minimum of participants to keep health risks as low as possible. I opted for working with only two subjects at a time, where each of them took one role after the other. I also tried to mainly work with subjects, that had already taken part in the *Virtual Jump* simulation, which allowed me to keep testing time short since this eliminated the need for a first "baseline"-run before doing the simulation (again) with external triggers. This way I was not only able to evaluate the effects on immersion most effectively and securely, but also observe possible group-dynamic behavior while still keeping the number of people to work with at a minimum.

After the user study was carried out, I aggregated the gathered data and drew my conclusions. The evaluation of the results led me to a lot of expected answers, but there were also a few key points that I would never have expected to see when I started working on this topic.

1.4 Overview of the Thesis

This section provides a short overview of all chapters contained in this thesis and their respective content.

Chapter 2 gives some background on virtual reality in general as well as on VR- and mixed-VR gaming. The historical development will be discussed in short and a broader overview of the current state-of-the-art will be given.

Chapter 3 carries the research questions, referenced by their number in the corresponding list, over to concrete requirements to soft- and hardware implementations for my applications and test setup. Then, the design and implementation process that took place and lead to the final test setup will be presented. A number of details are provided relating the decisions made which lead to using specific software versions and frameworks, followed by a description of the physical test setup. In the end, there will be a short mentioning which concessions had to be made to expedite the time schedule and perform all necessary tests for the user study as quickly as possible, in order to avoid delays caused by the *COVID-19* pandemic.

Chapter 4 contains all information about the small-scale user study I conducted. The exact process of how test jumps were performed will be described, as well as all hygienic measures I had to take to minimize health risks in light of the *COVID-19* pandemic as much as possible. This chapter also includes my personal observations during testing and the first "lessons learned" that surfaced during the evaluation itself.

Chapter 5 showcases the results of the user study. The gathered data will be evaluated and interpreted as well as some possible conclusions from my work will be laid out. Finally, there will be a short summary and conclusion of my work, as well as an outlook on possible implications of my results.

CHAPTER 2

Background

2.1 History of Modern VR

Immersive experiences fascinated people since the beginning of photography. Long before the invention of computer graphics, as we know it today, people enjoyed stereoscopic still images by using relatively simple viewing devices. The basic principle from back then is still the same today: each eye is presented with a slightly different image. The images are usually either taken one after the other by moving the camera to a slightly different position in between shots, or both pictures are captured simultaneously by using two cameras at once. This results in two photographs, depicting the same scene, just from a slightly shifted position, thus mimicking the slightly shifted position of the human eye.

This technique was initially invented by Sir Charles Wheatstone in 1838 who designed a stationary mirror stereoscope [Whe38]. Sir David Brewster improved the device by using lenses instead of mirrors, which allowed him to rearrange the setup and create his lenticular stereoscope which was notably smaller and portable [Bre56]. As can be seen in Figure 2.1, this device already looked very familiar to simple Head Mounted Displays, or HMD for short, like *Google Cardboard* [goo].

Viewing images this way never really took off, but never really disappeared either. Probably the most popular and most famous stereoscope, which is how these devices are called, is the famous *View-Master* created by William Gruber [Gru14]. It was introduced in 1939 and featured a cardboard wheel with two translucent images on opposite sides of the wheel [Sel17]. When the viewer looked through the stereoscope, each eye was presented one of the two images, thereby creating a stereoscopic impression.

Although this technique is rather straight-forward for still images, realizing it with video is a much more complex topic. But with moving pictures being displayed in a stereoscopic way, immersion was much greater than it was the case with still images. In 1956, Morton Heilig created the famous *Sensorama* motion picture system [Hei62], which was ahead

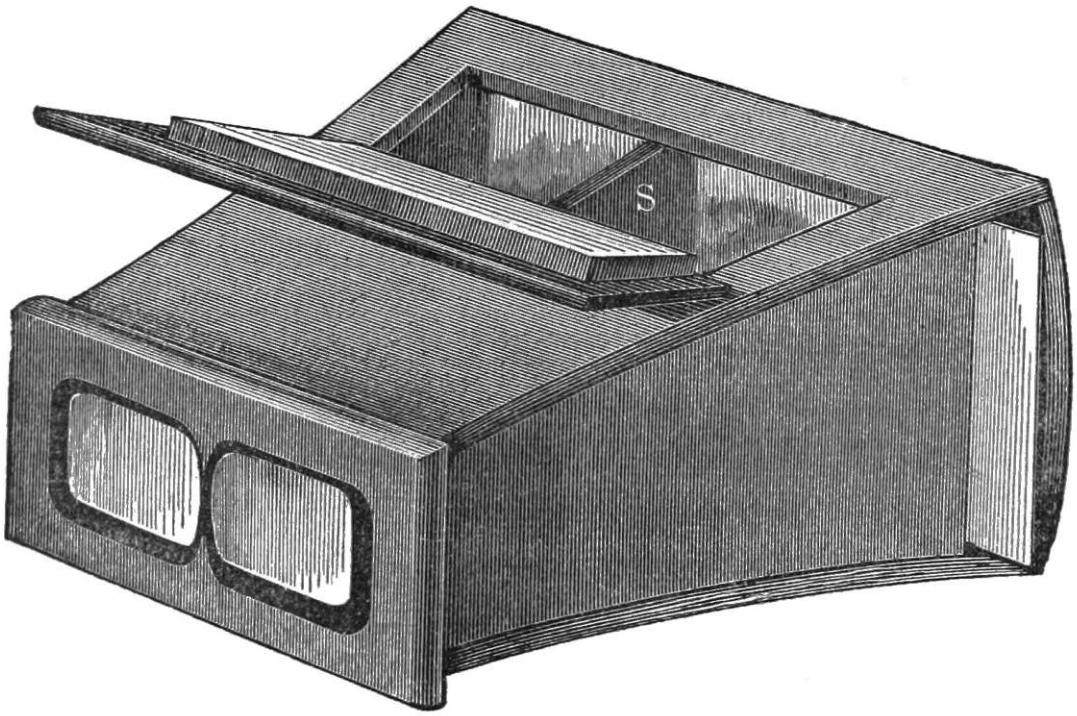


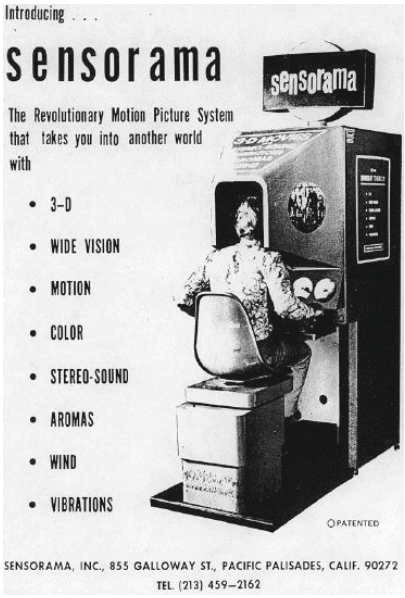
Figure 2.1: Illustration of a Brewster-type stereoscope. Source: [oC82]

of its time in many ways. This was a large, stand-alone device with a chair built into it, with which the user could experience very convincing immersion at that time. It used stereoscopic video images paired with stereo sound, dispensing pre-defined scents, wind effects and a vibrating chair. Morton Heilig was a motion-picture camera man by trade, so he produced 6 short films to be viewed and experienced through his device [AAM+13]. In 1960 Morton Heilig also filed a patent for the first Head Mounted Display called *Telesphere Mask* [Hei60], which was still only playing back pre-recorded content.

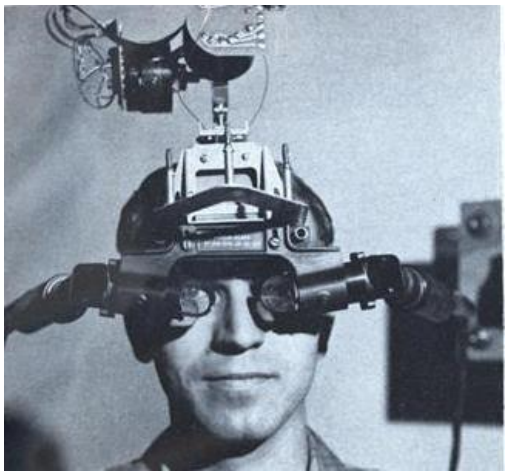
In 1965 Ivan Sutherland formulated his idea of *The Ultimate Display* [Sut65] which should, by definition, be able to show a world inside a computer, that was almost indistinguishable from the real world. In 1968 he created a head-mounted three-dimensional display, which allowed the user to look into a computer-generated world, by rendering individual images for both eyes separately [Sut68]. This HMD was actually partially see-through and already allowed the wearer some basic interaction with his or her virtual environment. The position of the headset was tracked via a mechanical tracking system nicknamed "The Sword of Damocles" [VK07], which mechanically connected the headset directly to the ceiling in order to support its weight and track its movement [Sut68]. Although this unit was only ever built and used in a lab, with this device Ivan Sutherland is commonly known as the inventor of the first VR headset as we would define it today [Bar19].

1979 the company McDonnell-Douglas first integrated an HMD into their *VITAL* helmet

for military use [SJH80]. With the integrated head tracking, this device presented jet pilots with primitive computer-generated images that changed according to the movements of their heads [Bar19]. According to some sources, this might very well have been one of the first applications of a computer-based VR HMD outside a science lab [Soc17].



(a) *Sensorama* device.



(b) Sutherland's VR headset.

Figure 2.2: Two early systems that were able to provide the user with an immersive experience. Source: [ADI17]

The term "Virtual Reality" made its first appearance around the year 1987, in connection with Jaron Lanier and his company *VPL Research* [Soc17] [Bar19]. This was the first company to sell VR goggles and data gloves, which made a public appearance in the Hollywood blockbuster *The Lawnmower Man* from 1992 [law92].

In 1989, *NASA* contracted the company *Crystal River Engineering Inc.*, newly founded by Scott Foster, to create a virtual audio system to be used with their VR based training program for astronauts [Bar19]. The result of this process was the first real-time binaural 3D audio processing system [WSFF90]. The underlying principle is still widely used today and an essential part of many immersive VR experiences.

In the early 1990s, a big VR hype began to manifest itself. In 1991, the *Virtual Group* launched their Virtuality system [vir20b], which was a line of real-time VR arcade machines. Around the same time, the video game company *Sega* started developing a consumer HMD for home use with their popular *Genesis* console. Although the release of *Sega VR* was originally planned for 1993, the device never made it to the market. There had already been difficulties during its development and the first test groups reported cases of headaches and children feeling sick [Wil19]. To avoid a possible PR nightmare, *Sega* decided to cancel the release of the device altogether. Curiously, when asked for the

reason for this, *Sega* allegedly stated that they were concerned about their players' safety [Wil19]. They claimed that their VR system was so realistic, that they were worried that people might hurt themselves when unconsciously physically moving while using the headset [Wil19].

In 1995, *Nintendo* released their take on VR that was meant to bring immersive gaming to the home user: their *Virtual Boy* [vir20a]. In contrast to Sega's headset, which had to be connected to their *Genesis* gaming console to work, the *Nintendo Virtual Boy* was a standalone device. It was also not intended to be attached to the player's head, but had to be placed on a table, mounted on the accompanying stand. The player then had to put his head onto the device which made the *Virtual Boy* rather a tabletop device rather than an actual HMD.

(a) *Sega VR* headset. Source: [Wil19] (b) *Nintendo Virtual Boy*. Source: [vir20a]

Figure 2.3: Sega's and Nintendo's attempts for VR devices in the 1990ies.

Another big difference of Nintendo's device was the display technology used. Since LCD displays only provided low resolutions and CRTs where too bulky and consumed too much power for a device like this, they opted to use a completely new technique. The *Virtual Boy* used a single row of red LEDs which, by means of using an oscillating mirror, where able to "print" an image directly onto the player's retina [Edw15]. The resulting image had a high resolution and was very sharp, but only monochromatic since it was produced by only red LEDs. Another issue with this technique was, that the two "displays" had to be precisely aligned or could otherwise cause eye problems when used by someone with a not fully-developed optical system [Edw15]. Although *Nintendo* tried to compensate for this by using a sturdy and rigid housing for the *Virtual Boy*, they decided to also display a warning that suggested taking a break, when the device was used for more than 15 minutes at a time. An additional problem was the table-mounted nature of the device which made it allegedly very uncomfortable to use [vir20a]. In the end, the *Virtual Boy*

proved to be a commercial failure for *Nintendo* and was discontinued just a year after its release [vir20a].

What happened to the *Sega VR* and the *Virtual Boy* can easily be generalized and represented the fundamental shortcomings of VR at that time: Virtual Reality was extremely hyped in the mid-1990s and customers' expectations were enormous. VR systems had some inherent shortcomings, many of which still are an issue today. Ultimately, technology at that time was not able to deliver what marketing promised. Display technology was not nearly as evolved as it is now and the images created by computer graphics where nowhere near what one would consider to be photo-realistic. Although an immersive experience consists of more than just images alone, they nevertheless are an essential part of it; especially when customers only get to try a device for a short amount of time e.g. at a retail shop or an exhibition.

After a variety of unsuccessful product launches in the consumer segment and a lack of major breakthrough developments on the horizon, it got quiet in the field of Virtual Reality for a few years. Then, in 2012, the *Kickstarter* [kic] campaign for the *Oculus Rift* prototype [Tea16] was started by Palmer Lucky [Bar19]. This prototype was capable of displaying high-quality computer-generated images which allowed for a previously unmatched degree of visual immersion, while still using relatively cheap hardware for the headset. The first version of the *Oculus Rift Developer Kit* was offered for a price of only $300 on *Kickstarter*, which was just a fraction of other headsets available on the market at this time, while still being technically superior [Tea16]. The original *Kickstarter* campaign aimed for a funding of $250.000 but in the end reached just over $2.4 million - over 9 times the amount needed [Tea16].

The *Oculus Rift* represented a completely new class of HMDs with a variety of key advantages over its predecessors. First of all, the development and realism of 3D graphics and especially real-time graphics, had come a long way since the last take on VR was made. The *Rift* was connected to a highly powerful gaming PC which needed to supply the internal displays with images. The built-in single screen, that displayed the image for the left and right eye side by side, was capable of a resolution of 1280x800 pixels (resulting in 640x800 per eye) [DDAM14]. Since the actual surface area was rather small, this display had a relatively high pixel density and was not unlike the screens commonly used in smartphones. These displays were already produced in high quantity and widely available for moderate costs. Another key advantage was the ability to use relatively cheap lenses in the headset which was made possible by modern rendering pipelines in computer graphics.

The biggest drawbacks of cheap lenses are the introduction of geometric, as well as chromatic distortions. An illustration of this effect can be seen in Figure 2.4. Thanks to the computational power available on high-end graphics cards and the versatility of modern graphic rendering pipelines, these effects can effectively be mitigated by counteracting the distortion for each color separately, by using simple shaders to compensate for that during the rendering process [var15] [WLL+11].

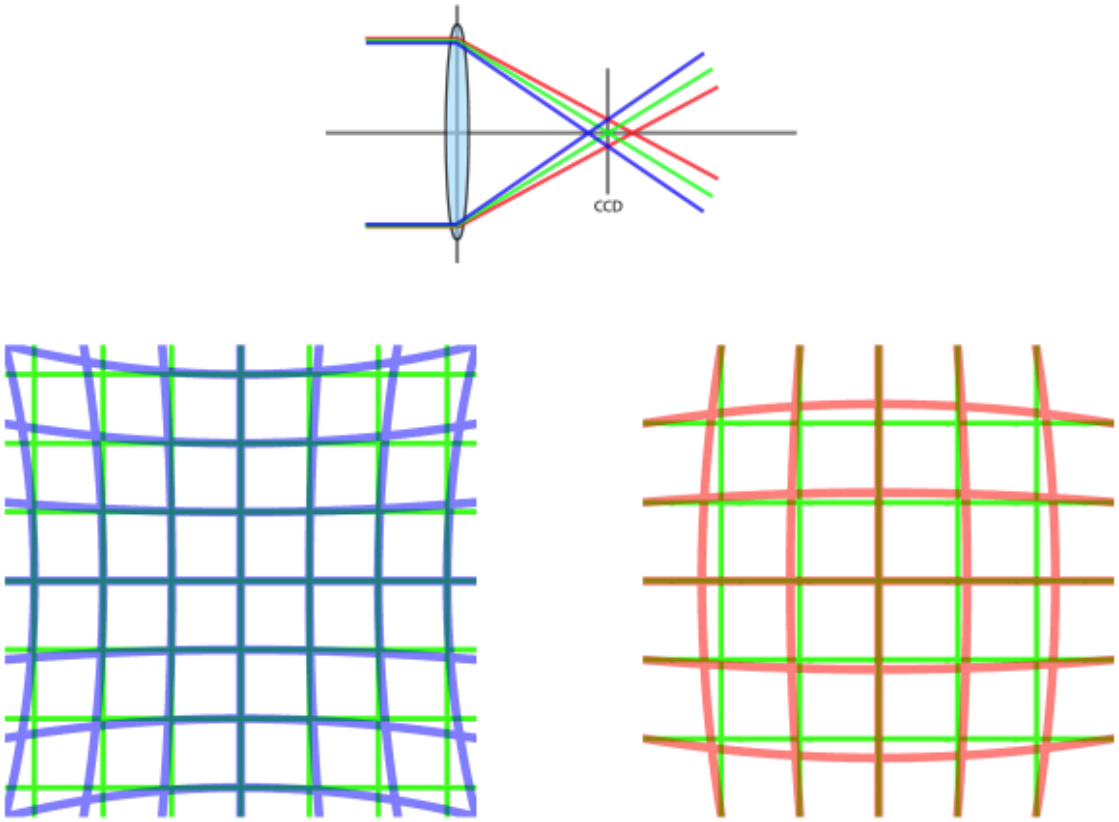


Figure 2.4: Chromatic aberration for different wavelengths: barrel distortion of the red channel, pincushion distortion of the blue channel, both relative to the green channel. Source: [WLL+11]

While modern VR was limited to the PC realm for a while, in 2016, *Sony* released its *PlayStation VR* system [pla20]. This VR headset was connected to the *PlayStation 4* or *PlayStation 4 Pro* console and relied on the *PlayStation Camera* for tracking [Bak16]. This system was basically a turn-key solution which did not require any prior knowledge on tracking technology or VR in general. The *PlayStation VR* headset cost about the same as the original *Oculus Rift Developer Kit*, but if bought together with a *PlayStation 4* console and camera, it was cheaper than buying the *Rift* and a high-end gaming PC. Buying a new PC was strictly speaking not necessary, but since only high-end graphics hardware was able to deliver VR content at sufficiently high frame rates, at least an expensive upgrade of the existing setup was almost always needed.

One disadvantage of going the console-route was, that possible applications were strictly limited to gaming. While PC-based headsets had a broad spectrum of applications in various fields, the *PlayStation VR* was solely marketed towards gaming. Due to its simplicity and affordability, while still delivering a good visual performance and a convincing feeling of immersion, *PlayStation VR* was adopted very well by the gaming

community [Gra20].

All these new devices have one thing on common: they use small, high-density displays which are very similar to those found in modern smartphones. Naturally, the idea of using smartphones themselves as a display and just build the viewing assembly around it was not too far-fetched. Smartphones were already commonly used and they all could basically play back stereoscopic images and videos. In most cases basic 3D rendering tasks were also possible, depending on the processing power of the specific device. Naturally, the VR experience highly depended on the screen resolution and performance of the smartphone used and nowhere near matched the quality of 3D images generated on a PC's high-end dedicated graphics hardware. Smartphone based HMDs are also usually limited to only 3 degrees of freedom (3DOF) [Bar18], meaning only orientation but not movement is tracked. This limits the degree of immersion, but nevertheless smartphone-based HMDs offer a cheap and easy way to take a peek into the world of VR and experience some kind of immersion first hand.

The pioneering solution was Google's *Cardboard* [goo] introduced in 2014, which was very affordable and can still be used today [VR 18]. Since the project was and still is very popular, *Google* decided to even release it as open source in 2019 [car19]. In its original form, it literally was just a piece of cardboard accompanied by two raisin lenses. This gadget was folded up around the smartphone and positioned the lenses right in front of the display, ready to strap it to your head like any other HMD. From the outside, this device closely resembled early portable lenticular stereoscopes like the device created by Sir David Brewster in 1870 [Bre56] or the commercially sold *View Master* from 1939 [Gru14].

The most amazing thing about this approach was, that the entry-barrier for experiencing VR at least in a simple form, was as low as never before: a basic pre-assembled cardboard unit cost around $15 [car]. In combination with any kind of smartphone, that most people owned by then anyway, everybody could get a taste of VR. After the release of the original *Cardboard* assembly, a variety of manufactures designed more durable polycarbonate counterparts which were based on the same concept. Many of them featured higher quality lenses and sometimes even looked like more expensive, professional headsets [vrh20]. Of course, ultimately the abilities of all these devices were still limited to the capabilities of the smartphone used, which served as display, processing- and rendering unit at once.

Although resolution and pixel density of displays have increased in recent years, VR headsets used today basically still work in exactly the same way. Despite even modern VR headsets having their shortcomings, like the infamous screen door effect [NSMS20] and tracking or latency issues, VR has also begun to slowly establish in today's gaming scene. It is still not very widely adopted, but over previous years many interesting games were released and the first Triple-A titles started to show up. Although the improved realism of games and thereby also an increasing degree of immersion are a compelling proposal for entering VR, there are still two mayor barriers to fully-featured VR:

For one, high-end, PC-connected VR headsets are already very expensive by themselves and do still require a highly capable gaming system in order to supply the headset with realistic 3D images, with a consistently high frame rate. The cost factor is less of an issue with console-based systems like Sony's *PlayStation VR* setup, but even these devices require a substantial investment. As development progresses over time, hardware capabilities increase and when aiming for a constant level of performance, prices drop further every year. While the hardware required to run basic VR-environments was considered absolutely high-end a few years ago, modern mid-range hardware can nowadays already run many popular VR simulations.

The other big entry barrier for fully-featured VR still is the need for a rather complex and spatially demanding setup for accurate tracking, like HTC's *Lighthouse* system used for the *Vive* series [NLL17] [Lan16a], and the necessity of a wired connection between the headset and a gaming PC or console. While there are wireless transmission solutions for various HMDs available [Bur19], they all have shortcomings of their own: they add weight to the headset, often have a noticeable impact on latency and sometimes even connection problems occur.

These two key barriers for VR gaming are not an issue with smartphone-based headsets, but there are other disadvantages with these devices: there is no tracking of translational movement (only 3DOF) and even orientation tracking can sometimes fail, since it is solely based on the smartphone's basic onboard sensors. In addition to that, the graphics performance is, of course, fairly limited. This is especially true, since VR headsets often use higher display refresh rates to fluidly provide the wearer with images allowing quick head movements and avoid stuttering.

To possibly combine the advantages of both systems, another class of VR headsets has emerged in recent years. In addition to the very high-end headsets connected to a PC and the simple smartphone-based solutions which are easier to use but lack visual performance, new purpose-built standalone devices were developed. These headsets have a built-in battery and usually use very power efficient processors based on the *ARM* architecture, like the ones used in smartphones, paired with high density displays for an improved VR experience. Additionally, they often use a wider array of sensors and some devices even use video recognition of objects in the user's surroundings to calculate the head position and improve tracking. Although these devices closely resemble smartphone hardware, they are optimized for delivering immersive VR as a standalone device. Most of these devices still only track orientation (3DOF), thus holding back immersion, but some standalone VR headsets like the *Oculus Quest* [wik20i], already support full orientation and movement tracking (6DOF) [VR18]. Of course, the graphics performance of these headsets is not on-par with the one achieved with a dedicated gaming PC. To overcome this, some manufacturers like *Oculus* implemented a solution to link up their standalone headsets to a gaming PC via USB cable [Dan20] in order to gain better graphics performance. Due to latency penalties and compression artifacts introduced when transferring the video image to the device, the visual quality does not entirely match those of natively connected, high-end desktop HMDs, but the sole option

of using a standalone headset as a fully-featured desktop VR device once in a while, sure is very compelling.

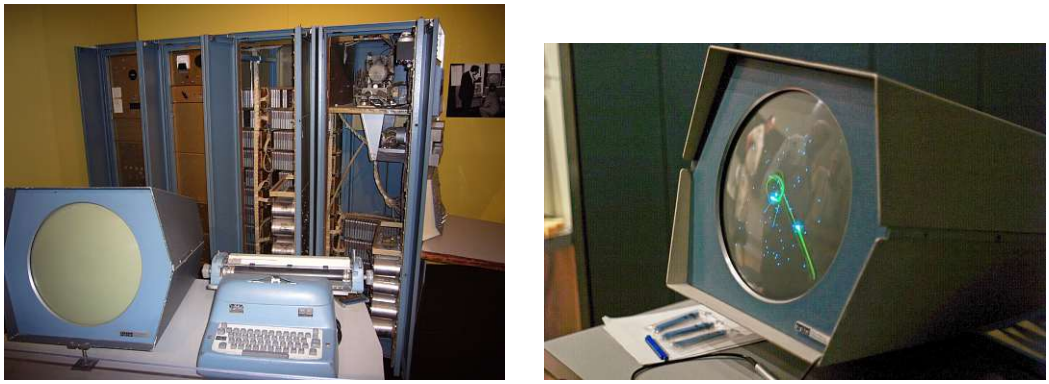
2.2 History of Multiplayer Games

Probably the earliest attempt known to create a fully interactive, electronic video game was made back in 1947 by Goldsmith and Ray with their *Cathode-Ray Tube Amusement Device* [GM48]. This system was inspired by World War II radar displays [Coh19a] and was based on a simple, hardwired electronic circuit [Don10]. It was designed to be played by a single person and allowed the player to set the trajectory of missiles and shoot them at various targets. The targets were stuck to the TV screen the device was connected to, in form of printed overlays and the game provided an explosion effect when a target was hit [Coh19a]. Due to a variety of circumstances the invention never resulted in a commercial product [Don10] and only a few handmade prototypes were ever built [Coh19a]. Since the *Cathode-Ray Tube Amusement Device* was purely mechanical and did not involve many of the characteristics we would attribute to a video- or computer game today, it is commonly not regarded as the first device of this category [Coh19a].

The next step in development was the tic-tac-toe-based game *OXO* (also known as *Noughts And Crosses*) by A.S. Douglas from 1952. It also was a single player game and was created to run on the *EDSAC* computer located at *Cambridge University*, England [Don10]. This game, reportedly, was the first to be graphics-based and allowed the player to compete against a primitive form of artificial intelligence (AI), which enabled the computer to react to the player's placement of X-es on the screen [Coh19b]. Although featuring many novel techniques and approaches, the game was almost completely overlooked at that time due to the fact, that the only system it ran on, the unique *EDSAC* computer at Cambridge, was never made accessible to the public [Coh19b].

The more commonly known history of gaming starts around 1958 with a multiplayer-only game. That year, William Higinbotham created a game called *Tennis for two* which ran on an analogue computer that was originally used to perform ballistic calculations for military purposes [Don10]. The game displayed a rudimentary side-view of a tennis court on an oscilloscope screen. It was played by two people, where each of them had a wired controller with a knob and a button. The knob controlled the angle at which the ball should be hit and the button triggered the smash to return it to the other side of the tennis court [wik21g]. While by some, this is considered to be first interactive video game in history, others oppose that claim: Due to using an oscilloscope display, some argue that this game cannot be considered a "video" game, since this term implies using a rasterized form of visual presentation like a television screen [Nyi16].

With *Spacewar!* Steve Russel created the first widely known computer game as we would define it today, back in 1962, at the *Massachusetts Institute of Technology* (*MIT*) [Don10]. It was another multiplayer game which allowed two players to perform a spaceship duel in outer space. It ran on a rather expensive "minicomputer" called *PDP-1* (short for *Programmed Data Processor 1*), which was roughly the size of a large car [Ken10]. This



(a) *PDP-1* Hardware: CRT display (left), console typewriter (right), processor frame (background). Source: [wik20k]

(b) The game *Spacewar!* on the CRT display of an *PDP-1* minicomputer. Source: [wik20n]

Figure 2.5: The Game *Spacewar!* in action on its original hardware.

machine used punched paper tape as its primary medium of storage and featured a CRT display. Due to the price of around $120.000, only 53 units of the *PDP-1* were ever sold and almost exclusively used at universities, secure military locations and research facilities [wik20n] [wik20k].

Originally, the players interacted with the game by using the built-in control switches of the *PDP-1*. Later, the team working on the game built dedicated controllers, which only contained the switches needed to play [Ken10]. The game showed two different rocket ships, each of which was controlled by one player. In the middle of the screen was a gravitational star which the players had to cleverly maneuver around, while trying to destroy the other rocket ship by shooting at it.

The game was very popular among students at *MIT* and the source code was shared freely across universities. According to several sources, Steve Russel briefly thought about making money off of it, but concluded that due to the expensive hardware required, it would not be profitable [Ken10] [Don10].

In the late 1960ies Ralph Baer started to develop a gaming system targeted towards TV owners. His invention ultimately resulted in the development of the famous *Magnavox Odyssey* [Gam08] which was released in 1972 [Don10]. This console put out rather crude graphics which basically only consisted of monochrome white shapes being displayed on a television screen [wik20e]. The system was able to run 12 different two-player games that where statically hard-wired into the console. Electronically, the game modes could be selected by inserting one of the supplied game cartridges, which in essence just were small circuit boards, which effectively worked like flipping dip-switches inside the console. This, in combination with the overlay attached to the television screen, allowed the user to select the game he or she wanted to start [Ken10].

The system was shipped with 6 different game cards, some of which could be used

for multiple games when combined with different screen overlays [Coh20]. Among the included games was a rudimentary version of a simple table-tennis game, that was later licensed and adapted by *Atari*, which was rooted in the coin-operated arcade business back then, and became more commonly known as "Pong" [Ken10]. Since the visuals of the *Odyssey* console were very simplistic, the manufacturer included a variety of transparent overlays to be placed on the TV screen in order to add visual detail and color to the image [Lan18].

(a) *Magnavox Odyssey* video game console (one controller attached). Source: [Lan18]

(b) Screen overlays that shipped with the *Magnavox Odyssey*. Source: [wik21c]

Figure 2.6: *Magnavox Odyssey* console and supplied screen overlays.

Hardware wise, the two players interacted with the system via two box-shaped wired controllers. Baer even developed an electronic rifle made out of plastic, that allowed a player to aim and shoot at the screen, allowing the system to recognize if the target was hit or not. This was the first peripheral add-on input device ever released for a video game console and also the first of its kind: a "light gun" [Lan18] [Hor11].

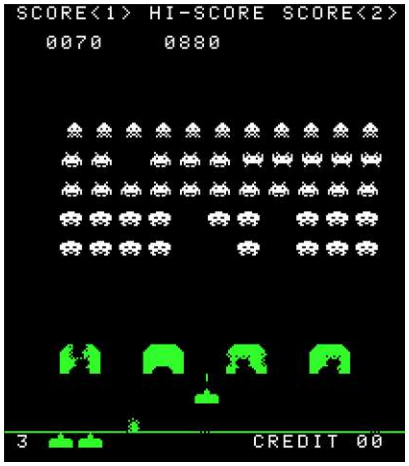
Although *Odyssey* was sold around 350 thousand times by 1975 [Gam08], the popularity of early video game consoles for home use remained mediocre, presumably because of their limited functionality and basic technology [wik20b]. The next big milestone for home gaming systems would have to wait until the second generation of gaming consoles which, by then, already featured programmable ROM cartridges which allowed users to play a variety of different games on their devices [wik20b].

The real breakthrough for video games however, can be dated back to 1980 when, for the first time, a very popular arcade game was licensed and ported over to a home video gaming system [Gam07]. *Space Invaders* had already been a huge success in arcades all across the world when *Atari* decided to push it as the "killer app" for their *VCS* (short for *Atari Video Computer System*, later also known as *Atari 2600*) [wik20m] [wik20b]. Although, strictly speaking, being more of a single player game that offered multiplayer, only by each player taking turns after one-another, *Space Invaders* was a true milestone in gaming history. Many later "giants" of the genre described it as their first contact with video games [wik20m]. Even Shigeru Miyamoto, the famous creator of the *Mario*

and *Legend of Zelda* franchises [wik20l], once named it after being asked for the one game that revolutionized the industry [Say07].

As the gaming industry slowly developed, many different games became popular. While a lot of titles were purely focused on single player modes, others allowed to be played by multiple people at once. Some of them worked on a turn-by-turn basis, where the players alternated in playing the game, while others allowed for true, real-time multiplayer gaming with two people playing simultaneously.

(a) The *Atari VCS* (later named *Atari 2600*), four-switch model from the early 1980ies. Source: [wik20b]

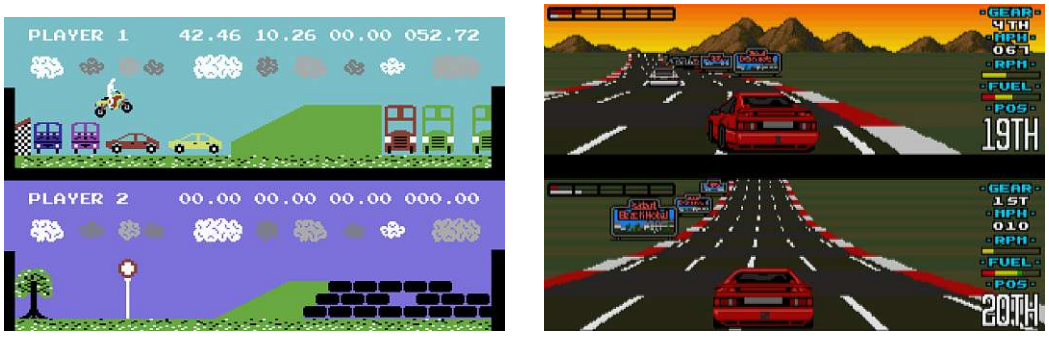


(b) Screenshot of the game *Space Invaders*. Source: [wik20m]

Figure 2.7: *Atari VCS* console and screenshot from the game *Space Invaders*.

When it comes to home video- and computer gaming, most of the real-time multiplayer games worked very well by providing the same view of the game for both players. One good example for this is the top- or side-view of a tennis court. Some games however, need different perspectives and/or dedicated areas of the screen to display information for each player separately. While the "split screen" technique was already used in studio-made films in the 1960ies [wik20q], it was not incorporated in video games until the 1970ies [wik20p]. Arcade machines like *Steeplechase* (1975) [ste] or *Drag Race* (1977) [dra] stacked player's views on top of each other, resulting in several horizontal bands or zones (one for each player) on the screen. This method, allowing for real-time multiplayer gaming on a single screen, was quickly adapted for home gaming consoles. The most common application for a setup like this is probably real-time racing games. One of the earlier examples is the game *Kikstart: Off-Road Simulator* from 1985, which was released for various video game and home computer platforms [mob] [kul]. Another, much newer and well-known example following the same concept was the famous *Lotus Esprit* video game series, the first part of which was *Lotus Esprit Turbo Challenge*, released in 1990 [Lem] [wik20d]. These split screen approaches on home gaming consoles were later facilitated by consoles like the *Nintendo 64* [wik20h] which allowed connecting up to four controllers to the same console [wik20p].

While split screen multiplayer was and still is a fundamental part of the local multiplayer landscape, especially on consoles, networked gaming was also developed quite early in the history of games. The game *Maze War* from 1973, is considered to be one of the first games to ever implement network gaming between different computers [wik20a] [wik18]. To be precise, it has to share this title as "joint ancestors" with the space flight simulator *Spasim* [wik20o] that was developed for the *PLATO* platform almost simultaneously [wik20f]. *Maze War* ran on the *PDS-1* graphics display system built by the *IMLAC Corporation* in 1977, which is considered to be the first commercially available low-cost implementation of Ivan Sutherland's *Sketchpad* system [Sut64]. The system was very popular in the 1970ies and was primarily used by universities and R&D organizations [wik20c]. The base version of *Maze War* was initially created to use two serially connected *PDS-1* units. A year later, the next iteration extended this feature by implementing a client-server networking feature which allowed it to be played by up to eight participants simultaneously [wik18] [wik20c]. A *PDP-10* computer served as the game host/server that also ran the *Maze War* AI program [wik20c]. The connection between the different machines used by the players was established through the famous *ARPANET*, which was already spanning across a number of universities at that time [wik21a].



(a) *Kikstart: Off-Road Simulator* in split-screen mode (*Commodore 64* version). Source: [mob]

(b) Split-screen multiplayer in *Lotus Esprit Turbo Challenge* on an the *Amiga* home computer. Source: [wik20d]

Figure 2.8: Two-player split-screen mode in *Kikstart* and *Lotus Esprit Turbo Challenge*.

Multiplayer games quickly grew more popular and began to establish well in the market. One markable breakthrough can be named with the first-person shooter game *Doom* released by *id Software* in 1993. First-person shooters had been around for some years by then, but *Doom* set precedence for many innovative technologies at the time. It featured network-based, peer-to-peer multiplayer gaming and also had a level editor, that let external authors create custom maps for multiplayer matches [Low20] [Don10].

Due to the growing popularity of networked multiplayer gaming, some console manufactures also wanted to enable users to play with others over the internet. This trend started as early as the 1980ies and manufacturers released modem add-ons for their devices.

In Japan, *Nintendo* originally positioned the *NES* (short for *Nintendo Entertainment System*), as it was called in the US and Europe, or rather the *Famicon* (short for *Family Computer*), which it was called in Japan, as a home computer system, rather than a gaming console. Due to that positioning, *Nintendo* could also market both a modem and an online service for their system in Japan [Gen96] [SE99]. With the *Family Computer Network System* users could gain access to game cheats, jokes and weather forecasts, as well as download additional content for games they already owned [wik20j]. The company also kept the system up to date, in order to work with their following products, but the network had limited success [Gen96].

Probably the biggest rival of *Nintendo* at that time, *Sega*, also tried to establish an online service which they dubbed *Sega Meganet* (sometimes also called *Sega Net Work System*). The system allowed downloading games via a subscription service and playing them with each other over dial-up connection [Red12]. Just as Nintendo's service, Sega's online subscription model never really took off, presumably due to the limited number of games supporting the service (only 17 titles) and the high price [Red12].

Although the combination of online services, subscriptions and gaming consoles was not a success from the get-go, their modern counterparts are an essential component of today's console gaming. Practically all major manufacturers of popular gaming platforms maintain an online service nowadays. *Microsoft* started their *Xbox Live* service back in 2002 [Cla02]. It was initially presented for the manufacturer's *Xbox* gaming console, which has since been discontinued from the network, but the service has always been updated to work with their latest products [wik20r]. *Sony* launched their *PSN* (short for *PlayStation Network*) in November 2006 to go live with their *PlayStation 3* console, which, alongside newer devices released by the company, is still supported until today [Nii06] [wik21f]. With the *Nintendo Network*, *Nintendo* launched their latest attempt at an online service in January 2012 which was first supported by their *3DS* handheld gaming device and now also supports the *Wii U* console [Goo13] [wik21e].

Since then, network gaming and especially internet gaming has become an essential building block of the modern gaming landscape. While there are still a lot of very popular single player titles released today, there are also games, that solely focus on online multiplayer gaming. Probably one of the most popular examples is the *MMORPG* (Massively Multiplayer Online Role Play Game) *World of Warcraft* [oEB16] by *Blizzard Entertainment*. With this game, users can only play when connected to the internet, since the whole virtual gaming environment is completely hosted on the servers owned and run by the developer and publisher.

This wide spread of online gaming opens up technical possibilities that could not have been thought possible, when the first attempts on network gaming were made. A lot of games these days are platform agnostic and it almost does not matter which device or system is used to join. One well known and recently popular example for this is the shooter game *Fortnite* by *Epic Games*. It can be played on a PC, several gaming consoles and even on most smartphones. Although users are playing on entirely different devices, they can compete against each other in various online matches. Since some input

methods come with system-inherent downsides, for example, with regard to the precision the player can aim at an enemy, developers often have to implement tools or algorithms to level the playing field with such an inhomogeneous audience. One example would be so-called "aim-bots", which try to account for system-inherent aiming and shooting inaccuracies on some consoles or hand-held devices.

Given such device- and system-independent gaming platforms, one might conclude that all participating players can be treated more or less equally, no matter if they are sitting in front of their TV with a gamepad, at their desk using mouse and keyboard or are using an expensive gaming system with an attached VR headset. Unfortunately, the story is not that simple since especially VR setups pose an entirely different form of interaction with the game environment and therefore, can usually not be treated like other, more conventional, input- and visualization-devices.

2.3 Competitive vs. Collaborative Multiplayer

Before taking a closer look at the differences between interaction paradigms of traditional interfaces and VR setups, some more general approaches on how multiple players are handled inside a game need to be explored.

To many, multiplayer video- or computer-games are basically an extension of traditional boardgames into the virtual space [wik20g] and can enrich the experience of playing a console or computer game substantially by adding social interactions into the mix. In electronic games, just like it is the case with physical board games, there are basically two different ways multiple players can interact with each other: they either are opponents competing against one another or working together to achieve a common goal or objective. In the video- and computer-gaming industry these concepts translate into the more traditional Player vs. Player (PvP) mode and the accordingly named cooperative (Co-Op) approach.

Historically speaking, two players competing against each other is the older concept of these two, since it has already been part from the beginning of multiplayer games back in the early days of computer games. In titles like *Spacewar!* or *Pong*, one person tries to beat their human opponent in direct combat, be it by destroying their spaceship or by winning against them in a table-tennis-like match. Even in games that are not designed to be real multiplayer titles like *Space Invaders*, people compete against each other by trying to reach a higher score and earning some glory by documenting their achievement by making an entry to the game's high-score list.

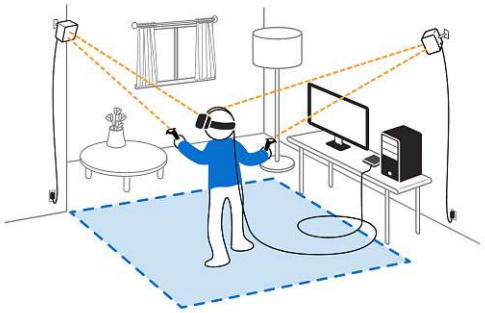
A completely different approach of integrating social human-human interaction in a game, is the so-called Co-Op mode. In contrast to one person competing against another, here the players have to work together in order to achieve a common goal or objective. This usually does not relate to another group of human players, but rather the game world itself or a computer-controlled enemy. This approach facilitates social interaction probably even more, since the players are more likely to communicate with each other

during playing the game to coordinate their efforts. In the game *Call of Duty: Modern Warfare* [cod] for example, players have to join their forces to complete special mission objectives and to defeated their computer-controlled counterparts [Hoh19]. In a scenario like this, each in-game character has their inherent advantages and disadvantages, which makes coordination and cooperation between the players key to beating the game. In order to coordinate such endeavors, players must be able to communicate with each other. In many online games this is done by using either a text- or voice-based chat feature, that can be integrated in the game itself but can also be run as a separate application in the background if a game does not offer such a feature natively.

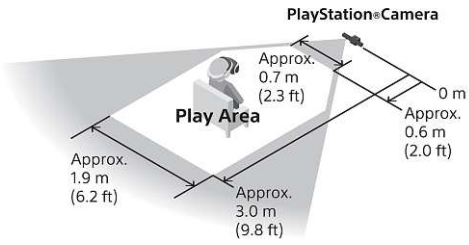
Of course, a need for an electronic communication channel only applies when the players are merely connected via network and are not sitting next to each other. While this is rarely true for PC gaming, this type of co-located multiplayer constellation is not uncommon for many co-op modes in console-based game. Here, several players might come together on a couch in front of a large TV and play a game with or against each other in split-screen mode. This specific setup is sometimes also referred to as "local co-op" or "couch co-op" [wik21b].

2.4 Challenges of Multiplayer VR

In contrast to most traditional games, making a VR title ready for multiple players is quite challenging. Probably the simplest way to realize this is to simply add a networked gaming option. This way, several players can easily join a game over the internet or any other kind of network, while still using their own VR headset and gaming PC or console. A multiplayer feature like that is relatively easy to implement in VR and therefore not uncommon since the computational overhead of this game mode is, when compared to single player VR, neglectable. Unfortunately, this drastically changes when the goal is to implement a local multiplayer setup where both players have to share the same room.



(a) Setup diagram for standing VR from the *HTC Vive* user manual. Source: [Viv16]



(b) Space requirements in sitting VR from the *PlayStation VR* manual. Source: [Shu17]

Figure 2.9: Two setup diagrams for standing and sitting VR modes.

The hardware requirements for supplying an HMD with a constant stream of stereoscopic,

in-game images are immense, especially when taking into account that a relatively high refresh- and framerate has to be kept, in order to minimize motion sickness [Nic99] [LKK17] [SSB+20]. Achieving this with more than one headset at the same time, is an even greater challenge to begin with. This is probably one of the main reasons, why this is hardly done. And unfortunately, when trying to use two or more HMDs side by side, even further difficulties arise:

For one, there are obvious spatial constraints, which make it quite challenging, if not impossible, for this to work without extensive preparation of the surroundings or a lot of space between the players. There have been some attempts to proactively manage the available space to increase flexibility for players, but generally speaking, a person in VR needs a given space to move around [YWC18]. Even when assuming a sitting or standing position for two players in VR next to each other, each of them needs at least both arms' lengths reserved all around them. Within this radius, which is even larger when considering fully-featured standing/walking VR, there must not be any objects that the player could trip over or damage. This zone should also be free of any obstacles like furniture or walls the player could hit and hurt him- or herself. Lastly, the spaces of the players must be kept separately and cannot intersect each other to avoid collisions between them, which makes two or even more players side-by-side in VR spatially very demanding.

Another key element that has to be taken into account is tracking. Not only do the HMDs need to have constraint and reliable tracking themselves [WPB+20], but all players usually also have input devices in their hands. In order to maximize immersion, this usually includes two separate controllers for each player (one in each hand), that are intended to be tracked wireless along all 6 (or at least 3) degrees of freedom [LPLK17]. With an increasing number of players, the number of devices to track therefore increases three-fold. An alternative would be to setup completely separate tracking areas for each player side-by-side, which would not only increase the required space drastically, but could also lead to interference between the setups e.g. when using infrared-based lighthouse tracking. There might be a way to make this work for two players, but it becomes increasingly complex when the number rises.

Although being highly impractical, some tinkerers still managed to get more than one VR headset working on a single PC. This does however, usually either involve a mixture of different VR headsets and manufacturers or heavy virtualization using a dedicated graphics card for each HMD [SMP+19]. Since the actual use cases are far and beyond, it is no wonder that console manufacturers handle things more restrictively in this regard. *Sony* for example, explicitly states that players cannot run a second VR headset off the same console for their *PlayStation 4* [Shu17] [Osa17].

A way around these problems all together, while still integrating at least some elements of VR in local multiplayer gaming, is to mix VR with traditional input- and display-methods. More specifically, only one player is wearing an HMD and uses a fully immersive VR setup, while the others interact with the game or simulation by means of more traditional input methods. In lack of additional VR hardware, these players would be constrained

to use a gamepad or mouse and keyboard and follow the game on a traditional 2D screen or display. These types of games are called "local asymmetric VR" [wik21d] or sometimes also "Party VR" [Mel19b] [VRG19]. With this approach, local multiplayer (or "couch-co-op" [wik21b]) and VR do not completely exclude each other and can still be an entertaining experience for multiple players without the added need for the additional hardware and space.

Figure 2.10: A group of people playing a "Party VR" game with Sony's *PlayStation VR* system. Source: [Shu17]

2.5 Different Interaction Paradigms

Before further discussing possible ways of mixing VR and non-VR, the differences between these two interaction methods need to be explored. The main challenges in mixing these two arises from the fact, that they follow fundamentally different interaction paradigms.

The traditional way of interacting with a game or any other kind of simulation for that matter, is by using one or more input devices. A typical example in the world of video games would be a gamepad (on consoles also often referred to as controller) [dL14]. Other input devices are more purpose-built and try to mimic their virtual counterparts. An interesting example would be a joystick that was almost omnipresent in arcade machines, gaming consoles and home computers alike. While it was reportedly originally developed to mimic the controls found in a real-world tank for a corresponding arcade game back in 1974 [dL14], it quickly became a real general-purpose input device. While the principles stay the same across all variants of these devices, many of the newer models are designed to better fit a specific use case, e.g. a flight stick to mimic the controls of a jet plane. Many other input devices like the famous light gun or a steering wheel with pedals

are, in contrast, strictly purpose-built by design and can hardly be used for different applications.

A completely different approach is ubiquitous when it comes to PC gaming or navigating through any kind of simulated environment or application on this system: keyboard and mouse. While gamepads are very common among PC players, probably the majority of them still use their keyboard and mouse as primary input devices for gaming. Although they where primarily built for completely different types of interaction, namely inputting text or navigating through "clickable" user interfaces, they can also be efficiently used for gaming. One major reason for the wide spread of the usage of these input methods when it comes to gaming might be, that these devices are used for other applications as well and therefore usually are already connected to almost every PC anyway. As diverse as the methods of user input may be, so consistent is the way of presenting the visual system output to the user. Be it people using a console or sitting in front of a PC, almost all of them watch the game on a traditional 2D display like a TV or computer screen. While some alternative solutions exist, they are sparsely used and highly uncommon, especially among gamers.

This is where things really change when we get into VR. While methods of hand-based user input often differs from one player to another, they still are all fundamentally based on the same principle of single, hand-held devices, controlled by buttons or other kinds of touch sensitive elements. In VR, every hand usually has its own controller-device, sometimes even detecting movement of each finger individually [Rob19], to facilitate immersion in order to create a more realistic experience. Usually the controllers are operated "blindly" as many games and applications only show a - more or less rudimentary - virtual representation of them in VR.

The major difference when using VR is, however, that each user is supplied with personalized visuals, specific to their head position and orientation, while the HMD also acts as an input device itself. Moving, tilting or rotating the head translates into camera movements that are mirrored by its virtual counterpart, thus creating the feeling of "being in there". In addition to the visuals presented to the player, other senses of a person in VR are also stimulated (e.g. with spatial 3D sound), leading to a high degree of presence and immersion which is often considered to be a key characteristic of VR when compared to other forms of human computer interaction [DBJ+19].

This type of "full body interaction" and immersion into a virtual world comes with some major differences when compared to traditional input- and display-methods. When in VR, a person is almost entirely encapsulated and usually not able to perceive his or her surroundings unobstructedly. This facilitates immersion, but it also introduces the risk of a person hitting something in the room or hurting themselves by falling over furniture, since players simply cannot see their immediate surroundings. Also, a VR headset needs to be supplied with much more frames per second in order to keep up the appearance of fluid motion and to prevent cybersickness. Stuttering or low framerates are usually much more noticeable in VR than they are on a regular screen or display and can easily cause discomfort, motion sickness or vertigo [Nic99] [LKK17] [SSB+20]. This translates

(a) Joystick from a classic *Atari* video game console. Source: [Bru13]

(b) Controler from a classic *NES* (*Nintendo Entertainment System*). Source: [Bru13]

(c) Modern controller from a *Sony PlayStation 4* console. Source: [Bru13]

(d) Modern controller from a *Microsoft XBox One* console. Source: [Bru13]

Figure 2.11: Traditional input devices for various video gaming consoles.

to higher requirements in graphics performance when using a VR headset, compared to traditional display devices.

2.6 Working with Mixed Input Methods

All before-mentioned problems and constraints paired with the current impossibility to use more than one VR headsets on a single gaming console, raise the need for an alternative approach. If the goal is achieving a couch-co-op-like experience involving at least one player in VR, mixing input methods is a possible solution. Since there are such key differences in using more traditional input devices paired with a television screen or computer monitor to using a VR setup, this can be quite challenging. When settling for just a single player in VR and others, at the same time in the same game, on gamepads looking on a TV screen, one has to re-evaluate how these players can interact with each other while still providing an enjoyable gaming experience.

Figure 2.12: Handheld VR controllers from the companies *HTC* (left) and *Oculus* (right). Source: [Lan16b]

Due to the inherent differences, many asymmetrical local VR games are co-op games. The most likely reason for this is the substantial difference in the amount of control between players using different input methods. Because of the more natural form of interaction a VR setup allows, a single player in VR, competing against one using traditional methods, might not be balanced enough to make a game fair and enjoyable. More traditional players teaming up against one in VR on the other hand, like it is done in the cross-platform game *Acron: Attack of the Squirrels!* [squ] [Mel19a], is more feasible but can also potentially lead to problems. This setup could in some situations facilitate an "all against one" mindset which might not be desired for an entertaining evening with friends. When designing a co-op-based game with mixed VR and non-VR participants however, each player can take a different role, thus allowing the player in VR to take full advantage of their setup and join the others to achieve a common goal. In many games all "traditional" players form a team, that either has to support the player in VR to achieve a mission objective, or which are, in turn, supported by the player in VR in achieving a given goal inside the game.

When looking at a game of this kind, the non-VR players are more or less subject to ordinary game mechanics and can therefore be seen as "traditional" players in regard to their gaming experience and degree of immersion. The far more interesting point to observe is the player in VR. In contrast to the more commonly used single-person experience with pre-programmed, scripted events inside the virtual environment, there are usually no effects, events or other stimuli controlled from outside the simulation or game. In this case however, actions inside the game are triggered by other players which

might be part of the gaming world, but will not always be visible to the player in VR. Actions from other humans are, generally speaking, much less predictable and might therefore require more adaption and flexibility from the person in VR when compared to pre-scripted ones that can eventually be "learned" over time.

On top of any in-game interaction, there is also an additional, many times even subconscious, "side channel" communication involved. The person in VR might, for example, unconsciously show signs of stress or give other indications on how he or she is handling the current events in the game. This is especially true for competitive games, where players compete against each other. In an asymmetric scenario like this for example, the team of traditional players could easily deduct if their VR opponent can manage to fight off virtual attacks just fine or if he/she starts to struggle and might lose some metaphorical ground soon.

Although currently mainly game-related settings were discussed, the same of course also applies to any other form of virtual interaction. In mixed virtual environments for meetings, collaborative work or learning experiences [SSO+12] [JKK+20] [GKK+17] for example, there might be users participating with a VR headset as well as people on a desktop PC, all joining the same virtual environment. Strictly speaking, even a traditional flight simulator like the one used to train jet pilots or astronauts, is based on the same principle, just lacking the VR component: the "player" is inside a simulation, which contains elements that can be triggered from people outside the simulation, that are not part of the virtual environment themselves. In traditional and well-established setups like this, the effects on immersion might not be that important since the focus lies almost solely on the learning process, but when this is applied to VR, either in form of a game or any other kind of simulation, immersion starts to become increasingly important.

The question presenting itself is, in how far the - direct or indirect - interaction with a person outside an ongoing simulation and therefore outside of the perceived virtual world, can influence the degree of immersion for a person in VR. These "external" people might be part of a given game or simulation themselves, like co-players that just use gamepads and a TV instead of a VR headset and just have different roles. They might also be completely outside the virtual world like the person operating the controls in a simulator-setting. In order to evaluation this rather facete-rich question, I created a basic soft- and hardware setup, implemented it and carried out a small-scale user study to research the influence of outside interaction on a person's immersion when in VR.

2.7 Related Work

Surprisingly, the topic of immersion in local asymmetric VR is not covered extensively in scientific literature yet. There is a variety of publications dealing with asymmetric collaboration, many of which also involve Virtual- or Augmented Reality setups. Many of these publication however, focus almost entirely on collaboratively working on projects, educational use cases or telepresence systems. With these rather functionality-based approaches, it is no surprise that immersion is not overly emphasized in these publications.

Peter, Horst and Dörner for example, propose an asymmetric VR/non-VR setup for visualizing potential flooding issues as consequences of heavy rainfall for future home owners [PHD18]. In their setup, a civil engineer acts as *VR-Guide* in front of a laptop. The guide then leads a layperson wearing a VR headset through the simulation.

Although the human *VR-Guide* is, strictly speaking, not part of the virtual world that is perceived through the HMD, he or she can directly interact with it. The guide can view everything the person in VR sees and can also fly around the virtual environment by using a simulated camera drone. The external user can change certain simulation parameters and manipulate objects inside the simulation. He or she can also highlight objects or areas to steer the VR user's attention towards a specific region. During the whole process, the *VR-Guide* can constantly talk to the user. This way, the traditional concept of audio-tour-guides based on pre-recorded audio files, can be extended by adding personal interactions and information tailored to the specific situation.

An additional benefit of this setup by Peter et al. is the possibility for the *VR-Guide* to monitor VR hardware and movements of the person using it. The guide can for example potentially keep track of the maximum rotation speed of the person using the headset and the accumulated rotations performed, as well as monitor the framerate and latency of the HMD, which could possibly serve as an indicator of arising cybersickness. Although their setup is much like the one created for this thesis, they focus their work on their novel in-simulation highlighting technique which uses a particle system to create a spotlight [PHD18].

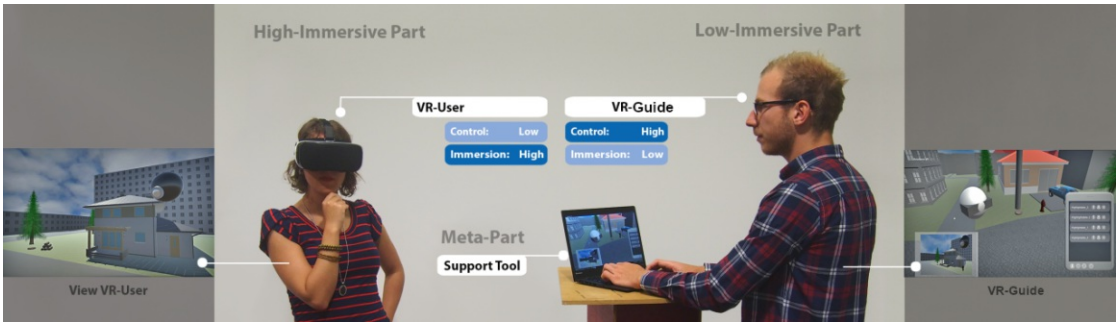


Figure 2.13: Diagram of the *VR-Guide* setup discussed by Peter et al.. Source: [PHD18]

They implemented their prototype in the *Unity* game engine and conducted a small-scale user study to test it. Their results showed, that their interactive *VR-Guide* approach worked better than pre-recorded audio files, due to a more individually guided tour for the VR user. Interestingly, they also noticed that the presence of a *VR-Guide* reduced the movement and head rotation of the person in VR drastically, thus also reducing the risk of cybersickness [PHD18]. They concluded that for their specific use case, an external user who was not fully part of the VR simulation but could still interact with it and its users, had a positive effect on the person in VR [PHD18].

Ibayashi et al. took a very similar approach and developed a tool for asymmetric

interaction between architects and designers on one side and inhabitants and owners on the other. With the system presented in their paper [ISS+15], they create a "virtual dollhouse" where designers can change floorplans or furniture positioning inside an apartment or commercial space by using a touch screen. The future occupants are inside a fully immersive VR simulation of that space and can see all changes in real-time.

The designers and/or architects interact with the virtual world via a large, touch enabled table-top display. On the screen, the designers can see the apartment or commercial space (e.g. a restaurant) from a top-down view. The current position and view direction of the HMD user is visualized by a virtual avatar ("doll") [ISS+15].

The future occupant, tenant or owner of the space to be designed is wearing an HMD and is fully immersed into the simulation. In order to be able to communicate with the designers using the table-top display, the ceiling of the virtual apartment is rendered transparent. Instead of an opaque ceiling, the user is presented with an image captured by a camera attached to the tabletop screen, looking upwards.

The designers can interact with the environment via touch gestures, visualized to the HMD user as a giant virtual hand coming down from the ceiling. The person in VR in turn, can "point" at objects by touching a smartphone-display mounted to the front of his or her VR headset, that serves as a touch input device

Ibayashi et al. also implemented a prototype using the well-established *Unity* engine, featuring two separate applications, both running on the same computer [ISS+15]. One application supplied a connected tabletop display, while the other one supplied a connected HMD. The smartphone for the VR user's touch input was linked up via WiFi.

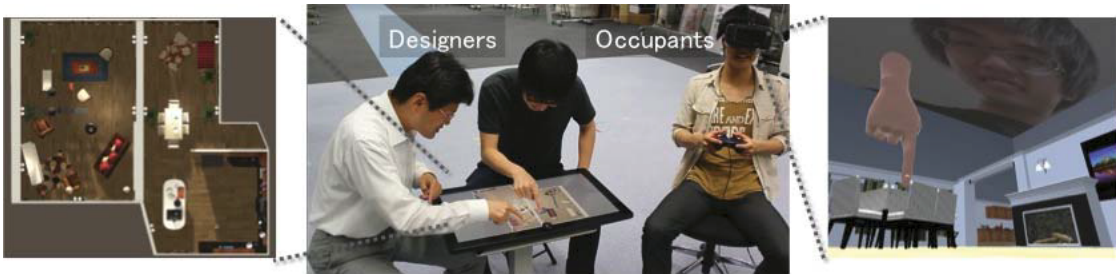


Figure 2.14: Virtual Dollhouse VR system by Ibayashi et al.: Top-down view for designers (left); All three collaborators using their devices (middle); VR view of occupant (right). Source: [ISS+15]

Although these two publications are based on a setup very similar to the one created for this thesis, the interaction between persons outside the virtual world and the VR user is an essential part of the concepts here. Due to that fact, these scenarios can merely serve as examples of related work, since it is difficult to compare them to the setup described in this thesis directly. Any "outside" person, be it a virtual guide or designers on a tabletop display, take an active role in communicating with the user in VR directly, which provides a completely different baseline for immersion. Nevertheless, these papers

provide valuable insights and showcase two completely different approaches and practical use cases for asymmetrical VR setups.

Thomsen, Nilsson, Nordahl and Lohmann take a completely different approach by focusing on asymmetric collaboration for learning purposes [TNNL19]. They borrow some concepts from the gaming world and take gamification into account, but in essence, aim to create a teaching tool for use in a classroom-like environment. Since their paper is declared part of a larger research project, they do not implement a concrete prototype but rather deal with the applied concepts in a more abstract and theoretical way. A special focus seems to lie on asymmetry itself, as well as on its levels and implications thereof.

The paper follows a VR enhanced teacher-classroom approach where the authors define the terminology of "actor" for the person in VR and "assistant(s)" for one or more non-VR users. They categorize asymmetry in three different degrees, depending on the possibility of interaction of non-VR participants:

Low asymmetry is defined as non-VR users operating an app to join and interact with the virtual world. They can interact with the world itself, with objects inside it, as well as communicate with the person in VR. The "external" user can therefore be seen as almost equal to the VR "actor", but without the same degree of immersion. Instead of a fully immersive VR system, this person or group of people, use a traditional 2D display (e.g. a smartphone or tablet) which serves as a "window" into the virtual world.

Medium asymmetry is a setup in which the person outside of VR can send digital information to the person wearing the VR headset, but cannot directly interact with the virtual environment or objects within it.

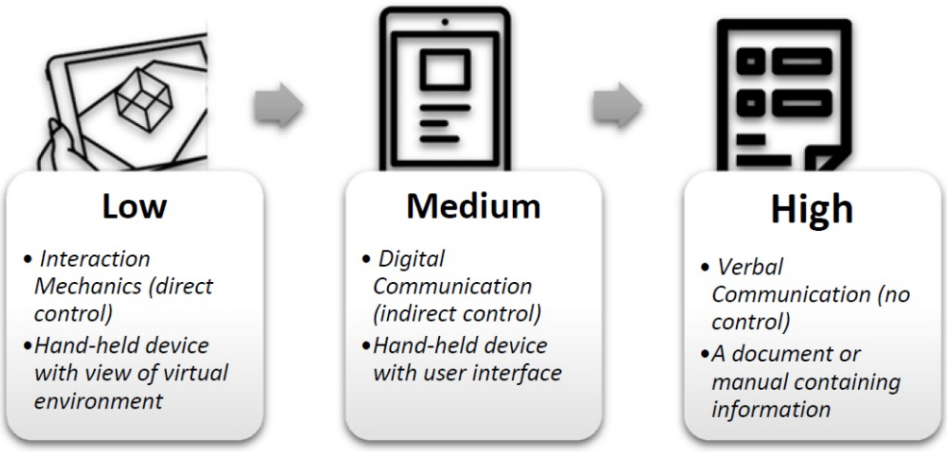


Figure 2.15: Three degrees of asymmetry as defined by Thomsen, Nilsson, Nordahl and Lohmann. Source: [TNNL19]

In high asymmetry, the person outside of VR does not even have a digital way of

communicating with the VR user and can only do this in an analog way. In this case the "outside" user solely has a passive (not-connected) app or even just a piece of paper as their tool.

The experimental setup for this thesis, would arguably mostly fit into the "low asymmetry" category, but does not include all of its characteristics. In some regards, it could also be seen as "medium asymmetry", since the external user can interact with the simulation but not directly with the player in VR. Also, there is no contextual collaboration intended in this thesis' setup and no digital way of bidirectional communication in place either.

In their special use case, the authors of this paper came to the conclusion, that VR in general, can serve well as an educational tool [TNNL19]. There are some benefits, like the ability to perform field-trip-like exploration to otherwise inaccessible areas (e.g. ancient Rome). In other areas, primarily when learning "simple knowledge" (e.g. vocabulary), using VR-based systems might even become an obstacle in the didactic process as it might pose a possible distraction to the learners [TNNL19]. This can partly be overcome by implementing small rewards into the virtual environment, therefore possibly connecting this topic loosely to gamification as well.

According to the authors, the approach discussed was intended to be tested with classes of primary school students in Denmark [TNNL19]. In a typical real-world classroom setting there might be some additional constraints like a limited budget and/or limited availability of VR headsets, thereby consequently increasing the diversity of devices used to join a virtual world during a session. Due to their narrow target use case, the authors centered their attention more on systematic benefits and possible problems occurring in a learning or teaching environment, rather than on maximizing immersion. With regard to their focus, they concluded that VR can be didactically beneficial in some cases but can also become a distraction for learners in others [TNNL19].

A more practical take on the thoughts and concepts described above was made by Borst, Lipari and Woodworth [BLW18]. They designed a VR-based framework for virtual fieldtrips named *Kvasir-VR* for use in high school classes. With this asymmetric system, only students wear VR headsets while the teacher is located in front of a traditional 2D display interacting with them. This usually translates into the teacher standing in front of a large TV while his or her 3D image is captured by a depth camera (in this case a Microsoft Kinect) and audio is recorded over a microphone. The avatar of the teacher can then either be streamed live to the virtual world of the students, which can see and hear them through their HMDs and headsets, or recorded and played back as a virtual avatar performing short VR quizzes.

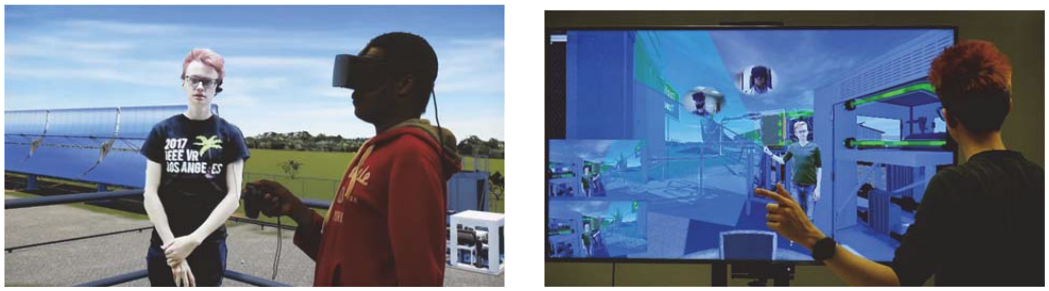
The authors tested their system on real students in order to find out, if there is any substantial learning gain in using such a virtual field-trip framework and compare live (networked) guides/teachers with pre-recoded versions. For their field test, they digitally recreated a field trip to a solar power plant and enhanced it with additional educational content and interactive elements.

Borst et al. implemented this system with the *Unity* engine and tested it on 88 high

school students whose learning gain was assessed by comparing the results of a pre- and post-fieldtrip multiple-choice test [BLW18]. The teacher was the same in both the live and the pre-recorded variant, as well as the content, talking points and script for the teacher.

While students were immersed inside the virtual world, they were physically seated and all shared the same virtual viewing location [BLW18]. Other students were shown in the virtual environment as generic avatars, positioned to the student's left or right side. Movement was realized via teleportation, that was performed by only one student who was assigned this task by the teacher [BLW18].

The teacher stood in front of a TV, showing a mirrored image of him- or herself, which allowed pointing to objects in the virtual world, even if they were located behind him or her. The teacher's interface also featured a live camera feed of each student wearing their VR headset. The student's images were positioned on the teacher's screen in a way, that allowed the teacher's avatar to look at the student he or she addressed virtually, for a more natural communication and proper eye contact [BLW18]. The interface also included small live feeds of the current field of view of each student to provide the ability to check what students were looking at and to avoid miscommunication when students were asking questions. While the live teacher was able to ask the students questions in an interactive manner, the pre-recorded version was scripted to ask these questions too and allowed users to select an answer virtually.



(a) 3D visualization of teacher in VR, overlayed by footage of a student.

(b) Teacher interacting with teacher-interface on a large TV.

Figure 2.16: Images showing the *Kvasir-VR* framework by Borst et al.. Source: [BLW18]

The authors of this paper came to the conclusion, that although contrasting studies exist, the learning gains during their tests were substantially better with live teachers compared to pre-recorded ones [BLW18].

In contrast to the previously discussed publications, Liszio and Masuch look at the topic of asymmetric collaboration and immersion in VR from a gaming perspective [LM16]. Their main focus lies on immersion, which can be destroyed all to easily by any outside distraction reminding the player of the "real world". Arguably, this can pose a difficult challenge for designers aiming to create a local multiplayer game with social interaction and high immersion alike. While, according to Liszio and Masuch, some authors believe

that these two elements contradict each other, some researches see social interaction as a fundamental key element of any entertaining game scenario. Cairns et al., for example, show in their non-VR test-setup, that immersion of players was higher when playing against one another, compared to playing against the computer [CCD+13]. Interestingly, they also found that there is hardly any difference to the degree of immersion, whether two players are in the same physical location or if they are only connected via a local network or the internet.

In search for a solution for this problem, the authors decided to implement an asymmetric local multiplayer game, that makes communication between players an essential part of the game mechanics. Due to the integration of the "real world" communication into the game, the extent of disruption of immersion should be reduced or even eliminated completely [LM16].

The game Liszio and Masuch designed is played by three players at once, where each of them has a designated role [LM16]. They all have a common goal, namely to find parts of their crashed space ship in order to re-assemble it and have to work together to achieve it.

One player wears a VR headset and his or her motions are tracked by means of special controllers. This player pilots a virtual robot (more precisely a so-called "Mech"[LM16][wik19]) that has to collect parts of the wreckage, as well as fend off enemy attacks. All needs of the Mech robot, like a constant supply of energy and steady ammunition level, as well as weapon selection, are handled by the co-pilot.

The co-pilot can, in turn, not directly interact with the game's environment him- or herself, but uses a tablet PC to manage all internal systems of the Mech. This player has to control energy as well as ammunition levels and also generate energy from minerals. The latter is done by repeating visual patterns on the tablet's display, which are becoming more complex over time. The co-pilot also is the only player that sees a mini map of the environment and has to give the Mech pilot directions on where to go.

The third player operates a scout drone, also by means of using a tablet. The tablet's screen shows an isometric view of the drone and the ground beneath it while the device is also used to steer the drone by tilting the tablet in the appropriate direction. The person controlling this scout drone has to find missing parts of the wreckage and also has the ability to collect minerals from the ground, which are then converted into energy by the co-pilot. Additionally, the scout drone can launch fireworks in order to distract enemy units.

In this specific game setting, all players are forced to collaborate with each other, since otherwise the goal of the game would not be achievable. Unfortunately, the authors of this paper do not seem to have carried out a user study or survey to assess their results but rather assume that their concepts invoke high levels of immersion and social presence [LM16].

As can be seen in this section, the possible approaches to asymmetric collaboration in

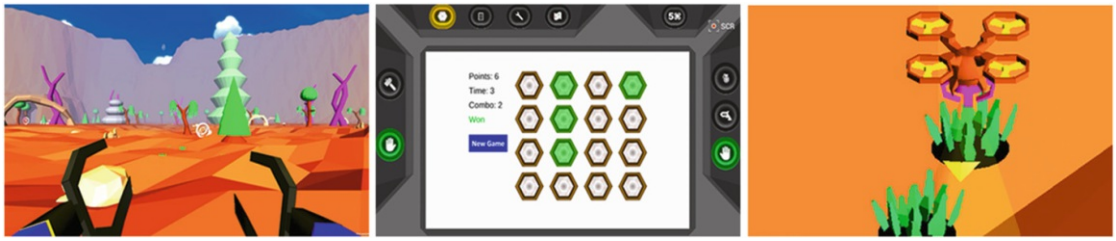


Figure 2.17: Three-way asymmetric multiplayer game by Liszio and Masuch: VR view from pilot(left); Visual pattern to solve for co-pilot (middle); Isometric drone view from scout (right). Source: [LM16]

VR are diverse. The spectrum stretches from highly practical use cases, like visualizing flooding for future home owners, where immersion is of lesser concern, all the way to classic gaming scenarios, where immersion is a key element since it is a convincing sales argument. Interestingly, none of the described publications seem to concentrate solely on the immersion of the player in VR when influenced by outside, humanly controlled interaction. Maybe the results gathered in the course of this thesis can provide some useful insights and may lead to some interesting results.

CHAPTER 3

Design and Implementation

3.1 Requirements: External Interaction Console for a VR Simulation

Since I could not find any answers to the questions I was interested in, I had to implement a system that would allow me to conduct a user study and draw my own conclusions. In order to properly evaluate the influence of humanly controlled outside interaction on immersion, I had to design a system in which external influences could be triggered selectively and in an isolated way. The person in VR would have to be completely immersed in an ongoing, game-like simulation. There also had to be a way for another person to interact with the VR simulation and trigger effects or actions inside it, without the need for direct communication between both subjects.

My first two research questions (number 1 and 2) concerned the immersion of the person in VR and in which way it is affected by externally triggered effects. Since immersion is highly subjective and hard to quantify, let alone to measure objectively, I needed baseline data of immersion without any externally triggered stimuli. This was one of the main reasons why I chose the *Virtual Jump* simulation to base my project on. The *Virtual Jump* project was an already fully functional VR simulation that only had to be adapted and extended to fit the needs of my requirements. In addition to that, I was able to use data already gathered by my supervisor during previous projects, that also contained results of an immersion-related user study without any externally triggered effects. Since the *Jump Cube* was not only used with its original content but there are also a lot of different versions and additionally other simulations, I would only be able to use a subset of the available data based on the original version. Nevertheless, they formed a helpful foundation to compare my results to. With the fundamental VR simulation already implemented, I had to evaluate which additional features I needed to implement in order to answer my previously listed research questions.

Since I wanted to research the influence of externally triggered effects, I, of course, had to add additional effects to the simulation or at least make some, that were already present, triggerable from the outside. In agreement with my supervisor, I decided to define three different categories of effects: simple, moderate and complex.

1. Simple effects are interactions from a person outside with the simulation directly. They might be quite subtle and may not even be instantly recognized by the user in VR as being outside-controlled or triggered. An example for this type of effect would be rain starting to fall inside the simulation or the deployment of a subtle scent inside the simulation.

2. Moderate effects are events that are instantaneously recognizable as being triggered in some form, most likely from the outside, and have direct impact on the simulation or the simulated environment. Examples for this type of effect would be strikes of lightning, explosions or other very distracting and probably even shocking events that occur in the simulation.

3. Complex effects have an inherent duration and consist of several other, smaller effects. These sub-effects can all be categorized in one of the before-mentioned categories themselves, and form a complex effect when being arranged or grouped together. An example of this would, for example, be a thunderstorm. It does consist of rain, falling over a certain timespan, as well as thunder and lightning.

With the categorization in place, the next part was defining how many and which effects needed to be implemented in order to answer my questions. Looking at the categories above, it becomes clear, that effects of different categories might affect a person in VR in fundamentally different ways. While short and instant effects like an explosion might startle the player and distract him or her a lot, effects of the "complex" category are much more subtle and might in some cases even go completely unnoticed at all. Effects with such different characteristics and a variety in their perceived origin, could have drastically different effects on immersion. So, in order to be able to evaluate the impact of events obviously manually triggered from the outside, as well as ones that could easily be able to be part of the simulation, I had to implement at least two different types. This would also allow me to evaluate if manually triggered effects would be perceived differently than pre-programmed ones and possibly help answering my research question number 3. Therefore, the simulation needed to contain at least one "moderate" effect that would be instantly recognizable as externally triggered and also one or more of either the "simple" or "complex" category, that might possibly appear to be part of the pre-programed simulation for some people. This setup would allow me to find out, if there are any effects on immersion at all, and whether they are positive or negative.

By working with different kinds and categories of effects, it would also be possible to blur the line between obviously externally triggered and more abstract ones, that could also be part of the simulation themselves. The questionnaire filled out by the "Jumper"

after their virtual parachute jump contained questions with regard to which externally triggered effects could be recognized. This, in combination with allowing the "controller" to trigger both types of effects independently, would allow me to assess which effects were perceived as externally triggered and which would possibly be overlooked or appeared to be part of the original simulation.

The last research question concerning the person in VR, was whether externally triggered effects would interfere with the content itself (question 4). The answer to this will likely differ on an effect-basis since they all cause distractions to a very different degree. This fact makes it rather hard to evaluate that question on a general level. Although the variation of the original *Virtual Jump* content did originally contain some narrative content, I decided to remove it. The presence of a narrating voice accompanied by images which passed by the jumper, in combination with several triggered effects, could have led to the simulation being extremely cluttered and "overloaded" with content. According to the insight I gathered during operating the *Jump Cube* on several exhibitions, the experience of a virtual parachute jump and the flight over parts of the city of Vienna would still be exciting, enjoyable and offer the player a view over the city, which one usually would not get. Effects like explosions obstructing the view or flashes of lightning accompanied by thunder sounds could therefore easily be distracting or may in some cases even annoy the person in VR.

The next block of research questions regarded the interaction between the jumper and the person operating the external interaction console (number 5 and 6). In order to allow at least unilateral non-verbal communication between the two participants, I needed to design my setup in a way, that would allow the external user to see what the person in VR was seeing through their headset, and also how he or she reacted to it. An interesting point of observation was if a possible loud or shocked reaction from the jumper would pursue the controller to trigger the effect in question more or less often. These kinds of interactions would of course also inevitably lead to the manifestation of some kind of power dynamic since one person was literally more "in control" than the other. This dynamic would build up even more, when the participants would switch places after the first jump. In order to facilitate this, and at the same time reduce the number or participants per test-date in order to increase *COVID-19* safety (more on that later), I decided to make the two participants switch roles after the first jump. This way the likelihood of the formation of an entertaining experience for both participants was much higher since the person in VR might seek some form of "revenge" if he or she perceived the triggered effects to be annoying or disturbing.

One of my previously stated research questions concerned the use of the interaction console and the different version of it (question 7). As already outlined in the introduction and the definition of the research questions, I planned to create at least two types of interfaces available for my interaction console. One using a traditional button user interface and one utilizing the *Leap Motion Controller* based on hand movement. In order to realize this, I had to create two different applications, or at least frontends, for the console part of my setup. In order to make this as easy as possible, I choose to establish

a simple network-based communication channel between the VR simulation and the console application. This would not only allow easy switching between the two variants, but theoretically also allow using more than one interaction console simultaneously.

The last question remaining, number 8, was about the entertainment-factor of a setup like this. With the current complexity it would hardly be feasible for any commercial application, but could maybe still be an interesting and enjoyable experience for events or in arcade-like settings. The answer to that question would summarize all the implementation efforts above and give me an indication of how enjoyable and how accessible the experience was for the participants.

All these requirements defined above regarding the soft- and hardware parts of my test setup led to the features that needed to be implemented during the development phase of my thesis. Since most of the hardware part was concerning the actual VR simulation which was already up and running, I was able to concentrate my efforts mainly on the software side. In summary, I needed to take care of the following points:

1. Implementation of several different types of effects,
2. including at least one "obviously triggered" and one more subtle one.
3. At least one of the effects should be highly distracting in order to evaluate its influence of the perception of the content.
4. The controller has to be able to see what the person in VR sees and also his or her reaction to the triggered effects.
5. Ideally, both participants should switch roles to facilitate group- and power-dynamic interactions.
6. The remote interaction console should at least offer two different types of user interface for the operator to choose from.

With all research questions formulated into concrete requirements to the hardware setup and necessary software implementations, I could now proceed with the actual work on creating the project. I needed to create an application for the interaction console and modify and add more effects to the existing *Virtual Jump* VR simulation.

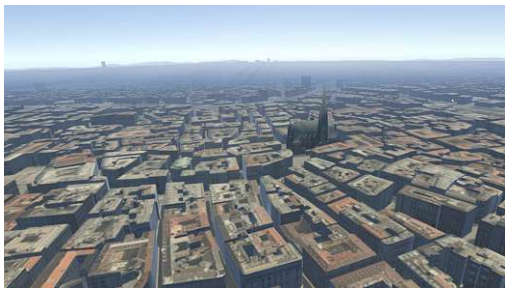
3.1.1 The *Virtual Jump* Simulator

The *Virtual Jump* simulator was originally created for the 200-year anniversary of the *Technical University of Vienna* (*TU Wien*) within the context of a research project [EM15]. The original content allowed a person to perform a virtual parachute jump out of an airplane over Vienna. To accompany this virtual experience and to create a highly immersive experience, the user did not only perform this "jump" virtually but also physically. For this, the user or player was equipped with a specially designed,

tailor-made harness that was hooked up to a steel frame construction (the *Jump Cube*) by a variety of ropes. This unique setup allowed the user to physically jump off a small pedestal and ended up suspended in mid-air inside the *Jump Cube* apparatus. By using a specially created adaptive counterweight assembly, the system was able to deal with a vast range of people with different weights and sizes. The setup also contained a traditional desktop monitor, which was positioned inside the *Jump Cube* and was connected to the computer running the simulation. This screen was visible for the operators, as well as for the audience and showed what the person in VR was currently seeing in real-time. This allowed monitoring the player as well as the current state of the simulation and simultaneously helped to engage and entertain external bystanders.

As soon as the player was suspended in mid-air, air was blown from beneath him or her to simulate wind that would, in reality, be experienced during an actual jump when falling towards the ground. The *Jump Cube* also included a water sprayer system, that allowed a fine mist of water droplets to be injected into the airway coming from installed fans. The purpose of this implementation was to physically complement the experience of falling through (virtual) rain clouds by adding the sensation of moisture or wet air. After a predefined falling time, the virtual parachute opened and the person suspended in the ropes was pulled up to transfer from a horizontal to a more up-right position. This was done to simulate hanging under a real, opened parachute while slowly descending back down to the ground.

(a) Screenshot while looking out of the virtual airplane.



(b) Screenshot while flying over the city of Vienna.

Figure 3.1: Two screenshots from the *ECR 2017* version of the *Virtual Jump* content [EM15].

Although not initially part of the *Virtual Jump* simulator, later on, a system for scent dissipation was added. This system allowed to activate one of six pre-defined scents at any time during a simulation. In the case of the parachute jump, there was for example a gasoline scent that was activated after a small propeller airplane passed by the "jumper" on his or her way down.

In the original *Virtual Jump* content, the player finally landed at the *TU Wien* and was greeted by virtual representatives of the university. On the way towards the ground, the player passed several objects, including images showing historical milestones from the

history of *TU Wien*. This trip through the university's history was also accompanied by a narration.

This content was later adapted to be shown at the *European Congress for Radiology* (*ECR*), where the landing site was changed to the *Vienna International Center* (*VIC*), where the congress was held. In this version, the history of *TU Wien* was also replaced by the history of radiology and the virtual greeting at the end of the simulation was changed to be given by representatives of the *ECR*.

Over time, in addition to these purely passive jump experiences, a variety of other simulations were created, including very entertaining and interactive simulations like air-races and missions in outer space. Each of these contents was tested and enjoyed by hundreds of people during many occasions when the *Jump Cube* was on display at exhibitions and congresses.

As foundation for my project, following the requirements outlined in 3.1, I needed a well-tested, immersive simulation that players could immerse themselves into. Since my testing did not require any interaction on behalf of the player, I concluded that using a purely passive content would be the best option. Following this line of argumentation, I decided to use the original, entirely passive *Virtual Jump* content as foundation for the VR simulation to be used in my project. Choosing this content allowed the user (the "jumper") to concentrate on the simulation itself and on possibly triggered effects, since it did not require any active action, like steering, to be performed on the jumper's behalf. Since the initial content where the user landed at *TU Wien* was rather short, I decided to use the version created for the *ECR*. In this scenario, the player jumped out of the plane over the university, but as soon as the parachute opened, the player floated not directly down to the ground, but was carried over to the VIC. This extended simulation time was a positive advantage for my use case, since it gave the user handling the external interaction console, more time to interact with the player and the simulation. Since the history of radiology was of no concern for my project, I removed the audio narration as well as the accompanying images completely, in order to avoid any distractions or interferences.

By building up on that specific simulation, I was confronted with some peculiarities that had to be dealt with. Although the *ECR* content from 2017 was newer than the original *Virtual Jump*, it was still based on an older *Unity* version. Due to various reasons which will be discussed later, it was necessary to raise the project to a newer version, which opened up some challenges or even problems that needed to be dealt with.

For the interaction console I needed a completely separate system, that would still be able to interact with the running simulation. Since the VR player's view was already visible from the outside via a small desktop monitor, I just needed to create an interface for the console operator to trigger the desired effects. My intention was to create both a traditional button-based interface, as well as a hand gesture-based input system with the help of the *Leap Motion Controller* [Ult], which is able to track hand and even finger movements in real-time. Concerning the choice of which framework or platform to use

for this part of the project, the decision very straight forward. As the VR simulation was using the *Unity* engine, establishing a connection between two applications within the same engine and framework, should be relatively straightforward. Even more so, as *Unity* offers novice users to easily create multiplayer games without any in depth knowledge of the underlying technology. Another key advantage was that *Leap*, the manufacturer of the hand-tracking device I wanted to use as user input, released a development kit for the *Unity* platform which I could use. And last but not least, *Unity* also makes prototyping as well as implementing basic simulations and interactions relatively easy, by providing the user with a graphical editor. This would reduce the necessary time for development, since only the custom logic and other, more specialized content had to be implemented in code. The versatility of the *Unity* engine also theoretically allowed me to deploy my project for a variety of operating systems, including *Windows* and *Android*. Since I intended to allow triggering actions not only from a desktop computer but from handheld, touch-enabled devices too, this system was an ideal choice for my project.

3.2 Implementation and Challenges

At the beginning of my development process the choice of the appropriate tools and frameworks had to be made. As previously described, the decision to use the *Unity* game engine as the foundation for my project was already set. The next decision was to find out which version was best to use in my situation.

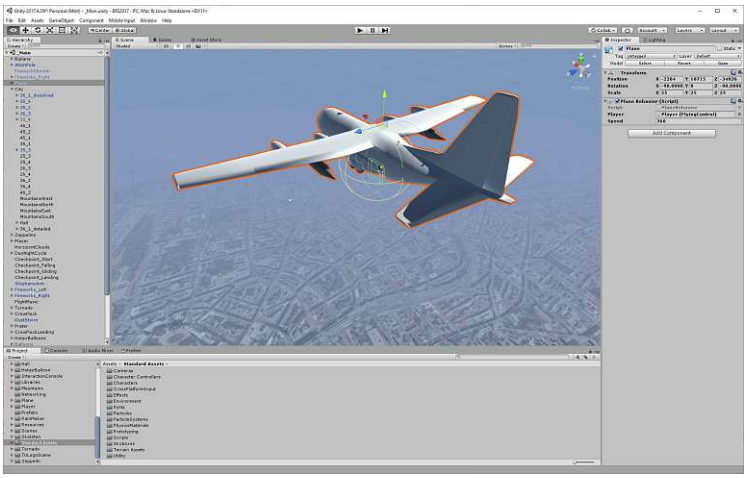


Figure 3.2: Original *Virtual Jump* content opened in the *Unity* editor.

Although the latest version of the *ECR* was newer than the original *Virtual Jump* simulation since that was created even earlier, *ECR 2017* still used a *Unity* version from 2017. The content was built for *Unity* v5.5.1, which was already quite dated when I started my thesis. For all I knew, this content could either have been created from scratch or be based on the original, even older, project files from the initial *Virtual Jump* content, that had just been lifted to a newer version. Since I experienced the *Jump Cube* in action

with the *ECR* content first hand on several occasions during exhibitions, I could be sure that there were no major bugs or problems with it. This way I was confident enough that it would be a safe choice for basing my project on.

My setup did not only consist of the VR simulation, but also of a completely separate application for the remote interaction console. Since the whole concept of externally controlled or triggered actions relies on these applications interacting with each other, establishing a robust and reliable communication was key in order to meet the requirements defined in 3.1. Since the VR simulation content I intended to use was created with the *Unity* game engine, I decided to use the same platform to build my console application. During my studies, I already worked with the *Unity* system, so I knew this choice would allow me to create my project in a reasonable amount of time. In addition to that, using *Unity* for both applications would also make communication between the applications easier. *Unity* is often used to create simple multiplayer games and has many pre-built networking elements already integrated. Since it is so widely used, there are also a number of online resources that I could use for research and leverage for debugging problems that might occur.

Figure 3.3: The *Leap Motion Controller* next to its original packaging.

One of the first steps was to make myself familiar with the software development kit (SDK) for the *Leap Motion Controller* to learn more about possible requirements and suggested ways of implementation. The *Leap Motion Controller* is a small device, that allows the user to interact with a computer system by using hand gestures. In order to track a user's hands, the device would either be mounted vertically on the frontside of an HMD or be placed flat on a surface (e.g. a table) in front of the user. In my setup I would place the *Leap Motion Controller* on the table in front of the person operating the interaction console. This way, I was able to offer users a second type of interface by allowing them to trigger actions via moving their hands rather than clicking or touching buttons.

Initially, I intended to use *Unity*'s built-in networking function for establishing a connec-

tion between both applications. In order for this to work, both instances had to be based on the same *Unity* version. After encountering various problems when trying to raise the *ECR* 2017 *Virtual Jump* project to the most up-to-date version, I had to establish a "middle ground". I needed to find a *Unity* version where possible occurring problems caused by the version lift would be fixable in a reasonable amount of time but which also worked with one of the SDKs released by *Leap* for their *Motion Controller*. After extensive testing, I settled for the *Unity* game engine version 2017.4.35 and the *Leap Motion Developer Kit* v4.4.0.

After the *Unity* versions were set, my next step was to establish a connection between the two applications I needed to work with: the actual VR simulation on one side and the remote interaction console on the other. In order to be able to meet all requirements defined earlier, establishing a network-based connection between the two programs at play was the most efficient solution. Initially, I intended to make use of *Unity*'s native networking implementation – *Unity Networking* (*UNet*) here, but unfortunately, this turned out not to be as feasible as I hoped.

The networking implementation in *Unity* seems to be primarily focused on being used in multiplayer games, which makes sense, given the popularity of that game type. It allows every client to spawn their own "player" (in form of a so-called "prefab") and transmit its movements and actions over the network and a server to the other clients. One key element here is, that the spawned object and by extension also all other objects it should interact with, has to be present in every client's local scene. This is, generally speaking, the case with typical multiplayer games, where all instances basically run the same application, containing the same assets that can be loaded if necessary. Each client therefore has its own "player" avatar as well as locally spawned representations of remote players that can replicate movement and actions of their remote counterparts.

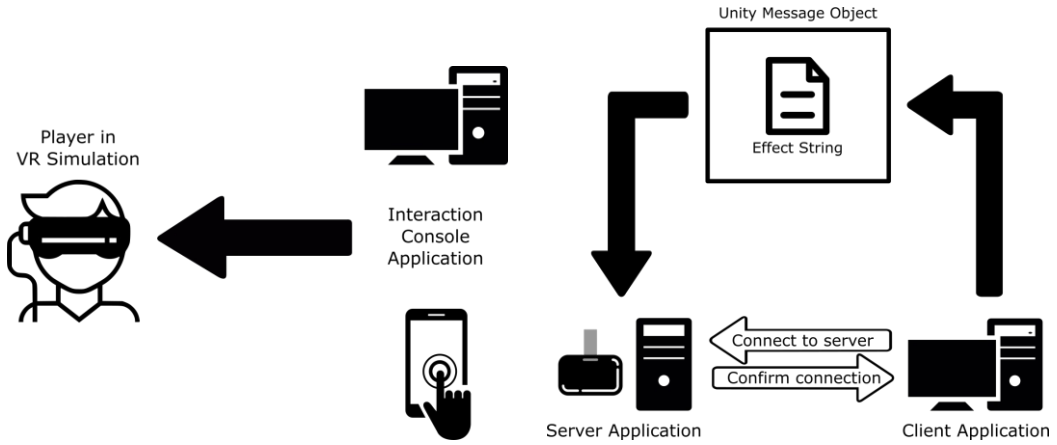
This is a very effective and convenient way to allow creation of standardized multiplayer games quickly and easily, without the need for deeper knowledge of networking fundamentals on the game creator's side. Due to the submission of all movements and actions in real-time, this kind of networked synchronization is rather bandwidth intensive. For my use case, this was completely unnecessary, since I only required a single "trigger" impulse to be sent and had no need for a constant real-time synchronization of any kind. While using *Unity*'s native networking capabilities would not have been a very efficient solution for my use case, it would theoretically still be possible, if it wasn't for the need of "identical" client scenes.

In my case, simulation assets were only present in one part of the linked applications: the VR simulation. The remote console did not have any and could therefore also not spawn them correctly on the VR simulation's side. First, I tried to solve this with empty dummy objects, which rudimentary provided elements and methods of the actual objects to spawn inside the simulations, but ran into further problems when trying to trigger connected sound-effects and particle systems.

After researching the capabilities and working requirement of *Unity*'s native networking

implementation, it became obvious, that it was not sufficiently suitable for my case. In contrast to the real-time synchronization of identical clients which is commonly found in multiplayer gaming, I needed a more message-based approach for my project. Considering my requirements from 3.1, there was no need for a bidirectional communication between the applications at all. In order for the console operator to observe the jumper, he or she had to be positioned in viewing distance of the *Jump Cube*, thus also allowing the operator to look at the live view shown on the desktop monitor connected to the simulation host. This way, the operator of the remote console could closely observe what the jumper was currently seeing or looking at, without the need for implementing a live video feed into the interaction console's interface. Without such a video stream, an unidirectional communication channel would satisfy my requirements and was also the most efficient approach to take. Establishing a custom, message-based communication between my applications would also result in much less network overhead, since all I really needed was to send and receive a simple message once an effect was triggered.

The console would only be required to send messages while the server only needed to receive and decode them. The minimal approach that needed to be implemented would be sending a message containing only an integer value, representing the effect that had to be triggered. Instead of using this minimalistic approach, that could possibly lead to the wrong effects being triggered if e.g., the number of the desired effect was not mapped correctly, I decided to use a string-based message content. This way the risk of misidentification was reduced to almost zero and the overhead in comparison to using an integer was neglectable. Since this approach perfectly covered my needs for triggering effects and was also remarkably efficient, I decided to implement the communication between my applications this way.



(a) Setup of VR simulation and interaction console.

(b) Client-Server comunication between applications.

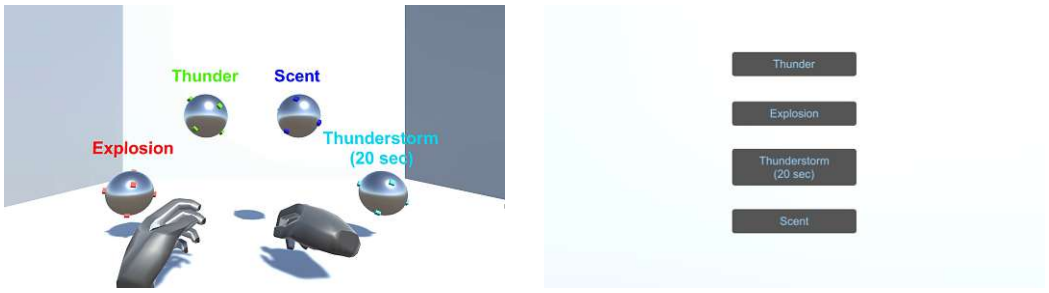
Figure 3.4: Schematic diagrams showing the interaction between VR simulation and remote interaction console.

After being fully implemented, my VR simulations took the role of a server which listened for any incoming commands on a pre-defined TCP port. The interaction console was connected to the "server" before the simulation was started and could send message-based commands to the VR simulation. For simplicity's sake and to keep versatility high, the messages I used were simple strings that where encapsulated in dedicated *Unity* objects and sent through the *Unity Network Manager*. For each action triggered, the remote console sent a predefined string which was received and read by the VR simulation that, in turn, then spawned, for example, the requested explosion. This simple and yet very effective and robust implementation of network communication allowed me to change the way triggers worked on the console side at will, without having to make any adaptions to the message receiving end in the VR simulation.

3.2.1 Implementation of the Interaction Console

As stated in my requirements at the beginning of this chapter, I already knew I needed at least two different versions of my remote console application for the user to choose from. Before going into more detail on the differences between the implementations of the user interfaces however, I wanted to make sure one crucial key component would be working properly. Whatever the interface was, each had to establish a connection to the server running the VR simulation and had to be able to send commands to it. Since I needed some rudimentary form of user interface to test sending trigger messages anyway, it was only logical to start with the simpler one of the planned interfaces.

Buttons provide a simple way of triggering effects and are familiar to almost every user. They can be used both in desktop applications, where they are clicked by using the mouse, but also on touchscreen interfaces where they are simply tapped with the finger or a special pen. Since I wanted to be able to use both these ways of interacting with the buttons on my interaction console, I made sure that the *Unity* project was building correctly for both *Windows* and *Android* and that it would run and work without any problems on both of them.



(a) Hand gesture-based user interface. (b) Traditional button-based user interface.

Figure 3.5: Screenshots from the two different versions of the remote interaction console user interfaces.

For the actual user interface, I used a simple canvas that showed 4 buttons to the user

- one for each action. While most of the actions that could be triggered were almost instant and did not last very long, others, once activated, lasted for several seconds. In order to avoid them being triggered over and over again while they were still active, I had to implement a way of blocking the trigger for a given amount of time. I decided to equip each effect button with a "cooldown"-timer representing the duration of the triggered effect and disabled the button for this timespan by using a timer. This way the corresponding button would be blocked for as long as the triggered effect lasted and could only be clicked again afterwards. To visualize this for the user, I did not only disable the functionality of the button but also shifted the color to a darker grey, indicating the current unavailability of this effect. After the pre-defined cooldown time the color of the button changed back, the functionality was restored and the effect could be triggered again.

The second version of my interaction console was meant to be controlled by hand gestures and/or movement. In order to achieve this, I used the *Leap Motion Controller*, that uses an infrared tracking system and is able to recognize the position of one or two hands with decent precision. Since the device was not only capable of tracking user's hands but in many cases even managed to track each finger individually, I originally wanted to implement a gesture-based interaction system. My intention was to allow the user to trigger effects by performing pre-defined movements or gestures with his or her hands and fingers.

Unfortunately, during my initial testing, I soon realized that tracking accuracy and robustness was not on the level I had initially hoped for. Although the position of the hands was tracked reasonably well, their orientation was not always correct. This was especially true when only one hand was in the tracking area instead of both being "visible" for the device. The system often flipped the tracked hand-model around, thus completely switching which physical finger was mapped to which of their virtual counterparts. I generally found that although the *Leap Motion Controller* had two operating modes to choose from (mounted on an HMD and tabletop usage), that the first one seemed to be used as default and probably also as a fallback. Although the device could be switched to tabletop mode in the settings, it often assumed to "see" the back of the user's hands, rather than the insides of them. Due to that major orientation- and consequently also tracking issue, accompanied by some problems under different lighting conditions, the tabletop tracking mode I wanted to use did not nearly work as well and as reliably as I had initially hoped for.

After trying to improve tracking precision for a while with limited success, I decided to abandon the gesture-based interaction model and moved to one more suited for the tracking accuracy I had to work with. Since hand tracking worked quite well as long as the hands remained reasonably static or moved as a whole, I decided to track them as a whole. I created virtual spheres that had to be touched, hit or even slapped with the user's virtual hands. Since the *Leap* SDK already included some abstract hand models as virtual representation of their physical counterparts, I decided to use those in order to visualize hand movement and tracking for the user.

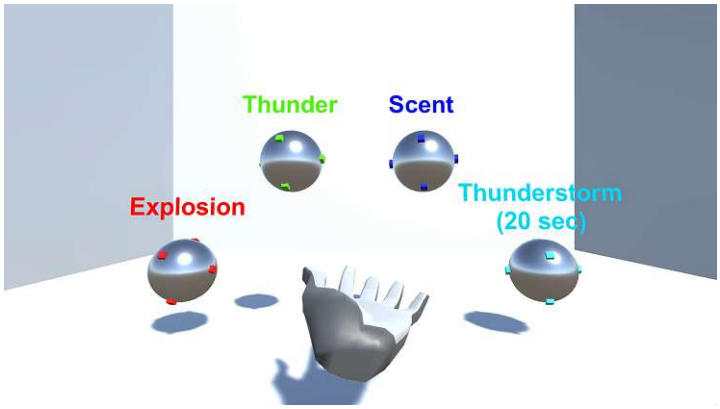


Figure 3.6: Incorrect tracking showing an inverted hand position.

My first idea for implementing the interaction with the "trigger spheres" was to position the player inside a cube with the spheres bouncing around. This way, the user would have to try and "catch" them in order to trigger an effect. Although this would certainly have made a nice mini-game, it would most likely also distract the operator of the interaction console from their actual task; to intentionally trigger effects in the VR simulation. Since my goal was to evaluate the impact of humanly controlled outside interaction on immersion, and if possible, also any kind of manifesting power-shift between the two test subjects, I decided to position the spheres right in front of the user on fixed coordinates. This would eliminate the distracting "game" element on the console side and would also make triggering effects easier for users that were not used to this kind of motion interface.

Inspired by one of the SDK's sample projects, I decided to use reflective spheres with a metal-like surface, accompanied by colored markers visible on their outside. Although the spheres' positions were fixed, they could still spin around freely in order to let the user know he or she was interacted with them successfully. The colored markers not only helped to make the spinning more clearly visible, it also allowed me to color code the effects for better association with the effect name. Each sphere had the name of the effect that would be triggered shown hovering above it. As the text always was the same color as the corresponding's sphere's markers, this allowed the user to intuitively associate the sphere with the effect he or she wanted to trigger. As already discussed during the description of the button interface, I also needed to implement and visualize a cooldown functionality here. In order to do this, I stopped the sphere's rotation and switched the color of its markers and of the corresponding text to grey when the effect was triggered and the corresponding sphere was disabled.

The connection to the server application, in this case the VR simulation, was established manually after initially starting the application. In the upper left corner, the user was presented with a simple text input field for the IP address of the computer hosting the simulation. Right above it, there was a simple "Connect" button which established the connection to the VR simulation. As soon as the link was successfully created, both user

interface elements where hidden since they were no longer necessary for operating the console.

(a) Client side of the connection interface (*Leap* version).

(b) Server side of the connection interface upon application start.

Figure 3.7: Cropped screenshots of client and server application windows, showing the network connection user interface.

The intended use case of my setup was for one VR simulation to always be paired up with exactly one instance of the remote interaction console application. Since the remote console did only trigger effects and was, strictly speaking, not required for the VR content to run, the content could of course still be used without an interaction console connected, but as the console application was missing, no additional effect could be triggered.

Although not intended initially, the highly scaleable client-server architecture of the connection between the two components made it possible to connect more than one interaction console to one and the same VR simulation. In order to keep versatility as high as possible from the start, I tested this scenario and ensured it would work without any major problems. But since this was not the use case I created this setup for, there needed to be some limitations to the interface, more precisely to the visualization of the effect duration when running it this way.

For simplicity's sake and to keep the transferred data low, the timer for the visualization of the button and trigger-sphere cooldown was directly implemented in the interaction console application. In order to avoid an effect being triggered shortly after one another by using a second console application, I also implemented a simple check if the effect was currently being executed in the VR simulation. As long as the requested effect was active the simulation would not respond to a message that would otherwise trigger the desired effect.

Due to the missing backchannel however, the second interaction console would not "know" if the effect was actually triggered or if it was currently blocked due to it already being executed. And if the effect was already active, the remote console was not informed of how much time was left for it to finish. Without that information, the second console just started its local cooldown timer and blocked out its local trigger for the full duration

of the pre-set effect cooldown. This would of course not represent the actual status of the effect in the VR simulation, but likely run out after the effect had already finished.

Although I tested this during development, I decided to not put more work into that feature, since connecting several console applications did work if desired during my tests (e.g. one PC with the *Leap Motion Controller* and an additional tablet computer with the button interface), but was not part of my intended test setup.

3.2.2 Extending the Original *Virtual Jump* Content

While the remote interaction console served as the client, the actual VR simulation took the role of a server hosting the simulation. The user interface for establishing a connection was, just like with the client application for the interaction console, reduced to a minimum and made as simple as possible. Since no server address had to be entered here, the only user interface element was a single button that was shown in the upper left corner to start the server. As soon as the server was started, the button was removed from the interface, no matter if a client was connected or not.

The simulation received messages from the remote interaction console as simple string messages, wrapped in a *NetworkMessage* class. Depending on the message's content, the string sent by the remote interaction console, the implementation on the receiving side (the VR simulation) started the requested effect. When a command was received, the simulation first checked if the requested effect was currently active or not. If it was indeed currently active, the received message was simply discarded, otherwise the corresponding prefab would be instantiated and the effect would start.

Implemented Effects

After the basic setup, including a working communication channel and triggering logic, was created, effects had to be implemented in the VR simulation. As defined in the requirements in 3.1, I needed at least one moderate effect, which would immediately be recognizable as being triggered and possibly also be very distracting. Since I wanted to provide the console operator with more than just the mere two effects I needed as per my requirement to choose from, I decided to create more than just a single one of this type.

The first effect I implemented was thunder and lightning. Its integration was quite easy, since the *Virtual Jump* content already used these effects during the player's "fall" right after he or she jumped out of the airplane. In order to create a triggerable effect, I just needed to create a prefab containing the effect, that would be spawned upon receiving the appropriate trigger message from a client. The existing implementation of the lightning was just a flat, semi-transparent texture the jumper originally fell through, accompanied by the sound of thunder. For simplicity's sake, I decided to first just position the existing lightning effect directly in the player's viewport and refine the effect later on. Just as the visual effect, the thunder sound was directly attached to the player's avatar and played back to the jumper's headphones without any additional 3D effect.

The second effect was an explosion that was spawned at a randomly chosen position within a bounding box around the player. Since the player was moving throughout the whole simulation with varying speeds and looking in various directions, I decided to randomize the precise position of the explosions in order to keep them as unpredictable as possible. This way, I was able to avoid explosions always spawning at a fixed location in relation to the player, but was also able to keep it near him or her in order to have a highly surprising and distracting impact. Randomizing the explosion's position this way, could of course lead to explosions a few meters away to the side, explosions occurring behind or above the player or just make them appear right in front of him or her, leading the player to fall or fly right through it.

In order to achieve a realistic explosion effect, I decided to use the free *Detonator* explosion framework from the *Unity* store [unia]. It contained several different kinds of explosive effects of various intensity. I picked one of them and integrated it into my explosion prefab. In order to keep the effect contained to visual and audio content only, I removed the pre-existing collision and force components that were part of the asset. These were part of every explosion in the asset pack and were responsible for moving elements in the scene away from the explosion. The inclusion of these functionality aimed to simulate the physical force of a real explosion and allowed the user to achieve realistic effects. A typical example for this would be crates or other loose objects that were located in close proximity to the explosion, that would then be pushed away from it. Since I had no need for this force simulation in my use case and it would, if applied to the player, even push him or her off course, I decided to remove these components altogether.

(a) Lightning effect. (b) Explosion effect.

Figure 3.8: Two of the triggerable effects as seen from the jumper inside the VR simulation.

As stated in my requirements, I did not only need moderate effects, but also some more subtle ones that could possibly even go unnoticed for the player. The effect I wanted to implement to represent this "simple" category was the deployment of scents. Due to the inherent delay in the perception of scents by the human olfactory system, they can, if they are subtle enough, easily be perceived unconsciously and might not be directly associated with an externally triggered effect.

Conveniently, although it was not part of the original *Virtual Jump* simulation scent had been added later on and was already part of the content version I was using (*ECR*

2017). The scent deployment system was dubbed *Vragrancer* [Eid18] and worked with a physical magazine of pre-filled scents that would be dissipated into a constant flow of air that was passing by the player's nose through a little tube with an opening that was attached to the HMD. Since these magazines held up to six different scents, I initially intended to allow triggering different scents separately, which would, when triggered shortly after each other, also allow mixing them to a certain degree. Unfortunately, due to time constraints brought about by the *COVID-19* pandemic, which will be discussed later, I had to stick with my initial implementation of only a single scent. The scent I ended up using for my user study was part of a cartridge, originally intended for use with another content and was called "Seebrise" (which translates to "Sea Breeze"). The scent was rather intense but not unpleasant and could possibly be described as fruity and lemon-like. Due to the intensity of the scent used, it might, arguable, belong more to the "moderate" effects after all, since it was definitely perceived as triggered when it was used too often. As with every other effect, I assigned a short cooldown after triggering the deployment of the scent which was set to 2 seconds. But when the effect was triggered too often, it quickly became very distracting.

Lastly, I also added a complex, longer lasting effect that could possibly, in some cases, even be perceived as part of the original simulation, depending on the time it was triggered. It was a thunderstorm which was a combination of rain accompanied by a variety of lightning strikes and thunder sounds. For the rain effect, I resorted to using another free *Unity* asset pack called *Rain Maker* [unib]. This not only contained realistic sound effect for rain but also created convincing visual effects looking like falling raindrops. Upon triggering this complex effect, I spawned the rain prefab and set it to last for a given timespan that would be calculated from the number of lightning strikes and the time between them.

Thunder and lightning I used were the same as the ones previously described, with the only difference being, that in this case, there would be more of them. I set the number of lightning strikes for a thunderstorm to 5 and implemented them to be triggered with a random amount of time in between. After experiencing some edge cases during testing, where two or even tree lightning strikes occurred directly after one another, I decided to implement a minimal amount of time that had to pass between them. To this minimum of 2 seconds, a randomly chosen amount of time was added before triggering the next lightning strike, in order to keep the effect as unpredictable as possible. As a whole, the thunderstorm lasted for a duration of 15 to 20 seconds. Since I had to define fixed duration for the corresponding button or trigger-sphere cooldown on the client side in the interaction console, I set this to the maximum duration of 20 seconds.

3.3 Impact of *COVID-19* on Time Schedule and Project

At the beginning of 2020, the *COVID-19* pandemic reached Austria and a lot of preventive measures had to be taken by the government to keep it from spreading further. During the summer, the situation cleared up a little and infections went down again. Nevertheless,

it soon became apparent that there would be a second wave of infections, possibly again accompanied by a hard lockdown and the recommendation not to leave home if not necessary.

Unfortunately, during my test jumps for the user study I could hardly avoid a certain degree of proximity and in addition to that, also the VR headset had to be passed around between subjects taking the virtual jump. Of course, a higher number of infections occurring in the population would increase the possibility of one of my participants being infected and a country-wide lockdown would have made all test jumps needed for my user study out-right impossible. Consequently, in order to keep the risk of exposure to the virus for my test subjects as low as possible and before the country-wide situation would take a turn for the worse, I had to carry my study out as quickly as possible.

Since my project was almost finished by the end of June 2020, following the recommendation of my supervisor, I started my user study although the applications were not completely finalized yet. In order to perform the test jumps as soon as possible, I had to use the current project version which meant I had to make some compromises regarding content, functionality and minor flaws in the user interface.

The first concession affected the number of scents that could be triggered. As already mentioned, I initially intended to include several different scents to choose from. My intention was to make each one of them triggerable individually and possibly even do not set a general cooldown for effects of this category at all, but only one for each scent individually. Due to the delay in the perception of scents and them sometimes "hovering around" for a few seconds, triggering several scents simultaneously or closely after another, would have given the console operator the ability to mix several scents to a certain degree. Since with this approach a wide variety of combinations would have been possible, the palette of olfactory stimuli would have been enriched tremendously and would have been diversified far beyond only a small number of pre-defined scents. Although this would certainly have made the effect more unpredictable and enriched the experience for both the player and the controller a lot, for the purpose of my thesis a single scent would still serve as a good representation of an externally triggered, olfactory event.

Another possible point of critique might be the implementation of the lightning strike. In fact, this was also a point several subjects of the user study mentioned as part of their feedback in the questionnaire. Since this effect was the first I implemented, it initially only served as simple "proof of concept" for the effect, which was scheduled to be refined later on. In its initial implementation, the static, semi-transparent image of a lightning was just blended in and faded out, directly attached as full-screen overlay to the player's viewport. This way of implementing an effect resulted in the image moving with the viewport when the player moved his or her head since it was statically fixed to it. As a lightning would not move along with you when you turn your head in real life, this was of course very unnatural and probably also immersion breaking. Unfortunately, this implementation stayed in my final version of the project for the user study and was never replaced. Although I was aware that this could possibly have a negative impact

on immersion for the jumpers, it was still a reasonable concession to make in order to ensure carrying out the user study as early and therefore as safely as possible.

One feature that was still intended to be included in the interface of the remote console, was a visualization of the cooldown duration of the different effects. While this might not make a relevant difference for shorter effects like the explosion with a cooldown of just three seconds, it would certainly be beneficial when using longer-lasting effects like the thunderstorm. In these cases, the console operator did not really know when the effect would become available again and just had to wait until the button or sphere became active again. My original intention was to visualize this timer with a simple progress bar, that would linearly count down the remaining seconds, thus providing an easily readable and intuitive feedback of the current state of the effect. Since this feature would have been part of the remote interaction console and not the VR simulation, the evaluation of immersion for the person in VR was not affected here. Still, it would certainly have improved the remote console's usability, made the interface more intuitive by providing more information in a non-intrusive way and would thereby have increased the quality of the operator's experience significantly.

The fourth and last adaption that had to be made due to the rapidly closing timeframe for my user study to be carried out, was the number of participants. In my original proposal, I intended to conduct my tests with 20-25 people, which was hardly viable given the circumstances under which my study was actually carried out. In addition to reducing the number or test subjects, I also made a change to the process of assigning roles to the participants. I decided to work with two people at a time and opted for making them switch places for the second run. This way I was able to reduce contacts between participants and contain possible infections to only a small number of people instead of potentially risking a wider spread. In addition to that, I also gained the opportunity to observe and evaluate possible effects of small-scale group dynamics between the two users and in some cases observe the use or maybe even abuse of power in conjunction with switching roles.

Bearing these concessions to the completeness of my software implementation as well as the amount and extent of features in mind, the setup I created was not as complete and refined as I would have liked it to be, but still met all requirements defined in 3.1 perfectly. With my final software implementation in place and a precise plan of what hardware setup was necessary in order to conduct my user study, I was able to carry out the necessary test jumps.

CHAPTER 4

Evaluation

After the implementation phase was completed, the next step was defining the methodology and procedure for carrying out the user study. In order to evaluate the impact of humanly-controlled outside interaction on immersion, I needed to perform a series of test jumps with voluntary test subjects. Afterwards, they had to fill out a questionnaire containing carefully formulated questions related to the topic I cover in my thesis. The answers given by my test subjects will then be aggregated and statistically evaluated in order to draw some conclusions which should answer my research questions as defined in section 1.1.

4.1 Setup and Testing Procedure

Each test had to be performed by at least two people at once. Both these persons had fundamentally different roles which had not only influence on what they would do during the test, but also which questionnaire they had to fill out.

- The *jumper* was the person in VR, performing the virtual parachute jump.
- The *controller* was the one operating the remote interaction console.

The part of the jumper was purely passive and did not require any active interaction with the simulation or the test setup as a whole. From the jumper's point of view, this was basically identical with an traditional experience of the *Virtual Jump* simulator with one of the before-mentioned, passive contents.

In contrast to that, the role of the controller included much more interaction since this person could trigger as many effects as he or she wanted without it having any impact on their part of the experiment.

Since these two roles were so fundamentally different, I created two entirely separate questionnaires, which contained relevant questions for each of them. After the test jump was performed, both test subjects were handed the corresponding questionnaire which they had to fill out to the best of their knowledge. After the first jump, after both participants filled out their questionnaires, they switched places and each of them took the role of their counterpart.

In the traditional *Virtual Jump* setup, the jumper would have been strapped into a tailor-made harness which would have been connected to the *Jump Cube*'s frame via ropes and an adaptive counterweight system [EM15]. Due to a variety of reasons, this was not the case during my test jumps. For reasons described later, I opted for replicating the experience as closely as possible without the need for additional personnel or participants working in close proximity to each other while putting on the harness and strapping the jumper into the assembly. Instead, a chair and a detached stepping-pedestal were placed in the center of the *Jump Cube*, which the jumper laid down on. This way, the position was almost identical to the one the person reached during the "falling" part of the jump, while being suspended in the ropes but improved the simplicity and practicality of getting in and out of this role and position significantly.

The remote interaction console application was running on a separate computer which was positioned on a table next to the *Jump Cube*. The computers were connected via a common Ethernet cable which handled the communication between the applications. The operator of the interaction console used either the mouse or the *Leap Motion Controller*, depending on the interface that was chosen.

(a) Setup for test jumps: remote interaction console (front), simulation PC (back).

(b) Jumping area of the *Jump Cube* with stepping-pedestal and VR headset.

Figure 4.1: Images of the setup used to perform the test jumps for the user study.

Originally, I intended to let the test subjects choose which interface they wanted to use for the remote interaction console. At the beginning of my user study, I realized that due to using a much smaller test group than initially planned, it would be better to mostly define the used interface myself in order to avoid one of them not being used at all.

4.2 Questionnaire

The two questionnaires I created had some more general parts in common, but of course contained different questions regarding the role the test subject fulfilled (jumper or controller). Both questionnaires can be found in the appendix to this thesis. Since, in my case, all participants spoke German, I decided to create the questionnaire in German, to eliminate possible misunderstandings or misinterpretations.

The general part included some statistical questions like gender or age. In my case, I used three checkboxes for the gender (male, female, other) and a free number field for giving their age. Next, there were two questions concerning previous experience of the person with VR and the experience with the feeling of immersion. The answers to these questions could help me to eliminate possible bias and also potentially help me to better understand responses to later questions. Since some of them concerned subjective feelings and comfortability of being in VR, these questions would possibly be answered differently, whether a person has experienced VR a lot or not at all. After that, there was a question whether the participant liked the simulation or not. The aim of this was to see, if one role would be generally more enjoyable than the other one. This was then followed by two checkboxes representing the two roles a participant could take, of which of course always exactly one should be selected. This was added to show the person filling out the form that the following questions would be role-specific, as well as to simplify distinguishing the forms during processing. The last question, which both versions of the questionnaire had in common, was whether the person felt comfortable in his or her role on a scale from -2 to +2. The answers to this question would of course have very different meanings for persons being in VR or operating the remote interaction console. Nevertheless, I chose to formulate this question as openly as possible in order to pose an easy entry to the role-specific part of the questionnaire.

The first role-dependent question for the controller was whether he or she felt “in control” of the situation (on a scale from 0 to 4). The reason for this question was to be able to evaluate possible power- and group-dynamic effects occurring between the two participants.

Next, I asked for the number of effects that were triggered. The answer I expected was the number of totally triggered events in order to compare this to the number of effects perceived by the jumper. Interestingly, during writing the question, the wording seemed quite clear to me, however I realized in the course of the user study that in reality it was sightly ambiguous. This did not pose any kind of problem, since due to my close involvement in carrying out the test jumps and the small test groups, I could just clarify the meaning of the question verbally, but it definitely was a “lesson learned” when it comes to creating questionnaires.

The next two questions focused more on the power dynamic between the two participants. First, I asked the controllers if they had a feeling of superiority (compared to the jumper). This was directly followed by a question whether an outside observer would say that they exploited their power or not. The question was formulated this way, in order to put the

controller in a position in which he or she should try to look at what happened more objectively, but also inevitably included the controller's guilt or even shame, if he or she did in fact exploit the situation for his or her own enjoyment.

After that, there was a simple question of which interface was used by the controller (Button, Leap Motion or Tablet), followed by a more general block, which, again, both questionnaires had in common.

Instead of these questions only affecting the role of the controller, the role-specific part of the jumper's questionnaire contained similar ones, tailored to his specific situation. The first, and probably most relevant one, was how intense their immersion was on a scale of -2 to +2. After that, a simple yes/no question had to be answered whether the jumper felt at the controller's mercy or not. This answer would make an interesting point when looked at in correlation to the controller's answers to this feeling of power over the jumper.

Next, the jumper had to name how many effects had been triggered during his virtual parachute jump. This was directly followed by an open question where the test subject had to name all different effects he or she observed. The answers to this question could potentially be biased as a result of switching the two participant's roles for the second run. In order to account for that, I decided to mark each survey form with one or two lines, representing if it was this person's first or second run.

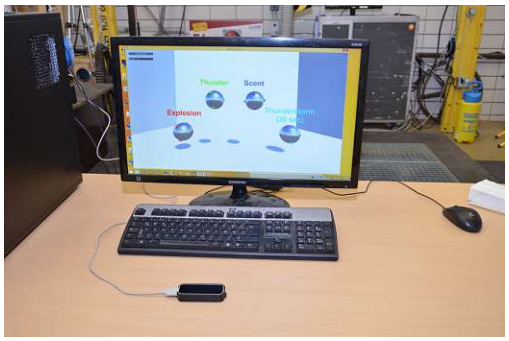
Another very import question followed, namely whether the triggered effects did affect the simulation in a positive or negative way. More precisely the question was whether the triggered effects were distracting, which they partially were intended to be, or if they aided to the livelihood of the simulation. This was directly followed by a question in how far the indirect interaction with a human being, instead of pre-scripted events, affected the jumper's experience (on a scale of -2 to +2).

Continuing, there was again a more general block of questions, which both questionnaires had in common. First, the participants were asked if they wanted to take the role of their counterpart. I originally added this question to see if the test subjects would find the other role desirable or not. Due to the simplified and reduced testing procedure however, each participant did have to take the other role either way.

Next, the participant was asked if he or she would characterize the experience and the setup as a "fun group activity", followed by a question whether they would like to do it again e.g., with a group of friends. With these two questions I tried to evaluate if and how the participants enjoyed the test run in more detail. These two answers, especially the second one, would also allow me to evaluate if this approach could be transformed into a commercial product. Although this was not planned by any stretch of the imagination, I wanted to know if people would see it as viable anyway.

The last two questions were open ones with very similar content. Firstly, the participants were asked if they wanted to tell the developers something in a general sense. The second question asked, more precisely, for any recommendations or tips that would further

improve realism or immersion. Although the second question was more steered towards the person in VR, both survey forms contained it. Since the controller did see the content of the simulation as well as the reactions of the jumper, he or she could still have some valuable insights that would otherwise have been missed.



(a) Remote interaction console application with *Leap Motion* interface.

(b) Different desinfectants for hygenic operation.

Figure 4.2: Details from the test setup for the test jumps for my user study.

4.3 Impact of the *COVID-19* Pandemic

As mentioned before, I had to perform the test jumps in a relatively short window of time during the *COVID-19* pandemic. After the first lockdown was lifted, infections were moderately low in Austria during the summer. When the number of infections slowly started to rise again, the test jump had to be performed as quickly as possible in order to avoid being impossible due to another lockdown. In addition to this rush on the schedule, the test jumps also had to be performed as safely as possible with minimal risk for potentially spreading the virus among my test subjects.

Originally, I intended to work with a test group of around 20-25 people, but under the given situation caused by the *COVID-19* pandemic, I decided to reduce the test group to the smallest number possible. In the end, I worked with 8 people, which all doubled as jumpers as well as controllers, thus resulting in 16 filled out questionnaires in total.

The test jumps were performed in teams of two, with me supervising the test subjects switching places after the first turn. This allowed me to reduce the number of people in the testing area to only three at a time. This not only allowed enough distance between each pair, but also hugely increased the subjective feeling of safety. Only working with two participants at a time and also making only one test-appointment per day, allowed any floating particles in the air to dissipate thereby further decreasing potential infections between test groups.

In addition to only working with a single group per day, simultaneously reducing the number of people in the testing area and keeping distance between them, disinfection

was another important part of the safety concept. I provided paper towels, surface disinfectant, as well was a bottle of hand sanitizer and placed them clearly visible at a table in the testing area. Before the start of the test jumps and between each user, I thoroughly cleaned my hands with the hand sanitizer and the VR headset as well as the interaction console's table and input devices with the surface disinfectant and the paper towels. I also encouraged my participants to use the hand sanitizer before and after the test to reduce the risk of infections. I performed the disinfection of my hands and of the equipment clearly visible, and also told the participants what I was doing in order to assure everyone on site, that everything had been cleaned and disinfected as thoroughly as possible.

With this meticulous disinfection and reduced headcount, I was comfortable enough to carry out my user study without exposing my test subject to undue risks. Although the reduced number of participants might have introduced some kind of bias in my data as a result of the small sample size, I was still able to gather enough data to draw some conclusions with regard to my research questions.

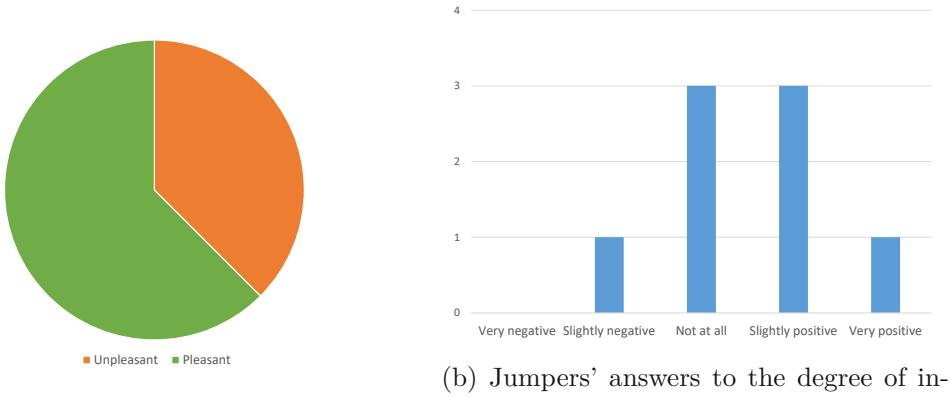
4.4 Overview of the Results

The answers to the questions of the survey proved surprisingly diverse. Many of them seem to provide mixed results. When looking closer at the questions however, this is hardly surprising since many of them are highly subjective.

The participants of my user study were all male and of ages between *26* and *32*. When categorizing the participants' ages, 62,5% where 30 years or younger, while the remaining 37.5% where over 30 years old. In the test group, most people also had prior experience with virtual reality and immersion. In my sample size of 8 people, 7 of them (87.5%) answered the question about previous contact with virtual reality with *yes*. This high number of test subjects with previous VR and immersion experience might lead to slightly biased results, since the level of comparability might be different when compared to someone who is new to VR altogether.

Interestingly, this correlation does not present itself with regard to the jumpers. The answers to the question about the level of comfortability were mostly positive for this group. 75% of the jumpers specified that they felt *comfortable* (25%) or *very comfortable* (50%). *Neutral* was never ticked, while *uncomfortable* and *very uncomfortable* were selected by one participant each. This shows that the experience is very polarizing and was received well by some test subjects, but felt rather unpleasant to others. The tendency towards the upper half of the comfortability scale might have its origin in the high number or participants with prior experience with VR after all.

The degree of how intense immersion was felt by the jumpers seems to be very individual and no clear trend can be seen. None of the test subjects named the intensity of immersion *not at all*, but 2 people (25%) each selected *almost none*, *moderate*, *strong* and *very*



(a) Jumpers' answers to the type of influence of triggered effects (37.5% *unpleasant*, 62.5% *pleasant*).

(b) Jumpers' answers to the degree of influence that the indirect interaction with another person had on a scale from -2 (left) to +2 (right).

Figure 4.3: Diagrams showing the influences of externally triggered effect to jumpers' experiences.

strong respectively. This shows how subjective immersion is and how differently people experience it.

Although the general degree of immersion seems to be quite equally distributed among intensities, the effects of externally triggered events do show more conclusiveness. No participant stated that indirect human interaction influenced their experience negatively. One person (12.5%) claimed it had affected the experience *slightly negatively*, three (37.5%) did not notice any influence, another three (37.5%) named the influence to be *slightly positively* and one (12.5%) said it influenced their experience *very positively*. These number can be categorized in 12.5% felt negatively influenced, 37.5% not affected at all and 50% rated the influence of outside, indirect, human interaction on the positive side of the scale.

Interestingly, when asked if the triggered effects where unpleasant and distracting or rather made the simulation more lively, only 62,5% felt the simulation was enriched with the effects. The remaining 37,5% rated the effects, in general, unpleasant and distracting. When comparing these numbers with the rating of the influence that these effects had to the VR experience, we see that some participants who rated the effects as unpleasant or distracting must have still described the influence positive or at least as neutral.

When asked for the number of effects that were triggered from the outside, seven jumpers' answers ranged from *30* to *100* while one person just answered "a lot" which has to be excluded here. When categorizing the valid answers into three groups, two jumpers (28.57%) claimed to have seen *100* externally triggered effects, one (14.29%) named the number to be below *50* while the majority of 57.14% (4 people) stated a number between *50* and *99*. With the exclusion of the person who could not pin-point a number, the average number given was 62.14. The rather broad range of the answers here is

hardly surprising, since a lot of effects were triggered and with numbers that high, people apparently do not count but rather estimate. In addition to that, when starting the simulation, it might not have been clear from the get-go which effects were pre-programmed and which ones were externally triggered. As the virtual parachute jump goes on for several minutes though, the externally triggered effects should have been identifiable more easily, due to the constant repetition all the way through the simulation.

When asked which effects the jumpers observed, the answers where given as free text, which could be mapped to the correct effects during evaluating the questionnaire. After the classification of various expressions and names to the actually implemented effects, the numbers show, that some effects where more obvious than others. The *explosion* was identified by all participants (100%), followed by *thunder and lightning* which was names by 75%. The *scent* was recognized as a triggered effect by 5 test subjects (62.5%) while the *thunderstorm* was only named by 3 out of 8 jumpers (37.5%). Two answers could not be classified as one of the implemented effects.

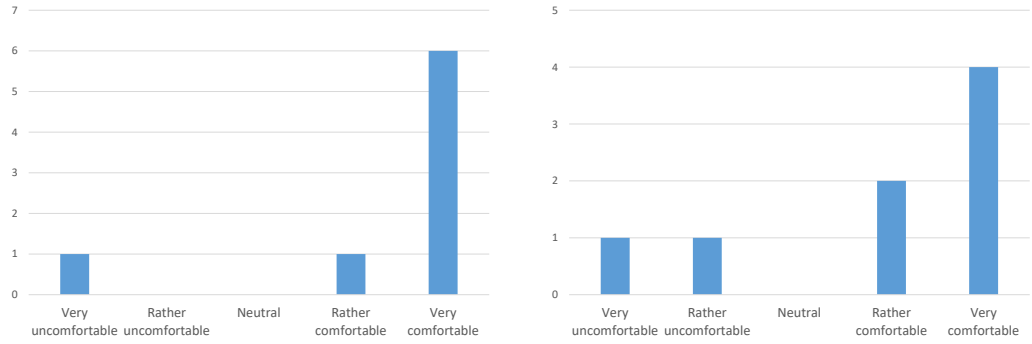
When the jumpers were asked if they felt at mercy to the controller, 75% agreed while only 25% did not feel that they were. To no big surprise, all of the jumpers (100%) answered the question whether they wanted to take the role of the controller with *yes*.

The question concerning the characterization of the test setup as a *fun group activity* was surprisingly positive. While only 6 out of 8 jumpers (75%) would like to experience a simulation like this again with e.g. a group of friends, 7 out of 8 (87.5%) would characterize this setup as a *fun group activity*.

The free text answers for feedback to the developer can be summed up by the notion of wishing for more realistic effects. A number of participants noted that some effects were distracting and interfered with the simulation content and also various inconsistencies were mentioned like the effects not fitting in the simulation scenario (more precisely the weather situation). Some bugs in the simulation were recognized and while the positive effect on immersion was explicitly noted and the feedback was mostly positive, one participant suggested to limit the number or occurrence of certain effects.

While this user study was initially focused on the role of the jumper, the evaluation of the controller's answers did prove to be very informative too. Since the participants were the same due to the switching of roles after the first jump, the general information regarding the composition of the test group with regard to gender, age and prior VR experience is the same as outlined before.

The first differences in answers show up when looking at the question whether the participants liked the game or simulation or not. In contrast to the jumper's questionnaire, 100% of the controllers answered this question with *yes*. This trend continues when looking at the answers to the question asking about the level of comfortability in this role. Only one person (12.5%) stated he felt *very uncomfortable*, another one felt *rather comfortable* (12.5%), while the majority (75%) stated to have felt *very comfortable*. The other options (*rather uncomfortable* and neutral) were not selected.



(a) Level of comfortability of controllers on a scale from -2 (left) to +2 (right).

(b) Level of comfortability of jumpers on a scale from -2 (left) to +2 (right).

Figure 4.4: Diagrams of subjective level of comfortability of participants.

This tendency towards a high level of comfortability is not very surprising, since the controllers where not only the ones controlling the triggered effects, but also the situation as a whole. This can also be seen in the question about how much "in control" the participants felt. With all other options not ticked at all, the majority (75%) answered *very*, followed by two people (25%) that selected *slightly*. Building up on this concurring impression of control of the participants taking the role of the controller, the next two questions to look at show very interesting results.

When asked if the controllers had a feeling of superiority of power during the test jump, the vast majority of 87.5% answered with *yes*. Only a single test subject answered this with *no* (12.5%). Interestingly, the question whether an external observer would think that they exploited and misused their power was unanimously answered with *yes* by 100% of the controllers.

Consequently, in contrast to the jumpers, the notion of taking the other person's place is, while still being high, much less pronounced among the controllers. Here, 75% wanted to switch places while the remaining 25% answered this question with *no*.

Another interesting point for comparison is posed by the number of triggered effects. In comparison to the triggered effects as perceived by the jumpers, the numbers given by the controllers are more evenly distributed. Like with the jumpers, one person just answered "a lot", an additional two wrote down "100+". When we categorize the answers, again leaving out the non-numerical one, 42.86% claimed to have triggered *less than 50* effects, one person answered with *50 to 99* and another 42.86% stated a number of *100 or above*. When only counting absolute numbers, this results in an average of 48 effects. When we interpret the *100+* answers as just plain *100* while still leaving out the *a lot* response, we get an average of 62,86 effects.

One question that was only present on the controller's questionnaire was the one concerning the preferred interface for interacting with the remote interaction console. The

responses from the participants show, that most of them liked the *Leap Motion* interface better (75%) with only two people (25%) preferring the button interface. In this special case though, these numbers have to be taken with a grain of salt since they are hardly representative. In four cases the participants were not allowed to choose the interface in order to make sure to be able to evaluate both interfaces in the course of the user study. The remaining test subjects were allowed to choose freely between the two variants. Keeping that in mind, we have to leave out the two participants who were instructed to use the button-based interface, as well as the two that had to use the hand movement-based system in order to get more realistic numbers. When we remove these cases and only look at the participants that were allowed to choose, we see that all of them opted for the hand movement-based interface.

As already mentioned with regard to the answers to the general questions in the beginning section of the questionnaire regarding gender, age and prior VR experience, the answers concerning the setup's potential as a *fun group activity* and the desire to participate in an experience like this again (e.g. with friends), coincide with the answers given in the jumper's questionnaire.

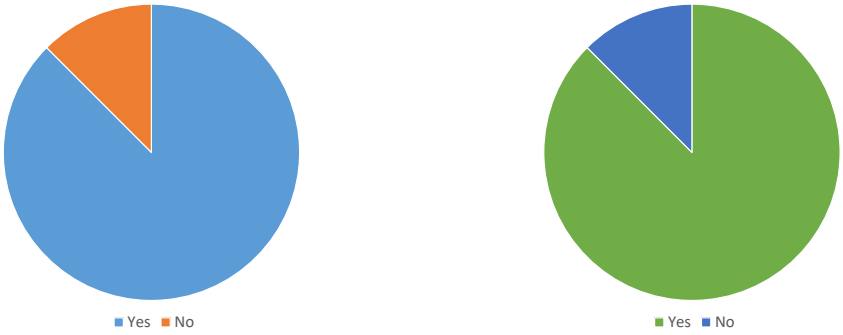
The free text answers of the controllers are of course focused on different issues than the ones of the jumpers. Among the participants taking the controller's role, the call for a broader spectrum of effects was very common. Also mentioned very often were some modifications to the interface, e.g. like the initially intended more detailed visualization of the effect cooldowns. Several times the wish for a video feed showing the jumpers directly on the console's interface in order to avoid having to look at a second screen was mentioned. Overall, the experience of using the interaction console, especially in combination with the *Leap Motion Controller*, seems to have been received very well and enjoyed a lot.

4.5 Interpretation and Discussion

After all results have been presented, there still is the need for further discussion and interpretation of these numbers; especially with regard to their meaning concerning my research questions.

As already mentioned, it is hardly surprising that previous experiences of participants with VR and immersion have an impact on their level of comparability while inside the simulation. The relatively high number of test subjects rating their comfort level positively also coincides with the answers whether the simulation or game was liked or not, where the majority of the participants responded positively.

Bearing the high percentage of prior VR experience among the participants in mind, it comes as no surprise that the degree of immersion is rated differently by every person, thus showing rather mixed results. Since every individual has its own preferences and visual cues they look out for in regard to realism, experienced VR users probably tend to



(a) Prior experience with VR among jumpers (87.5% *yes*, 12.5% *no*).

(b) Answeres to whether jumpers liked the game or simulation (87.5% *yes*, 12.5% *no*).

Figure 4.5: Correlation between prior VR experience of jumpers and liking of the simulation.

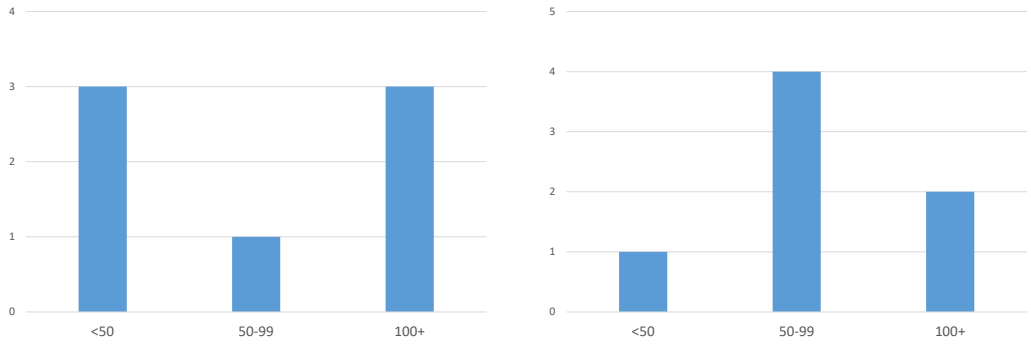
be less impressed with relatively simple simulations in visual terms, when compared with today's commercially available games, for example.

Interesting is however, that the jumpers claimed that the humanly controlled, externally triggered effects not only affected their immersion, but also did so in a positive way. Due to the distracting nature of many effects and the high number in which they were triggered by almost every controller, this is rather surprising. One could easily have assumed, that the effects would rather hinder immersion than enrich and add to it. Be it the effects themselves, the pure knowledge that they were triggered by another person, or the subliminal and indirect interaction with another participant, even without any direct communication, the test subjects in VR claimed this increased their immersion. With regard to my initial research questions if externally triggered stimuli affect immersion at all (1) and in which way (2), the results show clear answers to that. According to my test subjects, there indeed is an effect on immersion and it seems to be a positive one.

An interesting point of observation is posed by the number of observed and triggered effects when comparing the answers of the jumpers with the corresponding ones of the controllers. The number of effects the controllers triggered roughly coincides with the number of effects perceived by the jumpers, but still shows some interesting differences. While the jumpers generally observed between 50 and 100 effects, the controllers' answers were more diverse. When comparing these numbers, some of them seem to have understated the effects triggered, while others overstated them, even without being able to give a concrete number. The number of triggered actions as perceived by the controllers seems to heavily lean towards both ends of the spectrum rather than showing an approximated normal distribution.

One possible reason for this might be, that most controllers enjoyed the *Leap Motion* interface very much and therefore probably did not pay much attention to the effects they triggered. This could very well have led to unspecific answers like *a lot* or *100+* since the

participants did just know they hit the trigger numerous times without counting. On the other side, when triggering effects consciously, the manifesting power dynamic and possibility or even fear of extorting this power might have led to understating the number of actually triggered effects. If controllers did want to avoid (maybe just subconsciously) the appearance of having misused their powers, they might have given a lower number of effects, than what they had actually triggered.



(a) Number of effects supposedly triggered by the controllers.

(b) Number of trigered effects as percieved by the jumpers.

Figure 4.6: Divergence between triggered and perceived effects.

On the other side, while the majority of jumpers did recognize externally triggered effects, some of them classified simulation-internal effects as externally triggered effects too. So, the number of experienced effects given by the jumpers, could very well be off too. Although this might not account for a significant difference since the number of pre-programmed effects is fairly limited, it might still contribute to the difference observed here.

In light of this realization, it is hard to conclusively answer my research question about the difference in perception between pre-programmed and externally triggered effects (3). Since a few participants did list pre-programmed effects as "externally triggered", there might not be a clear distinction here. The main difference might very well only be the repeated triggering of these additional effects which can thereby be identified as externally controlled. Theoretically, the indirect communication could also be a factor here. If the jumper responds to an effect in a certain way (e.g. screaming), the controller might be inclined to trigger it again, but none of my results show a direct correlation like that. When taking this into account, externally triggered effects could indeed be described as being perceived differently from pre-programmed ones, not by the simulated actions but rather by the inter-personal communication between the participants. In how far this difference in perception is noticeable to the jumper, is hard to answer. In the end, the effects themselves are most likely perceived exactly like pre-programmed ones, but when looking at the manually triggered effects in the greater context, they might very well be perceived differently.

The answer to my research question about the interference of triggered effects (4) can only be answered when looking more closely at the critique and feedback the participants gave in the last two questions of the questionnaire. As mentioned before, some test subjects explicitly described the effects as distracting and interfering with the simulation. Although not all participants gave answers like that, probably also due to that fact that the degree of distraction is highly subjective, I think it is safe to say that externally triggered effect at least can interfere with the simulation content in some cases.

Another very interesting point of observation is the inter-personal communication and power dynamic between the two participants taking the roles of controller and jumper. According to my data, the vast majority of controllers felt some kind of superiority or power over the jumper. With regard to my research question whether a situation like this would provoke a dynamic shifting of power (6), it appears safe to say it did. Since the setup was designed to facilitate this, it is only logical that it manifested when both participants took their roles. An interesting and unexpected answer on the other hand, was the one about what a hypothetical external observer would say whether this power was extorted or not. One would have expected a more diversified field of answers, but at least in my test group, every controller had the opinion that he could easily be accused of extorting his power.

Sadly, research question number 5, how the console operators respond to seeing the reaction of the person in VR to their triggered effects, is not that easy to answer. Since this is hardly quantifiable and not practical to put into a questionnaire, we can only rely on the previously made conclusions and my observations while carrying out the test jumps for the user study. Every controller seemed to respond in a very different way, or in some cases not at all, to the jumper's reactions to triggered effects. As previously mentioned, some controllers had much fun using the interaction console from a stand-alone application standpoint and therefore rather just "noticed" the jumper's reaction instead of directly reacting to them. Others were completely focused on the jumper and his reaction and only used the interaction console as a tool in order to provoke him or her. In the end, in this case, the research question might have been formulated too broadly since it does not allow for a conclusive answer. Some test subjects operating the remote console did not respond at all, some closely observed how the jumper reacted and seemed to make provoking extreme reactions from the jumper their main goal.

Not very surprising were the answers in regard to switching places. With the controller being the "empowered" role, it is only logical that all jumpers did want to switch places. It is also a logical conclusion that the controllers, with just having stated that they might have extorted their powers over the jumper, did not want to switch roles that unanimously.

My research question about the interface of preference when it comes to the remote interaction console (7) is difficult to answer, taking the condition under which I performed my user study into account. Due to the rather small test group I had to work with, I decided to instruct two participants to use a specific interface, while the rest of the participants had the possibility to choose their preferred input method. This way I could

make sure, that the traditional button UI, as well as the *Leap Motion* interface would both be used at least twice and allowed me to evaluate both. As the numbers show, whenever possible, the test subjects chose the hand movement-based interface over the traditional button-based UI. The only exceptions in my test group where the ones, that were specifically instructed to use the button-based interface. Accordingly, I think it is save to conclude that the *Leap Motion* interface was preferred.

When it comes to rating the test setup and the simulation as a whole, the answers show a clear direction. The experience was mostly enjoyed very much and the majority of the participants would characterize it as a *fun group activity*. The more relevant question with regard to my research question of the potential of this setup for an exhibition or even a commercial product (8), was the next one to follow. When asked if my participant could imagine to re-experience the game or simulation e.g. with friends, the approval did go down a little. One reason for this might be the lack of visual realism of many effects or the obviously experimental system setup. Another one might also be that the experience is fun, but does not offer any form of engagement for the jumper at all. In order to be used more than once, typically both participants should have fun and be able to interact with the game as well as with each other. After all, inter-personal communication is usually an essential building block for almost any enjoyable group activity.

As a final remark about the interpretation of my results, it has to be noted that as a consequence of the relatively small test group, no further correlative evaluations were performed. With a sample size that small, some correlations, like the "fun-factor" of the controllers vs. the jumpers, would not make much sense and hardly produce valuable results. This is even more true, since in my case all participants ultimately took both roles, which could make it hard to distinguish which role brought them more joy, but would certainly introduce a bias since the ability to compare the experiences would affect the answers of the participants.

After carefully conducting my user study and gathering data, I was finally able to evaluate and reflect on the results. It is important to keep the relatively small size of my test group in mind. For the conclusions drawn in this chapter this means that not all of them will be representative and might carry a heavy bias. This is especially true since my test group was unusually homogeneous with all participants being male and within a relatively small age range, as well as almost everyone already having prior experience with VR. Bearing that in mind, while some interpretations might still leave room for discussion, other questions could be answered relatively conclusively. With all my research questions explored and most of them answered in a satisfactory way, the user study can be regarded as a success and leads this thesis to its conclusion.

CHAPTER 5

Conclusion and Future Work

After covering the origins of virtual reality and multiplayer gaming, exploring scientific literature concerning asymmetrical interaction methods involving VR, successfully implementing a setup where the effects of externally triggered stimuli can be explored and carrying out a small-scale user study, it is now time to summarize the results and draw some final conclusions.

During the introduction and background parts of this thesis, we saw that the concept of virtual reality is by no means a new one. Although ideas and concepts have already exited for a long time, modern technology boosts their practicality considerably. In many cases, back then and also today, immersion is one of the ultimate goals developers and designers strive to achieve. Interestingly, most cases cover the immersive experience of the person using the system in question alone (e.g. a VR headset) and do not take external stimuli into account. Most systems making use of asymmetrical interaction, like the ones discussed in the section *related work*, use it as a practical tool or means of communication, be it in a scientific sense or as a gameplay element. Supporting and fortifying the immersion of a player in VR is not that commonly covered, which is why my results shed light on many, previously only sparsely covered, aspects of asymmetrical interaction involving virtual reality.

In chapter 3, an outline of the project's development phase and implementation was given by covering various design decisions and why they were made. We carried the initially posed research questions over to concrete requirements and defined three different types of effects. The requirements lead to the implementation of a new remote interaction console application using the *Unity* game engine, as well as to the adaption of the original *Virtual Jump* content.

The next phase was the conduction of a user study with a carefully defined questionnaire for both roles that participants could take. Both, jumpers as well as controllers, had to answer several questions that ultimately would allow me to draw necessary conclusions

and to answer my initially defined research questions. Although my user study was only conducted with a small test group and the system was a basic implementation of the concept in play, I was able to gather valuable data and draw interesting and even surprising conclusions.

5.1 Results and Evaluation

The results of my small-scale user study revealed many interesting facts and were able to answer most of my research question conclusively. When summarizing the results of my user study, we always need to keep the small size of my test group in mind, which only allows drawing conclusions for my specific group of participants. Due to the reduced number of test subjects and the unusually homogeneous group composition, my results cannot be assumed to be true for any given group within a larger population.

In my specific test group however, I was able to witness the development of interesting group-dynamic effects between the participants. This probably was even facilitated by the methodology of a team of two test subject, switching their roles after the first jump. The most interesting part was the difference in the way the controllers reacted to the jumper's responses to the triggered effect. Some observed them keenly whenever an effect was triggered, other seemed to draw joy from provoking reactions to their effects, while some seemed to enjoy the hand-movement based interface so much, that they almost did not care if or how their partner reacted.

Generally speaking, in my user study, the controllers seemed to enjoy themselves more than the jumpers did. When it comes to the interfaces used by the controller, the more unusual hand-movement based controls have definitely been favored over the traditional input methods. Both these realizations have to be taken with a grain of salt though, since it might only be true for my specific test group. Since most of my participants had prior experience with VR, the impressiveness of a fully immersive VR simulation might be higher in the general public when compared to my participants. With regard to the hand-movement based interaction with the console application however, my results could very well be more representative, since there was no obvious bias or prior experience among my test group.

When it comes to the topic of immersion, the results showed some surprising realizations. My user study showed for example, that immersion was indeed affected by externally triggered effects. This was somehow expected, but the direction of the influence was surprising. Contrary to what could easily be assumed, the results show, that externally triggered effects enhance the impressiveness rather than hinder it. This might also only be true for my specific test group, since, again, the high degree of prior experience with VR could have affected this outcome. My results also show, that externally triggered effects could usually be identified as such by most of my participants. This is, of course, only true for my specific test setup and the effects I used. If the effects might be differently implemented, more fitting for the simulation context or simply not that repetitively triggered, they could also be mixed up with pre-programmed ones in my opinion.

While some of these conclusions were surprising, one of my research question was answered just like one would expect. My results show, that externally triggered effects, especially with the frequency they were triggered by my controllers, did indeed interfere with the simulation content. Given that I intentionally implemented some of them to be perceived as highly distracting, it came as no surprise that the jumpers also shared this impression. This can, of course, easily be mitigated by blocking out specific effects for a given part of the simulation, e.g. while a narration is going on, and also highly depends on the triggered effect, since some are implicitly more distracting than others. Although this does not need to hold true for every group or population, in general terms, one can conclude that externally triggered effect, when not restricted to specific parts of the simulation or to only non-distracting types of effects, are not suitable for story-based simulation contents. In VR simulations where there is no ongoing story or plot that has to be followed, like it was the case in my user study, externally triggered effects seem to enrich the experience and boost immersion for the player in VR.

The last of my research questions, whether this setup could be considered a "fun group activity" and hold potential that would warrant further development and refinement, was answered relatively positively by my participants. Most of them stated that they could imagine experiencing a simulation like this, e.g. with a group of friends. Like already noted before, with all participants having prior VR experience and also being technology-affine in general, the results could very well be different when looking at a more diverse and bigger group of people. Nevertheless, the performed test jumps seem to have been an enjoyable experience for most of my participants as well as for me, especially when keeping the external circumstances in mind.

5.2 Possible Practical Applications of my Work

As a result to the answer to my last research question, there could indeed be potential in this setup for some kind of commercial product. But before jumping to conclusions, a few things need to be taken into account. For one, like already discussed, the opinion of my small test group cannot be generalized and taken to be representative for a larger population. Technology-affine people, like my test subject, are, most likely, more prone to enjoy novel, experimental setups like this than the general public.

Another fact to keep in mind is the complexity of the setup. The *Virtual Jump* simulator alone exceeds the available space in most living rooms by far. Therefore, a setup like this would only be possible at events or in arcade-like settings. In this special case, the enjoyment surely also stems from the participants knowing each other. This is also the reason, why my question was targeted on repeating the experience e.g. with a group of friends. When we take that limitation to the user base into account, public events like exhibitions etc., become less relevant. Gaming arcades and similar commercial offers, would still fit this paradigm as they could be visited by a group of friends for an enjoyable afternoon or evening.

In addition to the previously discussed entertainment factor, there is also another aspect

that could bring joy to people. A group of bystanders or attendees at a conference where this setup is deployed for example, could surely be entertained by watching other people live through this experience. This is especially true, if the participants show their feelings very openly, as loudly laughing or screaming usually catches peoples' attention.

There is also a third use case for a system like the one I created for this thesis, even if it is only in a highly adapted form. When we look back, for example, at some commercially available, asymmetrical VR titles for home entertainment platforms like *PlayStation VR*, there are a lot of similarities in the way they are designed. The main difference is the focus on collaborative or competitive interaction, in contrast to unilateral action of just the controller with the simulation environment. The setup I created could easily be transformed into a system ready for home use, by using only a standard VR headset instead of the fully-immersive *Virtual Jump* simulator. The external remote interaction console, could in this use case easily be replaced by smartphones, operated by friends sitting next to the VR player, triggering actions via Wi-Fi. If we think of a setup like that, the system I implemented, in essence, can basically be reduced to a software application, that could easily be sold as a game for current VR platforms. Unfortunately, when reduced this far, one key aspect of this experiment would also be diminished drastically. Since a retail VR headset only provides the wearer with visual and audio stimuli, it will most likely not be able to reproduce immersion to the same degree as we saw during the user study covered in this thesis.

5.3 Outlook and Final Words

Summarizing these conclusions, it is doubtful, but not entirely impossible, that we see a system like this as a commercial system in the future. Transforming the system to a home-use practical variant would mean getting rid of most of its selling points, while modern VR arcades and theme parks already have much more interactive and exciting attractions to be used.

From an academic point of view though, this experiment proved to be highly interesting and showed surprising results. Since in scientific applications, immersion is usually not the primary concern, integrating externally triggered effects will probably not be the main goal when developing a system for practical use. In some situations however, the knowledge that immersion can indeed be increased by externally triggered stimuli could prove to be worth taking into consideration and could consequently help raise the quality of many VR experiences.

List of Figures

Bibliography

[AAM+13] Abrar Omar Alkhamisi, Saudi Arabia, Muhammad Mostafa Monowar, et al. Rise of augmented reality: Current and future application areas. *International journal of internet and distributed systems*, 1(04):25, 2013.

[ADI17] Asmaa Saeed Alqahtani, Lamya Foaud Daghestani, and Lamiaa Fattouh Ibrahim. Environments and system types of virtual reality technology in STEM: A survey. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 8(6), 2017.

[and] Android: The platform pushing what's possible. *https://www.android.com/*. Online; accessed 27-03-2021.

[Bak16] Jeff Bakalar. PlayStation VR is pretty damn cool. *https://www.cnet.com/reviews/sony-playstation-vr-review/*, Oct 2016. Online; accessed 29-11-2020.

[Bar18] Dom Barnard. Degrees of Freedom (DoF): 3-DoF vs 6-DoF for VR Headset Selection. *https://virtualspeech.com/blog/degrees-of-freedom-vr*, Sep 2018. Online; accessed 29-11-2020.

[Bar19] Dom Barnard. History of VR - timeline of events and tech development. *https://virtualspeech.com/blog/history-of-vr*, Aug 2019. Online; accessed 29-11-2020.

[BLW18] Christoph W Borst, Nicholas G Lipari, and Jason W Woodworth. Teacher-guided educational VR: Assessment of live and prerecorded teachers guiding virtual field trips. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 467–474. IEEE, 2018.

[Bre56] David Brewster. *The Stereoscope; Its History, Theory and Construction, with Its Application to the Fine and Useful Arts and to Education, Etc.* John Murray, 1856.

[Bru13] Grant Brunner. Shoulder buttons of giants: The evolution of controllers leading up to PS4 and Xbox One. *https://www.extremetech.com/gaming/156711-evolution-of-video-game-controllers-leading-to-ps4-and-xbox-one*, May 2013. Online; accessed 07-02-2021.

[Bur19] Chris Burns. HTC Vive wireless adapter review. *https://www.slashgear.com/htc-vive-wireless-adapter-review-16553910/*, Sep 2019. Online; accessed 29-11-2020.

[car] Get Cardboard. *https://arvr.google.com/cardboard/get-cardboard/*. Online; accessed 29-11-2020.

[car19] Open sourcing Google Cardboard. *https://developers.googleblog.com/2019/11/open-sourcing-google-cardboard.html*, Nov 2019. Online; accessed 29-11-2020.

[CCD+13] Paul Cairns, Anna L Cox, Matthew Day, Hayley Martin, and Thomas Perryman. Who but not where: The effect of social play on immersion in digital games. *International Journal of Human-Computer Studies*, 71(11):1069–1077, 2013.

[Cla02] Dan Clarke. Xbox Live. *https://www.gamingnexus.com/Article/71/Xbox-Live/*, Nov 2002. Online; accessed 06-01-2021.

[cod] Call of Duty©: Modern Warfare: Home. *https://www.callofduty.com/de/modernwarfare*. Online; accessed 04-02-2021.

[Coh19a] D.S. Cohen. Cathode-ray tube amusement device: The world's first video game? *https://www.lifewire.com/cathode-ray-tube-amusement-device-729579*, Mar 2019. Online; accessed 05-01-2021.

[Coh19b] D.S. Cohen. OXO aka Noughts and Crosses - the first video game. *https://www.lifewire.com/oxo-aka-noughts-and-crosses-729624*, Mar 2019. Online; accessed 05-01-2021.

[Coh20] D.S. Cohen. Magnavox Odyssey - the first gaming console. *https://www.lifewire.com/magnavox-odyssey-the-first-gaming-console-729587*, Sep 2020. Online; accessed 05-01-2021.

[Dan20] Benjamin Danneberg. Oculus Link im Test: PC-VR für Oculus Quest mit offiziellem USB-Kabel. *https://mixed.de/oculus-link-test/*, Apr 2020. Online; accessed 29-11-2020.

[DBJ+19] Ralf Dörner, Wolfgang Broll, Bernhard Jung, Paul Grimm, and Martin Göbel. *Einführung in Virtual und Augmented Reality*, pages 1–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

[DDAM14] Parth Rajesh Desai, Pooja Nikhil Desai, Komal Deepak Ajmera, and Khushbu Mehta. A review paper on oculus rift-a virtual reality headset. *arXiv preprint arXiv:1408.1173*, 2014.

[dL14] Alan Richard da Luz. Evolution of the physical interfaces in videogames as a support to the narrative and the gaming experience. In Aaron Marcus, editor, *Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments*, pages 688–698, Cham, 2014. Springer International Publishing.

[Don10] Tristan Donovan. *Replay: The history of video games*. Yellow Ant, 2010. ISBN: 978-0-9565072-2-8.

[dra] Drag race - videogame by Kee Games. *https://www.arcade-museum.com/game_detail.php?game_id=7634*. Online; accessed 05-01-2021.

[Edw15] Benj Edwards. Unraveling the enigma of Nintendo's Virtual Boy, 20 years later. *https://www.fastcompany.com/3050016/unraveling-the-enigma-of-nintendos-virtual-boy-20-years-later*, Aug 2015. Online; accessed 29-11-2020.

[Eid18] Horst Eidenberger. Smell and touch in the Virtual Jumpcube. *Multimedia Systems*, 24(6):695–709, 2018.

[EM15] Horst Eidenberger and Annette Mossel. Indoor skydiving in immersive virtual reality with embedded storytelling. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, pages 9–12. ACM, 2015.

[Gam07] Retro Gamer. The definitive Space Invaders. *Retro Gamer (Imagine Publishing)*, 41:24—33, 2007.

[Gam08] John E Gamble. Competition in video game consoles: Sony, Microsoft, and Nintendo battle for supremacy. *McGraw-Hill/Irwin*, 2008.

[Gen96] Next Generation. Getting connected. *Next Generation (Imagine Publishing)*, 2(19):29—35, Jul 1996.

[GKK+17] Scott W Greenwald, Alexander Kulik, André Kunert, Stephan Beck, Bernd Fröhlich, Sue Cobb, Sarah Parsons, Nigel Newbutt, Christine Gouveia, Claire Cook, et al. Technology and applications for collaborative learning in virtual reality. Philadelphia, PA: International Society of the Learning Sciences., 2017.

[GM48] Jr Thomas T Goldsmith and Estle Ray Mann. Cathode-ray tube amusement device, Dec 1948. US Patent 2,455,992.

[goo] Google Cardboard. *https://www.google.com/get/cardboard/*. Online; accessed 29-11-2020.

[Goo13] Owen Good. Nintendo Network announced; digital distribution of games envisioned. *https://kotaku.com/nintendo-network-announced-digital-distribution-of-gam-5879796*, Jun 2013. Online; accessed 06-01-2021.

[Gra20] Peter Graham. 5 million PlayStation VR's have now been sold. *https://www.vrfocus.com/2020/01/5-million-playstation-vrs-have-now-been-sold/*, Jan 2020. Online; accessed 27-12-2020.

[Gru14] Gretchen Jane Gruber. *View Master: The Biography of William B. Gruber*. Hillcrest Publishing Group, 2014.

[Hei60] Morton L Heilig. Stereoscopic-television apparatus for individual use, 10 1960. US Patent 2,955,156.

[Hei62] Morton L Heilig. Sensorama simulator, 08 1962. US Patent 3,050,870.

[Hoh19] Nate Hohl. Call of Duty: Modern Warfare: How to succeed in special ops co-op mode. *https://gamecrate.com/call-duty-modern-warfare-how-succeed-special-ops-co-op-mode/24541*, Oct 2019. Online; accessed 04-02-2021.

[Hor11] Vince Horiuchi. Ralph Baer, the father of video games, reflects on his career. *https://archive.sltrib.com/article.php?id=52850548&itype=CMSID*, Nov 2011. Online; accessed 05-01-2021.

[ISS+15] Hikaru Ibayashi, Yuta Sugiura, Daisuke Sakamoto, Natsuki Miyata, Mitsunori Tada, Takashi Okuma, Takeshi Kurata, Masaaki Mochimaru, and Takeo Igarashi. Dollhouse VR: a multi-view, multi-user collaborative design workspace with VR technology. In *SIGGRAPH Asia 2015 Emerging Technologies*, pages 1–2. 2015.

[JKK+20] Kisung Jeong, Jinmo Kim, Mingyu Kim, Jiwon Lee, and Chanhun Kim. Asymmetric interface: User interface of asymmetric virtual reality for new presence and experience. *Symmetry*, 12(1), 2020.

[Ken10] Steven L Kent. *The Ultimate History of Video Games: from Pong to Pokemon and Beyond - The story behind the craze that touched our lives and changed the world*. Three Rivers Press, 2010. ISBN: 0-7615-3643-4.

[kic] Kickstarter. *https://www.kickstarter.com/*. Online; accessed 29-11-2020.

[kul] Kikstart: Off-road simulator - testberichte vom Amiga Joker, ASM, Power Play, PC Joker, Play Time, Happy Computer. *https://www.kultboy.com/testbericht-uebersicht/3449/*. Online; accessed 31-03-2021.

[Lan16a] Ben Lang. Analysis of Valve's 'Lighthouse' tracking system reveals accuracy. *https://www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/*, Jul 2016. Online; accessed 29-11-2020.

[Lan16b] Ben Lang. Update opens the door for enhanced roomscale tracking with Rift and Touch. *https://www.roadtovr.com/oculus-rift-home-1-6-update-touch-four-sensors-roomscale/*, Jul 2016. Online; accessed 20-02-2021.

[Lan18] Mark Langshaw. A look back at the Magnavox Odyssey. *https://www.digitalspy.com/videogames/retro-gaming/a616235/magnavox-odyssey-retrospective-how-console-gaming-was-born/*, Oct 2018. Online; accessed 05-01-2021.

[law92] The Lawnmower Man. *https://www.imdb.com/title/tt0104692/*, Mar 1992. Online; accessed 29-11-2020.

[Lem] Kim Lemon. Lotus Esprit Turbo Challenge. *https://www.lemon64.com/?game_id=1548*. Online; accessed 05-01-2021.

[LKK17] Jiwon Lee, Mingyu Kim, and Jinmo Kim. A study on immersion and VR sickness in walking interaction for immersive virtual reality applications. *Symmetry*, 9(5), 2017.

[LM16] Stefan Liszio and Maic Masuch. Designing shared virtual reality gaming experiences in local multi-platform games. In *International Conference on Entertainment Computing*, pages 235–240. Springer, 2016.

[Low20] Henry E. Lowood. Doom. Encyclopædia Britannica, *https://www.britannica.com/topic/Doom*, Mar 2020. Online; accessed 06-01-2021.

[LPLK17] S. Lee, K. Park, J. Lee, and K. Kim. User study of vr basic controller and data glove as hand gesture inputs in vr games. In *2017 International Symposium on Ubiquitous Virtual Reality (ISUVR)*, pages 1–3, 2017.

[Mel19a] Kyle Melnick. Acron: Attack Of The Squirrels! - a surprisingly competitive VR party game designed for the masses. *https://vrscout.com/news/acron-attack-of-the-squirrels-launch/*, Aug 2019. Online; accessed 07-02-2021.

[Mel19b] Kyle Melnick. Five VR party games perfect for your upcoming halloween bash. *https://vrscout.com/news/vr-party-games-halloween-2019/*, Oct 2019. Online; accessed 07-02-2021.

[mob] Kikstart: off-road simulator (1985). *https://www.mobygames.com/game/kikstart-off-road-simulator*. Online; accessed 31-03-2021.

[Nic99] Sarah Nichols. Physical ergonomics of virtual environment use. *Applied Ergonomics*, 30(1):79–90, 1999.

[Nii06] Hirohiko Niizumi. PlayStation Network platform detailed. *https://www.gamespot.com/articles/playstation-network-platform-detailed/1100-6145981/*, Mar 2006. Online; accessed 06-01-2021.

[NLL17] Diederick C Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the HTC Vive virtual reality system for scientific research. *i-Perception*, 8(3):1–23, 2017.

[NSMS20] Jilian Nguyen, Clinton Smith, Ziv Magoz, and Jasmine Sears. Screen door effect reduction using mechanical shifting for virtual reality displays. In Bernard C. Kress and Christophe Peroz, editors, *Optical Architectures for Displays and Sensing in Augmented, Virtual, and Mixed Reality (AR, VR, MR)*, volume 11310, pages 200 – 210. International Society for Optics and Photonics, SPIE, 2020.

[Nyi16] Kristen J. Nyitray. Tennis for Two. *https://www.sunysb.edu/libspecial/videogames/tennis.html*, Apr 2016. Online; accessed 05-01-2021.

[oC82] Harry Houdini Collection (Library of Congress). *The Popular science monthly.*, volume v. 21 (1882). Popular Science Pub. Co., New York, 1882. Online version at *https://www.biodiversitylibrary.org/item/17938*; accessed 04-07-2021.

[oEB16] The Editors of Encyclopaedia Britannica. World of Warcraft. *https://www.britannica.com/topic/World-of-Warcraft*, Jun 2016. Online; accessed 06-01-2021.

[Osa17] Sohrab Osati. Can you connect two PlayStation VR headsets to a PS4? *https://sonyreconsidered.com/can-you-connect-two-playstation-vr-headsets-to-a-ps4-510d17ded811*, Jun 2017. Online; accessed 04-02-2021.

[PHD18] Mark Peter, Robin Horst, and Ralf Dörner. VR-Guide: A specific user role for asymmetric virtual reality setups in distributed virtual reality applications. In *Proceedings of the 10th Workshop Virtual and Augmented Reality of the GI Group VR/AR*, 2018.

[pla20] PlayStation VR: Erlebe deine Spiele in unglaublichen Virtual-Reality-Welten. *https://www.playstation.com/de-at/ps-vr/*, 2020. Online; accessed 29-11-2020.

[Red12] Adam Redsell. SEGA: A soothsayer of the games industry. *https://www.ign.com/articles/2012/05/20/sega-a-soothsayer-of-the-games-industry*, May 2012. Online; accessed 06-01-2021.

[Rob19] Adi Robertson. The Valve Index might have the most fun VR controllers I've ever tried. *https://www.theverge.com/2019/5/28/18639084/valve-index-steamvr-headset-knuckles-controllers-preview*, May 2019. Online; accessed 07-02-2021.

[Say07] Carolyn Sayre. 10 questions for Shigeru Miyamoto. *Time*, 41, jul 2007.

[SE99] David Sheff and Andy Eddy. *Game Over, Press Start to Continue: How Nintendo Conquered the World*. CyberActive Publishing, 1999. ISBN: 0-966-9617-0-6.

[Sel17] Mary Ann Sell. View-master history and value. *https://www.antiquetrader.com/collectibles/focusing-view-master-history-value*, Nov 2017. Online; accessed 29-11-2020.

[Shu17] Sid Shuman. PlayStation VR: The ultimate FAQ. *https://blog.playstation.com/2017/10/02/playstation-vr-the-ultimate-faq/*, Oct 2017. Online; accessed 04-02-2021.

[SJH80] Leonard T Suminski Jr and Paul H Hulin. Computer generated imagery (cgi) current technology and cost measures feasibility study. Technical report, COMPUTER SCIENCES CORP ORLANDO FL, 1980.

[SMP+19] Linus Sebastian, Ivan Metelitsa, Alexandre Potvin, Maxine Tamoto, Edzel Yago, Taran Van Hemert, and Colton Potter. 3 VR gamers, 1 CPU - ultimate VR setup! *https://linustechtips.com/topic/901880-3-vr-gamers-1-cpu-ultimate-vr-setup/*, Dec 2019. Online; accessed 04-02-2021.

[Soc17] Virtual Reality Society. History of virtual reality. *https://www.vrs.org.uk/virtual-reality/history.html*, 2017. Online; accessed 29-11-2020.

[squ] Acron: Attack of the Squirrels! VR game for Oculus Quest and Steam. *https://www.resolutiongames.com/acron*. Online; accessed 07-02-2021.

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

[SSO+12] Anthony Steed, William Steptoe, Wole Oyekoya, Fabrizio Pece, Tim Weyrich, Jan Kautz, Doron Friedman, Angelika Peer, Massimiliano Solazzi, Franco Tecchia, et al. Beaming: an asymmetric telepresence system. *IEEE computer graphics and applications*, 32(6):10–17, 2012.

[ste] Steeplechase - videogame by Atari. *https://www.arcade-museum.com/game_detail.php?game_id=9787*. Online; accessed 05-01-2021.

[Sut64] Ivan E Sutherland. Sketchpad a man-machine graphical communication system. *Simulation*, 2(5):R–3, 1964.

[Sut65] Ivan E Sutherland. The ultimate display. *Multimedia: From Wagner to virtual reality*, 1, 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, 1968.

[Tea16] Oculus Team. Oculus rift: Step into the game. *https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game*, Jan 2016. Online; accessed 29-11-2020.

[TNNL19] Lui Albæk Thomsen, Niels Christian Nilsson, Rolf Nordahl, and Boris Lohmann. Asymmetric collaboration in virtual reality. *Tidsskriftet Læring og Medier (LOM)*, 12(20), 2019.

[Ult] Ultraleap. Tracking: Leap Motion Controller. *https://www.ultraleap.com/product/leap-motion-controller/*. Online; accessed 27-03-2021.

[unia] Detonator explosion framework: VFX particles: Unity Asset Store. *https://assetstore.unity.com/packages/vfx/particles/detonator-explosion-framework-1*. Online; accessed 11-04-2021.

[unib] Rain maker - 2d and 3d rain particle system for Unity: Environment: Unity Asset Store. *https://assetstore.unity.com/packages/vfx/particles/environment/rain-maker-2d-and-3d-rain-particle-system-for-unity-34938*. Online; accessed 11-04-2021.

[var15] Oculus DK2 lens - characteristics - V-Rtifacts. *https://vrtifacts.com/oculus-dk2-lens-characteristics/*, Jun 2015. Online; accessed 29-11-2020.

[vir20a] Virtual Boy. *https://en.wikipedia.org/wiki/Virtual_Boy*, Nov 2020. Online; accessed 29-11-2020.

[vir20b] Virtuality (product). *https://en.wikipedia.org/wiki/Virtuality_(product)*, Aug 2020. Online; accessed 29-11-2020.

[Viv16] HTC / Vive. *Vive PRE User Guide*. HTC Corporation, 2016.

[VK07] Rick Van Krevelen. Augmented reality: Technologies, applications, and limitations, 04 2007.

[VR 18] VR Cardboard. About Google Cardboard. *https://www.vrcardboard.ch/en/about-google-cardboard/*, 2018. Online; accessed 29-11-2020.

[VR18] Oculus VR. Introducing Oculus Quest, Our First 6DOF All-in-One VR System, Launching Spring 2019. *https://www.oculus.com/blog/introducing-oculus-quest-our-first-6dof-all-in-one-vr-system-launching-spring-2019/*, Sep 2018. Online; accessed 29-11-2020.

[VRG19] Team VRGC. Best VR party games. *https://vrgamecritic.com/article/best-vr-party-games-vive-rift-psvr*, Oct 2019. Online; accessed 07-02-2021.

[vrh20] The ultimate guide to virtual reality headsets. *https://www.vrs.org.uk/the-ultimate-guide-to-virtual-reality-headsets/*, 2020. Online; accessed 29-11-2020.

[Whe38] Charles Wheatstone. Xviii. contributions to the physiology of vision.—part the first. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical transactions of the Royal Society of London*, (128):371–394, 1838.

[wik18] Maze War. *https://de.wikipedia.org/wiki/Maze_War*, Jun 2018. Online; accessed 06-01-2021.

[wik19] Mech / mecha. *https://en.wikipedia.org/wiki/Mech*, Aug 2019. Online; accessed 27-03-2021.

[wik20a] History of massively multiplayer online games. *https://en.wikipedia.org/wiki/History_of_massively_multiplayer_online_games*, Dec 2020. Online; accessed 06-01-2021.

[wik20b] History of video games. *https://en.wikipedia.org/wiki/History_of_video_games*, Dec 2020. Online; accessed 05-01-2021.

[wik20c] Imlac PDS-1. *https://en.wikipedia.org/wiki/Imlac_PDS-1*, Nov 2020. Online; accessed 06-01-2021.

[wik20d] Lotus Esprit Turbo Challenge. *https://en.wikipedia.org/wiki/Lotus_(video_game_series)#Lotus_Esprit_Turbo_Challenge*, Dec 2020. Online; accessed 05-01-2021.

[wik20e] Magnavox Odyssey. *https://en.wikipedia.org/wiki/Magnavox_Odyssey*, Dec 2020. Online; accessed 05-01-2021.

[wik20f] Maze War. *https://en.wikipedia.org/wiki/Maze_War*, Oct 2020. Online; accessed 06-01-2021.

[wik20g] Mehrspieler. *https://de.wikipedia.org/wiki/Mehrspieler*, Dec 2020. Online; accessed 04-02-2021.

[wik20h] Nintendo 64. *https://en.wikipedia.org/wiki/Nintendo_64*, Dec 2020. Online; accessed 05-01-2021.

[wik20i] Oculus Quest. *https://en.wikipedia.org/wiki/Oculus_Quest*, Nov 2020. Online; accessed 29-11-2020.

[wik20j] Online console gaming. *https://en.wikipedia.org/wiki/Online_console_gaming*, Dec 2020. Online; accessed 06-01-2021.

[wik20k] PDP-1. *https://en.wikipedia.org/wiki/PDP-1*, Dec 2020. Online; accessed 05-01-2021.

[wik20l] Shigeru Miyamoto. *https://en.wikipedia.org/wiki/Shigeru_Miyamoto*, Dec 2020. Online; accessed 05-01-2021.

[wik20m] Space Invaders. *https://en.wikipedia.org/wiki/Space_Invaders*, Dec 2020. Online; accessed 05-01-2021.

[wik20n] Spacewar! *https://en.wikipedia.org/wiki/Spacewar!*, Dec 2020. Online; accessed 05-01-2021.

[wik20o] Spasim. *https://en.wikipedia.org/wiki/Spasim*, Jun 2020. Online; accessed 06-01-2021.

[wik20p] Split screen (video games). *https://en.wikipedia.org/wiki/Split_screen_(video_games)*, Nov 2020. Online; accessed 05-01-2021.

[wik20q] Split screen (video production). *https://en.wikipedia.org/wiki/Split_screen_(video_production)*, Dec 2020. Online; accessed 05-01-2021.

[wik20r] Xbox Live. *https://en.wikipedia.org/wiki/Xbox_Live*, Dec 2020. Online; accessed 06-01-2021.

[wik21a] ARPANET. *https://en.wikipedia.org/wiki/ARPANET*, Jan 2021. Online; accessed 06-01-2021.

[wik21b] Cooperative gameplay. *https://en.wikipedia.org/wiki/Cooperative_gameplay*, Feb 2021. Online; accessed 04-02-2021.

[wik21c] Magnavox Odyssey. *https://de.wikipedia.org/wiki/Magnavox_Odyssey*, Jan 2021. Online; accessed 05-01-2021.

[wik21d] Multiplayer video game. *https://en.wikipedia.org/wiki/Multiplayer_video_game*, Feb 2021. Online; accessed 07-02-2021.

[wik21e] Nintendo Network. *https://en.wikipedia.org/wiki/Nintendo_Network*, Jan 2021. Online; accessed 06-01-2021.

[wik21f] PlayStation Network. *https://en.wikipedia.org/wiki/PlayStation_Network*, Jan 2021. Online; accessed 06-01-2021.

[wik21g] Tennis for Two. *https://en.wikipedia.org/wiki/Tennis_for_Two*, Jan 2021. Online; accessed 05-01-2021.

[Wil19] Chris Wiltz. The story of Sega VR: Sega's failed virtual reality headset. *https://www.designnews.com/electronics-test/story-sega-vr-segas-failed-virtual-reality-headset*, Mar 2019. Online; accessed 29-11-2020.

[WLL+11] Paul Wighton, Tim K Lee, Harvey Lui, David McLean, and M Stella Atkins. Chromatic aberration correction: an enhancement to the calibration of low-cost digital dermoscopes. *Skin Research and Technology*, 17(3):339–347, 2011.

[WPB+20] Rui Wu, Jeevitha Pandurangaiah, Grayson Morgan Blankenship, Christopher Xavier Castro, Shanyue Guan, Andrew Ju, and Zhen Zhu. Evaluation of virtual reality tracking performance for indoor navigation. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1311–1316. IEEE, 2020.

[WSFF90] Elizabeth M Wenzel, PK Stone, SS Fisher, and Scott H Foster. A system for three-dimensional acoustic 'visualization' in a virtual environment workstation. In *Proceedings of the First IEEE Conference on Visualization: Visualization90*, pages 329–337. IEEE, 1990.

[YWC18] Keng-Ta Yang, Chiu-Hsuan Wang, and Liwei Chan. Sharespace: Facilitating shared use of the physical space by both VR head-mounted display and external users. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 499–509, New York, NY, USA, 2018. Association for Computing Machinery.

Appendix: Questionnaires

Geschlecht: ☐ männlich ☐ weiblich ☐ sonstiges Alter: _____

Haben Sie bereits Erfahrung mit VR (Virtual Reality)? ☐ Ja ☐ Nein

Haben Sie schon einmal in einer immersiven Simulation teilgenommen, also an einer, in der Sie echter Teil des Geschehens waren, statt dieses nur als Zuschauer zu durchleben? ☐ Ja ☐ Nein

Hat Ihnen das Spiel / die Simulation gefallen? ☐ Ja ☐ Nein

Welche Rolle haben Sie in der Simulation eingenommen? ☐ In VR Simulation ☐ An der Konsole

Wie wohl haben Sie sich in Ihrer Rolle wohl gefühlt?

☐	☐	☐	☐	☐
Sehr unwohl	Eher unwohl	Neutral	Eher wohl	Sehr wohl

Wie hoch war für Sie das Gefühl der Immersion?

☐	☐	☐	☐	☐
Absolut nicht	Kaum	Etwas	Stark	Sehr stark

Haben Sie sich ihrem Gegenüber ausgeliefert gefühlt? ☐ Ja ☐ Nein

Wie viele Effekte wurden von Ihrem Gegenüber in etwa ausgelöst? _____

Welche einzelnen Effekte konnten Sie erkennen?

_____

Waren die Effekte für sie eher unangenehm und störend, oder haben diese die Simulation lebendiger gestaltet? ☐ Unangenehm ☐ Angenehm

In wieweit hat die indirekte Interaktion mit einem realen Menschen ihr Erlebnis beeinflusst?

☐	☐	☐	☐	☐
Sehr negativ	Eher negativ	Keinen Einfluss	Eher positiv	Sehr positiv

Würden Sie gerne einmal die Rolle Ihres Gegenübers einnehmen? ☐ Ja ☐ Nein

Würden Sie die Simulation als lustige Gruppenaktivität einstufen? ☐ Ja ☐ Nein

Würden Sie an so einer Simulation gerne öfter teilnehmen (z.B. mit Freunden)? ☐ Ja ☐ Nein

Gibt es etwas das Sie den Entwicklern mitteilen wollen, oder haben Sie persönliche Kommentare zu Ihrem Erlebnis?

_____

Gibt es Verbesserungsvorschläge, deren Umsetzung sich positiv auf den Realismus oder die Immersion auswirken würde?

_____

Figure A.1: Jumper questionnaire.

Geschlecht: ☐ männlich ☐ weiblich ☐ sonstiges Alter: _____

Haben Sie bereits Erfahrung mit VR (Virtual Reality)? ☐ Ja ☐ Nein

Haben Sie schon einmal in einer immersiven Simulation teilgenommen, also an einer, in der Sie echter Teil des Geschehens waren, statt dieses nur als Zuschauer zu durchleben? ☐ Ja ☐ Nein

Hat Ihnen das Spiel / die Simulation gefallen? ☐ Ja ☐ Nein

Welche Rolle haben Sie in der Simulation eingenommen? ☐ In VR Simulation ☐ An der Konsole

Wie wohl haben Sie sich in Ihrer Rolle wohl gefühlt?

☐	☐	☐	☐	☐
Sehr unwohl	Eher unwohl	Neutral	Eher wohl	Sehr wohl

Wie sehr haben Sie sich „in Kontrolle“ der Situation gefühlt?

☐	☐	☐	☐	☐
Gar nicht	Kaum	Etwas	Doch etwas	Sehr

Wie viele Effekte haben Sie in etwa ausgelöst? _____

Hatten Sie ein Gefühl der Überlegenheit? ☐ Ja ☐ Nein

Was würde ein außenstehender Beobachter über ihre Art der Interaktion sagen? Haben sie Ihre „Macht missbraucht“? ☐ Ja ☐ Nein

Welche Methode der Interkation haben Sie bevorzugt? ☐ Buttons ☐ LEAP Motion ☐ Tablet

Würden Sie gerne einmal die Rolle Ihres Gegenübers einnehmen? ☐ Ja ☐ Nein

Würden Sie die Simulation als lustige Gruppenaktivität einstufen? ☐ Ja ☐ Nein

Würden Sie an so einer Simulation gerne öfter teilnehmen (z.B. mit Freunden)? ☐ Ja ☐ Nein

Gibt es etwas das Sie den Entwicklern mitteilen wollen, oder haben Sie persönliche Kommentare zu Ihrem Erlebnis?

_____

_____

_____

Gibt es Verbesserungsvorschläge, deren Umsetzung sich positiv auf den Realismus oder die Immersion auswirken würde?

_____

_____

_____

Figure A.2: Controller questionnaire.