UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

# Natural Menu Interactions in VR with Leap Motion

Masterarbeit

Zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Informatik

vorgelegt von

Björn Zeutzheim

| | |
|---|---|
| Erstgutachter: | Prof. Dr. Stefan Müller<br>(Institut für Computervisualistik, AG Computergraphik) |
| Zweitgutachter: | Nils Höhner, M.Sc.<br>(Institut für Wissensmedien, IWM) |

Koblenz, der 6. September 2019

## Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

| | Ja | Nein |
|---|---|---|
| Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. | ✓ | ☐ |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

(Ort, Datum) (Unterschrift)

## Abstract

With the appearance of modern virtual reality (VR) headsets on the consumer market, there has been the biggest boom in the history of VR technology. Naturally, this was accompanied by an increasing focus on the problems of current VR hardware. Especially the control in VR has always been a complex topic.

One possible solution is the Leap Motion, a hand tracking device that was initially developed for desktop use, but with the last major software update it can be attached to standard VR headsets. This device allows very precise tracking of the user's hands and fingers and their replication in the virtual world.

The aim of this work is to design virtual user interfaces that can be operated with the Leap Motion to provide a natural method of interaction between the user and the VR environment. After that, subject tests are performed to evaluate their performance and compare them to traditional VR controllers.

## Zusammenfassung

Mit dem Erscheinen moderner Virtual Reality (VR) Headsets auf dem Verbrauchermarkt, gab es den bisher größten Aufschwung in der Geschichte der VR Technologie. Damit einhergehend rücken aber auch die Problematiken aktueller VR Hardware immer mehr in den Vordergrund. Insbesondere die Steuerung in VR war schon immer ein komplexes Thema.

Eine mögliche Lösung bietet die Leap Motion: Ein Hand-Tracking Gerät, welches ursprünglich für den Desktop-Einsatz entwickelt wurde, aber mit dem letzten größeren Softwareupdate an üblichen VR Headsets angebracht werden kann. Dieses Gerät ermöglicht ein sehr genaues Tracking beider Hände und aller Finger. Damit ist es möglich, diese vollständig in der VR Welt zu replizieren und zur Steuerung zu verwenden.

Ziel dieser Arbeit ist es, virtuelle Benutzeroberflächen zu entwerfen, die mit der Leap Motion bedient werden können. Dies soll eine natürliche Interaktion zwischen dem Benutzer und der VR-Umgebung ermöglichen. Danach werden mit Hilfe einer Demoanwendung Probanden-Tests durchgeführt, um ihre Leistung zu bewerten und mit herkömmlichen VR-Reglern zu vergleichen.

## Table of contents

# 1. Introduction

## 1.1 Motivation

In the last years, virtual reality technologies have made great advancements and high quality, affordable VR systems are now available on the consumer market. Not only VR headsets, but also more advanced tracking technologies to work with in virtual environments have been developed or improved significantly. One of those is the Leap Motion device. This device uses two infrared cameras and computer vision, to track the user's hands in 3D, allowing them to be replicated in the virtual environment. Compared to classic VR controllers which the user has to hold in his hands, this technology allows controlling virtual environments without any additional devices.

The aim of this work is to evaluate how this technology can be used to control virtual reality applications, by operating virtual user interfaces with the Leap Motion. In order to restrict the scope of this evaluation, the user interfaces will be restricted to menus with buttons, sliders and labels.

## 1.2 Outline

First, this work will describe the fundamentals, required to implement user interfaces in virtual reality. This includes information about virtual reality systems, the Leap Motion and Unity, which will be used to implement the user interfaces.

After that, the implementation will be explained step by step and how each aspect of the menu has been designed and which tasks it should handle.

Next comes a description of the demo application and the test setup. After that the test results will be presented and evaluated in detail.

Finally, possible future work will be discussed and a conclusion about our research drawn.

## 1.3 Other work

In the past, there have been various efforts to create virtual reality interfaces which could be operated with data gloves. These gloves are expensive devices with many sensors that track the hands pose and transmit this data to the computer. Regretfully, no scientific report could be found, but various sources report about those past efforts [1] [2].

Recently some organizations are researching modern data glove variants again which also include tactile feedback and force feedback, for example haptx [3] or BeBop [4]. This allows the user to actually feel and grab the virtual objects.

# 2. Fundamentals

## 2.1 Virtual reality systems

Virtual reality systems allow users to enter specifically created virtual environment. Compared to augmented reality, they completely cut off the user from the real world by simulating 3D vision with an HMD (head mounted display). 3D sound from a surround sound system or headphones may be used for additional immersion. Depending on the quality of both, the user may be provided with a quite real experience.

The HMDs used for virtual reality usually consist of one large or two smaller screens and a convex Fresnel lens in front of each eye. This setup allows showing each eye an individual view with a wide FOV (field of view). Another very important part of the HMD is any kind of tracking device. This device is responsible for tracking the user’s head movement within the real world. This information is then used to move the view or character within the virtual world as well. By making sure that every position and rotation change of the real world is replicated in the virtual world, the user will feel as if moving inside the virtual world.

There are many different tracking technologies which can be used for this purpose. In case of the HTC Vive, the SteamVR tracking technology is used. It generally uses two base stations which sweep the room with infrared signals. These signals are picked up by multiple sensors on the HMD or VR controllers, which allows the processor to triangulate the HMDs position and orientation in relation to the base stations [5]. The benefit of this technology is that there is theoretically no limit to the amount of tracked devices in the room, while also allowing a large area of interaction.

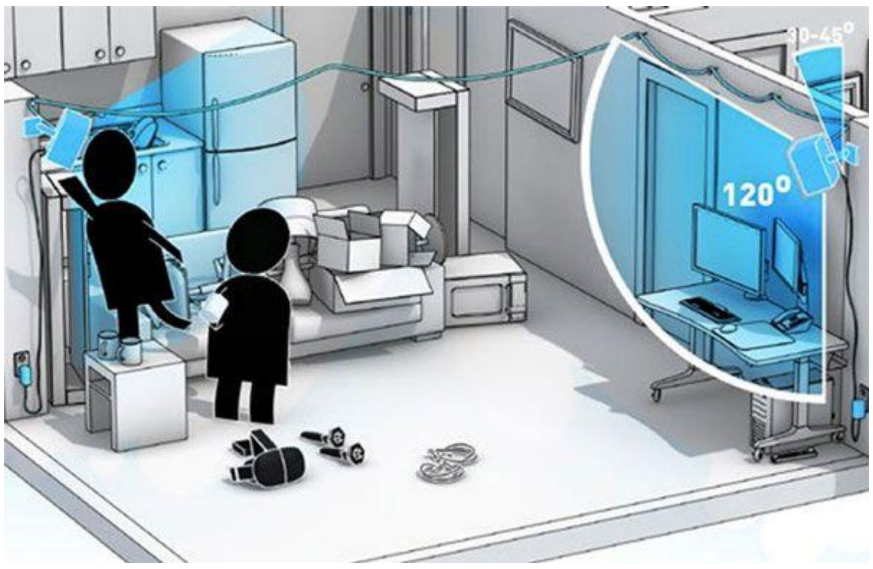

*Figure 1 SteamVR Tracking setup graphic*

Other than the HMD, every proper virtual reality system also provides a set of VR controllers. In general, they integrate with the tracking system of the HMD and therefore are precisely tracked in 3D space.

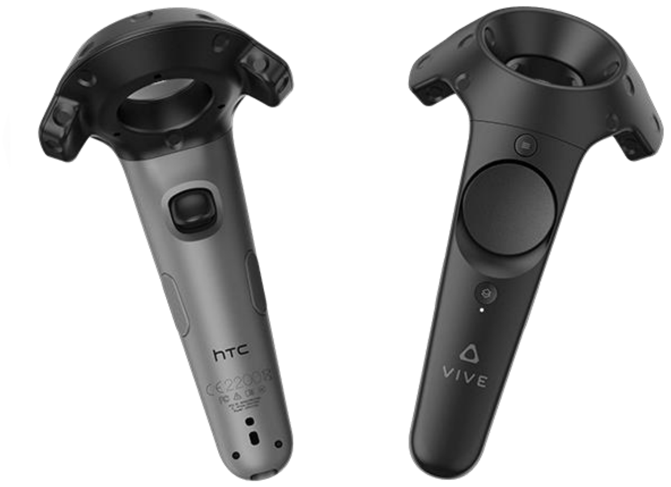

*Figure 2 HTC Vive VR controllers*

## 2.2 Leap Motion

The Leap Motion is a device developed by Leap Motion Inc. which was released in 2012 / 2013.

Its hardware consists of two wide angle (150°) infrared cameras plus a few infrared LEDs to provide lighting and is connected by USB. The LEDs are pulsating with the same frequency as the camera framerate and the camera's sensor is adjusted to capture primarily their wavelength. This allows using stronger IR light while optimizing power consumption, heat emission and lowering interference with other IR applications.

The grayscale video streams recorded by these cameras is then transferred to the Leap Motion tracking software running on the computer.

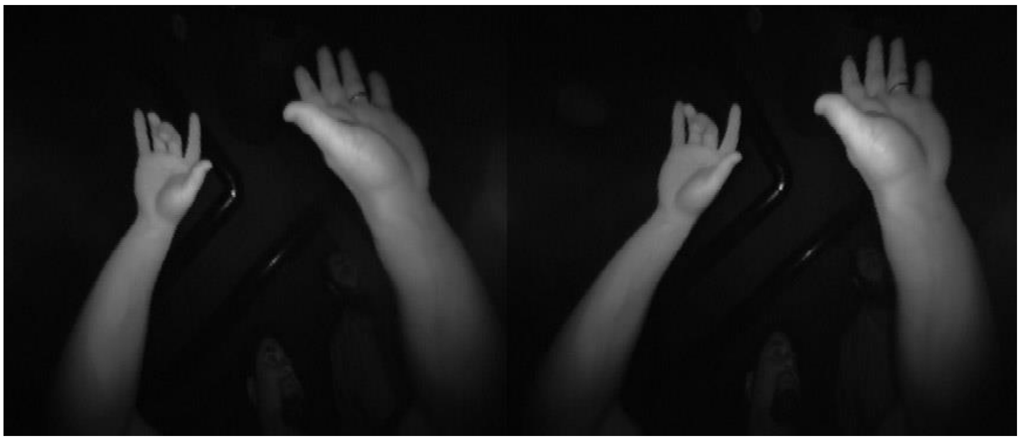

*Figure 3 Leap Motion Infrared Images [6]*

“The tracking software then uses computer vision to infer a realistic hand model from the incoming data. To achieve this, it first tries to isolate background objects and ambient lighting. After that, the tracking layer matches the data to extract tracking information such as fingers and tools. The tracking algorithms interpret the 3D data and infer the positions of occluded objects. Filtering techniques are applied to ensure smooth temporal coherence of the data. The Leap Motion Service then feeds the results – expressed as a series of frames, or snapshots, containing all of the tracking data – into a transport protocol” [6].

Originally, the Leap Motion was designed for use in desktop environments with the device being placed on a table. In February 2016 the new tracking software called Orion was released, which supports tracking the hands while the Leap Motion is mounted to the front of a VR headset.

## 2.3 Unity

Unity is primarily a game and 3D graphics engine developed by Unity Technologies since 2004. First commercial game engines were developed for a particular game or had to be customized for another game using it. This changed a lot when engines like the Unreal Engine allowed customizing most of the game logic as part of the engines editor itself by using user code or scripts.

Unity stores the game environment as scenes, which are a tree structure of so-called GameObjects. These GameObjects serve as containers for multiple, various types of components. Every component can have a different functionality with Unity providing a lot of basic components for tasks like rendering meshes, emitting sound, physics and more. Users can implement their own components by writing C# classes which define their properties and behavior.

In the past Unity used a Mono runtime to execute .NET code provided by the user on the different platforms that Unity supports. In its current version, the C# code is compiled to native C++ code, to boost performance [7]. Using the C# code it is also possible to modify the Unity engine’s editor itself, which is one of the reasons that make it one of the most popular game engines right now.

## 3. Concept

As already mentioned before, the user interfaces will be restricted to menus with buttons, sliders and labels. Implementing a whole user interface framework is a difficult and very work intensive task and would overpass the scope of this evaluation. Focusing on only simple, structured menus with only those 3 components is a manageable task and any more complex user interface component is usually composed of or an extension of these elements.

One of the main goals of the implementation is to allow attaching the menu to different anchors, especially to one of the hands. The idea is to make the menu appear when looking at the palm of the left (or right) hand. This would allow quick access to the menu and makes it possible to focus on the environment while still keeping the menu within the field of view.

Attaching the menu to the hand also gave the inspiration, to not only use standard rectangular menus, but to try finding an ergonomic design that would improve usability of the menu. The solution to that idea was implementing an arc shaped menu (Figure 4). This type of menu would be centered around the user's palm, allowing easy access to all UI elements. This however requires that the system can generate and handle these shapes. This includes click detection, mesh generation and layouting.

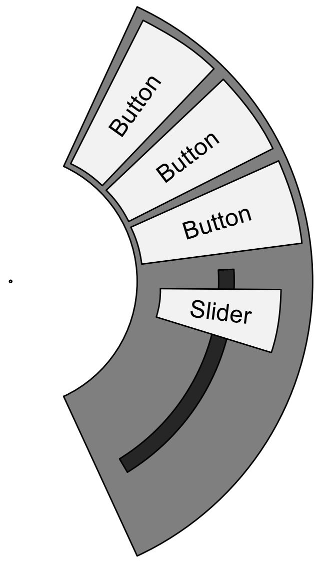

*Figure 4 Design concept for arc menu*

Another possible attachment point for the menu is a static point in the world. This is an option that could integrate well with games, for example by attaching the menu to a computer terminal. Another option would be attaching the menu relative to the HMD as a floating interface, which follows the users gaze.

# 4. VR Menu Implementation

In order to provide a dynamically adjustable, compact and easy to use user interface framework, various components are necessary which will be explained in the following sections. These components can be separated by the following tasks:

- Button component with touch handling
- Mesh generation
- Interface layouting
- Text and image rendering
- Menu management and transitions

In order to handle different shapes of buttons and UI layouts, most of the components have been implemented in a generic way so that their subclasses can handle these cases. In the demo application, the two shape variations of arc and rectangular menus have been implemented.

## 4.1 Button Component

Every pushable element of the UI uses an extension of the base MenuItem component at its root. This component is responsible for tracking the cursors, checking whether they are eligible for interaction with the button and various other smaller tasks.

In order to check for cursor interactions with the element, the following process is executed with each frame update: First it starts by creating a list of cursors that are facing it. This is determined by comparing the scalar product of the cursor direction with the depth axis of the button (negative Y-axis). Also, all cursors that are currently pressing the button, but not facing it anymore, are added to the list as well.

After that, the distance of each cursor to the borders of the button in the XZ-plane is calculated. If it is less than or equal to zero it means the cursor is inside the button, if it is greater than zero the cursor is besides the button. By comparing the distance with the cached values of the previous frame, it is possible to check if a cursor is pressing the button:

- The last and current frame’s XZ distance must be less than or equal to zero
- The last frame’s Y distance must be larger than zero
- The current frame’s Y distance must be less than zero

If these conditions are fulfilled, the cursor is considered active and the first two rules are not checked for this cursor any more until the cursor releases the button again.

Becoming active however still does not make the button enter the pressed state. Once at least one cursor is active – meaning it is touching the button – the button will take on the lowest Y distance of these active cursors as its press depth, capped by the maximum press depth constant of the component. This press depth value will also make the button mesh child object of the button shift in the Y axis, creating the visual effect of the button being pressed.

Once a button’s press depth reaches the maximum value, the actual button press is registered and a press event is fired. After that, the button will stay in pressed state and cannot trigger again until it is fully released, meaning no more cursors are active on the button anymore.

## 4.2 Mesh Component

The mesh component is responsible for creating dynamic meshes to visualize buttons as well as UI backgrounds. It also takes care of providing visual hints of the interaction state with button elements. In the demo application this is achieved by rendering a colored border around the edges of the shape. Depending on the distance of the closest cursor to the button, the border becomes more opaque, until it becomes fully visible and an additional highlight, if a cursor is in front of it. In the same way the highlight becomes more prominent once a cursor touches the button and eventually toggles it into the pressed state.

The mesh for each element is generated dynamically. In case of the rectangular menu it is a simple quad, for the arc menu it is an arc segment. By generating those meshes instead of using predefined ones, it is possible to combine any set of UI elements with any dimensions.

## 4.3 Menu Layout Component

In order to efficiently construct user interfaces – possibly even with varying number of elements or sizes – it is necessary to be able to dynamically position and resize each UI element. The most common type of layouts are simple directional containers, where all its child elements are placed in a row or column and are scaled to take up the available space. This task is performed by subclasses of the basic MenuLayout component, depending on the shape (arc or rectangular) and the direction (row or column) of the layout. Layouts also allow child elements to provide weights, which define how much of the available total space they get assigned.

When any of the variables affecting the layout are changed (number of child elements, their weights, size of the container, etc.), the layout component will automatically recalculate the position and sizes of each element.

## 4.4 Menu Canvas & Label Components

Finally, the canvas and label components are responsible for displaying textual information. In the same way as the layout components, they automatically ensure that the text is confined within the bounds of the button they are assigned to.

## 4.5 Menu manager component

The menu manager component is the parent component to multiple menus. As a root component for the menu, it takes care of providing them with some common data and is also responsible for handling multiple child menus and transitions between them.

## 4.6 Menu anchors

The menu anchor components are possible attachment points for the menu root. It allows the menu to be dynamically attached to different game objects.

# 5. Evaluation

## 5.1 Test setup

So far, only the underlying concepts and implementation outlines for the VR menus have been described. However, the aim of this work is the evaluation of interactions in VR with the Leap Motion in comparison to classic handheld VR controllers. For this purpose, a simple demo has been created, where testers are able to try out different scenarios with both interaction options.

The demo consists of a particle simulation where various parameters can be controlled with the VR menu in real-time. These options include

- Speed
- Lifetime
- Emission shape
- Color
- Stopping and pausing the simulation

These options are separated into various submenus, to demonstrate menu transitions and to restrict the menu size. If it were too large, the menu items would have to be very small or the whole menu would be too big to properly fit the screen. Either options would make it difficult for the user to use the menu efficiently. The demo scene menu structure is shown in Figure 5 Demo scene menu structure.

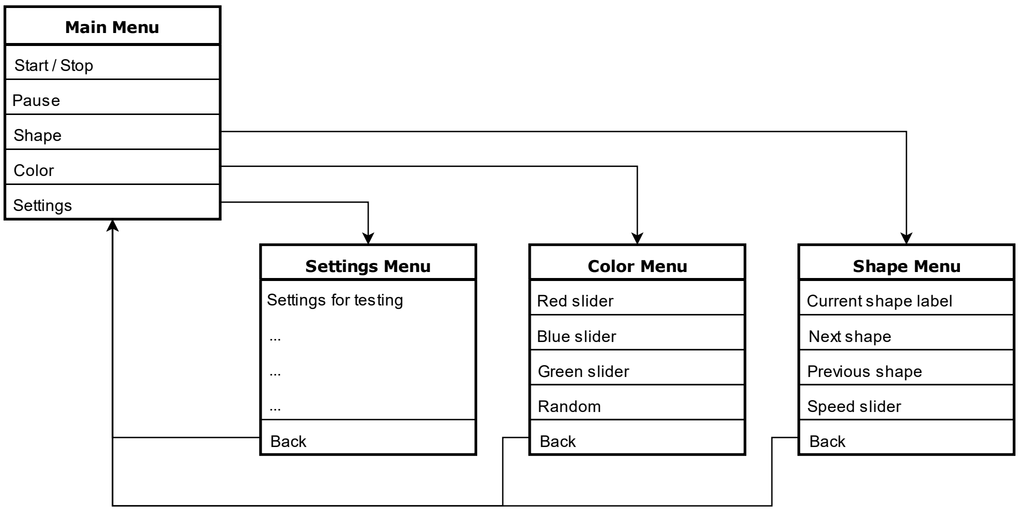

*Figure 5 Demo scene menu structure*

The test environment itself consists of a single room with the particle simulation and a nearby pillar, which is used as an example for a static menu anchor point.

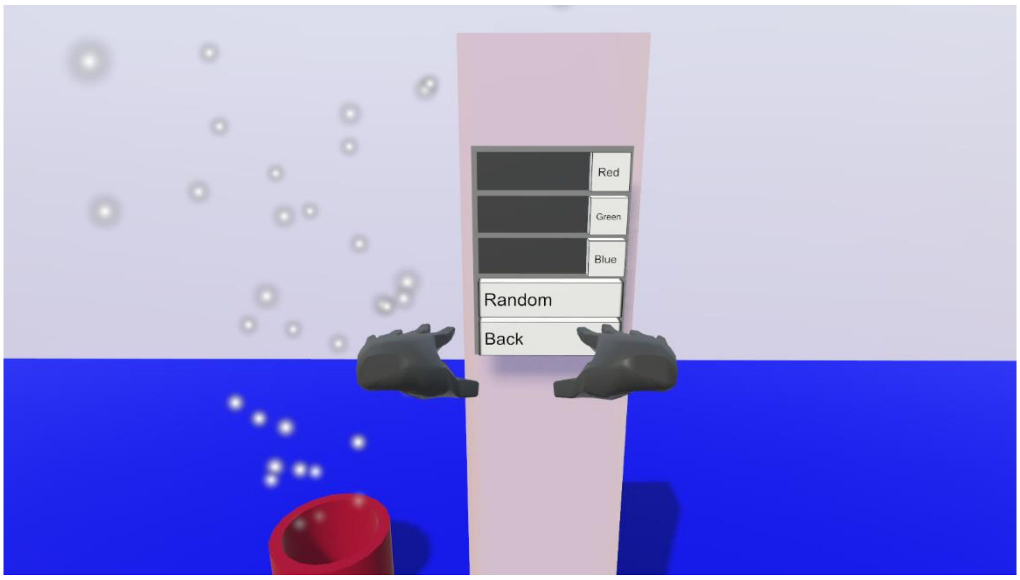

*Figure 6 Demo scene with menu pillar and particles*

## 5.2 Test case

Every test starts with a short briefing about what the tester may expect and what he is about to see. After that they put on the VR headset and the head straps are adjusted so the HMD properly fits the person's head. This is an important step to ensure the image quality is good and rendered text can be read properly.

After that the tester may freely interact with the VR environment and the menu, which is initially mounted to the static anchor on the pillar. This step allows the tester to get familiar with VR if it is their first time and get a feel for the Leap Motion control and virtual hand representation. During this time, the tester may also be provided with additional guidance, as there is no thing such as a usage manual built into the demo.

After getting used to the initial setup (rectangular menu anchored to the pillar), the tester can try out the different menu anchors as well as the arc menu variant. With each variation, the user should execute some simple tasks with the menu, like changing the color or speed of the particles.

After sufficiently testing the menu by using Leap Motion control, the VR controller is handed over to the user to try out interacting with the menu by using a classic laser pointer type of control. This is a very important step to make sure that users that never interacted with VR before can properly evaluate the Leap Motion control in comparison to VR controllers. It is also relevant for testers who are already familiar with VR, to make sure they can compare only the control method, while leaving out other VR menus they already know from their evaluation.

After this, the tester will remove the HMD and fill out the questionnaire.

## 5.3 Test Results Evaluation

The tests covered 13 people with an age between 23 and 47 years and 10 people had very little to no prior experience with VR, while 3 were very familiar with it. Furthermore, there was only a single person who had proper experience with hand tracking of devices such as the Leap Motion or Microsoft HoloLens.

Because the menu in its hand attached variant was fixed to the left hand and had to be operated with the right hand, the participants have been checked for left-handedness to evaluate potential issues. The responses in the questionnaire of the 3 left-handed testers reported few to medium issues when using the menu. Therefore, an option to swap hands for left-handed people should be included when using this type of user interface in a real-world application.

About half of the participants rated the experience with the Leap Motion control positive, the remaining half with a neutral score. However, this is only a subjective perception and not a scientific observation.

When comparing the hand attached menu to the statically anchored menu, the responses covered a wide spectrum, with a tendency to the hand attached menu. Pretty much the same result could be observed when comparing the arc shaped menu to the rectangular version. Most users found the size of the buttons to be appropriate, but for 2 testers they were too small and posed a problem during operation.

The next questions are centered around comparing the Leap Motion control option with the VR controller one. 9 out of the 13 testers (69%) found the Leap Motion to be the more intuitive control option (Figure 7) and the same amount – but not the same people – would prefer to have this option available in general for VR applications, if it’s applicable.

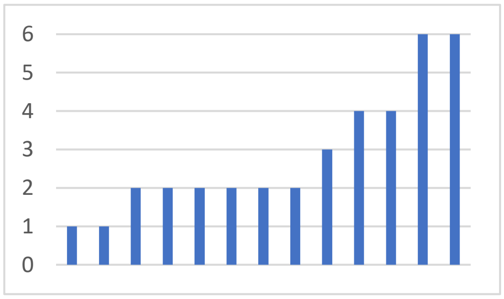

*Figure 7 User's preferred control method scored from 1 for Leap Motion to 6 for VR Controller. Each bar represents a single tester.*

In the next question, the participants were asked what they liked or disliked about the VR controller option. The most common pros were: A higher precision, allowing to see exactly which option will be clicked when pressing the controller button and no dependency on depth perception. One of the primary cons was that the controller had to be held in the hand at all times, which is uncomfortable for the user. Also, some users complained about not knowing which button to press at first.

Finally, the testers were asked about what they liked, disliked or would want to improve about the Leap Motion control option.

The first issue that users reported, which could also be observed externally, is depth perception. The problem is that, without a physical response of the menu, the testers often did not press the buttons deep enough to register a press, lost contact during slider movement or in the beginning not even touched the button at all. This often led to confusion in the beginning, but after some testing most participants were able to get the correct feeling for the depth quickly. A total of 5 users reported this issue, while 2 reported the missing physical feedback as troublesome.

The next issue is tracking problems originating from the Leap Motion tracking device. This issue is the problem that most participants experienced. Because the Leap Motion device with its cameras is mounted in front of the HMD, it is difficult to track any gesture where the user is pointing a finger (typically the index finger) directly in front of him, because the hand will occlude the finger itself. An example for this issue can be seen in Figure 8. Without the shadow – which is not a possible tracing input – it is impossible to tell whether a finger is extended or not. This will cause the Leap Motion Tracking Software to either track the hand as a closed fist or make the pointing finger to jump between extended and retracted state erratically, because the software has not enough reliable tracking data.

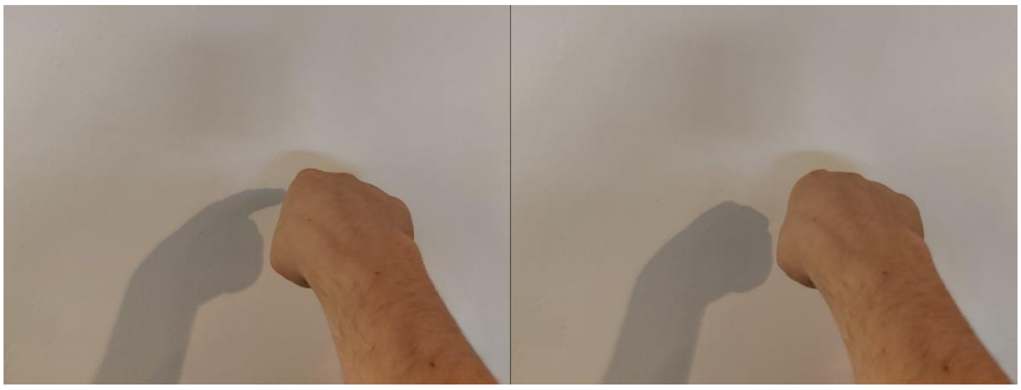

*Figure 8 Index finger tracking issue.*
*Only the shadow makes it clear whether a finger is extended or not*

One possible solution for this issue is to make users point their finger not directly away from them, but in a diagonal upward direction. This will adjust the angle towards the camera and makes it possible for the Leap Motion to properly track the fingers again (see Figure 9). Only a few users used that gesture by default, and many had issues at first and had to be instructed on how to circumvent this tracking issue.

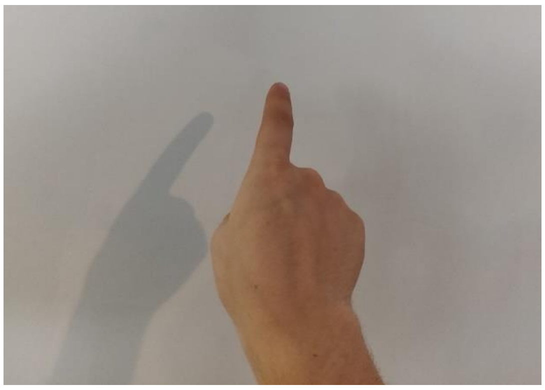

*Figure 9 Proper view to track fingers*

The other solution would be to mount the Leap Motion device higher on the HMD with a slight tilt downward. This would help to increase the angle of the camera towards the finger, making it possible for the Leap Motion software to distinguish between extended and retracted fingers again. However, the mounting toolkit provided with the Leap Motion is not suited for this type of attachment, making this a difficult option.

There were also a few positive things that users reported about the Leap Motion controls. A few testers liked, that getting rid of the need for a VR controller opened up the hands and increased the feeling of immersion, which was one of the aims of this work.

Another positive feedback received the hand attached menu: It allows users to operate the menus, while keeping their work area focused (in this case the particle simulation).

One of the improvements suggested by testers was to allow grabbing menus to place them into the world. Coincidentally, this was planned for the menus in the beginning, however it could not be realized due to implementation difficulties and time restrictions.

## 6. Conclusion

Using the Leap Motion to control user interfaces proved to be an interesting approach to handle VR interactions. By completely removing the need for additional handheld devices, while also allowing the user percept his own hands in the virtual environment, the feeling of immersion into the virtual reality can be improved significantly. This matches the initial feedback of many test users when they notice their own hands to appear in the VR environment.

However, the tests also showed a lot of issues that have to be resolved in order for the technology to be useful in more applications, especially the tracking issues with pointing fingers. This is an issue, where the hardware rather than the software needs to be adjusted. One potential improvement would be increasing the distance between the Leap Motion cameras and embedding them on the very top of the HMD, like for example with the Microsoft HoloLense.

Finally, there are a lot of applications where using VR controllers makes more sense, especially in the gaming sector. For example, a VR controller has many buttons that can be pressed instantly regardless of this situation. This is very important for games and applications that need real time interactions. Another example would be applications where the user is expected to hold something in his hands, like a sword.

## Bibliography

[1] "NASA Science: Whatever happened to ... Virtual Reality?," [Online]. Available: https://science.nasa.gov/science-news/science-at-nasa/2004/21jun_vr. [Accessed 03 09 2019].

[2] "Back to the 90s – Die bizarre Welt der Vintage VR," [Online]. Available: https://www.aspekteins.com/back-to-the-90s-die-bizarre-welt-der-vintage-vr/. [Accessed 03 09 2019].

[3] "haptx," [Online]. Available: https://haptx.com/technology/. [Accessed 29 08 2019].

[4] "BeBop Sensors," [Online]. Available: https://bebopsensors.com/. [Accessed 29 08 2019].

[5] "TRIAD Semiconductor SteamVR Tracking," [Online]. Available: https://www.triadsemi.com/steamvr-tracking/. [Accessed 28 8 2019].

[6] Leap Motion, "How does the Leap Motion controller work?," 9 August 2014. [Online]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/.

[7] Unity Technologies, "How IL2CPP works," 19 08 2019. [Online]. Available: https://docs.unity3d.com/Manual/IL2CPP-HowItWorks.html.