A Modular Avatar Animation System Based on Foot Tracking for Co-Located Virtual Reality Environments

Daniel Rammer

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2017

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 22, 2017

Daniel Rammer

Contents

Abstract

The so-called second wave of virtual reality (VR) is the result of technological advances and ongoing research over the past 50 years. It entails a variety of consumer devices and a great diversity of new content. While most current VR applications focus on single-user interaction, it is highly likely that multi-user applications are going to gain in importance. Hence, the representation of users in VR environments becomes a more crucial factor for such VR experiences. That especially applies to co-located settings, where multiple users share both the virtual and the real space. The way how avatars are visualized strongly influences their perception and therefore the feeling of presence.

The goal of this thesis is to develop a modular tracking and animation system for VR environments, which should reveal further insights within this domain. Expert interviews are conducted to detect possible improvements to the introduced approach. The outcome of this thesis should facilitate the examination of avatar visualization and presence in co-located VR environments.

Kurzfassung

Die so genannte zweite Welle von Virtual Reality (VR) ist das Ergebnis technologischer Entwicklungen und stetiger Forschung in diesem Feld der vergangenen 50 Jahre. Sie bringt eine große Bandbreite von Produkten für den Endverbraucher sowie eine Vielfalt an neuen Inhalten mit sich. Obwohl sich die meisten aktuellen VR-Anwendungen auf Single-User-Interaktion konzentrieren, ist es sehr wahrscheinlich, dass Multiuseranwendungen signifikant an Bedeutung gewinnen werden. Die Darstellung von Nutzern in VR-Umgebungen wird daher immer wichtiger. Das gilt insbesondere für Co-Located Settings, bei denen mehrere Benutzer sowohl den virtuellen als auch den realen Raum teilen. Die Art der Darstellung von Avataren beeinflusst ihre Wahrnehmung und damit auch die Social Presence, das Gefühl mit jemandem zu sein, stark.

Ziel dieser Arbeit ist es, ein modulares Tracking- und Animationssystem für VR-Umgebungen zu entwickeln, welches weitere Einblicke in dieses Gebiet ermöglicht. Um Weiterentwicklungsmöglichkeiten des vorgestellten Ansatzes zu ermitteln, werden Experteninterviews durchgeführt. Das Resultat dieser Arbeit soll die Untersuchung von Avatar-Visualisierung und Presence in Co-Located VR Settings erleichtern.

Chapter 1

Introduction

Most virtual reality (VR) technologies focus on single user interaction, therefore, the user's visual representation—the avatar—seems less crucial and is often neglected. This changes when it comes to multiuser applications. In any kind of virtual multiuser environment, the visual representation of a user or player—the avatar—is highly relevant for the perception of one another. Tracking devices, which are often available with present consumer-ready VR products, can be utilized to control avatars more realistically. For large areas, it would be feasible to utilize floor-based crowd tracking solutions to integrate tracked feet positions and improve the animation of users' avatars. Foot tracking, however, has been less explored in these contexts and therefore is investigated in this work.

1.1 Background

The multiplayer game *CARGO* (see figure 1.1), which was developed by Lorenz Krautgartner and the author of this thesis, served as the inspiration for this endeavor. It is a playful co-located hybrid VR experience in which two VR players cooperate with multiple non-VR players in the same room. The utilized infrastructure is equal to the one utilized for this thesis. *CARGO*'s avatars, in contrast to those in this work, are simple rigid models consisting of two parts, body and head. They simply follow the player's movement and rotation; in addition, an avatar's head pitch follows the player's basic head movements (nodding is enabled). During the development of *CARGO*, it was already noticed that the nodding of an avatar's head has a large impact on its perception.

PHARUS [16], the utilized 2D tracking system for *CARGO*, not only provides the positions of tracked users but also the positions of their feet. This information however was never used in former projects. That is where this work has its point of contact.

1.2 Objectives

The goal of the practical contribution of this thesis is to present a scalable, modular inverse kinematic avatar animation system (*MIKA*) for co-located VR environments which is capable of controlling avatars based on 2D foot tracking data. The option

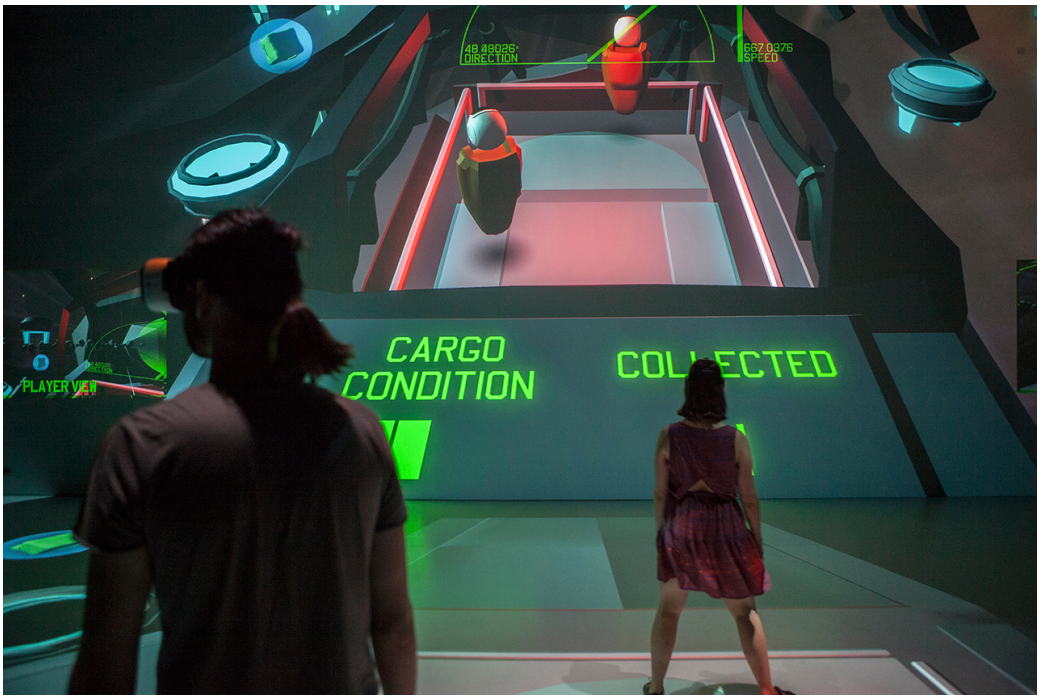


Figure 1.1: *CARGO* in action. Two VR players wearing *GearVRs* are controlling the spaceship. The two orange rigid avatars appear on the wall projections and follow the movements (and rotations) of the players.

to exchange tracking sources as well as the animated avatars represents the modular character of this work.

Two tracking solutions will be realized. First, a 2D tracking approach, which requires specialized solutions to compensate the lack of data, such as the actual height position of tracked feet, will be implemented. And second, a consumer-ready, six-degrees-of-freedom product will be added. Both implementations will be compared and evaluated in expert interviews.

1.3 Acknowledgement

The hardware and the infrastructure necessary for testing and evaluating the implementation was provided by *Playful Interactive Environments*, a research group of the *University of Applied Sciences Upper Austria*.[1][2] Special thanks to Jeremiah Diephuis for the consulting and supervision throughout the whole process and also to Dr. Christoph Anthes and Alexander Wilhelm for the valuable feedback and input given during the evaluation.

[1] http://pie.fh-hagenberg.at
[2] https://www.fh-ooe.at

1.4 Research Questions

The aim of this thesis is the validation of the capabilities of a custom foot tracking system as a data source for avatar animation in VR. It should be clarified to what extent it is possible and reasonable to utilize a floor-based 2D foot tracking system to control users' avatar walking animations. Furthermore, it will be explored if and how such an approach can be utilized to foster the social presence in a co-located virtual environment. Additionally, the performance of the 2D implementation will be compared to a state-of-the-art 6DoF tracking system.

1.5 Structure

This work is structured in seven chapters. Chapter 2 provides an overview of the development of virtual reality and shows present-day technologies and their limitations. Furthermore, the VR in multi-user scenarios is briefly discussed. Chapter 3 presents the concepts of presence in the context of VR. Chapter 4 explores various related visualization topics, reaching from character animations to optimized rendering for mobile VR headsets. Chapter 5 introduces implementation details. It shows which hardware is utilized and how the whole infrastructure is designed. The overall software architecture is outlined, and specific parts of the implementation are explained in detail. Some of the emerging advantages and disadvantages are already indicated in this part of the work. Chapter 6 presents the expert evaluation of the project. It contains the description and the results of the approach. Finally chapter 7 concludes this work.

Chapter 2

Virtual Reality

The term *Virtual Reality* first appeared in literature 1958 in the English translation of the book *Le Théâtre et son double* (1938 by Antonin Artaud), called *The Theater of its Double* [1]:

> "[...] between the world in which the characters, objects, images, and in a general way all that constitutes the *virtual reality* of the theater develops, and the purely fictitious and illusory world in which the symbols of alchemy are evolved."

In the 1980s Jaron Lanier was one of the figures who propagated the term virtual reality in the context of its today's understanding.

Neither virtual reality (VR), augmented reality (AR) nor tracking in these contexts are young inventions and its development started before the term virtual reality was established. The two most prominent technologies in VR are head-mounted displays (HMDs) and CAVE[1] systems to facilitate the immersion. First prototypes of such technologies were already developed in the 1960s.

2.1 History

Ivan E. Sutherland for example was one of the first researchers who experimented in these fields [20]. In 1968 he created a tracked augmented reality HMD called *the three-dimensional display* (see figure 2.1). The two built-in CRTs (one for each eye) project the images, just plain lines, on two half-silvered mirrors in prisms through which the user looks. Sutherland et al. realized two different tracking solutions. The first approach was a mechanical solution. For this rather uncomfortable method, the HMD was mounted to a mechanical arm that was mounted on the ceiling. This arm allowed translations and rotations in all directions. Encoders attached on each joint measured the current configuration and allowed the deduction of the head's position and orientation. The second method was an ultrasonic sensor consisting of three transmitters which are attached to the HMD and four receiver mounted in a square on the ceiling. The see-through rendering combined with the tracking allowed users to move around in a room while observing objects, which appear as they are hanging fixed in the air. The *Virtual En-*

[1] *CAVE* is a recursive acronym for *Cave Automatic Virtual Environment*.

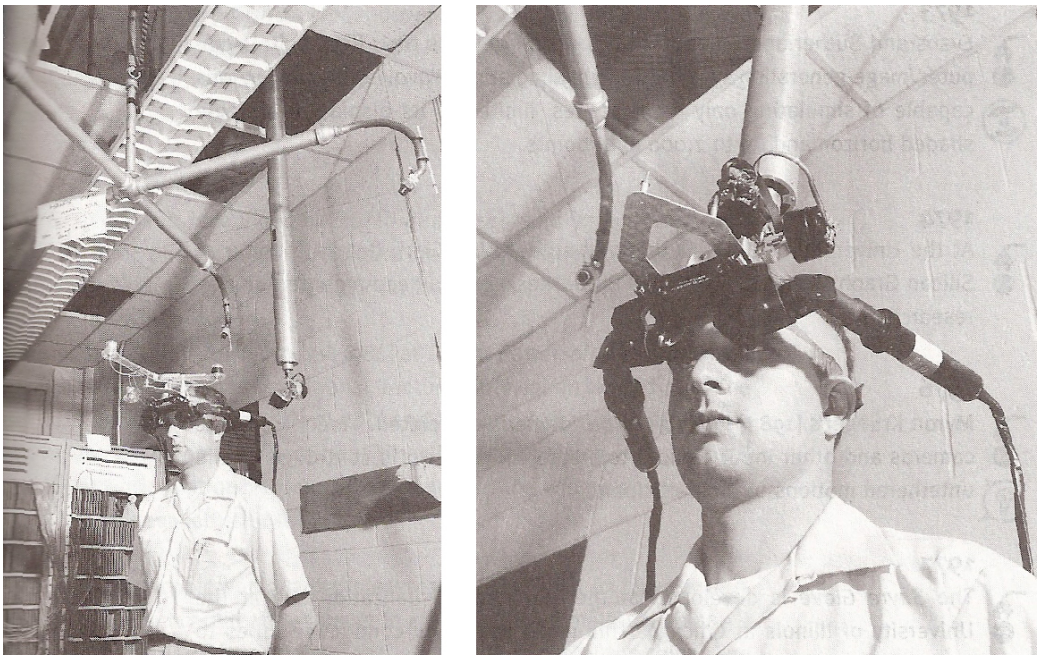


Figure 2.1: The *head-mounted three-dimensional display*, known as *The Sword of Damocles*. The right image shows the HMD mounted to the mechanical tracking device. The left image shows the same HMD in the ultrasonic setup.

vironment Display System (see figure 2.2) is one of the first virtual reality (in contrast to AR) projects and was developed by Fisher et al. at *NASA* 1985 [6]. It consists of an HMD with six-degrees-of-freedom (6DoF) tracking, glove-like devices for multiple degrees-of-freedom tactile input and speech recognition technology. As the binocular field of view of up to 90 degrees seemed quite advanced, the quality of the renderings, due to the minor processing power back then, was reduced to primitive, untextured 3D meshes with a rather low vertex count.

During the so-called first wave of VR, several consumer-level devices were released, some of them also provided head tracking or even hand tracking. These products usually provided sitting experiences only. The *Forte VFX1*, developed by *Forte Technologies*, is an especially noteworthy consumer product, which was released 1995. With a manufacturer's suggested retail price of US$695, it was one of the first affordable VR headsets for consumers. Several games natively supported the *Forte VFX1*. The HMD as well as the hand-held controller are tracked by internal sensors. However, Kuusisto [10] briefly describes its technical shortcomings and poor user experience. The horizontal field of view of 33.5° as well as the display resolution of 263x230 per eye are two clear weak spots.

To overcome limitations of VR (referring to HMD-based solutions) such as poor image quality, the isolation from the real world and inability to simultaneously share virtual experiences, Cruz-Neira et al. [4] invented the first *CAVE* (see figure 2.3) in 1992. They describe the *CAVE* as: "A new virtual reality interface, which consists of a room whose walls, ceiling and floor surround a viewer with projected images." They highlight

Figure 2.2: A later iteration of *NASA*'s *Virtual Environment Station* including the tracked HMD and the tracked glove-like input devices.

the *CAVE*'s capacity for stereoscopic rendering and to its capability of tracking the head as well as the hand of a user [5]. Tracking a user's head obviously is necessary to correct the stereo perspective dependent on the user's position and viewing direction within the *CAVE*.

About fifty years after Ivan E. Sutherland presented his vision of the *Ultimate Display* [21], which led to the development of the Sword of Damocles mentioned earlier, the current second wave of VR has introduced various new technologies and products. These current developments were preceded by the release of various VR products to the consumer market, the so-called first wave, over the past two decades. These first consumer devices suffered however from severe technological shortcomings, and the hype was largely overshadowed by these limitations.

In the past few years, a variety of more promising products using different technologies have emerged on the market. Most of today's products provide head tracking (at

Figure 2.3: The Cave Automatic Virtual Environment at EVL, University of Illinois at Chicago (2001).

least rotational tracking) but also support additional tracked controllers. The utilized approaches range from exclusive rotational tracking, realized with internal measurement units (IMUs) such as accelerometers, gyroscopes and magnetometers to marker-less inside-out systems which fuse IMUs with optical movement tracking and depth measurement (e.g., *Google Tango*[2] and *Structure*[3]). Besides their built-in sensors, marker-less systems do not need additional tracking hardware to achieve 6DoF head tracking. Large rooms, dark colored environments (e.g., black walls) and sunlight might interfere with the optical sensors and inhibit proper tracking. Tracking of limbs is not covered by these technologies. Despite certain limitations, especially inside-out technologies present new opportunities when it comes to scalability and flexibility.

2.2 Current Systems

A successfully crowd-funded project called *Oculus Rift* started in 2012 and released its first consumer version of a virtual reality HMD in the year 2016. One year earlier, *Samsung*, in collaboration with *Oculus VR*, released a mobile HMD named *GearVR* that utilizes a smartphone's display and processing power (see figure 2.4). As described

[2] https://get.google.com/tango
[3] https://structure.io

Figure 2.4: *GearVR* (left) and *HTC Vive* (right).

in chapter 5, the *GearVR* is used in this work. The *HTC Vive* (see figure 2.4) is similar to the *Oculus Rift* and was also released in 2016. It is the second VR product which is utilized in this work. The *Rift* and the *HTC Vive* are two of the most prominent wired HMDs. Besides them, the *PlayStation VR* is a third well-known wired device. Wired devices in contrast to mobile HMDs use a PC's processing power and therefore are able to render images of a much higher quality. Along with the *GearVR* (utilized for this work), the *Google Cardboard*[4] and the *Gameface Mark IV*[5] are two fairly successful products.

Compared to the *Forte VFX1*, current devices provide much higher display resolutions. The *HTC Vive* for example has two OLED displays with a resolution of 1080 by 1200 built-in.

2.3 Today's Limitations and Future Technologies

Today, technological shortcomings are still limiting the possibilities of VR experiences. Most displays of today's VR HMDs show a clearly recognizable *Screen Door Effect*. The screen door effect describes the visible gaps between a display's pixels. This effect becomes visible because the displays are viewed by the user through lenses. Diffuse screens, which blur the image, can reduce that effect. Increasing the resolution of future displays until the gaps become invisible will solve the screen door effect in the long term. Although higher resolution displays can of course facilitate higher-resolution renderings, and this development fosters the visual quality of VR experiences, it also increases the rendering workload.

The processing power needed for the rendering of VR scenes in a sufficient quality is generally available with current high-end consumer hardware. Rendering fully photo-realistic sceneries, however, remains out of reach. More powerful graphics cards have yet to be developed.

In contrast to adding processing power, increasing the efficiency of the whole rendering pipeline is also subject of current research. Built-in eye tracking in VR headsets

[4]https://vr.google.com/cardboard
[5]https://gamefacelabs.com

enables a technique called foveated rendering. The basic idea of foveated rendering is to reduce the image quality in the peripheral vision. Thereby, the rendering costs are reduced. The rendering power made available could then be utilized to enhance the image quality in the center of the view. Patney et al. at *NVIDIA Corporation*[6] [18] prove that this technique makes it possible to reduce the rendering workload and yet maintain the subjective visual quality to the user. Besides the better rendering efficiency, eye tracking (or gaze tracking) in VR opens new interaction and communication opportunities. The *FOVE 0*[7] is the first VR headset on the market with eye tracking built-in.

Locomotion in VR applications is an interesting but also problematic topic that is addressed in diverse ways. For experiences that enable users to walk around in virtual environments, two aspects are particularly important—a phenomenon called cybersickness and the naturalness of locomotion perceived by the user during roaming around in the virtual environment.

2.3.1 Cybersickness

Disorientation, headache, sweating, and nausea are just a few examples of the variety of possible symptoms that can occur due to cybersickness. In his work *A Discussion of Cybersickness in Virtual Environments*, Joseph J. LaViola [11] states that the effects of cybersickness may not only be triggered by entering a virtual environment but also after it is left. He illustrates probable causes and theories. The oldest and most prominent is the sensory conflict theory. It is assumed that a discrepancy between the visual and the vestibular system is the main reason for cybersickness. If cybersickness occurs and to which extent highly depends on the individual. In conclusion, motions in virtual environments and the associated accelerations are crucial factors that contribute to the cause of cybersickness.

2.3.2 Spatial Limitations

Apart from cybersickness, techniques utilized for locomotion also have a strong influence on the overall user experience. They reach from classic input devices, such as gamepads to omnidirectional treadmills as the *Cyberith Virtualizer*[8] or *Infinadeck*[9] (see figure 2.5). The most common locomotion methods in current consumer VR applications are either based on classic devices or on a combination of actual movement (users are tracked) and virtual teleportation. A 6DoF tracked HMD enables roaming virtual environments by walking around physically. The range of motion, however, is confined to the dimensions of the real space and the capabilities of the tracking system. In the case of the *HTC Vive*, the recommended maximal size of the tracked space is about 12 square meters. Although it is a very natural way of locomotion, the limitations of the real space as well the cables that connect current HMDs to PCs, severely constrain the freedom of motion. Virtual teleportation helps to overcome these limitations, at least partially. It is usually implemented as a point-and-click interaction. The user points at the desired

[6] http://research.nvidia.com
[7] https://www.getfove.com
[8] http://cyberith.com
[9] http://www.infinadeck.com

Figure 2.5: *Infinadeck* is an omnidirectional treadmill. It reacts to users' movements and keeps them in the center.

target (often an arc is traced) and presses a button to trigger a fast translation (often immediately) to the destination.

2.4 Multi-User VR

Most multi-user consumer applications are designed for remote interaction, such as online gaming. The way users are presented to each other, and the available communication channels, strongly influence the perceived presence (see chapter 3). Typically it is a combination of an avatar, which communicates in some way the position and movement of a user, and voice communication. In VR applications, 3D avatars represent users. If the users' HMDs are tracked, the avatars can directly imitate the motions. The same applies to the hands, if hand tracking—enabled by tracked controllers—is available. The level

of detail reaches from simple floating heads to realistic humanoid models.

2.4.1 Co-Location

The term co-location describes settings in which users, who are interacting in virtual environments, also share the same real space. Such settings pose new challenges. Ensuring the users' safety is the most evident and important one. Because the field of view (FOV) of most HMDs is significantly narrower than a human's natural FOV, the users' peripheral vision is significantly restricted. Hence, other methods must ensure that users stay aware of their proximity to others. When users share a tracked space, wired HMDs are a potential tripping hazard and therefore completely unsuitable.

In co-located settings, users perceive each other in the virtual world and also in the real environment. If not isolated by headphones, users can hear each other. Additionally, they can touch and feel one another. Nevertheless, they see the virtual representations. These aspects lead to the assumption that the overall feeling of presence, particular social presence (see section 3.2), becomes stronger.

Chapter 3

Presence

Fostering presence in co-located environments may be achieved by the utilization of foot tracking data. A user's actual movements—in case of this work, the feet—directly control the feet of an associated avatar and, derived from foot and head movement, indirectly control the whole virtual representation. Thereby, the *MIKA* system might be a viable vehicle for future studies which explore various fields of presence in computer-mediated virtual realities.

3.1 Definitions

Presence is an essential concept in human computer interfaces that is of particular importance for video games or immersive virtual reality applications. In a generalized and simplified sense, presence describes the sense of being somewhere, being with someone or something is there. Presence in computer-mediated environments is often subdivided into different subconcepts such as physical, social or self presence.

Telepresence, a general term for a computer-mediated type of presence, could be described as "the sense of being there" with the extension "in a virtual environment". In 1980 Minsky introduced the term [15] as a specialized form of presence. From here on, however, the shorter and more common term—presence—is used.

Lombard and Ditton formally describe presence as "the perceptual illusion of non-mediation" [22] which occurs when a person cannot perceive or acknowledge the existence of a medium in his or her communication environment. According to them, presence is a perceptual illusion and the property of a person. It is determined by the formal and content characteristics of a medium and the characteristics of the user. Lombard and Ditton define six dimensions of presence:

1. *Presence as Social Richness*
2. *Realism*
3. *Transportation*
4. *Immersion*
5. *Social Actor Within Medium*
6. *Presence as Medium as Social Actor*

They state that these six conceptualizations share a central idea. Each of them represents

one or more aspects of a perceptual illusion of nonmediation.

Lee defines presence as "a psychological state in which the virtuality of the experience is unnoticed" [12]. He comes up with three different subtypes.

1. *Physical presence*: a psychological state in which virtual physical objects are experienced as actual physical objects. It occurs when users do not notice the artificial nature of objects.
2. *Social presence*: a psychological state in which virtual social actors are experienced as actual social actors. Users of the technology do not notice the para-authenticity of mediated actors (human or artificial).
3. *Self presence*: a psychological state in which the virtual self is experienced as the actual self. It occurs when users do not notice the vitality of their own selves.

Both definitions, Lombard's and Lee's, provide a similar description of presence in a mediated context. However, Lee [13] augments the scope in his definition. He states that presence can be felt even in a non-mediated environment (e.g. interaction with an anthropomorphic robot—social presence). But both describe presence as a state in which users do not consciously perceive the mediating technology during the interaction, though they know of its existence.

M. Garau et al. [8] researched the impact of visual and behavioral realism in avatars on perceived quality of communication in VR. In the experiment's conditions, they use avatars with two different levels of visual realism. A very elementary, blocky, untextured and genderless model was used to represent both male and female for the lower-realism conditions. For the higher-realism conditions, a higher tessellated and textured model is used. To reflect different levels of behavioral realism, they either used random eye gaze or eye gaze inferred from voice.

Although the MIKA system would easily allow the avatar to be exchanged, for this work the avatar is identical for both conditions, hence the visual realism is invariable but the behavioral realism alters. Although the avatar itself, consisting of 3D model and IK rig, remains constant, its movement behavior changes. The two different tracking implementations, 2D tracking (see section 6.3.1) and 6DoF tracking (see section 6.3.2) differ. Similar to the hypothesis of Garau et al. it is expected that differences will be detected in the perception of the different movement behaviors.

3.2 Social Presence

Of all concepts of presence, social presence is the most relevant for this work. In an earlier definition of social presence, Biacco [2] suggests that social presence should be seen as a simulation of another intelligence rather than as a partial replication of face-to-face communication. Biacco explains that this simulation is then run in the body and mind of the perceiver. It models the internal experience of some other moving, expressive body.

3.2.1 Dimensions of Social Presence

Biocca et al. [3] distilled three dimensions of social presence: co-presence, psychological involvement and behavioral engagement. Especially the concepts of co-presence and

behavioral engagement are interesting aspects. Co-presence is described as [3]:

> "... the degree to which the observer believes he/she is not alone and secluded, their level of peripherally or focally awareness of the other, and their sense of the degree to which the other is peripherally or focally aware of them."

This definition does not presume that the interacting individuals are located within the same room but only that they are connected by a computer-mediated technology. However the concept of co-presence is still applicable even if the specific scenario has to be taken into account. Psychological involvement is defined as following [3]:

> "... the degree to which the observer allocates focal attention to the other, emphatically senses or responds to the emotional states of the other, and believes that he/she has insight into the intentions, motivation, and thoughts of the other."

Because locomotive properties of an avatar are related to it, the third dimension of social presence, behavioral engagement also appears quit relevant to this thesis and is defined as [3]:

> "... the degree to which the observer believes his/her actions are interdependent, connected to, or responsive to the other and the perceived responsiveness of the other to the observer's actions."

It is assumed that the inclusion of further user tracking information, such as foot tracking, to create more realistic avatar animations helps to foster social presence in co-located environments.

Roth et al. [19] investigate the effects of reduced social information and behavioral channels. They used faceless wooden mannequin avatars. It was found that social interactions tend to be impeded when non-realistic avatars are utilized. However, they state that the absence of behavioral cues, such as gaze and facial expressions, can be partly compensated by shifting the attention to other behavioral channels, such as body movement. These findings support the assumption that a higher degree of realism of an avatar's animations, including the animations level of detail, may positively affect social presence in co-located settings. Single statements of this work's evaluation (see chapter 6) indicate the validity of this assumption to a certain extent. Although it is not measured, the level of co-presence in this work (see chapter 5) is expected to be rather high, due to its co-located nature.

Furthermore, the prototype developed for this thesis may be used as a tool to examine presence in future work, and the findings of this thesis should help to create further iterations of such systems.

Chapter 4

Visualization of the Virtual Environment

4.1 Avatar

The *Unity* game engine provides a game character named *Ethan* (see figures 4.1, 4.2 and 5.13) which fulfills the requirements of this project. Both the option of using other available solutions as well as creating a new custom character were taken into consideration. Creating a new avatar with a proper rig and a custom inverse kinematics system (see section 4.1.2) may help to optimize the resulting animation but is not within the scope of this thesis. *Ethan* consists of a 3D model with a low vertex count (see section 4.3) and has a rig (skeleton) built in. It provides both forward and inverse kinematics.

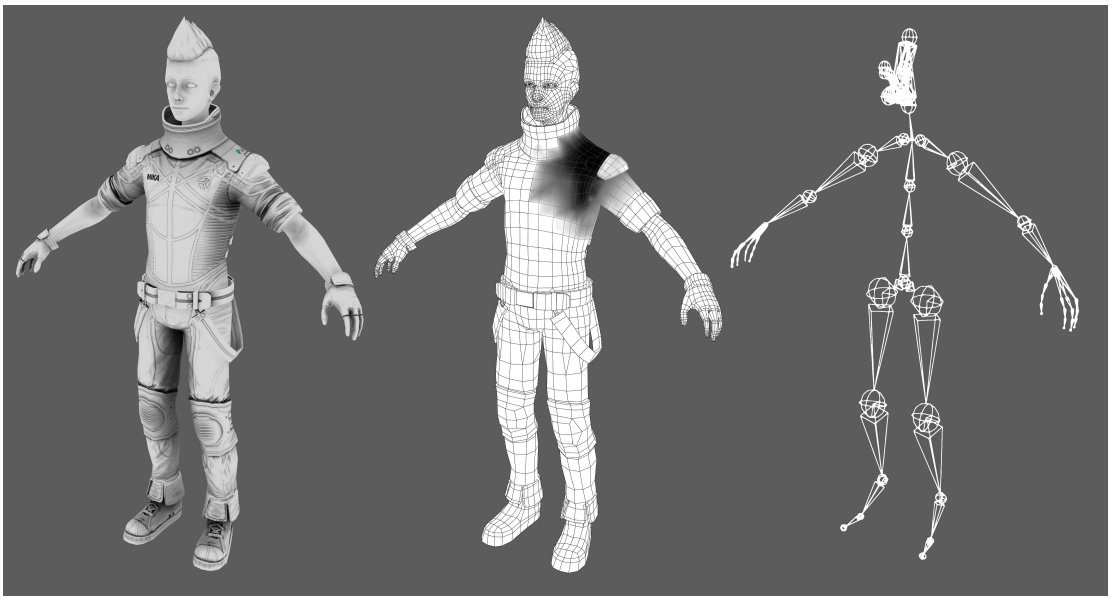


Figure 4.1: The left rendering shows the avatar model with applied normal map and occlusion map. The center rendering shows the avatar's wireframe and the vertex weights of the left shoulder joint (black area). The right image shows the skeleton (rig) of the same model.

Additionally, it provides many predefined animations (e.g., running or jumping) and the ability to blend them. These animations, except for the idle animation, are not used for this work.

4.1.1 Animation

Before a 3D model can be animated properly, it needs a rig, and the rig has to be connected to the mesh—the actual geometry. When a single joint is rotated, it is desirable to have the whole area around it affected. When the shoulder joint rotates, the vertices of the mesh near the shoulder joint have to be manipulated individually and in a meaningful way. Vertices that are located on the chest, for example, are less affected than those closer to the joint. The definition of this joint-mesh relationship usually happens manually in a 3D modeling tool and is called weight painting. Figure 4.1 shows a weight painted shoulder as well as the avatar's rig.

Characters in animation movies are animated manually by an animator (with the exception of motion capture). To achieve natural behavior, every motion is created individually. For 3D game characters, in contrast, all animations are predefined and prepared to be played in loops. They are triggered by a game engine and can be blended into each other to achieve smooth transitions. The production process of a single loop, like a walk cycle, is similar to the animation process of a movie character. In both fields, as well as in this work, inverse kinematics are utilized to achieve natural motions.

4.1.2 Inverse Kinematics

In addition to inverse kinematics (IK), a second technique that is also often used for animations in computer graphics is forward kinematics (FK). In FK, an end-effector's (e.g. hand or foot) pose (position and rotation) is calculated from the values of all joints which affect this pose. In the case of a foot, it could be the ankle, knee and hip-joint's rotation and the hip's position in space (if the model is rooted at the hip). An animator who wants to reposition this foot has to change every joint rotation manually until the foot reaches the desired pose. In the case of the tracking/animation setting of this work, the angles of all related joints of a user's body would have to be measured and then applied to the avatar's joints. This is one reason why IK is utilized for such tasks. In contrast to FK, where the pose (position and rotation) of an end-effector is computed by the parameters of each individual joint, the desired pose is given in IK, and the joint parameters are calculated. An IK rig consists of rigid segments (bones) that are connected with joints. This is called a kinematic chain and its configuration determines the current pose of a character.

For example, when a foot is lifted from the floor (illustrated in figure 5.13) its position, and perhaps also its orientation, may change. In the simplest case, all connected joints from the foot up the kinematic chain to the hip joint have to adjust their rotations (ankle, knee, hip and maybe the midfoot or the toes). A more complex rig may also try to balance the mass and therefore attempts to move all joints accordingly.

In the implementation of this work, a class named `IKControl` controls the rig. It receives positional data and rotation data of the feet and hands, and the look-at point for the head. The rig's feet and hands continuously follow these data. The head constantly looks at the look-at point, which is derived from the HMD's orientation and located

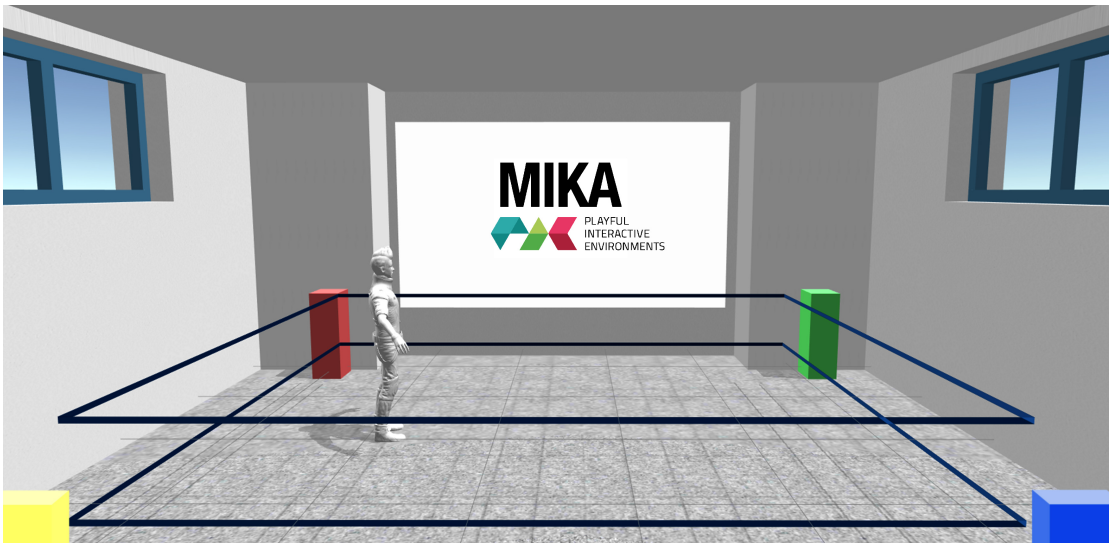


Figure 4.2: A rendering of the VR client scene. This is how users see the virtual environment. The room's geometry is an exact copy of the actual room in which the participants of the evaluation (see chapter 6) were during testing the system. The dark blue lines mark the borders of the tracked space (2D and 6DoF). Four colored cuboids are located near the edges of the tracked space (red: southwest; green: northwest; blue: northeast; yellow: southeast).

approximately five meters away from the avatar's head. The avatar is rooted at the hip joint. The avatar as a whole, rooted at the hip, is translated to the current average position of both foot tracking data and rotated to a value that depends on the velocity vector as well as on the current head orientation. The combination results in a full body animation.

4.2 The Environment

The virtual space (see figure 4.2), perceived by the VR users is a copy of the real room the users are located in. All dimensions are exactly the same, down to the very centimeter. The walls texturing is subtle, and window frames are not textured. Because users do have contact to the floor, its virtual tiles resemble those in the real world in structure and size. The interaction space measures about four by five meters and is enclosed with a virtual fence. To ease spatial communication, one of four colored cuboids is located on each corner of the fence ("I am going to go from the red to the green cube."). At the western wall, a virtual projection screen shows the logo of *Playful Interactive Environments* and the project's name *MIKA*. To reduce computation costs, the overall geometry is kept as simple as possible.

4.3 Rendering

Because of the limited rendering performance and the requirement that the application's frame rate must remain at 60 frames per seconds (fps) or higher, an economical use of the graphics power is required. This is achieved by obeying the following rules.

4.3.1 Geometry Optimization

A high number of draw calls can lead to frame rates below 60 fps. A draw call is an early step in the render pipeline; the CPU prepares data for the graphics unit and expensive computations are performed on the CPU. The number of draw calls is affected by the number of separate objects and the count of used materials. Keeping both of them low is necessary and also an effective method, which supports maintaining the target frame rate. Reducing the number of drawn objects can simply be achieved by combining meshes before they get added to the project (the *Unity* project in the case of this work). A good practice is to combine all meshes that share the same material. The counts of both, objects and materials, are reduced to a minimum by this measure. All windows and the door of the virtual space, for example, are a single object and share the same material. Meshes with a high vertex count cause a noticeably higher workload for the graphics chip. Keeping the vertex count low, by deleting unnecessary vertices of a 3D model, is an easy but important step to optimize graphics performance on mobile devices. The vertex count of all objects in the VR environment, excluding the avatar, is at 512. The 3D mesh of the avatar consists of approximately 6896 vertices. After the reinterpretation of *Unity* both add up to a total of about 10 000 vertices that has to be rendered by the mobile graphics chip.

4.3.2 Further Economizing Steps

Post-processing effects, such as anti-aliasing or screen space ambient occlusion (real time), require a considerable amount of rendering power. Therefore it is recommended to omit such procedures, if possible. However, anti-aliasing remarkably enhances the visual quality, especially in VR-applications, and therefore it is recommended to seek its activation. The goal of ambient occlusion is to give the impression of indirect lighting effects (corners and gaps appear darker than the surrounding). It also can be realized by applying a precomputed greyscale map to an object, which costs less processing power than calculating it in realtime. An occlusion map is applied to the avatar (see figure 4.1). Dynamic shadows can highly facilitate the realism of a 3D-scene. But, they are also an expensive technique and hence should be utilized very limitedly. On the one hand, even if multiple light sources are placed within a scene, not every single one should cause shadows. On the other hand, only selected objects, such as the avatars, should cast shadows. By keeping both low, shadow-causing light sources and shadow casters, the performance can be improved further. Because transparent objects and materials also cause substantial stress to the graphics processor, they should also be avoided if possible. In this work's implementation, no transparencies are incorporated at all. Furthermore, utilizing shaders that are optimized for mobile devices is a natural but also important option to reduce the rendering effort.

4.4 VR Client

Like the server application, the client implementation is also realized with the *Unity* game engine. The `NetworkPlayer` class described in section 5.2.5 represents the interface to the server application. When the client application boots, it immediately starts listening for broadcasts from potential *MIKA* servers within the network and connects when one is found. It also gets assigned to a tracked entity (see section 5.2.2) automatically by the server. All avatars assigned to other users (i.e. not one's own avatar) are rendered on a client. If another client is already connected to the same server, as well as when a new client connects, the respected avatar is instantiated instantly and therefore becomes visible for all other users. Tracking data of one's own instance is used to translate the virtual camera dependent on the movement of one's own feet and to visualize the positions of the feet.

4.4.1 Client Calibration

To align the rotation of the virtual space to the real space, so that the visualization corresponds to the tracking data, a simple calibration step is necessary. This orientational calibration of the HMD is the only explicit user interaction required by the client application. Since no automatic calibration technique is implemented, users have to trigger the calibration process manually by pressing the *Back Key* (see figure 4.3). When the user is tracked by the system and the HMD is connected to the server successfully, the calibration can be performed. Before triggering the process, the HMD's forward direction must be parallel to the Z-Axis (in the case of the *PIE-Space*[1] setup perpendicular to the western wall of the room). After the calibration was successful, the virtual space is properly aligned to the real space and walking around is safe. Even after longer sessions (>10 minutes), no significant drift has been observed, so recalibration is not usually necessary. In case the calibration would start to differ whilst interacting with the system and wearing the HMD, it is possible to recalibrate the orientation at any time.

[1] The *PIE-Space* is part of the *Playful Interactive Environments* facilities and equipped with LiDAR devices as well as a *HTC Vive*.

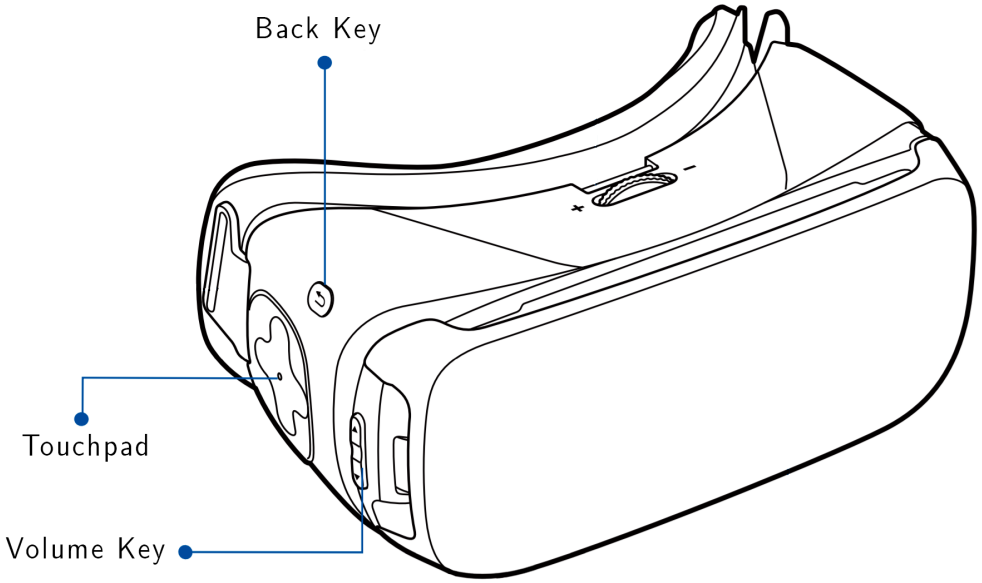


Figure 4.3: This sketch shows all input options of the *GearVR*. The touchpad can be utilized for basic navigation. The volume key (actually two keys) could also be used for menu navigation such as traversing a list. The back key, usually used to quit applications, triggers the calibration process.

Chapter 5

Implementation

The whole setup targets the needs of accessibility and scalability. On the one hand, accessing it should be possible with little effort and in a short amount of time; on the other hand, it should be capable of tracking multiple people. Entering the system should only take a few seconds and must not require any technical know-how. The combination of the utilized tracking system (see section 5.1.2) and the head-mounted display (see section 5.1.1) perfectly fit these requirements. Besides the obvious advantages, the chosen hardware has some drawbacks. This section describes all technical details of the application and the utilized hardware. Creating a system which utilizes a two-dimensional tracking technique is the main task of this thesis project. In the conducted evaluation, this setup is called *condition A* (see section 6.3.1). For *condition B* (see section 6.3.2) two six-degrees-of-freedom (6DoF) trackers are attached to a user's feet.

5.1 Hardware

Most of the required hardware is equal for both tracking implementations. Only the tracking devices and the corresponding infrastructure differs. For the setup of *condition A* (see section 5.2.3), four LiDAR devices and an additional Ethernet switch are necessary. *Condition B* requires a *HTC Vive* and four *Vive Trackers* instead. A detailed listing of the entire infrastructure follows in section 5.1.3.

5.1.1 Head-Mounted Display

Due to the fact that multiple users are moving around in a shared tracking space, avoiding harm to the users and also to the hardware has top priority. In order to reduce the risk of stumbling, it is a major requirement for the utilized head-mounted display (HMD) to be wireless. It is also considered that putting on and getting ready to use the application has to be doable easily and quickly. The imposed conditions narrow down the eligible devices. It is established that the *Samsung Gear VR*[1] [24] (see figure 2.4) headset in combination with a *Samsung Galaxy S6*[2] smartphone [23] fits the requirements. Therefore it serves as HMD in this project (from now on *Gear VR* refers

[1] http://www.samsung.com/us/mobile/virtual-reality/gear-vr/gear-vr-sm-r322nzwaxar
[2] http://www.samsung.com/at/smartphones/galaxy-s6-g920f

Figure 5.1: A SICK LMS-100 2D LiDAR sensor.

to the combination of *Samsung Gear VR* and *Samsung Galaxy S6*). The low weight of just 456 grams and the resulting wearing comfort is a further benefit of the *Gear VR*.

The limited processing power, compared to VR systems that utilize a personal computer (PC), places restrictions on the design of the software as well as to the visualization. However, heavier computations, such as filtering of tracking data, is processed on the server and the results are sent to the HMD. By following some computer graphics and graphics design related rules (see section 4.3), a sufficient visual style, which is computable on a smartphone, can be achieved. The *Gear VR* features acceleration sensors as well as a gyroscope. This enables rotational head tracking. Positional head tracking is not available though. The information of a user's position is provided either by the laser ranging system (see section 5.1.2) or by the the *HTC Vive tracking*. Pal et al. [17] conducted a study, in which they compare translational head tracking, rotational head tracking and complete (translational and rotational) head tracking. They found no significant difference between complete head tracking and rotational head tracking, in terms of task errors, usability, and presence. Therefore, they recommend rotational-only HMDs for creating usable and immersive VR experiences.

5.1.2 Position Tracking

As mentioned at the beginning of this chapter, the capability of tracking multiple people in a shared tracking space and its easy access are two important requirements. Sufficient accuracy as well as low error-proneness are also vital for such a system.

In the test setup, four *SICK LMS-100* [25] (see figure 5.1) light detection and ranging (LiDAR) devices are utilized for the scanning process. The LiDAR devices are connected via Ethernet to a PC on which the *PHARUS* tracking software (see section 5.2.3) is installed.

Light Detection and Ranging Devices (LiDARs)

LiDARs exist in different forms and serve many different needs. The areas of application reach from scanning the earth surface topologies for the fields of geology or archeology to the detection of the environments of self-driving cars and many more. In the case of the thesis, the LiDARs are utilized as tracking devices and are capable of determining the users' feet/leg positions within a given space (tracked space). To achieve a proper tracking, in terms of accuracy and reliability, multiple LiDARs are in use (four in case of the evaluation setup). Adding further devices is a natural and valid way to overcome occlusion problems, which potentially emerge through an increasing number of users. Further LiDARs can also help to maintain the tracking quality when the tracked space becomes larger.

Alternatives Considered

In addition to the combination of *Gear VRs* and the *PHARUS* system, other technologies also has been considered. Utilizing inside-out tracking devices, for example, would be another approach that may fit the needs of this thesis project. Although accurate positional and rotational head tracking would be beneficial, the short range (when it comes to tracking) such devices usually provide may not work properly in large empty rooms. Also the color and texture of walls and interior may impede the performance of various inside-out tracking technologies. Compared to the chosen *Gear VR* approach, consumer-ready VR systems, such as *Oculus Rift* and *HTC Vive*, have the advantage of a much higher processing power. The main drawback of these products is that the HDMs are wired to a PC. That could be overcome by letting the user carry it like a backpack. Although the limited interaction timeframe, caused by the necessity of batteries, is tolerable, another disadvantage, smaller tracking spaces, declines the applicability of such systems. Future products may support bigger tracking spaces. The basic *HTC Vive* setup, for example, allows tracked areas of up to a maximum of five by five meters. It is also noteworthy that the effort that is necessary to enter such a system includes putting on additional hardware (trackers and backpack) and therefore is significantly higher compared to the effort of entering the system introduced in this work.

5.1.3 Infrastructure

The infrastructure consists of one server PC (*AMD FX-8350 Eight-Core Processor*, 12GB RAM, *GeForce GTX 970*), four *SICK LMS-100* 2D LiDAR sensors, a *HTC Vive* with four trackers and two base stations. The LiDARs are connected to a *ZyXEL ES-108A* Fast Ethernet switch, which then is plugged into a LAN adapter of the server machine. The *HTC Vive* is connected to a USB port of the server machine as well as to the graphics card via HDMI. As mentioned earlier, two *Samsung Gear VRs* with two *Samsung Galaxy S6* phones serve as HMDs. The HMDs are connected to a wireless router (*NETGEAR Nighthawk X6*), which is also linked to the server on a second LAN adapter (see also figure 5.2).

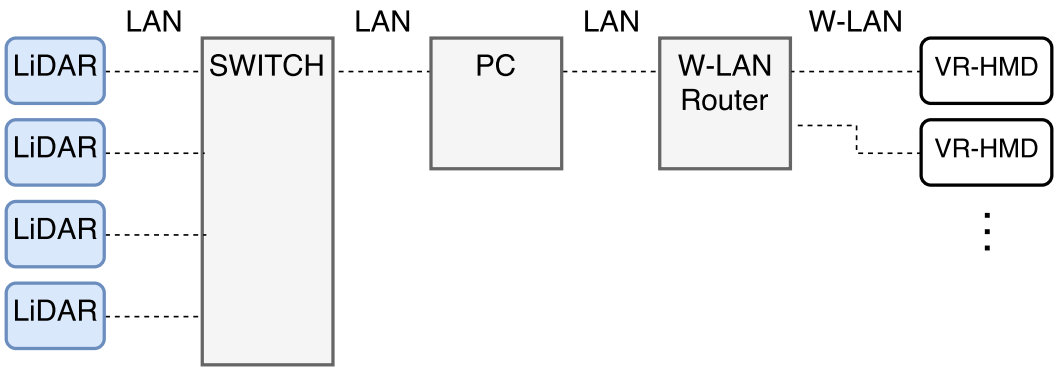


Figure 5.2: An illustration of the network infrastructure.

5.2 Software: Modular Inverse Kinematic System

The modular inverse kinematic system (*MIKA*) is the outcome of the practical contribution of this thesis. It consists of two main parts, the server application and the VR client application. A basic intent was to create a modular software architecture. The providers of tracking data, as well as the animated avatar, should be decoupled from the basic functionality and therefore exchangeable. Although both the server and the client application were developed with the *Unity* game engine [26]. The software, especially the server application, was developed so that it is highly independent from the game engine. This means that the classes only derive from *Unity*-specific types or use *Unity*-specific types if necessary or if avoiding them is exceedingly intricate. Those parts of the *MIKA* system which are related to *PHARUS* tracking, *Vive* tracking, data filtering, foot height estimation and avatar animation, are specialized implementations. However, the basic structure remains general. The functionality that handles how data has to be provided and how data can be received is determined by interfaces and abstract classes that have to be implemented by specialized classes.

As figure 5.10 illustrates, the implementation of the project is structured in three main modules. While the input module (tracking module) and the output module (animation module) can and have to be specialized for the specific needs of a project, the module in between, the part which holds and provides all available data (data management module), is designed to fit the needs of a wide range of tracking and IK animation scenarios. The following section describes the general implementation of each module as well as specific implementations needed for the thesis project. Section 5.2.2 refers to the input module, section 5.2.3 to the data management module and section 5.2.4 refers to the output module.

5.2.1 Server Application

The server application consists of

1. the receiving and processing of tracking data,
2. data management,
3. avatar animation and

4. networking.

That means that the server application is capable of almost all functionality introduced in this chapter. Only features that are exclusively concerned with rendering and calibration on the HMD are excluded.

All tracking related implementations, including the foot height estimation (see section 5.2.4), as well as the data management module (see section 5.2.3), are executed on the server only.

5.2.2 Tracking Implementations

To test the modular abilities of the system and to establish a basis for the planned evaluation of the project, two different tracking technologies have been implemented. In addition to the 2D tracking (see section 5.2.2) implementation, which utilizes the *PHARUS* system as data source, a 6DoF tracking solution (see section 5.2.2), which receives tracking data from a *HTC Vive*, has also been developed. The two described tracking implementations directly serve as conditions for the conducted evaluation (see chapter 6). As previously mentioned, neither the data management module (see section 5.2.3) nor the avatar animation module (see section 5.2.4) differ between those two implementations.

2D Tracking with *PHARUS*

Tracking data received from the *PHARUS* (see section 5.2.3) application initially is received and processed by classes provided by the *PHARUS* system. The `PharusPlayerManager` is the interface to the *MIKA* system. It instantiates a `TrackingEntity` when a user enters the tracking space and is recognized by `PHARUS`, and removes it if a tracked user disappears. `TrackingEntity` is provided by the *PHARUS* system as an abstract class that can be specialized. In the case of this thesis project, the specialization for the 2D tracking approach is called `UnityPharusFootTracking` (see figure 5.8 and 5.14). It contains all data the *PHARUS* application has generated related to the current entity (tracked user) and sent via the network. In total, an entity contains an ID, an absolute position, the next expected absolute position, a relative position, the orientation, the speed and the so called echoes. Except for the echoes, all data is already processed and filtered by the *PHARUS* application. However, the only data utilized by the *MIKA* system are the echoes, a list of unfiltered two-dimensional positions representing a user's feet.

The code fragments of the classes `AComponentData` (see figure 5.7), `LeftFootData` (an example of a specialized data providing class)

```
public class LeftFootData : AComponentData {...}
```

and `UnityPharusFootTracking` (see figure 5.8) in combination with figure (see figure 5.10) should point out the correlations of the classes responsible for the tracking functionality.

On the opposite end of the pipeline, the `IDataReceiver` interface,

```
public interface IDataReceiver {
  void Data(float[] position, float[] rotation);
}
```

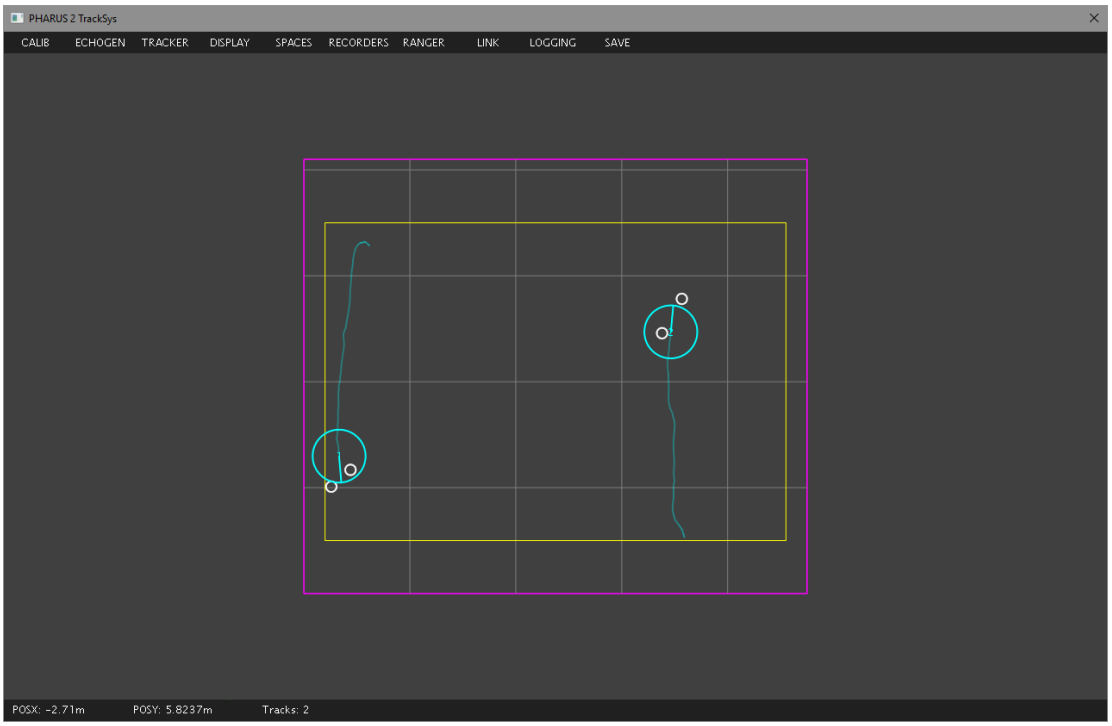


Figure 5.3: A screenshot showing *PHARUS TrackSys* tracking two users. The smaller white circles are the so-called echoes and indicate the users' feet positions.

which is the base interface for all receiver types, needs to be implemented to receive data from the `ModelDataManager`. *ILeftFootReceiver*

```
interface ILeftFootReceiver : IDataReceiver {...}
```

is one of the specialized *IDataReceiver* interfaces. The `IKModelController` (see figure 5.9) implements all *IDataReceiver* interfaces that are needed for the present IK-model.

Echo sorting: Users' foot tracking data, the echoes, are not filtered or sorted by the *PHARUS* application. As a result, the data received via UDP is first very jittery and second ambiguous; the foot assignment is unknown and can flip at any time. It also happens that one tracked entity contains fewer or even more than two echoes. Figure 5.4 shows typical tracked data received from the *PHARUS* system before any processing. Short losses of one or both echoes are implicitly handled by the filtering process (see section 5.2.2). Extrapolation techniques have been considered during the design and development phase. Since a suitable and reliable prediction of a user's actual foot movement is probably hard to achieve and after first tests have shown that missing echoes occur just sporadically and only over short time periods, extrapolation of missing data became obsolete. If more than two echoes appear, the two most reasonable ones get selected. That simply happens by checking distances between the last position to all possible new positions and mapping the closest one to the respective foot. Next, the echoes have to be mapped to the proper foot. Because in every frame a flip can occur, the echo-foot assignment has to be monitored and corrected constantly. A user's

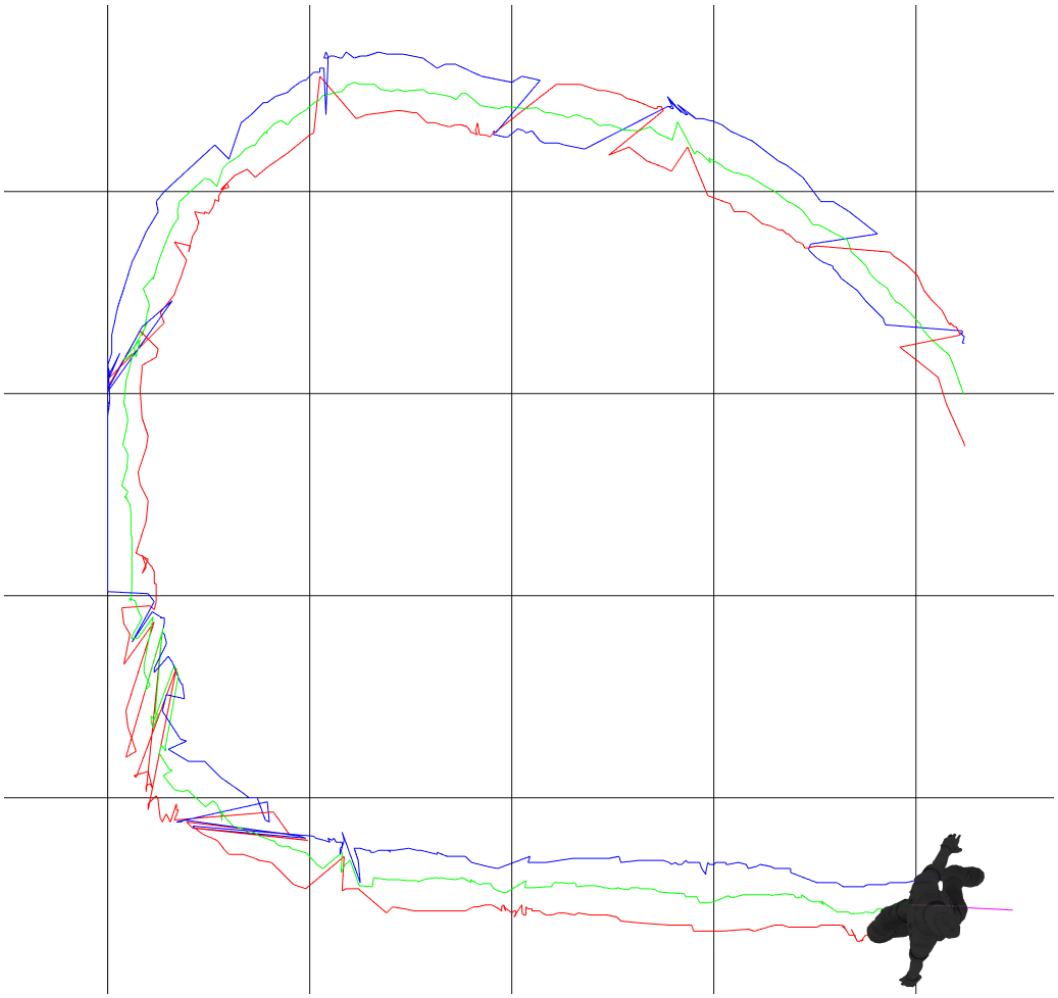


Figure 5.4: This screenshot shows the resulting foot and body movement with unfiltered data without foot correction. Blue: left foot; red: right foot; green: center by average of left and right foot; silhouette in the right bottom corner: the avatar in an orthographic projection; grid spacing is one meter.

current movement direction as well as the head rotation (received from the HMD) are taken into account to prevent wrong foot mappings and thereby foot crossings of the avatar. Basically, every tick it is checked which echo is closest to the last position of a certain foot in the last tick. Then it is assumed that the best fitting echo represents the corresponding foot. Under consideration of a user's assumed current body orientation and movement direction, a decision is made regarding which echo represents the left and which the right foot. After this step, the basis for the following smoothing process is created. By this measure, the problem that the avatar's feet may cross during walking is solved regardless if a user walks forwards or backwards. Figure 5.5 shows the 2D path of the single feet after the sorting algorithm has been applied. The data still appears jittery and inaccurate but all echoes are properly linked to the respective foot.

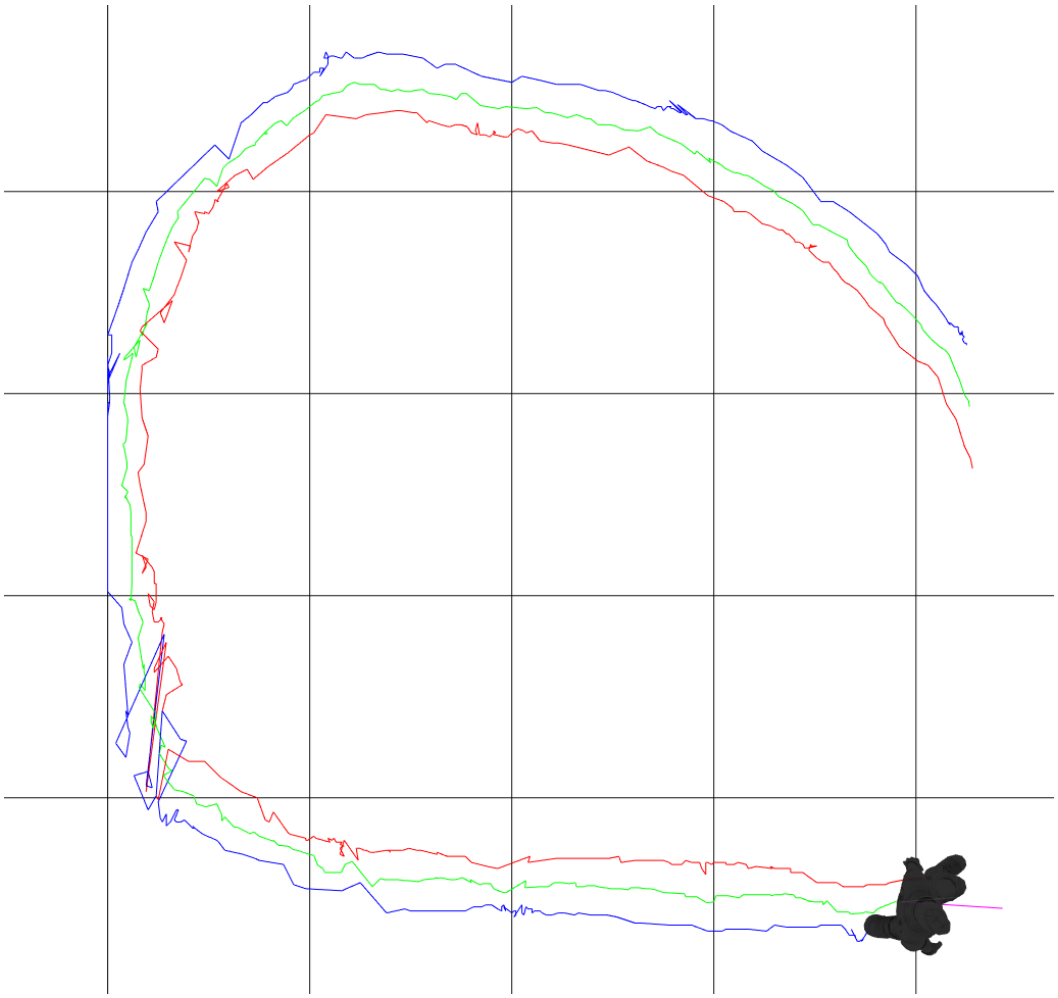


Figure 5.5: This screenshot shows the same data as figure 5.4 with foot correction applied but with the filtering still turned off. Blue: left foot; red: right foot; green: center by average of left and right foot; silhouette in the right bottom corner: the avatar in an orthographic projection; grid spacing is one meter.

Filtering: After checking and correcting the echo mappings, the still jittery tracking data gets filtered. After the filtering process, the data (motions) should not only by smoothened but also still be accurate and responsive. The alternation of the speed of each foot is vital for the following estimation of their heights and the associated walking animation (see section 5.2.4). If the filtering is too strong, the movement becomes too even and also latencies increase. If it is not strong enough, the jitter remains when the received data's quality is poor. Finding proper values for the utilized filter's parameters is a very important step in the whole process. As it is commonly used in various technologies and fields, such as aeronautics (radar) or autonomous self-driving cars (radar or LiDAR), in this project also a *Kalman filter* [9] is implemented to minimize the data's noise and approximate the objects' (feet) actual positions. In the early work *Laser-Based People Tracking* (2002), Fod [7] et al. successfully used a standard

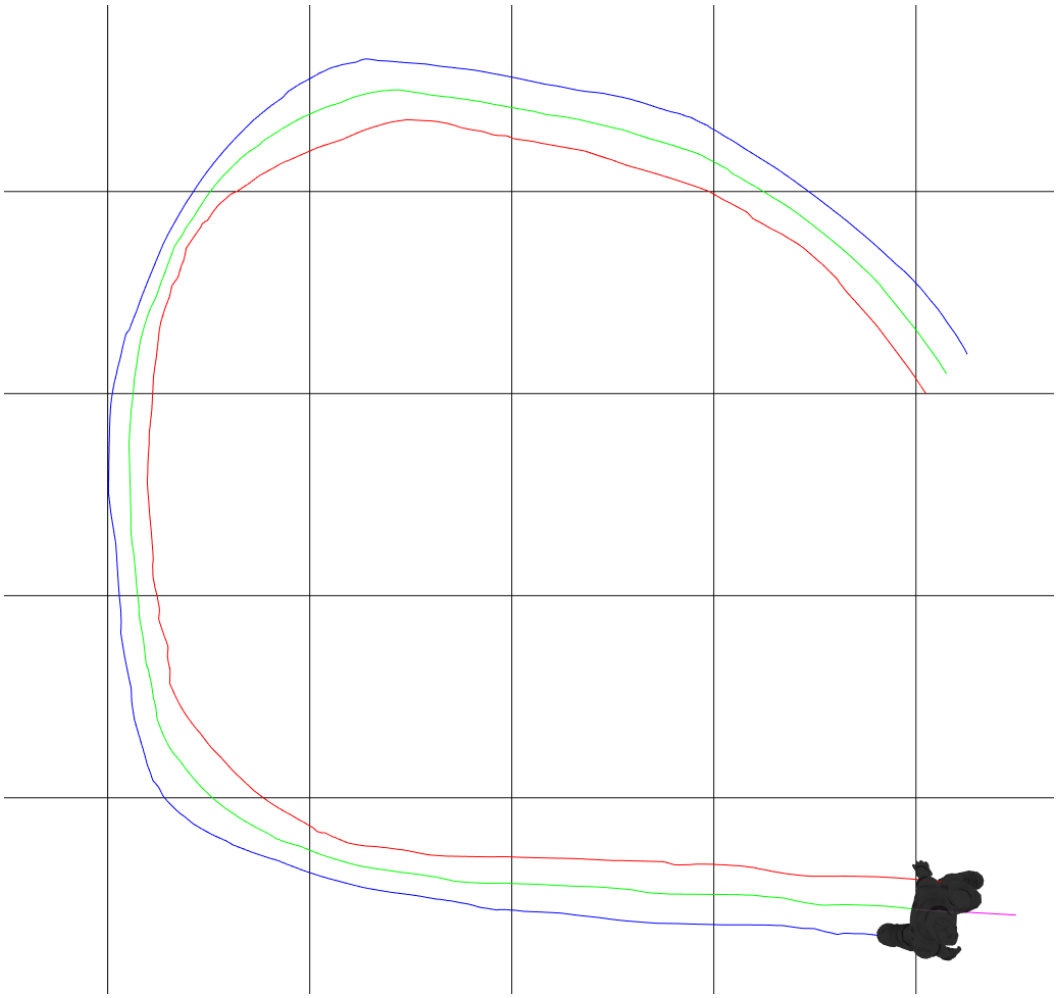


Figure 5.6: A screenshot showing the resulting foot and body movement (using the same data as figures 5.4 and 5.5) with foot correction and the kalman filter turned on. Blue: left foot; red: right foot; green: center by average of left and right foot; grid spacing is one meter.

laser ranger-finder to track people and utilized a *Kalman filter* to smooth paths. They evaluated the performance of a system similar to *PHARUS* in a series of experiments and showed its effectiveness in presence and occlusion. Also, the *PHARUS* application implements a *Kalman* filter to smooth tracked objects movements before sending them via Ethernet. However, this part of *PHARUS*, which combines echoes to entities and filters the created entities' position as well as their orientation, is not relevant to the project of this work. As mentioned afore, only the echoes are used.

6DoF Tracking with *HTC Vive*

6DoF tracking is, in addition to 2D tracking, the second implemented tracking technology. Compared to the 2D tracking solution, only the tracking module (see figure 5.10) is modified. The data management module (see section 5.2.3) and the animation

```
public abstract class AComponentData {
    protected Func<float[]> position, rotation;

    private bool usePosition, useRotation;
    public bool UsePosition {...}
    public bool UseRotation {...}

    public AComponentData(Func<float[]> pos, Func<float[]> rot {...}

    public float[] Position() {...}
    public float[] Rotation() {...}
}
```

Figure 5.7: *AComponentData* is the abstract base class for all data providers.

```
class UnityPharusFootTracking : ATrackingEntity, IMikaTrackedEntity {
  ...
  private void InitFeet() {
  lFoot = new LeftFootData(() => GetLeftFootPosition(), () => GetLeftFootRotation())
    ;
    ...
    UnityModelDataManager mdm = GetComponent<UnityModelDataManager>();
    mdm.AddProvider(lFoot);
    ...
    }
}
```

Figure 5.8: This code shows how a data providing method gets added as a callback. When a `UnityPharusFootTracking` gets instantiated, it adds itself as a provider to the `UnityModelDataManager`. This class is the connection between *PHARUS* and *MIKA*. It derives from the abstract class `ATrackingEntity` and implements the interface *IMikaTrackedEntity*.

module (see section 5.2.4) remain untouched. The VR client application is completely equal for both tracking types. *Steam* and *SteamVR* must be installed on the server to run the *Vive Trackers* (6DoF trackers). The required *SteamVR* plugin is also added to the *Unity* project. In the 6DoF implementation, the provided positional and rotational data is passed to the data management module (see section 5.2.3) without any filtering and nearly no alternation. Because the trackers are attached to a user's feet (see figure 6.2), the only correction applied is a approximated physical offset of a tracker (position and rotation) to the ankle. Although the utilized hardware is consumer-ready and usually very accurate (sub-millimeter in positioning), the trackers attached to the feet often show unpredictable behavior and errors, such as a steady position offset in height or fast and far drifts for a short timespan. Compared to the 2D solution, the required effort and thus the needed amount of time prerequisite to enter the system is higher. The trackers must be attached tightly to the feet and if necessary the assignment of the trackers must be adjusted on the server. The biggest benefit of this approach is full accurate tracking without any noticeable jitter or latency.

```
class IKModelController :  ILeftFootReceiver, IRightFootReceiver, ... {
...
  void ILeftFootReceiver.VectorData(float[] position, float[] rotation) {
        ikControl.leftFootPosition = position;
        ikControl.rightFootRotation = rotation;
    }
    void IRightFootReceiver.VectorData(float[] position, float[] rotation) {...}
...
}
```

Figure 5.9: *IKModelController* operates the actual IK-model (*IK-Control*).

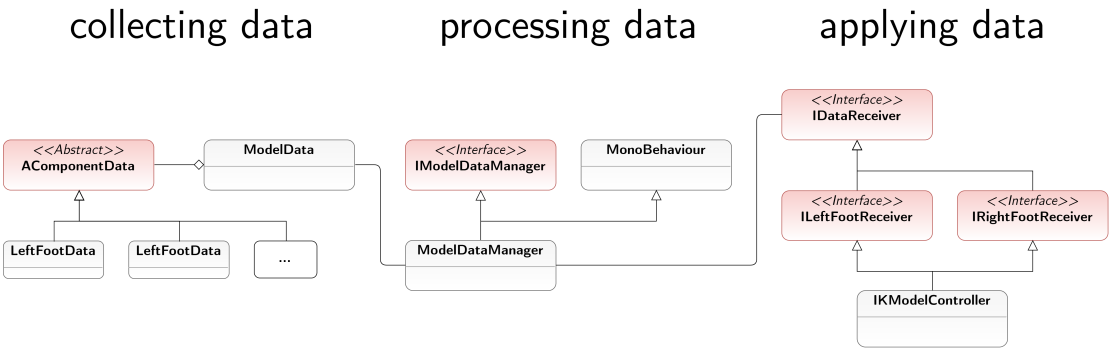


Figure 5.10: This sketch of the software architecture illustrates the modular structure of the implementation.

5.2.3 PHARUS

PHARUS [16] was developed by Otto Naderer for a permanent installation in large room called *Deep Space* in the *Ars Electronica Center Linz* (AEC)[3]. It is capable of tracking multiple people in large co-located environments. While in the AEC installation, six LiDAR (see section 5.1.2) devices are tracking an area approximately 140 square meters (about 16 by 9 meters) the area of this project's test setup measures approximately 24 square meters (about 6 by 4 meters) and is tracked by four LiDAR devices. The *PHARUS* system as it is used in the thesis project consists of two parts.

The first one, *PHARUS TrackSys* (see figure 5.3) receives data from the LiDARs and processes it. The results then are provided via TCP and UDP and consist of all tracked entities (usually tracked users). Each tracked entity consists of a variety of data—an entity's position, its current orientation (movement direction), its velocity and most important for this project, unsorted and unfiltered single foot data named echoes.

The second part is the client implementation. *PHARUS* already provides necessary C# classes which derive from the *Unity* base class `MonoBehaviour` and are able to establish network communication with the *TrackSys* application and to manage tracked entities within a *Unity* project. By deriving from `ATrackedEntity`, which gets instantiated by `UnityPharusManager`, all available data of a tracked user can be accessed.

[3] https://www.aec.at

```
public interface IModelDataManager {
  void SubscribeReceiver(IDataReceiver obj);
  void UnsubscribeReceiver(IDataReceiver obj);

  void AddProvider(AComponentData provider);
  void RemoveProvider(AComponentData provider);

  // update providers
  void UpdateModelData();
  // update receiver
  void UpdateCallbacks();
}
```

Figure 5.11: The interface *IModelDataManager* collects all tracking data and distributes it to subscribed components. It also controls the update loops of both.

`ATrackedEntity` is the exact point of intersection on which *MIKA* docks.

Data Management

The `ModelDataManager` is the link between the tracking and collecting module and the animation module. Its interface, shown in figure 5.11, illustrates how the implementing class has to process data received from various providers on the one hand and invoke callbacks of subscribed receivers on the other hand. `ModelDataManager` uses the class (`ModelData`) to store all tracking data. Each user in the system causes the instantiation of a `ModelDataManager` and thereby all other related modules are created.

The data management module is designed to accept different data providers. In the implementation of this work, for example, data from the currently used tracking system is combined with the rotational data of the HMD. It is also supported to subscribe several data receivers. Thereby one user could control an arbitrary number of avatars simultaneously.

5.2.4 Avatar Animation

Several possibilities to animate avatars using tracked data are under consideration. They vary in the extent of their dependency on the data's quality and their ability to provide synchronous (to the user) motions. Triggering predefined animations exclusively based on a user's velocity and orientation is an obvious and also established approach in video games and similar interactive applications. Since the actual positions of a user's feet are completely neglected, this method has been fundamentally excluded. On the other end of the spectrum, no pre-created animations are utilized at all but only tracked data is used to control an avatar. In between those two extremes, a broad variety of hybrid solution exists (e.g. control a walk cycle conditionally to foot movement). In the thesis project, avatars are animated almost only based on foot movement. To make them appear slightly more lively, an idle animation is added continuously. Slow and subtle movements affect the whole body. When a user stands completely still and the avatar would not move at all, the idle animation becomes most evident, because then only

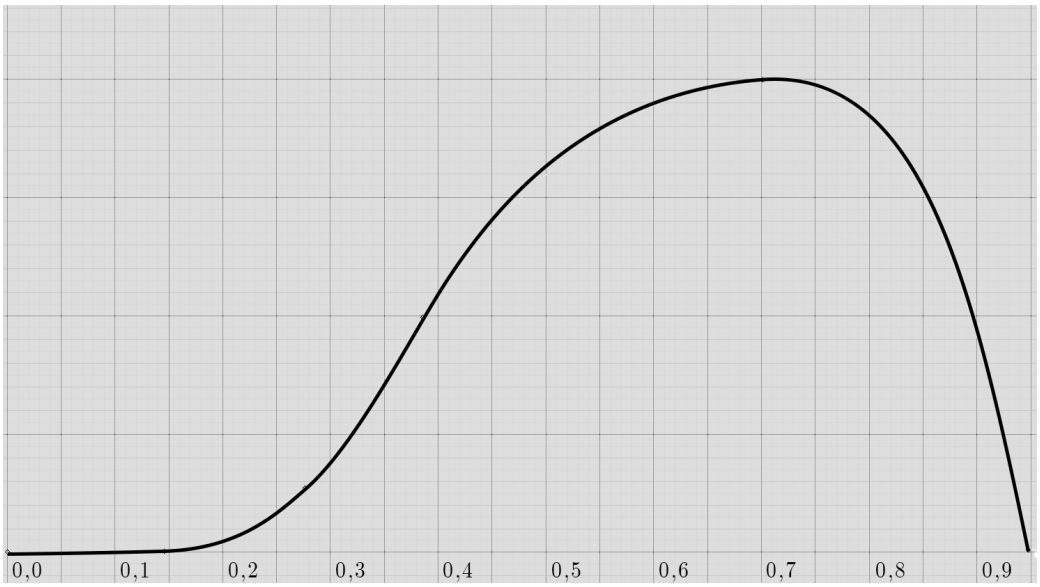


Figure 5.12: This is an approximation to the curve used to determine a foot's height dependent on its current velocity.

the idle animation is visible. Single foot position data (and derived velocity) is utilized to animate the avatar's feet and hands as well as to determine a general movement direction of a tracked entity. The HMD's rotational data determines the head pitching and slightly affect the body's orientation.

Foot Height Estimation with 2D Data

Since the 2D data provided by *PHARUS* just provides 2D position data, the height of an avatar's feet has to be estimated. This is achieved by deducing a probable and reasonable height dependent on the current velocity of a foot. The current velocity is derived from the knowledge about the last two positions and the timespan in between and is calculated by an implementation of the following equation.

$$\begin{aligned} \boldsymbol{v} &= (\boldsymbol{x}_n - \boldsymbol{x}_{n-1}) \cdot dt \\ \text{where } \boldsymbol{v} &= \text{velocity}, \\ \boldsymbol{x} &= \text{position}, \\ dt &= \text{elapsed time} \end{aligned}$$

The result is clamped to a maximum velocity, scaled between 0 and 1 and then filtered, so the values over time result in a smooth curve. Before the estimated height can be provided to the *MIKA* system, a lookup table (see figure 5.12) is applied to determine the final height value. Similar to this, a slight rotation is applied to the ankles. The procedures just explained are not necessary for the 6DoF approach and therefore omitted.

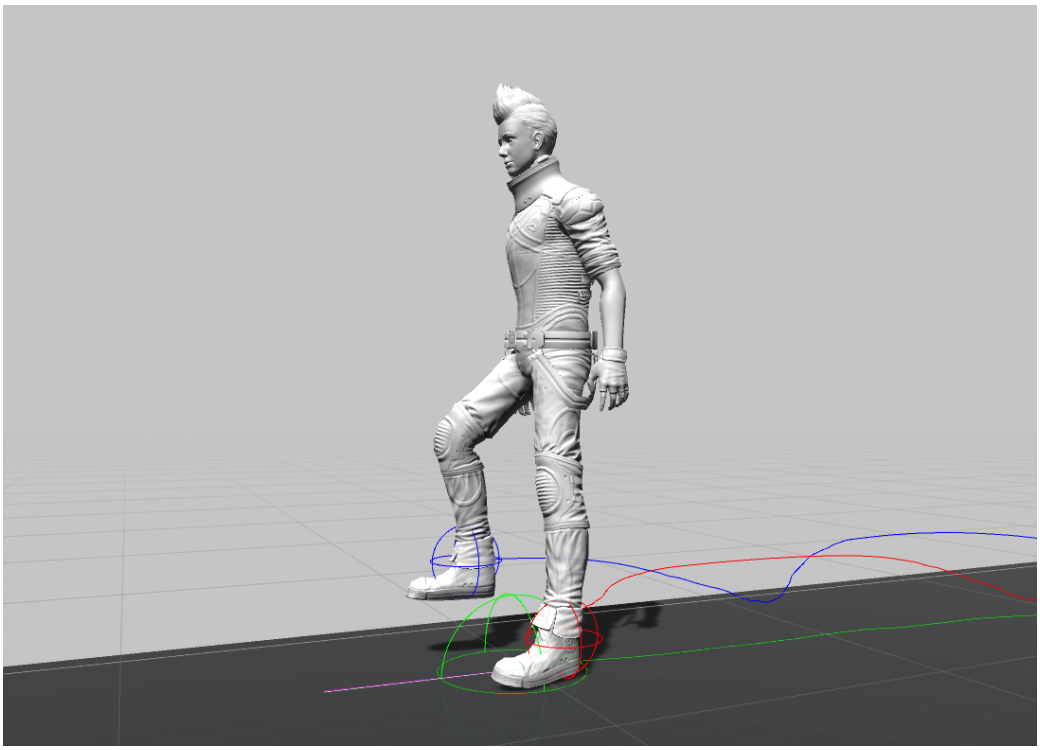


Figure 5.13: The utilized avatar with applied foot tracking data. The foot lifting height this screenshot shows is extremely exaggerated. Foot tracking data (augmented with height estimation) is displayed as red and blue lines. The green line indicates the mean value of both feet clamped to floor level and the pink one the velocity vector.

5.2.5 Networking

The communication between the *MIKA* server applications and the VR-clients is realized with *Unity*'s built-in networking API—*UNET*. *Unity*'s built-in networking application programming interface (API), *UNET*, provides a high-level API (HLAPI), which enables a rather quick implementation of networking functionality but also offers control over the underlying abstraction layers and protocols. In general, UNET is based on UDP. Within the HLAPI, the rules and parameters of the protocol can be set. To facilitate a low latency and ensure a correct order of incoming data, in this project tracking data is sent to the headsets unreliably but sequenced. Sending data unreliably means that if packages get lost, they are not resubmitted. Sequenced ensures that packages arrive in the same order as they are sent. For example, a package that is older (sent before) than an already accepted package does not get accepted by the receiver. All network communication between the *MIKA* server and the VR-client is implemented in the `NetworkPlayer` class. Figure 5.16 illustrates the usage of `SyncVars`. `SyncVars` are variables which, when changed on the server, are automatically synced to the client. Depending on whether a `NetworkPlayer` instance (client-side) belongs to a certain user or not, IK data is processed and the corresponding avatar visualized. If a `NetworkPlayer` belongs to a user, only spheres indicate one's own feet positions but the avatar is not

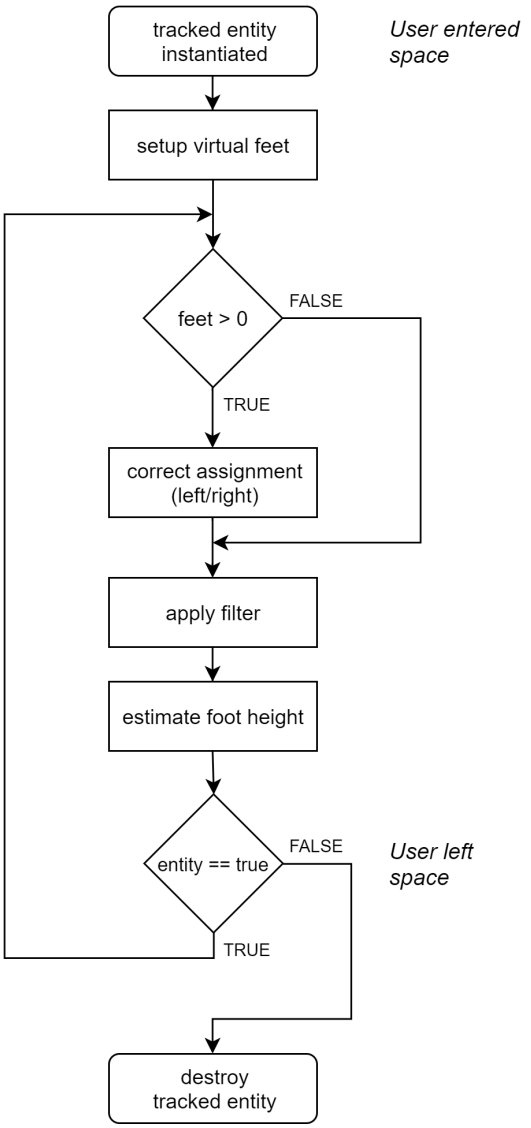


Figure 5.14: A simplified flowchart that illustrates the lifecycle of `UnityPharusFootTracking`—the 2D tracking implementation of `IMikaTrackedEntity`.

```
public float EstimateHeight(Vector3 position, Vector3 lastPosition) {
  ...
  speed = Clamp((position - lastPosition).magnitude / dt, 0, maxSpeed);
  speed01 = speed / maxSpeed;
  speed01 = kalman.UseFilter(speed01);
  height = curve.Evaluate(speed01);
  return height;
}
```

Figure 5.15: The foot height estimation procedure of the 2D tracking implementation.

shown. This tracking data (foot position and head rotation) is utilized to determine the virtual camera's transformation. The `ServerManager` derives from `NetworkManager` (a *UNET* class) and is responsible for the networked instantiation and destruction of all entities, including `NetworkPlayer` objects. In the setup utilized for the thesis project, positional and rotational data of both feet, both hands and the head orientation of a user is sent to the clients. However, the head orientation is provided locally by the HMDs. Before it can be processed on the server and then sent back the clients, it must be transmitted from the clients to the server.

Automatic Connection

To facilitate quick testing during the development and also to allow easy usage in general, an automatic connection functionality between the *MIKA* server and the *MIKA* clients has been implemented. Without this feature, the server's IP-Address must be set in the client application at some point, either before building the binaries (.apk files), before start-up within a configuration file or during the startup process in a setup menu. Due to the automatic connection feature, this configuration step becomes obsolete. It is achieved by network broadcasting and listening on the network. When a *MIKA* server starts up, it immediately begins broadcasting. Broadcasts are dispatched in a 1000 millisecond interval and include data, a string, which helps to identify the correct application. Different applications could be broadcasting simultaneous in the same network using the same port. The client listens to broadcasts on a defined port, receives them, checks if the data is correct and connects to the broadcasting server. When the connection is established, the client stops listening (further broadcasts are irrelevant). However, when the connection to the server is lost, the client resets itself and starts listening again. Because there is no determined maximum number of clients, the server continues broadcasting until it gets shut down.

Since there is nearly no configuration needed, starting up and running the whole system is quite simple even though several steps are needed.

1. All devices must be turned on (LiDAR, switch, wireless router, one ore two PCs—depending if the *PHARUS* application and the *MIKA* server are running on the same machine—and the smartphones.)
2. Starting up *PHARUS* application.
3. Starting up *MIKA* server.
4. Starting up the smartphones.

The order of 2,3,4 is not crucial and can therefore be interchanged. However, the suggested order represents the logical structure and the data flow.

```
public class NetworkPlayer : NetworkBehaviour, ILeftFootReceiver, ...
{
    [SyncVar]
    private Vector3 leftFootPosition, ...
    ...

    // Server-sided
    void ILeftFootReceiver.Vector(float[] position, float[] rotation)
    {
    leftFootPosition = new Vector3(position[0], position[1], position[2]);
    leftFootRotation = Quaternion.LookRotation(new Vector3(rotation[0], rotation[1],
     rotation[2]), Vector3.up).eulerAngles;
    }

    // Client-sided
    private void ProcessIK()
    {
    ikControl.leftFootPosition = leftFootPosition;
    }
}
```

Figure 5.16: An overview of the *NetworkPlayer* class implementation. Tracking data is received by the server and is written to *SyncVar*. These are synced to the client object and applied to the IK controller on the client side. `ProcessIK()` is only called if the *NetworkPlayer* instance does not belong to the current user.

Chapter 6

Evaluation

The implementation was evaluated by the conduction of expert interviews. It was intended to gather qualitative feedback from qualified specialists rather than measuring user experience in a more general way. The experimental and prototypical character of the project requires the expertise of people with a technical background related to the thesis project. The questions asked during the interview did not aim to receive definite and precise answers. They rather should trigger further thoughts and reflexions of a participant. Both interviews were recorded and are available as audio files (see section A.2).

6.1 Procedure

The experiment consisted of the following procedure: first, the interviewer welcomed the participant and provided an overall introduction to the thesis and a quick overview of the evaluation's procedure. The system is explained, aided by a short video (2 minutes and 54 seconds), which shows recordings and renderings of the system in action (see figure 6.1 and section A.3). While the interviewer commented on the video, the participant had the opportunity to ask questions. Two scenarios are illustrated within the video. First, a comparison of filtered and unfiltered 2D data (condition A) appears and second the filtered 2D data is compared to 6Dof data (condition B). After the video, the participant was asked two open questions regarding the shown footage and was invited to comment on his or her impressions on the demonstrated video. After the server, set up for condition A (2D tracking), and both VR clients were started, the interviewer issued one HMD to the participant and gave a short explanation for the following performance, including the calibration procedure of the HMD (see section 4.4.1). The participants were encouraged to communicate their thoughts during the performance, similar to the think-aloud method Lewis and Mack [14] introduced. First, the participant and second the interviewer entered the tracking space, started to walk around and watch each other's virtual representations. Since the focus was mainly on the subject's perception of the avatar, the experimenter walked around the participant and performed the following tasks: walking forward, walking backwards, walking sidewards. The same procedure followed with condition B. To enable 6DoF tracking for condition B, additional trackers were attached to the interviewer's and the partici-

pant's shoes (see figure 6.2). Each of both rounds took approximately five minutes. The experiment was finalized by an interview in which the participant was initially asked two questions about the perception of the avatars. Subsequently, the participant was requested to review the performance and describe his impressions of each condition.

6.1.1 Questions

The following questions were presented at the end of each stage of the interview. As mentioned before, the participants were not required to answer every question completely.

After the Video Review

- Where do you see advantages of these conditions?
- Where do you see disadvantages of these conditions?
- What else do you want to remark at this point?

After the Performance

- How realistic and natural did you perceive the avatar's movements?
- To what extent does it appear like a person is controlling the avatar?
- To what extent does it feel like a person is controlling the avatar?
- What else do you want to comment on at this point?

6.2 Participants

Two participants were invited to the expert interview separately. It was intended to recruit a specialist with a profound technical background in the field of virtual reality as well as one who is an expert in the fields of character animation. Dr. Christoph Anthes is specialized in AR and VR as well as in networked and collaborative environments, and he is a professor of AR and VR. He is the VR expert in this evaluation. The recruited animation expert is Alexander Wilhelm. He is a professor of character animation and motion graphics and researcher at *Playful Interactive Environments* (a research group of *R&D FH Upper Austria*).

6.3 Conditions

The two different tracking implementations (see section 5.2.2) were utilized for two different conditions of the expert interview. As explained in the chapter implementation 5, the data management module and the avatar animation module were the same for both conditions. Hence, the avatars' appearance was identical. Because of the two different tracking sources, the animation did differ noticeably. For both conditions, the avatars' animations are controlled by a combination of single foot tracking data and the head rotation data.

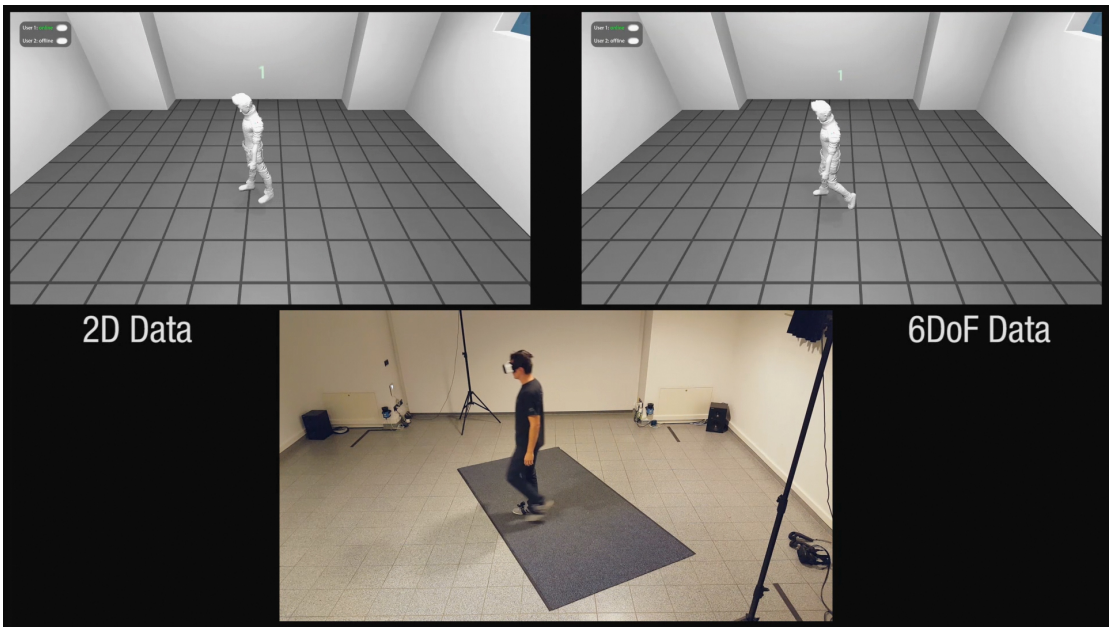


Figure 6.1: This figure shows one frame of the video shown during the evaluation. It is divided into rendered footage of the first condition (top left), rendered footage of the second condition (top right) and film footage of the recording session (bottom). The *HTC Vive* base stations are mounted on the tripods, which are visible in the filmed footage. Two of the four utilized LiDARs, mounted near the corners of the walls indentation, are also visible on the filmed material.

6.3.1 Condition A

The first condition served as the trial of the 2D tracking implementation. The position tracking is handled by the LiDARs, and therefore it is only necessary to wear the HMD, which is put on at the beginning of the first round. Especially noteworthy is the fact that the height of a foot and its rotation is estimated by its current velocity. Lifting or rotating a foot while standing still is not recognized by the system. This impends detecting if a user is walking sidewards or forwards. Therefore the visualized foot pose naturally differs from the real one.

6.3.2 Condition B

The second condition tested the 6DoF implementation. The position tracking is handled by the *Vive Trackers*. Hence, in addition to the HMD, the trackers are strapped to the participant's and the experimenter's shoes. These trackers enable more precise and realistic foot movements of the virtual avatar. Lifting and rotating a single foot is possible in this condition. Also, sidesteps are visualized correctly in this condition.

Figure 6.2: This is how the 6DoF trackers were attached to the user's and the interviewer's feet.

6.4 Results

The participants have different specializations and therefore distinct point of views on the presented material. Similarities in their feedback as well as especially noteworthy findings are treated in the summary of this chapter (see section 6.4.3). The following sections are chronological analyses of both interviews.

6.4.1 VR Expert

During the video review, the participant detected that the avatar's feet do not roll off during walking and the feet's rotation thereby reach excessive angles. He correctly assumed the lack of a foot joint (midfoot). That is especially noticeable in condition B, where 6DoF data utilized.

The 6DoF trackers show two different kinds of flaws which could be observed in the video, as well as during the performance. First, the tracking occasionally is several meters off for short time spans (one to two seconds) but then continues working accurately again without any intervention. The second problem is a height error which sometimes occurs and remains for longer timeframes. It happens that the tracked positions are about 15 to 30 centimeters too high. It was assumed that occlusions may be the reason for these issues. The two base stations of the *HTC Vive* are located opposite of each other (see figure 6.1). When a user walks away from one base station and towards the other one, the one in the back is probably occluded totally.

One important aspect that was noticed was that the avatar's feet are further away from each other than the user's. Thus, the avatar's hip and therefore its whole upper body moves further down than the user's when the feet's distance increases. That correlates with the fact that the avatar is smaller than the user who controls it. Therefore, the step range of the user is too far for the smaller avatar to behave naturally. By scaling the avatar to the size of the user, the leg's length will match better, and the negative effects just described would be reduced.

Concluding the Video Review

After the video was watched, it was asked to point out advantages and disadvantages of both conditions. It was mentioned that the tracking losses of the 6DoF approach seem to be problematic. In condition B, no considerable jitter was noticed. Regarding condition A, it was stated that it appears more stable compared to condition B, although a little jitter is noticeable. Although the walking in condition A behaves like sliding, it appears more natural than condition B.

During the Performance

Two kinds of exaggerated movements were noticed during the performance of condition A. First, the avatar's head sometimes tilted extensively forward. That happened when the corresponding user tilted the head. The fact that the avatar is not able to follow the changing look-at-point with eye movement but only by rotating the head, and the way the rotational data gets applied, led to exaggerated nodding. Second, in condition A, it happened that the avatar repeatedly flipped 180 degrees. That can occur when the user nods his head too far; thus the system interpreted backward walking.

The concurrency of the avatar's feet touching the ground and the noise produced by a user's feet during walking was perceived very positively during the performance of condition B. Strong foot tilting that was already observed during the video review seemed less problematic in the live experience. The distance between the feet were stated again. Tracking problems of the 6DoF trackers were recognized but not addressed in detail.

Participants Conclusion

After the performance, jitter was stated as the main shortcoming of condition A. The correlation of the virtual environment's visualization and real world sounds, such as footsteps and talking, were very supportive for the experience. Compared to watching the video (in the first part of the interview), a visual comparison of the virtual and real environment was impossible. Therefore the avatar's behavior was perceived more convincingly.

It was stated that it felt like another user was controlling the avatar. Yet, the immersion was not strong enough to facilitate the perception of the avatar itself as another person. This level of immersion was prohibited by unnatural movements, caused by flawed tracking.

The probably very positive effect of an appropriately scaled avatar according to the user's height was stated repeatedly. If the avatar's height would match, the head height

in the virtual environment and the real environment would also correspond. Besides the positive effect to animations, the direction of sounds when a watched user speaks would be more correct and help to foster the naturalness of the experience. 6DoF tracking may be improved by adding further base stations. That is supported natively by the second generation of the utilized hardware. The implementation of various sanity checks[1] was suggested to prevent impossible movements, such as moving immediately from point A to point B.

Finally, concerns were addressed about the system in general regarding cybersickness. The constellation of the utilization of wireless HMDs combined with the implemented tracking solutions tend to apply jitter and latencies to the virtual camera of VR users and therefore may cause adverse effects.

6.4.2 Animation Expert

This participant describes his ideas, opinions, and suggestions from an animator's standpoint but also tried to apply them to the evaluated real-time scenario. During and after the video review, flaws of both tracking methods were already noticed, and considerable efforts to understand the underlying technical and implementation related reasons were made. Potential solutions for further developed were also identified.

In the beginning, with a focus on condition A, the drift of the feet was questioned. Drift refers to the phenomenon in which a user's foot has contact with the floor during walking, but the corresponding avatar's foot does not stand completely still. The assumption that some kind of interpolation is responsible for this behavior was partly correct. It was explained that interpolation and filtering facilitate the error but a second factor also influences it. To reduce the chance of overstepping the tracking system, the LiDARs are mounted at approximately 30 centimeters above ground level and therefore track the shins instead of the actual feet or ankles.

First Approach—Rotoscopy

From an animator's point of view, it was described how the generation of animations would work by performing rotoscopy on the footage of the 2D-tracking rendering. First, the feet must be locked when they are actually touching the ground—so the drifting is prevented. This step establishes a basis for the animation process. Finding the points in time on which a user's foot touches the ground and locking the avatar's until the user's foot lifts again was highlighted as highly relevant. Additional data may be helpful to determine the correct moment and solve the locking issue.

After locking the feet, the hip's height, which is derived from the foot movement, is animated. In the course of this, it was mentioned that the avatar's overall hip position appears to be three to five centimeters too high. In addition to the hip's horizontal movement, vertical movement follows the standing leg. That emulates the natural behavior of balancing the body's weight. The extent of that vertical movement depends on the walking speed (the faster the walk, the weaker the pendular movement).

[1] Sanity checks in this context refer to a method of validating whether a set of data can lead to feasible movement or posture of the corresponding avatar.

Second Approach—Hip First

Next, a second approach, the procedure for a manually animated action sequence (e.g., fast running, jumping and rolling over), was described. First, just the hip is animated, followed by the torso. Only after that are the steps animated. That is done by the utilization of the IK chain. At the beginning of the movement, the animator determines which foot makes the step and which remains grounded. Then the hip's position is transformed. When the hip is translated forwards, due to the nature of the IK chain, at a certain point the feet will lose contact to the ground. This lifting indicates the proper moment for initiating the first step (the participant called that "giving the beat"). The hip motion always leads and therefore determines when the feet have to follow. From this point of the interview onward, the participant stopped talking from the perspective of an animator and started showing a stronger consideration to the applicability of the stated contents to real-time scenarios.

A Triangle and Springs

The geometry of the avatar can be described mathematically and mechanically. Thus, the constellation of the hip and both feet can be seen and treated as a triangle. The maximal hip height occurs when a person stands on one leg and balances the weight in this position (except jumping). During walking, the lowest hip position occurs when both feet have contact to the ground; more exactly, after the second foot hits the ground (this is the so-called contact position). Because the legs have to absorb the energy (body's weight), the actual height change of the hip is probably greater than the one given by the triangle approximation. The hip bounces to a deeper position after the contact. This behavior can be accomplished by the introduction of spring mechanics. Using springs is a considerable approach to improve animations in general and the real-time avatar animations of this work in particular. Springs can be described mathematically and help to simulate the human body's bounciness. A spring could help to control the horizontal hip movement as well. The fact that such sine-shaped motions are slightly behind in time after leading motions is crucial but handled automatically due to the characteristics of the spring mechanic.

Concluding Questions After the Video Review

Condition A seemed to be more stable, and it also appears more consistent. Condition B provides a more natural spatial fixation (no drift), and the stepping is clearer. Also in condition B, the heel movement is exaggerated (ankle pitch).

The following part of the interview was discussed after the performance.

Avatar Realism

After seeing the project in action, avatar realism and pitfalls were addressed. Glitches that occurred during the test were described as very irritating. The avatar's degree of realism intensified that negative effect. Realistic avatar models, such as the one of this work, require very realistic animations. When a realistic avatar acts in an unpredictable or unnatural manner, it leads to a high level of disturbance. A more abstract avatar (e.g., low-poly style) could help to generate more tolerance to flaws. In a co-located

setting, such as the one in this project, an avatar's motions do not have to be exact copies of the corresponding user's ones because users would never sense the difference anyway (due to the worn HMDs).

Final Questions and Participants Conclusion

It was asked how realistic and natural the avatar's movements (both conditions separately) were perceived.

It was repeated that regarding condition A, the avatar appeared more stable and more consistent. More jitter was experienced in condition B. On the one hand, condition B could allow precise and realistic motions; however, it resulted in extreme glitches (tracking positions are meters off for short terms). The accelerations that occur during walking in condition B help to generate a natural appearance.

The question: *To what extent does it appear like a person is controlling the avatar?*, was difficult to answer. The acoustics, for example, to hear someone walking strongly supports the presence of the other user. Frequent intense artifacts (fast and unrealistic movements) were very irritating.

In the participant's opinion, it would be a good solution to generate more animations utilizing the previously discussed techniques. The received tracking data therefore may not be used continuously, but the generated animation just reacts according to them. That may lead to a more stable experience, and therefore it may foster the feeling of presence. Unlikely poses should be ignored first and only if the user remains in such postures should the avatar's animation react to that with slow transitions instead of the often immediate and fast transitions of the current implementation.

6.4.3 Summary

First, it must be noted that the performance phases of both interviews suffered from technical problems that did not occur in earlier tests. The PC that usually runs the server applications was not capable of operating all necessary USB devices anymore. Therefore, a different machine was used. The replacement machine had only one LAN adapter built in, which was occupied by the LiDAR hardware. Hence, the PC was connected to the router via wireless LAN. It is suspected that this circumstance is the reason for poor data rates, which resulted in additional jitter, during both interviews.

Both participants stated that the concurrency of the avatar's footfalls and the acoustic feedback of the real feet touching the ground strongly fosters the naturalism of the avatar's movements. That effect, compared to the utilization of predefined walk cycles (see section 4.1.1), which is typical for avatars current VR games and interactive experiences, is beneficial in such co-located settings.

Although it may be possible to noticeable optimize the tracking process by adding further hardware to the system, introducing further sanity checks to the animation process may remain necessary. Reworking the whole animation implementation under consideration of the animation experts suggestions (see section 6.4.2) seems to have great potential.

Chapter 7

Conclusion

The developed system has shown itself to be scalable and modular and therefore reaches the specified goals. The exchange of the tracking module during the project implementation was as uncomplicated as intended. During earlier stages of the project as well as across the evaluations, it became apparent that *PHARUS*, a 2D tracking technology, can be utilized to foster presence in a co-located virtual environments by providing data for controlling avatars.

Each of the implemented tracking solutions has its advantages and disadvantages. Improving the hardware may eradicate, or at least alleviate some of the more serious flaws. An improvement of the software, especially of the animation module (see section 5.2.4) including further sanity checks and animation techniques, such as those the evaluation has revealed (see section 6.4) would be quite feasible.

7.1 Hardware-Level Issues and Improvements

Most issues could potentially be eliminated, or at least be minimized to a reasonable extent without any changes to the hardware setup. Possible measures and the benefits are discussed in the following section. Future HMDs with a significantly higher processing power will allow greater freedom in the visualization of such virtual environments and thereby make some of the design rules in section 4.3 obsolete.

7.1.1 2D Setup

It has been shown that the *PHARUS* can be utilized as a data source for avatar visualization in virtual environments. Without changing the hardware setup, the experienced flaws could probably be solved or at least minimized by implementing the suggested improvements of the experts of the experts (see chapter 6). Section 7.3 also addresses these measures.

The most significant weaknesses of the utilized 2D tracking are the lack of data (foot orientation and foot height are not measured) and the inaccuracy. The introduction of additional acceleration sensors, which are attached to a user's feet could be a low-cost approach to tackle this issue. The combination of these sensors' acceleration data and the already existing data of the tracking system would just allow better approximations instead of accurate or perfect data. However, the whole filter and preparation process

would become more complex. The results, however, are expected to foster more natural animations. Inaccuracy and jitter of the 2D setup could be enhanced to a certain extent by adding more LiDARs. The additional devices would also help to avoid occlusion problems, which correspondingly increase with the number of users within the tracked space. Furthermore, more precise LiDARs could be introduced to improve the overall accuracy.

7.1.2 6DoF Setup

The *HTC Vive* setup offers sub-millimeter precise real-time tracking and theoretically an extreme high scalability. It is straightforward by design to add many trackers to the system. But when these trackers are attached to users who walk around in the tracked space, occlusion problems practically increase with the number of users. That was already relevant in the two-user scenario of this work. Short-term tracking losses and the wrong height data are the two drawbacks of the 6DoF solution. Therefore it is barely applicable to the targeted use case. The second generation of the *HTC Vive Lighthouse* will natively support a larger number of base stations in one setup. Using more base stations is an effective way to counteract the occlusion problem. The VR expert also suggested the introduction of further base stations. Beforehand, the 6DoF approach was expected to outperform the 2D approach. At a point in time in which the previously addressed flaws have been amended, that also may be the case.

7.2 Avatar Aesthetics and Animation

The chosen aesthetics and the quality of this work's avatar animations turned out to generally meet the requirements and are suitable to create a feeling of presence in the tested settings. Nevertheless, in summary, the goals are met but only in their minimum requirements. The shortcomings of both tracking solutions—concerning hardware as well as the implementation—impaired the outcome. The character animation built upon did not cancel out the stated deficiencies. Inspired by the interviews, a set of measures regarding the rig and its controls, which are expected to have a great impact on the avatar's appearance, were defined and are described in detail in the next section.

7.3 Future Work

The lessons learned during the creation of this work, including the implementation of the *MIKA* system, are valuable for future studies and developments of animation systems. Regardless if the same avatar (the same model and the same rig) is utilized in future projects, the following measures can achieve significant improvements to an avatar's animations. Clamping the foot position when it is presumed to have floor contact is one major measure that is essential for the 2D tracking implementation. The drift problem described in section 6.4.2 can also be solved. The implementation of spring mechanics to control the hip height during walking is also considered. To generate a more dynamic behavior in general, spring mechanics may also be installed to several other body parts of the rig. The activation of the foot joint will allow the avatar's feet to roll off. This feature can be implemented rather easily, but it potentially affects the naturalism of

the avatar's behavior positively. Even if resizing the avatar to the user's actual height requires a kind of calibration process, it is a good way to adjust the avatar's proportions to the user's. Further sanity checks should be added to the implementation to prevent unrealistic movements.

Appendix A

CD-ROM/DVD content

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 PDF Files

Path: /

Rammer_Daniel_2017.pdf Master's Thesis

A.2 Audio Files

Path: /interviews

Christoph_Anthes.mp3 Recording of the VR expert
Alexander_Wilhelm.mp3 Recording of the animation expert

A.3 Video Files

Path: /video

MIKA_TrackingComparison.mp4 A short introduction video

A.4 Figures

Contains all figures of this work.

Path: /images

*.jpg, *.png . . . . . Raster images
*.pdf . . . . . PDF files

A.5 Online Sources

Path: /online

LiDAR_SICK_LMS1xx.pdf [25]
At the Heart of It All.pdf [22]
Unity3D_Engine.com.pdf [26]
Samsung.com-GearVR.pdf [24]
Samsung.com-GalaxyS6.pdf [23]

References

Literature

[1] Antonin Artaud. *The Theatre and its Double*. Trans. by Mary Caroline Richards. New York, NY, USA: Grove Press, 1958 (cit. on p. 4).

[2] Frank Biocca. "Lighting a Path While Immersed in Presence: A Wayward Introduction". In: *Immersed in Media: Telepresence Theory, Measurement & Technology*. Ed. by Matthew Lombard et al. Cham: Springer International Publishing, 2015, pp. 1–9 (cit. on p. 13).

[3] Frank Biocca, Chad Harms, and Jennifer Gregg. "The Networked Minds Measure of Social Presence: Pilot Test of the Factor Structure and Concurrent Validity". In: *4th Annual International Workshop on Presence*. Philadelphia, Pennsylvania, USA, May 2001, pp. 1–9 (cit. on pp. 13, 14).

[4] Carolina Cruz-Neira et al. "The CAVE: Audio Visual Experience Automatic Virtual Environment". *Communications of the ACM* 35.6 (June 1992), pp. 64–72 (cit. on p. 5).

[5] C. Cruz-Neira et al. "Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment". In: *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*. IEEE Comput. Soc. Press, 1993, pp. 59–66 (cit. on p. 6).

[6] S. S. Fisher et al. "Virtual Environment Display System". In: *Proceedings of the 1986 Workshop on Interactive 3D Graphics*. I3D '86. Chapel Hill, North Carolina, USA: ACM, 1987, pp. 77–87 (cit. on p. 5).

[7] Ajo Fod, Andrew Howard, and MJ Mataric. "A Laser-Based People Tracker". *Robotics and Automation* 3.May (2002), pp. 3024–3029 (cit. on p. 28).

[8] Maia Garau et al. "The Impact of Avatar Realism and Eye Gaze Control on Perceived Quality of Communication in a Shared Immersive Virtual Environment". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 529–536 (cit. on p. 13).

[9] R E Kalman. "A new Approach to Linear Filtering and Prediction Problems". *Transaction of the ASME—Journal of Basic Engineering* 82 (Series.1 (1960), pp. 35–45 (cit. on p. 28).

[10] Finn Kuusisto. "VR Head-Mounted Displays". *XRDS: Crossroads, The ACM Magazine for Students* 22.1 (Nov. 2015), pp. 65–65 (cit. on p. 5).

[11] Joseph J. LaViola and Joseph J. "A Discussion of Cybersickness in Virtual Environments". *ACM SIGCHI Bulletin* 32.1 (Jan. 2000), pp. 47–56 (cit. on p. 9).

[12] Kwan Min Lee. "Presence, Explicated". *Communication Theory* 14.1 (2004), pp. 27–50 (cit. on p. 13).

[13] Kwan Min Lee and Clifford Nass. "Designing Social Presence of Social Actors in Human Computer Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 289–296 (cit. on p. 13).

[14] Clayton Lewis and Robert Mack. "Learning to Use a Text Processing System: Evidence from 'Thinking Aloud' Protocols". In: *Proceedings of the 1982 Conference on Human Factors in Computing Systems*. CHI '82. Gaithersburg, Maryland, USA: ACM, 1982, pp. 387–392 (cit. on p. 38).

[15] Marvin Minsky. "Telepresence". *OMNI Magazine* (May 1980), pp. 45–53 (cit. on p. 12).

[16] Otto Naderer. "Crowd Tracking and Movement Pattern Recognition". PhD thesis. Universität Linz, 2015, p. 100 (cit. on pp. 1, 31).

[17] Swaroop K. Pal, Marriam Khan, and Ryan P. McMahan. "The Benefits of Rotational Head Tracking". In: *2016 IEEE Symposium on 3D User Interfaces, 3DUI 2016 Proceedings*. Greenville, SC, USA: IEEE, Mar. 2016, pp. 31–38 (cit. on p. 22).

[18] Anjul Patney et al. "Towards Foveated Rendering for Gaze-tracked Virtual Reality". *ACM Transactions on Graphics* 35.6 (Nov. 2016), 179:1–179:12 (cit. on p. 9).

[19] Daniel Roth et al. "Avatar Realism and Social Interaction Quality in Virtual Reality". In: *Proceedings of the 2016 IEEE Virtual Reality Conference (VR)*. Vol. 2016-July. Greenville, SC, USA: IEEE, Mar. 2016, pp. 277–278 (cit. on p. 14).

[20] Ivan E. Sutherland. "A Head-Mounted Three-Dimensional Display". In: *Proceedings of the 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, pp. 757–764 (cit. on p. 4).

[21] Ivan E. Sutherland. "The Ultimate Display". In: International Federation of Information Processing IFIPS Congress, vol. 2. New York, May 1965, pp. 506–508 (cit. on p. 6).

Online sources

[22] Matthew Lombard and Theresa Ditton. *At the Heart of It All: The Concept of Presence*. 1997. URL: http://dx.doi.org/10.1111/j.1083-6101.1997.tb00072.x (cit. on pp. 12, 50).

[23] *Samsung Galaxy S6*. Sept. 2017. URL: http://www.samsung.com/uk/business/business-products/smartphones/smartphones/SM-G920FZWFBTU (cit. on pp. 21, 50).

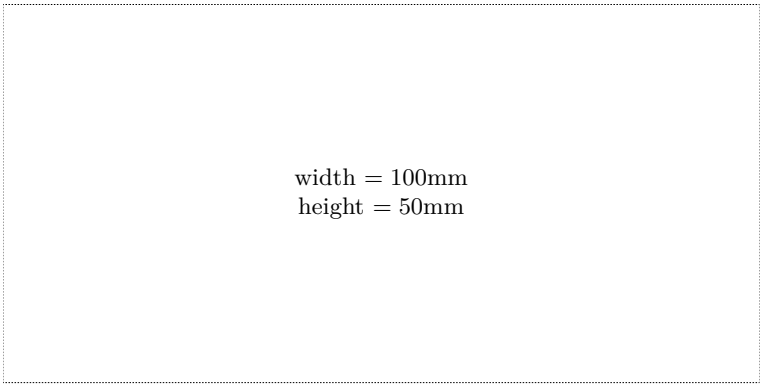
[24] *Samsung Gear VR*. Sept. 2017. URL: http://www.samsung.com/us/mobile/virtual-reality/gear-vr/gear-vr-sm-r322nzwaxar (cit. on pp. 21, 50).

[25] SICK. *SICK_LMS1xx*. 2017. URL: https://www.sick.com/media/dox/5/15/415/Product_information_LMS1xx_2D_laser_scanners_en_IM0026415.PDF (cit. on pp. 22, 50).

[26] *Unity3D Game Engine*. Sept. 2017. URL: https://unity3d.com (cit. on pp. 24, 50).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —