TU WIEN Informatics

Distributed Multi-User VR With Full-Body Avatars

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Oskar Perl
Registration Number 01625757

to the Faculty of Informatics

at the TU Wien

Advisor: Projektass. Dr.techn. Iana Podkosova, BSc MSc

Vienna, 28th December, 2020

Oskar Perl	Iana Podkosova

Abstract

Social virtual reality applications have the potential to provide a unique way to convey a sense of social presence when compared with other ways of communication in the area of computer mediated communication. Having a virtual body in a social virtual reality application can not only heighten the sense of embodiment of the user but can also convey a sense of presence to other users. General interest in social virtual reality applications is rising partly due to virtual reality devices become more affordable, including input technologies like hand tracking. This thesis aims to create a multi user virtual reality application that heightens social presence by representing users with a full-body avatar utilizing current entry level consumer grade virtual reality hardware. Hand tracking in combination with inverse kinematics is used to enhance perceived social presence. Those technologies provide a sufficiently convincing representation of the performed action of the user in a multi user context, while being significantly less cumbersome as solutions using additional trackers or controllers to realize a full body avatar.

Contents

CHAPTER 1

Introduction

In the last years Virtual Reality(VR) technology shifted from expensive prototype hardware, only used in research and industrial applications, more and more to an affordable consumer grade product. There is a multitude of head-mounted displays in low and middle consumer price ranges available. On the lower price point featuring the PlayStation VR headset [1], the Oculus Go [2] and Oculus Quest [3], the Oculus Quest and the lately released Oculus Quest 2 [4] stand out due to them not requiring a connected PC to operate and their cheap price point. The mid-range also features a wide variety of product like the older HTC Vive [5] and Oculus Rift S [6], or the newer Valve Index [7], HP Reverb G2 [8] and HTC Cosmos Elite [9]. In the mid-range the Valve Index stands out due to its innovative input system, where the controllers allow for individual finger tracking.

VR enables users to experience virtual worlds in 360 degrees and allows the user to interact with the environment in a natural and intuitive way. In most commercial systems controllers are used for interacting with the virtual environment. Controllers usually come in pairs and represent the users hands and abstract complex hand interactions, such as grabbing, with buttons and triggers. An alternative to controller input for VR interaction is hand tracking. Hand tracking allows the user to manipulate the virtual environment in a more intuitive way. Hand tracking has always been a research topic for various interaction use cases, where early prototypes used gloves [10]. Recent hand tracking solutions do not require gloves, instead they use cameras to track the user's hands. Leap Motions hand tracking technology follows this approach, as well as the integrated hand tracking of the Oculus Quest and Oculus Quest 2. Both are due to their affordability broadly available to consumers.

There are VR applications that allow multiple users to share the same virtual space. While many VR games focus on emulating the competitive multiplayer aspects of traditional multiplayer games, some VR games and applications feature a cooperative or purely social multi-user experience. Those social VR applications feature a unique way to interact

and collaborate with other users. By having an avatar controlled and animated by the user's movement in the real world, a multi-user VR application can convey a sense of social presence that cannot be conveyed in other interactive media or traditional ways of communication. After the start of the pandemic in 2020, many people had to work from home and were discouraged from meeting other people in person. The increased demand of a way of communication that conveys a sense of social presence and the broad offer of entry level VR hardware have renewed the interest in social VR platforms. There are already different social VR applications that allow multiple users to connect to a shared virtual environment, like Mozilla Hubs [11] and VRChat [12] and other platforms are being developed like Facebook's Horizon [13].

Social VR applications have the potential to fill the gap in the modern communication landscape, by providing a way of communication that conveys the presence of the communication partner and featuring unique ways of interaction. VR hardware is getting more affordable than ever before and social VR applications are rising in popularity. Considering those factors this thesis aims to create a multi-user social VR application of high fidelity, where users are represented with full-body avatars and hand tracking is utilized as a basis for an intuitive and natural interaction scheme.

1.1 Goal of the thesis

The goal of this thesis is to create a social VR application. This VR application aims to enhance the social of presence experienced by the clients in the virtual environment by representing the users with a full-body avatar. This full-body avatar shall display the actual movement of the user in a convincing fashion, as far as consumer grade VR technology allows, without having to rely on cumbersome and expensive capture technologies such as motion capture. To achieve this goal, a hand tracking system is used to capture the detailed hand movement of the user and applying it to the avatar. Further movement that cannot be captured by the head mounted display (HMD) or the hand tracking system, shall be generated procedurally. To procedurally animate the body of the avatar inverse kinematic solver need to be utilized to approximate and transform the remaining bones of the user's avatar to match hers or his posture. Furthermore should the clients be able to interact with the environment, allowing them to pick up objects and pass them to another. To achieve the social aspect of this VR application, the user's movement as well as their interaction with objects in the environment has to translate to the other clients using a networking solution. Finally, another important aspect is to choose technologies to realize this functionality in a way that the application can be experienced in a convenient fashion by limiting the amount of devices used to reduce setup time.

CHAPTER 2

Related Work

Multiple aspects are important for the implementation of a distributed multi-user social environment with full-body avatars that utilizes hand tracking as the main input system. To start with, embodiment plays an important role in representing different users in the scene to create a sense of social presence. Further, hand tracking is the key technology as it provides an intuitive way of interaction. This section will feature a selection of related work covering the mentioned key areas of this thesis.

2.1 Sense of embodiment and social presence

VR applications are unique when it comes to the potential of creating the sense of presence as they immerse the user into a virtual environment. Using an avatar to represent the player is a good way to increase presence. To achieve this heightened sense of presence the user needs to perceive the virtual body to be their own body by enhancing the sense of embodiment [14]. A social VR application has the potential to convey a similar social presence, defined as the feeling that the virtual environment is shared with others [15], as found in face to face communication.

Kilteni et al. describe the sense of embodiment as perceiving properties of a virtual body similarly to how one's own biological body is perceived [14]. In their work they further divided the concept of sense of embodiment into three sub-components: sense of self-location, sense of agency and sense of body ownership. They describe the sense of self-location as the sensation of the self being located inside a virtual body. Sense of agency describes how the effect of a performed action mirrors the expected outcome of this action. This implies that, if an expected consequence of an action matches the actual outcome of the action, the sense of agency is reinforced [14]. Finally, the sense of body ownership is described as perceiving the virtual body as one's own body trough visual and haptic feedback, as well as identifying oneself with the appearance of the virtual body.

Furthermore, Kilteni et al. suggest that the enhancement of the sub-components of embodiment leads to an increased overall sense of embodiment [14]. They propose several approaches to enhance the different sub-components. Self-location is highly dependent on perspective to strengthen the sense of being located in a virtual body. Therefore, using the first-person perspective where the user's virtual camera is placed at an eye level in a convincing fashion can improve the sense of self-location. To improve the sense of agency the virtual body has to respond to the actions the user performs in a quick and predictable way. This can be achieved, for example, by mapping the tracked movement executed by the user to the movement of the avatar in a convincing fashion. Finally, the sense of body ownership can be enhanced by providing convincing haptic feedback for performing actions in the virtual world and by allowing the user to choose or customize their virtual body.

Oh et al. investigated social presence and highlighted factors that contribute to conveying social presence in virtual environments [16]. In their findings they classify the predictors of social presence into immersive qualities, contextual properties and individual traits. One of the immersive qualities that contributes to a heightened perceived social presence is the visual representation of the communication partner. Oh et al. emphasized that the behavioural realism is one of the most important parts of the visual representation to enhance social presence [16]. Behavioural realism describes how convincingly the communication partner is able to act in a non-verbal way, be it through gestures, facial expressions or plausible eye contact. Other than behavioural realism, the quality of the visual representation in aspects to human likeliness and photo realism seem to have less impact on social presence [16]. Another immersive quality is haptic feedback. Providing the users haptic feedback while interacting with the environment or with each other can enhance to perceived social presence. Furthermore, Oh et al. describe the influence of contextual properties on social presence. Contextual properties rely, other than the immersive qualities, on more psychological aspects within the context of communicating in a virtual environment. For example, when the avatar of the user matches or is similar to his or her real appearance social presence can be increased [16]. A similar effect can be achieved by providing identity cues such as usernames or profile pictures. Displaying agency, by conveying that the avatar is controlled by an actual human, can increase social presence for the communication partner. Therefore, if the communication partner can perform convincing human-like actions a higher level of social presence can be achieved. This fact has been demonstrated experimentally [17]. Other contextual properties such as perceived physical proximity can enhance the social presence. This can be achieved by seeing the communication partner within a shared virtual space. The virtual space itself can also be enhanced for social presence by providing cues that indicate the presence of other users than the current communication partner, implying a social context [16].

2.2 Hand interaction

As mentioned in Section 2.1, increasing the sense of agency experienced by the user can increase the sense of embodiment. Tracking the hand of the user to animate the hands

of their virtual body can enhance the perceived agency, compared to controllers where different controller interactions would only trigger an animation of the performed action. However, using hand tracking to interact with objects or navigate user interfaces provides additional challenges.

Tscharn et al. conducted a study where participants would navigate a 3D map using hand tracking provided by a Leap Motion Controller and Space Navigator®, a physical control device [18]. While using hand tracking the navigation of the 3D map is controlled by gestures recognized by the hand tracking system. The efficiency of the Space Navigator® surpassed the hand tracking; however the participants found the hand tracking to be more enjoyable to use. Masurovsky et al. [19] found similar results in their study, where participants had to manipulate objects using hand tracking and the Oculus Touch controllers. Grabbing the objects using hand tracking was initialized by performing a grabbing gesture. The controller grab interaction was initialized by pressing a button. In this study, the Oculus Touch controllers outperformed hand tracking in usability. Additionally to the default grabbing interaction of the hand tracking, Masurovsky et al. provided a modified version where the virtual hand representation of the user would provide visual feedback when in grabbing range and while the grabbing interaction is successfully performed. This modified implementation performed slightly better than the default hand tracking solution.

One of the main reasons of efficiency loss in the study by Masurovsky et al. [19] was the frequent unintentional dropping of the virtual object. The interaction that initialized the grabbing interaction in [19] was based on recognizing gestures. Another approach for interacting with physical objects, without relying on gesture recognition, would be to model a physic based interaction model as proposed by Höll et al. [20]. Their VR interaction system simulates friction between the users tracked hands and the physical objects in the virtual environment. In their study they compared their system with the Interaction Engine provided by Leap Motion (which was also used as the default hand tracking based interaction in [19]). The participants found the solution by Höll et al. to be more realistic compared to the gesture based system.

2.3 Inverse Kinematics in VR

Inverse kinematics (IK) is an approach that allows to reconstruct the pose of a body based on several known body part poses. In particular, it can be utilized to approximate arm movement relative to the body using only the hand positions and rotations. This additional control over the virtual body's movement can, similar to hand tracking, heighten the perceived agency of the user. Parger et al. animated the upper body of a virtual avatar using multiple IK solvers and compared them to real-time motion-captured animations [21]. In their study participants performed different tasks while being embodied into an avatar representation that consisted of either only floating hands or arms animated by motion capture or inverse kinematics. They found that the IK-controlled arms were preferred to the floating hands. Parger et al. further concluded that the IK solution was indistinguishable from the motion capture, making this approach for animating a virtual avatar responsive to player input a good candidate for enhancing agency and embodiment without the need for additional sensors or motion capture suits.

CHAPTER 3

Methodology

This chapter describes the design of the implemented system and provides details about the choice of the used technology.

3.1 System design

Our multi-user application consists of three main components: tracking (including head and hand tracking), inverse kinematics and distribution. The workflow that connects these components is demonstrated in Figure 3.1. Head and hand tracking data is received on each user's computer and distributed to all other client machines over the networking system. There, head and hand tracking data is used to animate the avatar representing the user with the methods of Inverse Kinematics (IK). Similarly, each user interaction is distributed to other machines over the network.

Head and hand tracking

To animate and distribute the user's movement over the network some positions need to be tracked by the VR system. The most important positions that need to be tracked by this application are the head and hand positions. In this application to animate head movement the position and rotation of the Head mounted Display (HMD) needs to be tracked. To translate the users hand movement into the virtual environment the hands positions are captured by a hand tracking system, instead of tracking a controller. The hand tracking system allows further to track individual finger movement that will be used to animate the hands of the avatar procedurally. There are many ways how hand tracking can be realized.The main two hand tracking solution used in consumer grade VR are Ultraleap's(former Leap Motion) and the integrated hand tracking of the Oculus Quest and Quest 2 [22]. Both solutions use cameras and visual computing techniques to find key points that are used to generate a skeletal structure that can be used to animate

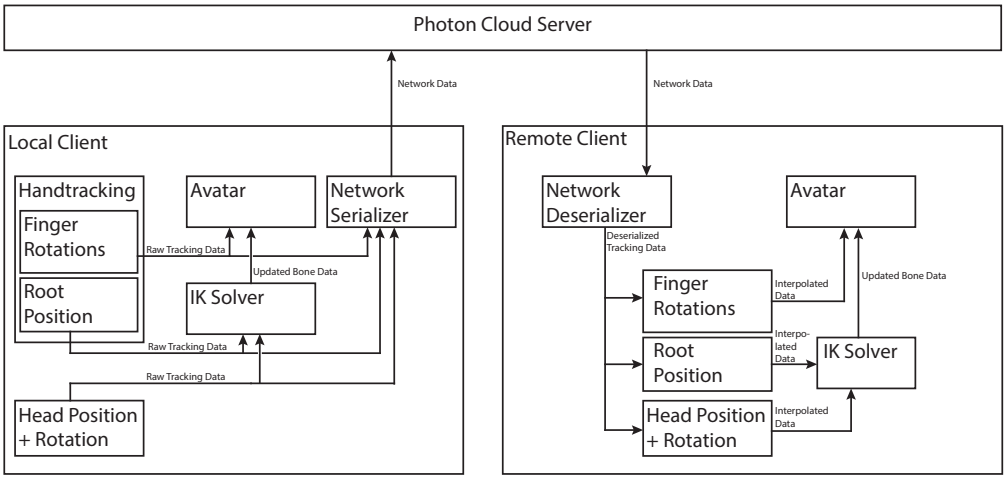


Figure 3.1: Schematic overview of how the different tracking-data is captured and processed by the IK solver to animate the local Avatar. The schematic further illustrates how raw animation data is distributed and used to animate the avatars of remote clients

a hand model. The hand tracking positional data captured by the hand tracking system is mapped to the corresponding bones of the chosen avatar, so that the avatar's hands mimic the users hand movement, as seen in 3.2. The positions tracked by the HMD and hand tracking system are synchronized over the applications network system to translate the movement of the user to all other connected users.

Figure 3.2: Hands of the avatar model animated with the real-time hand tracking input of user hands.

Inverse Kinematics

Since the avatar models used in this project have a humanoid skeletal structure, the limited amount of tracked positions is not sufficient to animate the avatar in a convincing fashion. To fill in the gaps, an inverse kinematic (IK) solver is needed. The inverse kinematic solver is then used to approximate the position of all other bones of the avatar that are not covered by the tracked positions. Other than in forward kinematics (FK), where the given joint rotations are used to calculate the end-effector position, IK uses the end-effector position to calculate the joint rotations, as displayed in Figure 3.3. Therefore, using IK is in our case useful since we have the end-effector positions (represented by the hand positions) and need to find the rotations for the remaining bones of the full-body avatar.

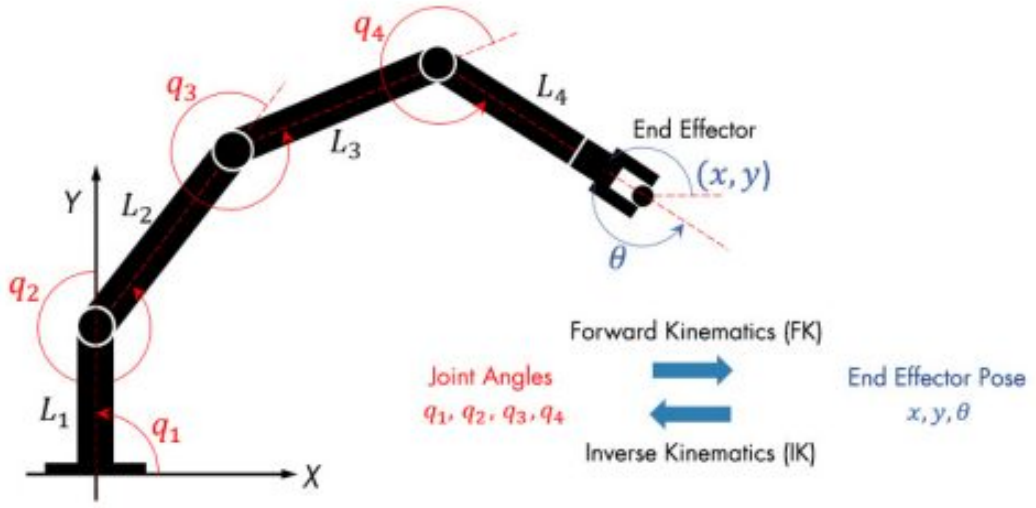


Figure 3.3: Illustration of FK and IK applied on an robotic arm.[23]

To ensure that the animations are immersive for the user, the avatar must be calibrated to the height of the player. Otherwise the arm's bone position derived from the inverse kinematic solver cannot position the arms in a way, such that the calculated hand position reaches the tracked hand target positions. The approach chosen for this project is to scale the avatar size proportionally to the height of the user. Another approach would be to scale the camera rig in relation to the user height. This would have the effect of resizing the user proportionally to the size of the avatar, however, real world height differences between users would not be preserved using this approach.

Distribution

Tracking input data, including the hand tracking data, is synchronized between all users in the environment. Body pose values derived from the inverse kinematic solver are

calculated locally for each client, including the avatars of remote players. An alternative approach would be to calculate IK-driven poses only for the local player and then distribute those poses to the other players. We chose our approach of calculating IK poses on every client because it reduces the amount of data that needs to be sent over the network. A possible drawback is that the positions of bones may vary between local and remote users, however, we never observed large discrepancies between the poses of players on different clients.

3.2 Choice of technology

3.2.1 Game Engine

In this Project we use the Unity3D engine. Unity is one of the more popular game engines and is used by some of the more popular VR applications like VR Chat [12] or Beat Saber [24]. Through various assets Unity provides VR development tools for different input devices.

3.2.2 Oculus Quest

The Oculus Quest is unique when it comes to VR systems. The Oculus Quest has the hardware to run VR applications and track user input built into the headset. Different from other HMDs the Oculus Quest does not need a PC or a console, eliminating the need for a cable. Furthermore, the Oculus Quest tracks controllers using the front mounted cameras of the headset, therefore not needing dedicated tracking stations such as the lighthouse system of the HTC Vive. Combining those two attributes, the the Oculus Quest is not restricted to a dedicated playing area, like other HMDs, making the headset highly portable and allowing usage in multiple different locations.

Additionally, the Oculus Quest can be connected to a PC via a single USB-C cable. This enables the Oculus Quest to run PC-VR applications and features unique benefits during development, allowing for rapid iteration cycles without having to build for the device.

The Oculus Quest also features a built in hand tracking system, eliminating the need for third party hand tracking solution like the Leap Motion Controller [25].

All in all the unique features of the Oculus Quest allows the user to enter the in this project realized application using only the headset without the need for a PC running the application or additional input and tracking devices.

3.2.3 Final IK

Final IK [26] is a Unity3D asset featuring a variety of IK solvers. To animate the full-body avatar used in this project, the VR IK solver featured in this package was used. This IK solver animates the whole body, including arm and head movement. Furthermore, the VR IK solver has a built-in locomotion solver that procedurally animates leg movement

according to the user's movement. This locomotion system works best for small areas, however being less suited for room-scale travel [26].

3.2.4 Photon Networking

Photon Unity Networking (PUN) [27] is a Unity3D asset that provides a network infrastructure and additional network functionality to Unity3D. It features a room based matchmaking system that allows objects and Unity3D scenes to be synchronized between players. It offers functionality for synchronizing common Unity3D properties like transforms, but also provides options for custom data synchronisation. Additionally PUN provides Remote Procedure Calls (RPCs) useful for distributing infrequent updates to other clients. PUN also features add-ons to support chat and voice chat functionality. Servers to run applications powered by PUN can be self-hosted, which is ideal for local testing. Additionally PUN offers a cloud based server structure to host the applications and created rooms. This allows for easy network testing, especially when the clients are not in the same location.

CHAPTER 4

Implementation and Use

This chapter covers the implementation and its components in detail. Assets used in this project are described here as well. Connection management, as well as all required components to use the players and intractable systems are described. Furthermore, instructions on how additional avatars can be created are provided in this chapter.

4.1 Used assets and packages

4.1.1 Unity3D

As explained earlier Unity is used in this project, since the available assets make it possible to create applications for special hardware such as the Oculus Quest. The Unity3D version used for this project is 2019.4.1f1.

4.1.2 Oculus integration

The Oculus integration provides Unity3D with tools to enable development for the different Oculus devices. Apart from the basic components like the camera rig and the interaction system, this project also utilizes the provided hand tracking features [25]. The version used in this project is v.16 [28].

4.1.3 PUN 2

To realize the network capabilities of this project Photon Unity Networking 2 is used. It provides a server connection system, as well as high level abstraction for serializing and de-serializing network data continuously, like player positions. It also features tools for sending RPCs for updating infrequently changed data. The version used in this project is v.2.16 [29].

4.1.4 Final IK

For enabling the earlier explained procedural arm and leg movement of the full body avatar the Final IK asset is used [26]. Final IK features multiple IK solvers. The solver used in this project is the VR-IK solver. The VR-IK solver is optimized for full-body avatar for VR applications and can be used on most avatar models outfitted with a humanoid bone structure. The version used in this project is 1.9.

4.1.5 Animation Rigging by Unity

The Animation Rigging by Unity was used as the IK solver before we switched to Final IK. It provides some basic IK solvers allowing to animate body parts such as arms procedurally. This package features also some tools. The one tool used by this asset is the Bone Renderer Component. This component highlights bones in the editor, which makes assigning bones to the correct fields of IK solver easy, even when using Final IK. The version used in this project is preview - 0.2.6.

4.1.6 Microsoft Rocketbox Avatar Library

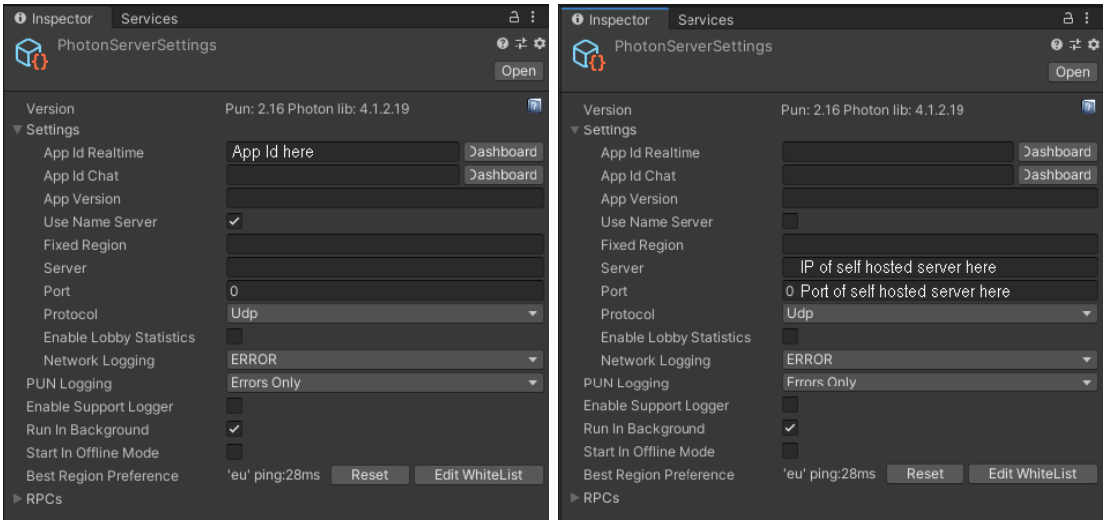
Microsofts Rocketbox Avatar Library is a collection of humanoid avatars freely available for research purposes [30]. Those avatars are fairly detailed and feature a standardized bone structure shared between all included avatars. Examples of virtual characters contained in the Rocketbox library can be seen in Figure 4.1.

Figure 4.1: Avatars from the Rocketbox Avatar Library used in this project.

Name	Type	Description
maxPlayersPerRoom	byte	Maximum number of players (default = 2)
playerPrefab	GameObject	The avatar instated in the scene representing the local player

Table 4.1: Serialized fields of the Network Manager.

4.2 Connection management



(a) (b)

Figure 4.2: Setting for a cloud server connection (a) and local server connection (b).

To connect to a server hosting the application, the Photon server settings have to be located and configured for cloud hosted servers or self-hosted servers, as described in Figure 4.2. To connect to a cloud server a app ID has to be generated with a Photon account and entered in the app ID field. Additionally the check box “Use Name Server” has to be checked. To connect to a self-hosted server server and port of the server have to be entered and “Use Name Server” has to be unchecked. Connection settings for both variants can be seen in Figure 4.2.

4.2.1 Network Manager

The Network Manager connects the clients to the Photon server and instanciates the referenced game object in the playerPrefab field in the scene. The Network Manager tries to connect the client to a room where the player limit has not been reached; otherwise it creates a new room. User-defined fields of the Network Manager are summarized in Table 4.1.

4.3 Player setup

4.3.1 Handtracking Map

The Handtracking Map is a class containing all possible bones of the OVRBone enum [25]. It is a part of both the left and right hand controller empty game objects in the avatar prefab. It is used as a reference map in the Handtracking Mapper, where the bone data of the OVRSkeleton is used to update the position of the referenced bones of the avatar. Not all bones need to be assigned, depending on the concrete implementation of the Handtracking Mapper. The interface for bones assignment is shown in Figure 4.3.

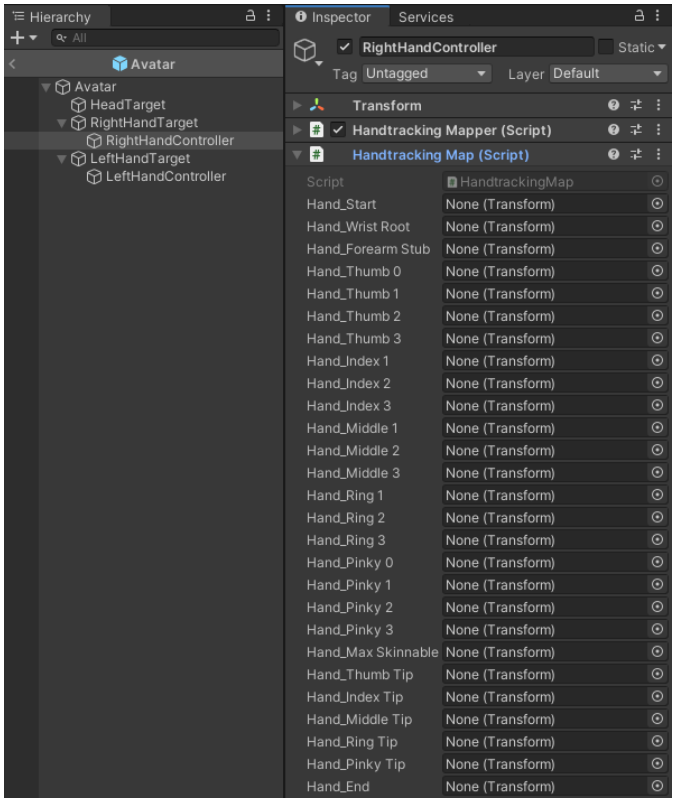


Figure 4.3: The Handtracking Map custom component used for mapping the Oculus hand tracking input to the referenced hand bones

4.3.2 Handtracking Mapper

The Handtracking Mapper is responsible for updating the rotations of the individual finger bones of the avatar hands, as well as distributing the updated rotations over the network. The rotations are updated for the local player instance using the OVRSkeleton data. Here, the OVRSkeleton data updates the local rotation of each finger bone reference from the Handtracking Map.

Name	Type	Description
_ovrSkeleton	OVRSkeleton	Reference for the OVRSkeleton script used to generate the bone data for this hand
_map	HandtrackingMap	Reference map of the avatar bones where the bone data of the OVRSkeleton should be mapped to
HandType	Hand	Enum used to set the hand used for this script, can be HandLeft, HandRight and None. Default None
hand	OVRHand	Reference of the current hands OVRHand to determine tracking confidence

Table 4.2: Serialized fields of the Handtracking Mapper.

Name	Type	Description
calibrationSize	float	Base scale for calibration
ik	VRIK	VRIK object holding the transform references for calibrating the height

Table 4.3: Serialized fields of the Height Calibrator.

The rotations of all referenced transforms in the Handtracking Map are serialized using OnPhotonSeriallizeView. The received data from networked instances other than the local player is then applied to the corresponding avatar instead of using the OVRSkeleton data. The received rotation data is also interpolated, so that the finger animations are displayed smoothly.

A summary of serialized fields that need to be assigned for this component can be seen in Table 4.2

4.3.3 Height Calibrator

The prototype features a basic height calibration for the hand tracking input. The system's reserved left hand pinch [25] is used to trigger a height calibration, as can be seen in Figure 4.4. In the calibration process the avatar is scaled to the height of the player at the time of triggering the action. This new scale information is also distributed to all remote players. Base scale can be adjusted and a reference to the Final IK solver need passed as can be seen in Table 4.3

4.3.4 Networked Player

The Networked Player script is used to initialize the avatar on creation as well as updating the target points for the inverse kinematic system, including head and both hand target positions. The targets are attached to the corresponding anchors in the OVRCameraRig

Figure 4.4: System reserved left hand Pinch gesture ([25]) used to trigger the height calibration

and are passed as references to the IK solver. If the confidence level of the hand tracking is low or lost, the Networked Player script saves the last tracked position and moves the hand target position to the last known position of the hands. If tracking is re-established, the OVRSkeleton is again used to update the hand position. A summary of serialized fields that need to be assigned for this component can be seen in Table 4.4

Name	Type	Description
customHandR	GameObject	References right hand game object containing an OVRHand script
customHandL	GameObject	References left hand game object containing an OVRHand script
headTarget	GameObject	The head target for IK solver, gets parented to the centre eye anchor of the OVR camera rig
rightHandTarget	GameObject	The right hand target for IK solver, gets parented to the right hand anchor of the OVR camera rig
leftHandTarget	GameObject	The left hand target for IK solver, gets parented to the left hand anchor of the OVR camera rig

Table 4.4: Serialized fields of Networked Player

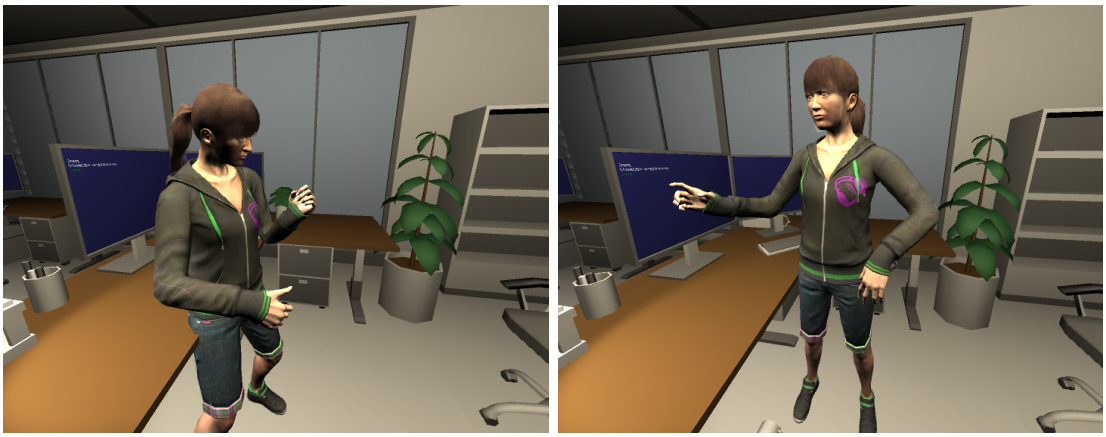


Figure 4.5: The avatar controlled and animated by a remote client, from the local users point of view

4.3.5 Avatar prefab

The avatar prefab represents the player in the scene. An avatar prefab consists of a head target, right hand target and left hand target, which are used for the inverse kinematic solver. Additionally a right- and left hand controller are used for the hand tracking input and the grabbing system and an avatar model is used to represent the player in the scene.

The avatar prefab uses a VR IK script from the Final IK asset to animate the avatar model using the targets within this prefab. A Networked Player from the view of the local player can be seen in Figure 4.5.

Name	Type	Description
velocityFactor	float	Modifies the strength of the velocity needed to throw the object on grab end. Default 1.0f
pinchThreshold	float	The threshold used to determine when the grab action should be triggered. Default 0.7f

Table 4.5: Serialized fields of the Handtracking Grabber

4.4 Distributed interactions

The distributed interaction system in this project is derived from the OVR grabbing system and expanded to work in a networked environment, using PUN as well as supporting hand tracking input.

Figure 4.6: Remote client interacting with an object from the local users point of view

4.4.1 Handtracking Grabber

The Handtracking Grabber is derived from OVRgrabber. It continuously checks the pinch strength of the OVRHand and calls the grabBegin function from the base OVRgrabber script, when the pinch threshold is exceeded by the observed pinch strength. The grab

end is triggered when an object is currently held and the current pinch strength does not exceed the pinch threshold. User-defined fields of the Handtracking Grabber are summarized in Table 4.5. Here the pinch threshold and the velocity multiplier for throwing can be adjusted. A Networked Player from the view of the local player can be seen interacting with objects in Figure 4.6.

4.4.2 Networked Grabbable

The Networked Grabbable derives from OVRGrabbable and overrides the GrabBegin and GrabEnd method. The Networked Grabbable aims to preserve the functionality of the original OVRGrabbable and applies the functionality to a networked context. The script transfers the ownership of the grabbed object as soon as someone initiates a grab action and distributes the physic properties of the object over the network.

4.5 New avatar integration

Adding a new avatar is simple by design, however there might need to be some adjustments depending on the skeleton of the desired avatar model. The skeleton of the hand portion of the model needs at least three bones in the pinkie and thumb, otherwise the whole range of motion of the hands cannot be represented in the chosen avatar.

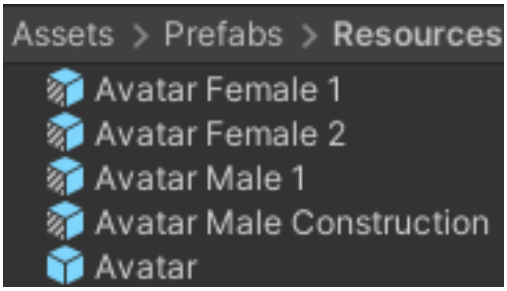


Figure 4.7: Location of the avatar prefab and its variants in the project folder.

There are already four different avatar prefabs in the project, all using models from Microsoft's Rocketbox avatar pack, featuring over 100 humanoid avatars from Microsoft. The scripts used in this project are tested using those models. For different models, adjustments to the Handtracking Mapper are required. In case of a different bone structure in the hand portion of the model the Handtracking Mapper needs to be adjusted to update and network all, potentially additional, bones provided by the model. Additionally, the orientation of the bones may vary between different models; in this case the positional and rotational offset needs to be taken into account.

To create a new avatar, locate the avatar prefab and create a prefab variant, illustrated in Figure 4.7. Then, add the desired character model to the root of the prefab. To rig the avatar to the inverse kinematic system, open the VR IK script located in the prefab root, as seen in Figure 4.8. Then drag and drop all bones from the avatar in the reference list

of the VR IK script. Some of the fields are optional, as hinted by a tooltip. Fields that do not feature an optional tooltip are required. All required fields need a reference.

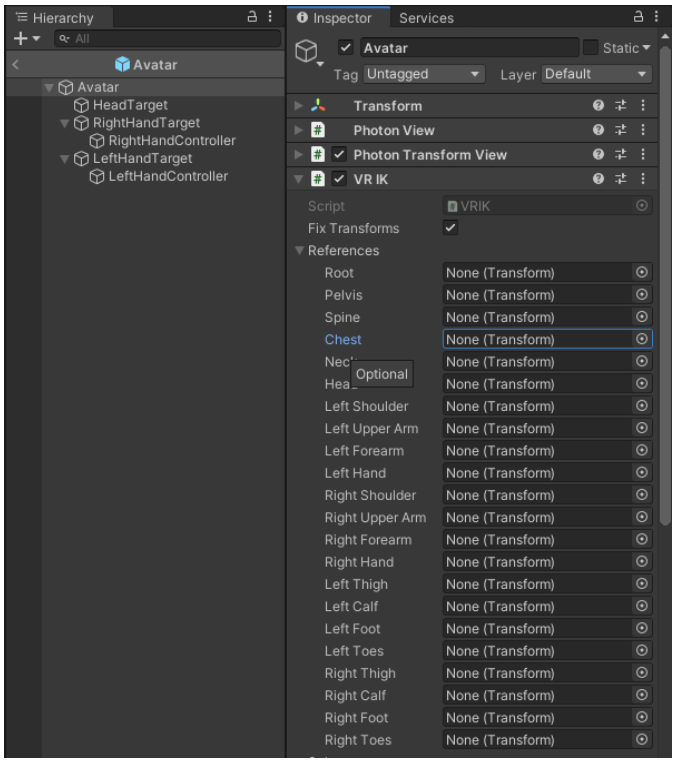


Figure 4.8: Fields of the VRIK component responsible for managing the bone references used by the IK solver.

To rig the hands, the Handtracking Map has to be located in the game object /RightHandTarget/RightHandController and /LeftHandTarget/LeftHandController respectively, as described in Section 4.3.1 and shown in Figure 4.3. Like in the VR IK script, the bones of the avatar need to be assign to the corresponding field in the Handtracking Map. The Handtracking Mapper is then responsible for updating and networking the hand tracking input on the avatar model.

As mentioned before, the Handtracking Mapper is designed for the Rocketbox Avatars and is not guaranteed to work with other skeletons. For models with a different skeletal structure a different implementation of the hand tracking mapper is needed to cope with the different bone structure and deviating offsets. To use the newly created avatar in the test scene, drag the avatar in the playerPrefab field of the Network Manager.

4.6 New grabbable objects integration

To the objects in the scene, that should be grabbable by the players, where all connected players should see the updated position, a Networked Grabbable script needs to be attached. All players in the scene need a Handtracking Grabber or an OVRGrabber attached to their hand anchors as well as a sphere collider set to trigger. The avatar prefab is already outfitted with all required scripts to utilize the grabbing system.

CHAPTER 5

Results

In this project, a multi user VR environment was created where the virtual body of each player is displayed and animated in a convincing fashion. With the combination of the Oculus Quest hand tracking and the Final IK solver, the movement of multiple users can be realistically and smoothly portrayed. Overall, this approach to user representation in VR is immersive and conveys a good sense of presence of other users in the same virtual environment and their scale, enabling plausible multi-user interactions. An example of two users interacting in a natural way is presented in Figure 5.1.

Figure 5.1: This figure shows two clients interacting in the virtual environment, displaying the animated full body avatars and the possible hand interactions. On the left side the scene is displayed in the Unity editor. On the right side the scene is shown from the local perspective of the local client.

5.0.1 Future improvements

Those benefits come with challenges and limitation. When it comes to hand tracking, the tracking range of the Oculus Quest is quite limiting. This requires the user to always keep their hands in their field of view to ensure that the hands are tracked correctly. Tracking becomes also difficult when the user performs rapid hand movement. Those limitations make actions like throwing an object quite difficult, since it requires fast movement and potential leaving of the tracking area while taking a swing. Another limiting factor is that tracking is lost when the hands are occluded, making actions where one hand needs to be in front of the other difficult to perform.

Using hand tracking limits the interaction possibilities compared to a controller. Conventional locomotion systems like using an analog stick for movement become not usable. Therefore, a room-scale play area is recommended to allow users to traverse distances in real space, limiting the potential size of the virtual environment. An option to allow for distance travel would be a teleport system, where a hand gesture would initialize a ray cast to the desired location and another gesture would confirm the teleport action. Another challenge can be found in UI interaction. A good approach would be to opt for a physical world space interface, like touchscreens in the scene, a wrist mounted interface to provide the users with additional options or buttons, to make up for the limited input system.

This illustrates that there are challenges to consider while creating a virtual environment designed for hand tracking and there are many possibilities when it comes to implementing VR interactions using hand tracking.

5.0.2 Conclusion

The results of this project show that using an all in one systems like the Oculus Quest, can provide a convincing full body avatar experience without the need for controllers or even a a designated playing area. Sufficiently accurate procedural animations of the avatar can be achieved using an IK solver, eliminating the need for any additional tracking technologies. Even though there are interaction limitations when it comes to hand tracking, immersive social VR interactions, utilizing full body avatars, can be achieved using entry level consumer grade VR technology making.

List of Figures

List of Tables

Bibliography

[1] Sony Interactive Entertainment. Playstation vr. https://www.playstation.com/en-us/ps-vr/. Accessed: 2021-03-29.

[2] Facebook Technologies. Oculus go. https://www.oculus.com/go/. Accessed: 2021-03-29.

[3] Facebook Technologies. Oculus quest. https://www.oculus.com/quest/. Accessed: 2021-03-29.

[4] Facebook Technologies. Oculus quest 2. https://www.oculus.com/quest-2/. Accessed: 2021-03-29.

[5] HTC. Htc vive. https://www.vive.com/us/product/vive-cosmos-elite/overview/. Accessed: 2021-03-29.

[6] Facebook Technologies. Rift s. https://www.oculus.com/rift-s/. Accessed: 2021-03-29.

[7] Valve. Valve index. https://www.valvesoftware.com/en/index/headset. Accessed: 2021-03-29.

[8] HP. Hp reverb g2. https://www8.hp.com/us/en/vr/reverb-g2-vr-headset.html. Accessed: 2021-03-29.

[9] HTC. Htc cosmos elite. https://www.vive.com/eu/product/vive/. Accessed: 2021-03-29.

[10] Thomas Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. volume 17, pages 189–192, 05 1986.

[11] Mozilla. Hubs by mozilla. https://hubs.mozilla.com/. Accessed: 2021-01-10.

[12] VRChat. Vrchat. https://hello.vrchat.com/. Accessed: 2021-01-10.

[13] Facebook. https://www.oculus.com/facebook-horizon/. Accessed: 2021-01-10.

[14] Konstantina Kilteni, Raphaela Groten, and Mel Slater. The sense of embodiment in virtual reality. *Presence: Teleoperators and Virtual Environments*, 21(4):373–387, 2012.

[15] Matthew Lombard and Theresa Ditton. At the Heart of It All: The Concept of Presence. *Journal of Computer-Mediated Communication*, 3(2), 09 1997.

[16] Catherine S. Oh, Jeremy N. Bailenson, and Gregory F. Welch. A systematic review of social presence: Definition, antecedents, and implications. *Frontiers in Robotics and AI*, 5:114, 2018.

[17] Wen Hai, Nisha Jain, Andrzej Wydra, Nadia Magnenat Thalmann, and Daniel Thalmann. Increasing the feeling of social presence by incorporating realistic interactions in multi-party vr. In *Proceedings of CASA 2018*, page 7–10, 2018.

[18] Robert Tscharn, Philipp Schaper, Jan Sauerstein, Sarah Steinke, Sebastian Stiersdorfer, Carsten Scheller, and Huy Tan Huynh. User experience of 3d map navigation – bare-hand interaction or touchable device? In Wolfgang Prinz, Jan Borchers, and Matthias Jarke, editors, *Mensch und Computer 2016 - Tagungsband*, Aachen, 2016. Gesellschaft für Informatik e.V.

[19] Alexander Masurovsky, Paul Chojecki, Detlef Runde, Mustafa Lafci, David Przewozny, and Michael Gaebler. Controller-free hand tracking for grab-and-place tasks in immersive virtual reality: Design elements and their empirical study. *Multimodal Technologies and Interaction*, 4(4), 2020.

[20] M. Höll, M. Oberweger, C. Arth, and V. Lepetit. Efficient physics-based implementation for realistic hand-object interaction in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 175–182, 2018.

[21] Mathias Parger, Joerg H. Mueller, Dieter Schmalstieg, and Markus Steinberger. Human upper-body inverse kinematics for increased embodiment in consumer-grade virtual reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, VRST '18, New York, NY, USA, 2018. Association for Computing Machinery.

[22] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. Megatrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics*, 39, 07 2020.

[23] MathWorks. Inverse kinematics (ik) algorithm design with matlab and simulink. `https://www.mathworks.com/discovery/inverse-kinematics.html`. Accessed: 2021-02-05.

[24] Beat Games. Beat saber. `https://beatsaber.com/`. Accessed: 2021-03-9.

[25] Oculus Documentation - Hand Tracking in Unity. https://developer.oculus.com/documentation/unity/unity-handtracking/. Accessed: 2020-12-28.

[26] RootMotion. Finalik. https://assetstore.unity.com/packages/tools/animation/final-ik-14290. Accessed: 2021-01-10.

[27] Exit Games. Pun v2. https://assetstore.unity.com/packages/tools/network/pun-2-free-119922. Accessed: 2021-01-10.

[28] Oculus developer release notes: v16. https://developer.oculus.com/blog/oculus-developer-release-notes-v16/. Accessed: 2020-12-28.

[29] Pun v2 version history. https://doc.photonengine.com/en-us/pun/current/reference/version-history. Accessed: 2020-12-28.

[30] Mar Gonzalez Franco and Eyal Ofek. Microsoft rocketbox avatar library now available for research and academic use. https://bit.ly/3uy0b9p. Accessed: 2020-12-28.