universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Virtual Reality Conferencing"

verfasst von / submitted by

Patrick David Pazour, BSc

angestrebter Akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 935
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik UG2002
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs

"Any sufficiently advanced technology is indistinguishable from magic."

Sir Arthur Charles Clarke

Acknowledgements

I would first like to thank my thesis advisor **Helmut Hlavacs**. He always helped me out when I got stuck with the direction of my research and presented new interesting viewing angles to solve technical problems to me. I could always rely on him to help me steer this thesis in the right direction.

I would also like to thank my work colleagues **Daniel Martinek** and **Andreas Janecek** who helped me by refreshing my latex knowledge and never hesitated to give me valuable feedback on the structural work of this thesis.

Finally, I must admit that without my partner **Julia Otte** I couldn't have managed to study successfully alongside work and in conclusion have written this thesis. She has my deepest gratitude for providing me with continuous encouragement and love throughout the last five years. Without you this could not have been accomplished. Thank you.

Patrick David Pazour

Curriculum Vitae

General

Name	**Patrick David Pazour**
Date of birth	October 12th, 1987
Place of birth	Gräfelfing, Germany
Nationality	Austria
Contact	patrick.david@pazour.at

Education

10/2016 - now	**University of Vienna** Master studies Media informatics
03/2013 – 07/2016	**University of Vienna** Bachelor studies Computer Science graduated with the degree of Bachelor of Science
10/2009 – 02/2013	**University of Vienna** Bachelor studies Musicology
07/2009	**Bundesrealgymnasium Baden, Biondekgasse** Matriculation

Work Experience

10/2017 - now	**Robimo GmbH** IT-Projectmanagement, Software Development
11/2016 - now	**IT-Freelancer** Software Development, Server Adminstration
03/2017 - 08/2018	**Research Group Multimedia Information Systems, University of Vienna** Student Staff
03/2017 - 08/2018	**Research Group Software Architecture, University of Vienna** Student Staff
02/2013 - 11/2016	**Ledl.net GmbH** Server Administration / Development, Technical Support
10/2011 - 01/2013	**Chillydomains (Hanival GmbH, Benefit Partner GmbH)** Technical Support
01/2011	**Bericht über die Jahrestagung der Österreichischen Gesellschaft für Musikwissenschaft 2010 "Sound recording. Musikalische Interpretationen im Vergleich** Report
09/2010 - 02/2011	**Faculty of Musicology, University of Vienna** Student Staff
07/2010 - 08/2010	**Internetradio Emap.fm** Internship

Contents

List of Figures

List of Tables

List of Abbreviations

AR	**A**ugmented **R**eality
FPS	**F**rames **P**er **S**econd
GCW	**G**lobal **C**hange **W**orld
HUD	**H**ead **U**p **D**isplay
HMD	**H**ead **M**ounted **D**isplay
SEM	**S**tructursl **E**quation **M**odeling
VE	**V**irtual **E**nvironment
VR	**V**irtual **R**eality

Dedicated to my father who
always supported me to
achieve my goals.

Abstract

Patrick David PAZOUR, BSc.

Virtual Reality Conferencing

The development and availability of immersive virtual reality devices has risen significantly in recent years. Combined with the latest milestones in computer graphics and the current motion tracking devices a certain feeling of presence inside the virtual world can be achieved which is aiming (or wishing) to be indistinguishable from the real world in the long term. Although this aspiration may be visionary at the present day, the possibilities of collaborative human interactions within a virtual reality system are already manifold.

This master's thesis focuses on the development and evaluation of a virtual reality conferencing application prototype supporting personalized user-based avatars for up to four persons joining remotely in a virtual conference room. The prototype has been evaluated in terms of the feeling of presence experienced inside the applications virtual environment.

For the evaluation two groups of five people each were individually joining a presentation inside a virtual conference room hosted by a photogrammetry-based 3D avatar of their experiment supervisor. The first group was equipped with virtual reality headsets, while the second group participated the presentation in front of a computer monitor without headset. Afterwards the two groups were asked to complete a form based on the *igroup presence questionnaire* scale to measure the spatial presence, involvement and the sense of realness inside the virtual conference room.

The results of the conducted experiments indicate a positive impact on the feeling of presence by users wearing virtual reality headsets as compared to the group without headsets.

Chapter 1

Introduction

"virtual reality, n. A computer-generated simulation of a lifelike environment that can be interacted with in a seemingly real or physical way by a person, esp. by means of responsive hardware such as a visor with screen or gloves with sensors; such environments or the associated technology as a medium of activity or field of study;" [21]

OED-Online (Oxford Dictionary)

1.1 Motivation

The recent years have shown a rapidly growing interest in virtual reality devices and means to integrate them into our daily lives.[1] A huge advantage nowadays as compared to the 90s or early 2000s are the technical graphical capabilities of modern head mounted displays[2] together with motion and body tracking[3]. With the means to create believable virtual reconstructions of known environments[4] and the emphasis to deliver an experience of presence inside this artificial reality to the user, the next missing link is to make effective collaboration possible.

To overcome the limitation of different regional homelands without spending a fortune on transportation, a modern and interesting solution can be network based virtual reality conference systems. Targeted at international corporations who rely on physical meetings and presentations as well as all matters of education of practical work flows a virtual reality based conference system can cut down the expenses of the most important resources: time and money.

Current sales and nearly weekly announcements of new virtual reality devices depict a strong influence on the consumers market. While the current number of virtual reality device users is comprehensible, further lowering of acquisition prices for displays and devices combined with enhanced comfort and acceptance by the user base might soon significantly enhance the audience for virtual reality environment software solutions.[5]

Researching the papers [7][13][12] which were published over the last decades illustrating the positive collaborative effects and possibilities of virtual reality environments and especially the immersive ones really demands for research on new solutions using the latest hardware technologies in this sector in our opinion.

A huge pitfall which should not be overlooked is to motivate users to explore and learn the interaction possibilities offered by immersive virtual reality devices and environments overcoming possible first antipathy and malaise.

[1]Playstation VR, htc VIVE, Oculus Rift etc.

[2]i.e. 2160 x 1200 pixel resolution with 90Hz for the htc VIVE head-mounted OLED display

[3]Accelerometer, magnetometer, gyroscope, lighthouse laser tracking system, constellation tracking camera, inside-out tracking camera, etc.

[4]i.e. photogrammetry based reconstructed environments

[5]current prices for full htc VIVE setup for example at 600 USD

We see virtual reality conferencing using the latest virtual reality devices as a beneficial choice for enabling cheap and environment friendly collaboration for distributed team members or companies.

1.2 Contribution

The main contributions of this thesis are:

- A working virtual reality conferencing application prototype
    - enabling meetings and collaborative work inside a virtual reality environment
    - with the possibility to present slides to all attendees and and emphasize certain parts by marking them
    - providing the integration of external 3D models and means to manipulate them within the virtual reality conferencing room
- An evaluation of presence inside the virtual reality conferencing application prototype
    - measured by experiments with two groups of five people joining a virtual reality conferencing meeting, where one group was equipped with virtual reality visors and the other group participated with their computers monitor

An overview of all the implemented features of the virtual reality conferencing application prototype followed by a closer look at each mechanic will be given at the beginning of chapter 3.

1.3 Problem Statement

The development of the virtual reality conferencing application prototype including the first survey for related work started at the end of 2016[6]. Despite technological VR breakthrough announcements seemingly every week most self acclaimed virtual reality conferencing applications were still experimental web based playgrounds with no integration of VR head mounted displays or state of the art motion tracked controllers. One of the most hindering facts

[6]time of the completion of the written thesis July 2018

to develop virtual reality business solutions is that the necessary hardware to render realistic environments is costly[7] and in addition a virtual reality headset with motion controllers is needed for each device including tech support in the current alpha/beta phase of the included device drivers[8].

A secondary goal of the virtual reality conferencing application prototype was to evaluate the effect of personalized avatars on the feeling of immersion and presence inside the virtual reality environment. To our knowledge at the time of working on the prototype no virtual reality conferencing tool or platform could be researched which offered personalized user-based 3D avatars.
The big challenges for the virtual reality conferencing application prototype were:

- Designing a realistic 3D environment
- Modeling realistic character faces based on photos taken beforehand for user-based 3D avatars
- Creating a robust network environment[9]
- Implementing collaboratively usable interactions (i.e. manipulating a 3D object together
- Syncing interactions, voice chat and animations
- Ideally still being able to render at 90 frames per second

1.3.1 Scope of the Work

The scope of this thesis was to implement a virtual reality conferencing prototype application and to test it later on with a small group of people to evaluate the presence and immersion while joining a presentation inside the virtual reality together with their supervisors 3D avatar. While developing the concept for the software implementation and getting to know the virtual reality headset and motion controller hardware many limitations arose for the simultaneous tracking of the mouth of the users while wearing virtual reality headsets and the unique face model generation for each user.

[7]minimum Nvidia GTX 1060 graphics card, 16 GB RAM memory, i7 latest generation processor
[8]i.e. Steam VR
[9]Peer to Peer local network in the scope of the prototype

This is why an exact tracking of the mouth of the user with a synced animation replication soon went out of scope, but is not forgotten for future implementation work since a lot of time was spent with research and experimentation in this topic.

Though the later chapters "Methodology And Implementation" and "Summary and Future Work" will mainly focus on the final virtual reality conferencing prototype and its evaluation, all the steps and branches which were finally left out in the scope of the project will still be summarized with focus on the lessons learned from them and their future outlook for the prototype application.

1.4 Structure of the Work

The following chapters are structured to start with an overview of the state of the art in "Related Work" 2 which covers a literature survey and analysis of the topics related to collaborative virtual reality applications and conferencing.

It is followed by an in-depth look into the methodology and structure of the virtual reality conferencing prototype application inside the chapter "Methodology and Implementation" 3.

The chapter "Experimental Evaluation" 4 describes the used igroup presence questionnaire, the experimental setup and finishes with a critical reflection on the execution and evaluation.

The last chapter of this thesis named "Summary and future Work" 5 summarizes the results and future outlooks of this work.

Chapter 2

Related Work

The collaboration of multiple users within virtual reality environments has been put to the test for a few decades now, as seen in several papers and publications from the 1990s until now[1], showing truly good results. However with the recent big leaps in the development of virtual reality headsets and motion tracked controllers[2,3] the price gap for the purchase of current high end equipment becomes smaller and smaller in favor of more immersive experience capabilities available to a broad consumer market.

In the scope of this research the surveyed papers and articles are classified into two main schemes: non-immersive and immersive virtual reality collaboration. The focus of this chapter is to compare results in different traditional non-immersive virtual reality collaboration scenarios with the experiences gathered from immersive ones.

The table 2.1 lists all surveyed papers ordered by their classification and furthermore the categorized usage scenarios. To the best of our knowledge the presented papers and articles on immersive collaboration in virtual reality environments were the most recent meaningful to survey.

[1]examples shown in the following table
[2]HTC Vive https://www.vive.com/
[3]Oculus Rift https://www.oculus.com/

Classification	Scenario	Title	Year
Non-Immersive	Design/ Education	Using Virtual Reality for Developing Design Communication [7]	2010
Non-Immersive	Design	Development of a web-based collaboration platform for manufacturing product and process design evaluation using virtual reality techniques [22]	2007
Non-Immersive	Conference	Mixing real and virtual conferencing: lessons learned [31]	2013
Immersive	Education	Peer collaboration and virtual environments: a preliminary investigation of multi-participant virtual reality applied in science education [13]	1999
Immersive	Education	Collaboration and Learning within Immersive Virtual Reality [12]	2000

TABLE 2.1: Classification of immersive and non-immersive papers.

2.1 Literature Studies

2.1.1 Non-Immersive

Using Virtual Reality for Developing Design Communication

This paper [7] by Gisli Thorsteinsson focuses on the relationship between collaboration and the design process.

The three base questions of the research were:

"How can a virtual reality environment be used for cooperative idea generation?", "How do communications during the lesson support students' work?" and "How does cooperation relate to teaching and learning within these lessons?".

The four test subjects consisting of two randomly picked boys and two randomly picked girls out of a group of voluntary students were able to choose a fitting avatar inside a collection of adult and children 3D models. Figure 2.1 shows the the students and their teacher using the virtual environment.

FIGURE 2.1: Figure from "Using Virtual Reality for Developing Design Communication" [7] showing the students and their teacher while using the virtual environment

The virtual reality environment itself was a house consisting of several rooms with a garden offering the opportunity for the subjects to draw together, play videos, browse the Internet and show power point presentations. Communication was enabled by voice over IP audio, text chat and body gestures via the avatar. Assignments were presented by a teacher inside the virtual reality environment, who trained them beforehand on the handling within the virtual environment as shown in figure 2.2.

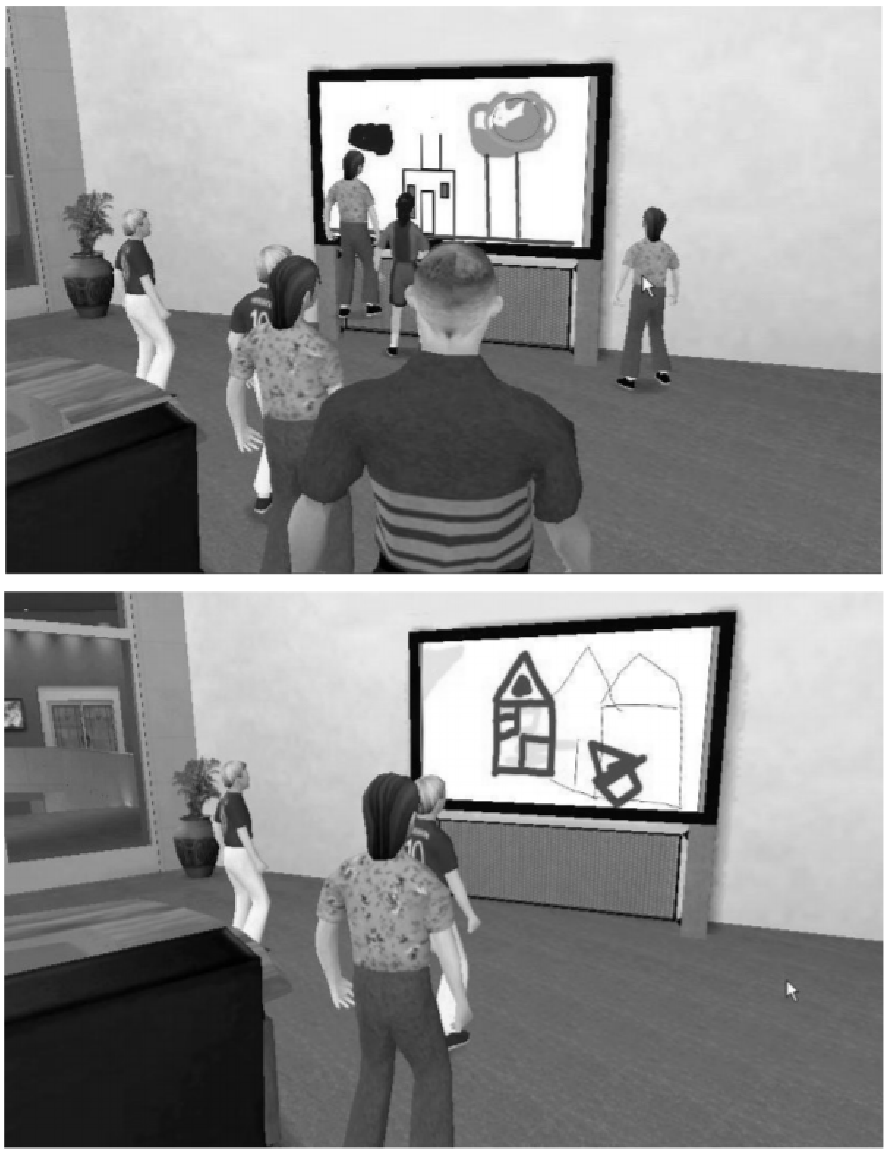


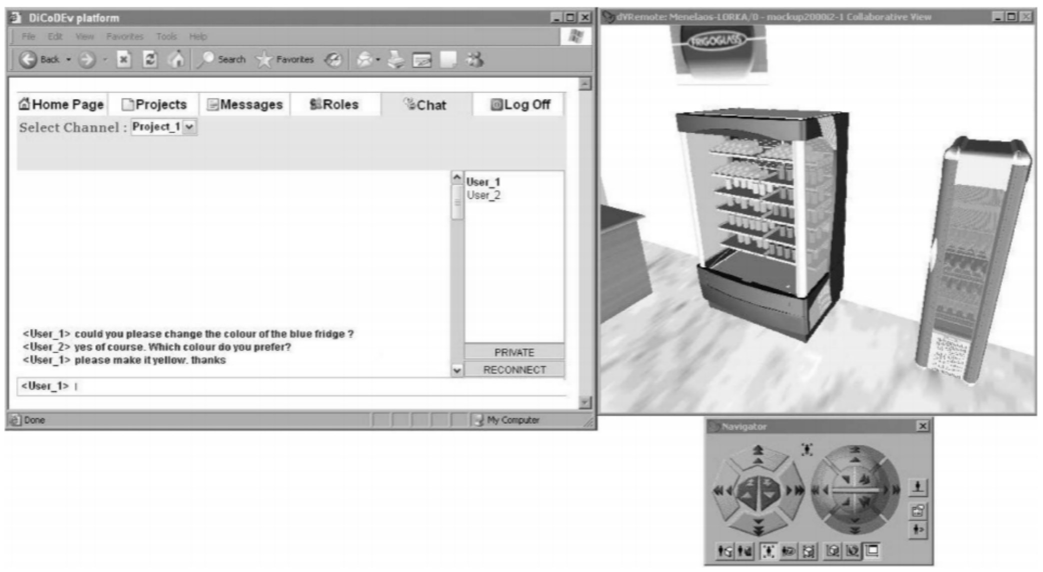
FIGURE 2.2: Figure from "Using Virtual Reality for Developing Design Communication" [7] showing the students and their teacher inside the virtual environment

The result of the collaboration was very silent communication and writing to each other. An effect by the chosen avatars was not investigated.

Development of a web-based collaboration platform for manufacturing product and process design evaluation using virtual reality techniques

This paper [22] by Pappas et al. describes a web-based platform for collaborative process and product design evaluation using a virtual reality environment.

The main goal was to enable real time validation of manufacturing processes and products collaboratively between distributed coworkers. The pilot case for research testing consisted of an virtual reality environment which was based on the requirements of an possibly actual manufacturing company. Figure 2.3 shows the real-time collaboration capabilities of the web-based platform.



```
<User_1> could you please change the colour of the blue fridge ?
<User_2> yes of course. Which colour do you prefer?
<User_1> please make it yellow. thanks
```

FIGURE 2.3: Figure from "Development of a web-based collaboration platform for manufacturing product and process design evaluation using virtual reality techniques" [22] showing the real-time collaboration capabilities

The abilities of the presented prototype were:

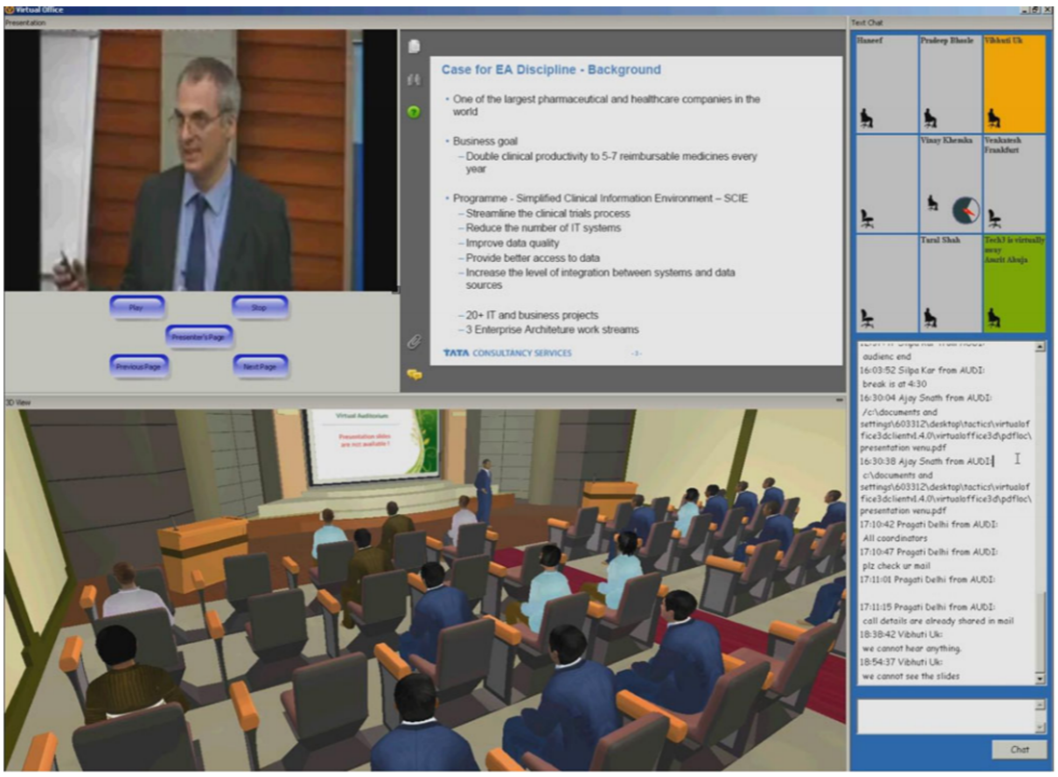
- cooperation among distributed designers and manufactures
- real-time multi-user interaction
- use/sharing and simulation of design and manufacturing data
- evaluation of the products using avatars
- activities in the virtual reality environment
- exchange based on the ideas of the 3D model of the product
- a product show room

Mixing real and virtual conferencing: lessons learned

An experiment of linking a conference with different remote location sites inside a virtual environment was conducted in this paper [31] by Sharma and Schroeder.

The research is based on data collected during an annual gathering of a global information technology consultancy company from India. The methods of the evaluation consisted of participant observation, surveys filled out by the remote attendees and interviews directed with the general audience consisting of the remote attendees and the real audience.

The virtual reality environment interactions included the camera view of the virtual lecture room, a virtual speaker avatar, user avatars identifiable with their names, navigation and avatar gestures, and recognition of questions from the virtual audience. Figure 2.4 illustrates the view seen by the remote audience.



Case for EA Discipline - Background

- One of the largest pharmaceutical and healthcare companies in the world
- Business goal
  - Double clinical productivity to 5-7 reimbursable medicines every year
- Programme - Simplified Clinical Information Environment – SCIE
  - Streamline the clinical trials process
  - Reduce the number of IT systems
  - Improve data quality
  - Provide better access to data
  - Increase the level of integration between systems and data sources
  - 20+ IT and business projects
  - 3 Enterprise Architeture work streams

TATA CONSULTANCY SERVICES -3-

```
audienc end
16:03:52 Silpa Kar from AUDI:
break is at 4:30
16:30:04 Ajay Snath from AUDI:
/c:\documents and
settings\603312\desktop\tactics\virtualof
fice3dclientv1.4.0\virtualoffice3d\pdfloc\
presentation venu.pdf
16:30:38 Ajay Snath from AUDI:
c:\documents and
settings\603312\desktop\tactics\virtualof
fice3dclientv1.4.0\virtualoffice3d\pdfloc\
presentation venu.pdf
17:10:42 Pragati Delhi from AUDI:
All coordinators
17:10:47 Pragati Delhi from AUDI:
plz check ur mail
17:11:01 Pragati Delhi from AUDI:

17:11:15 Pragati Delhi from AUDI:
call details are already shared in mail
18:38:42 Vibhuti Uk:
we cannot hear anything.
18:54:37 Vibhuti Uk:
we cannot see the slides
```

FIGURE 2.4: Figure from "Mixing real and virtual conferencing: lessons learned" [31] showing the view seen by the remote audience

One of the results findings was that at the time of the paper even low technical improvements would benefit the quality of the virtual reality conferencing experience greatly. Interestingly it was also revealed that the visual quality and fidelity of the virtual reality environment and the representing avatars of the members was far less important than the audio quality.

2.1.2 Immersive

Peer collaboration and virtual environments: a preliminary investigation of multi-participant virtual reality applied in science education

This paper [13] by Jackson, Taylor, and Winn examines peer collaboration inside an immersive 3-D virtual reality environment based on a preliminary study of 18 sixth graders.

The assignment of the students was to investigate the concept of global warming teaming up in pairs inside a 3-D model of Seattle called *Global Change World*. To encourage their collaboration the students were equipped with head mounted visors and an intercom system for voice chat communication.

The results showed that most students enjoyed working together inside this virtual reality environment and that the ability to work collaboratively played a significant role in terms of the individual engagement of the pairs.

Collaboration and Learning within Immersive Virtual Reality

The experiment in this paper [12] by Jackson and Fagan is a follow up to the previously described inside an everyday school social environment.

This time 56 ninth graders were tested in three different collaborative scenarios while again being inside the *Global Change World* to investigate the concept of global warming. The different experiences were conducted by singular subjects with minimal support, pairs of subjects working in collaboration and singular subjects working in collaboration with an expert as companion. Figure 2.5 shows a carried out selection inside the *Global Change World* toolkit.

Within the results no direct evidence could be found that the immersive virtual reality environment provided a beneficial collaboration effect. However a benefit on the educational effects was traceable and formed the conclusion

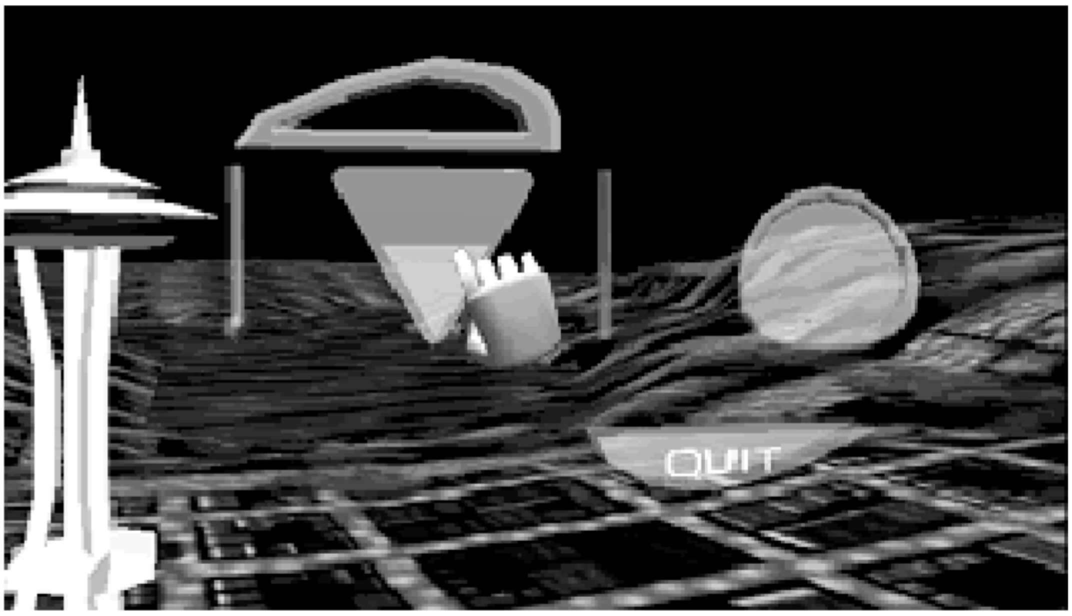


FIGURE 2.5: Figure from "Collaboration and Learning Within Immersive Virtual Reality" [12] showing a selection inside the *Global Change World* toolkit.

that immersive virtual reality collaboration can be successfully integrated into existing school curricula.

2.2 Analysis

A common thread through papers [7], [13] and [12] was that some of the test subjects had problems navigating or using the controls inside the virtual reality environment which was especially true for the immersive ones. A break through in easy to pick up interaction devices is yet to come which will enable instant adaptive mechanics to interact within virtual reality environments. Game based tutorials to motivate and guide the user through interaction tasks should be highly anticipated.

2.3 Summary and Result

A direct comparison between the non-immersive papers [7], [22], [31] and the immersive ones [13], [12] can not be drawn.

The classification used at the beginning of this chapter between the two categories is therefor only based on the fact of the applied technology with no further side effects which distinguish and remark the classification.

A mildly spoken common fact were mostly good results beneficing collaboration inside virtual reality environments, while the researches often predicted even better results without the technical limitations of the date of the publication.

Chapter 3

Methodology and Implementation

This chapter explains the details of the virtual reality conferencing application developed for this master thesis.

The final prototype includes the following features:

- Up to four attendees per session
  - A host initiates a session and up to three clients can join within the same local network.
- 3D avatars
  - Integration of user-based faces modeled by provided photos together with a generic model inside the application.
- 3D avatar facial animation
  - The facial data is tracked by the webcam and dependent on lighting, distance and orientation of the user. If a feature is matched it will be animated on the avatar. While wearing a headset, the facial animation is controlled by the speech of the user which is also a fall-back if no feature is matched without a headset.
  - Wearing a VR headset, the users head movement is tracked and animated on the avatar up to a natural movement angle. Without a headset, the user can move the head of the avatar by "looking around" with the mouse.
  - Additionally, an avatar idle and avatar speaking animation with gestures for the whole body is prepared.

- 3D VR room creation – conference
  - Realistic office room environment with immersive features like a ticking real time clock and traffic outside the conference room.
- 3D avatar placement /localization
  - Automatic round robin placement around the office table
- Voice transmission
  - Voice packages will be sent approximately every fully captured second, guaranteeing a short delay
- External data integration
  - Obj Files can be uploaded by the host within the menu, prior to initiating hosting the conference. The object can be seen by all participants.
- External data interaction
  - The object can be rotated, marked, scaled, grabbed and moved around by the host. The changes can be seen by all participants.
  - The host can mark slides and advance/return to pages in the slides. The changes can be seen by all participants. Additional every participant can display a zoomed in version of the current slide in front of him.
- Login
  - Different users will be able to login by provided user-names and passwords. The appropriate avatar is linked to a given user login and will be assigned automatically.

The further sections will present the controls, used technologies, frameworks and libraries being followed by a closer look at the character generation and the design of the virtual environment. At the end we will take a closer look at the face tracking, software and networking architecture.

3.1 Controls

The virtual reality conferencing application is designed to be usable with and without a virtual reality headset including motion controllers.

VR button	Non-VR key	function
-	Space	reset face tracker
-	Enter	enable/disable face tracker
-	1	spawn exclamation mark emote
-	2	spawn hash-tag emote
-	3	enable/disable close up HUD of presentation
-	0	disable/enable general HUD

TABLE 3.1: General controls of the virtual reality conferencing application

It is differentiated between hosts which start the conferencing room and optionally upload a presentation and 3D object and users who can join a hosts conferencing room. The controls are mapped in the table 3.1 showing the general controls for the host and the user and table 3.2 illustrating the host-only controls.

3.2 Used technologies, frameworks and libraries

The first time consuming step of this thesis was to find the fitting tools and libraries among the latest frameworks and technologies to reduce the necessity of implementing every link of the chain anew. Table 3.3 lists the used technologies, frameworks and libraries for developing the virtual reality conferencing application prototype.

3.3 Hardware

3.3.1 Requirements

To ideally target around 90 fps inside the virtual reality environment, the hardware requirements revolve around a mid tier and higher classified gaming PC or laptop with Windows 8 or higher.

As a fall-back the whole virtual reality conferencing prototype can be used without a virtual reality device but is recommended to be used with the HTC VIVE. The official recommended hardware requirements for the virtual reality headset HTC VIVE from the HTC Corporation are listed in table 3.4.

VR button	Non-VR key	function
controller trigger	left mouse button	grab 3D object
move controller with held trigger	move mouse with held left mouse button	move 3D object
-	return	reset 3D object
left and right direction of controller touch-pad	arrow keys	rotate 3D object
up and down direction of controller touch-pad	mouse wheel up/-down	enlarge/shrink 3D object
right motion controller grip button held	-	toggle laser pointer for painting
move controller with held trigger and right motion controller grip button	move mouse with held right mouse button	paint on 3D object or presentation
left motion controller grip button	middle mouse button	reset painting
right motion controller upper button	left ctrl	next presentation slide
left motion controller upper button	left shift	previous presentation slide

TABLE 3.2: Host controls of the virtual reality conferencing application

Title	Author	Category
HTC VIVE[1]	HTC Corporation	Hardware
Unity® Version 5.6.5-32 Bit[2]	Unity Technologies	Engine
Agisoft PhotoScan Standard Edition[3]	Agisoft LLC	Characters
FaceGen Modeller[4]	Singular Inversions	Characters
Adobe Fuse[5]	Adobe	Characters
ARToolkit[6]	Philip R. Lamb	Face-tracking
CSIRO Face Analysis SDK [19][7]	CI2CV	Face-tracking
VoiceChat [8] library[8]	Holmström	Voice Chat
Steam VR library for unity[9]	Valve Corporation	VR Interactions
Windows[10]	Microsoft	Platform

TABLE 3.3: Used technologies, frameworks and libraries

Component	Recommended system requirements
Processor	Intel Core i5-4590/AMD FX 8350 equivalent or better
GPU	NVIDIA GeForce GTX 1060, AMD Radeon RX 480 equivalent or better
Memory	4 GB RAM or more
Video Output	HDMI 1.4, DisplayPort 1.2 or newer
USB Port	1x USB 2.0 or newer
Operating System	Windows 7 SP1, Windows 8.1 or later, Windows 10

TABLE 3.4: HTC VIVE hardware requirements [5]

In order to use the voice chat a built in or external microphone as well as speakers or a headset are needed. The optional face tracking requires a common web cam to be enabled. For the tracking of head or hand movements with a virtual reality headset and motion controllers no additional camera is needed.

3.3.2 Virtual Reality Headset

The Oculus Rift[11] motion controllers launched in December 2016 [36] shortly after the first research ideas for this thesis evolved. Because the HTC VIVE shown in figure 3.1 launched earlier in 2016 [35] and already some experience with the hardware and its interactions in Unity® with the Steam VR unity plug-in from Valve Corporation was gathered beforehand, the virtual reality conferencing application prototype was developed from the start with the capabilities of the HTC VIVE in mind. Though one of the main features of the HTC vive is its room scale tracking feature the virtual reality conferencing application prototype was considered a seated experience from the start where the head and hands had only to be tracked in front of the used computer or laptop.

3.4 Virtual Environment Design

3.4.1 Conference Room Scene

An important aspect of this prototype was to deliver an immersive experience inside a believable environment. To recreate the feeling of being inside

[11]Homepage https://www.oculus.com/

FIGURE 3.1: Picture of HTC VIVE taken from https://www.vive.com/us/setup/vive/

of a meeting room a small business environment was built with the help of several selected individual assets and asset packages from the Unity® asset store. The final result is shown in figures 3.2 presenting the room from the right side, 3.3 presenting the room from the front-facing side and 3.4 giving a look at the street-facing side.

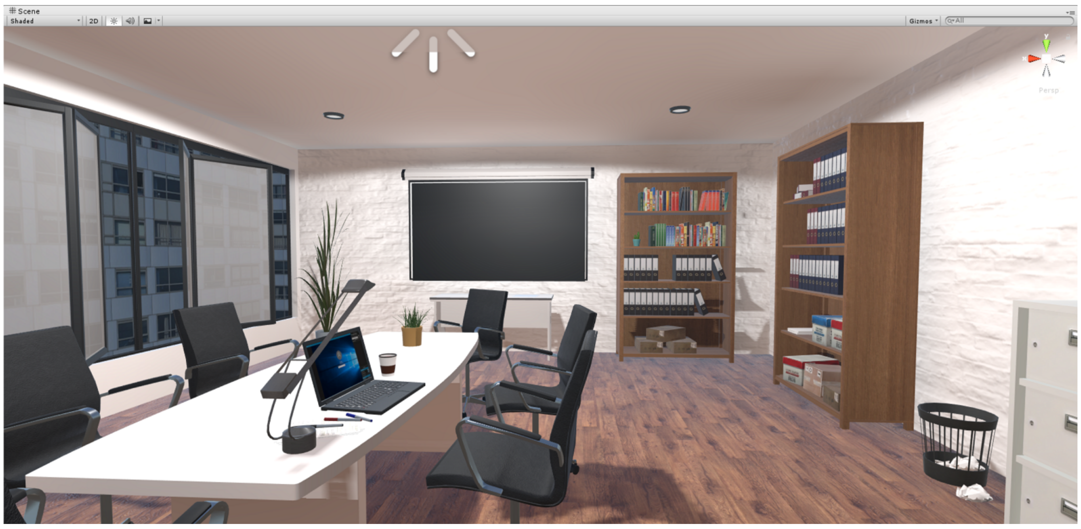


FIGURE 3.2: Screen-shot of the Virtual Reality Conference Room scene taken in the Unity® Editor

3.4.2 Immersive Add-ons

To further enhance the experience street noise in Vienna outside of the faculty of computer science was recorded and looped back into the scene positioned at the half opened windows in the conference room. Additional small but noticeable features like a clock showing the current real time with its moving handlers (figure 3.3) and cars passing by at street level a few floors below seen through the windows (figure 3.4) were carefully designed and placed inside the virtual environment.

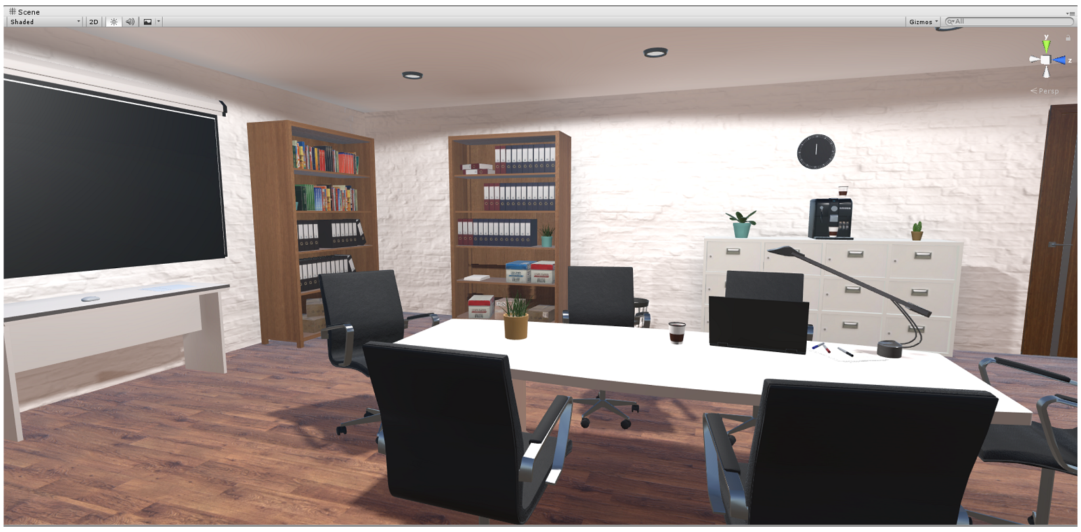


FIGURE 3.3: Screen-shot of the Virtual Reality Conference Room scene immersion details taken in the Unity® Editor

Subtle bloom and color grading image post processing effects were added to enrich the quality of the scenes with a focus on performance. To reduce aliasing artifacts Unity®'s FXAA [18] implementation was used.

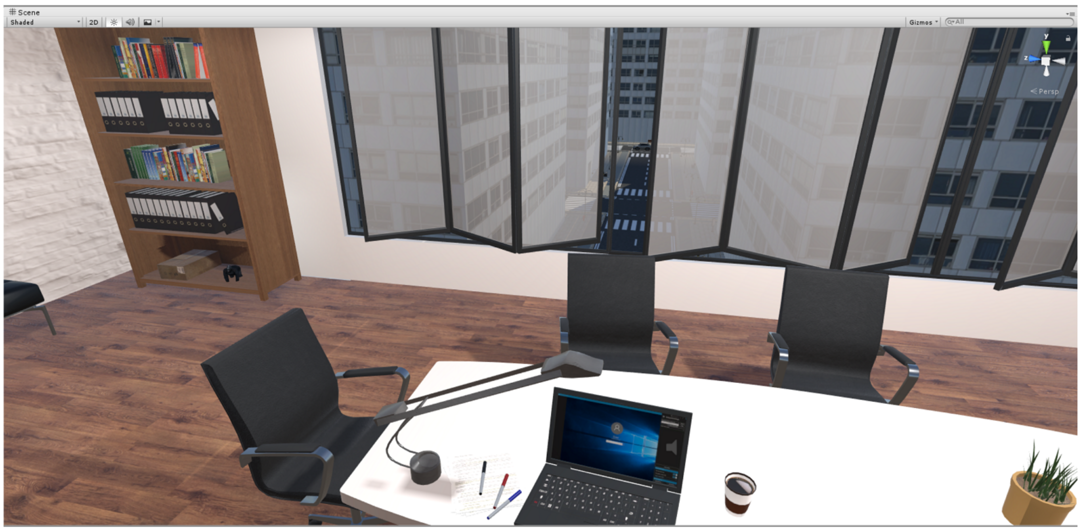


FIGURE 3.4: Screen-shot of the Virtual Reality Conference Room scene streets view taken in the Unity® Editor

3.5 Avatar-generation

3.5.1 First approach with Agisoft PhotoScan

The first approach used to generate realistic user based avatars was testing the capabilities of photogrammetry with the help of Agisoft PhotoScan, which is a software application to create 3D spatial data from photos. While there is no official statement on which papers and algorithms Agisoft PhotoScan is based on directly, a member of the Agisoft Technical Support named Dmitry Semyonov stated in 2011 in a forum post on Agisofts official community forum that the application is not based on the popular Bundler + PMVS2 + CMVS assembly based on the paper "Photo Tourism: Exploring Photo Collections in 3D" [33] by Snavely, Seitz, and Szeliski but described individual processing steps directly quoted as follows [30]:

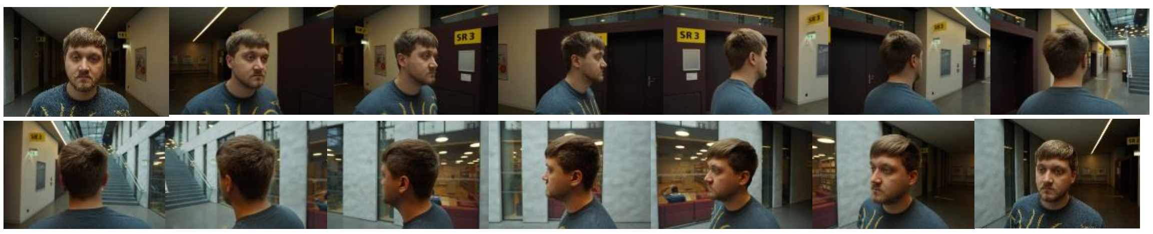
> *Feature matching across the photos.*
> *At the first stage PhotoScan detects points in the source photos which are stable under viewpoint and lighting variations and generates a descriptor for each point based on its local neighborhood. These descriptors are used later to detect correspondences across the photos. This is similar to the well known SIFT approach, but uses different algorithms for a little bit higher alignment quality.*
>
> *Solving for camera intrinsic and extrinsic orientation parameters.*
> *PhotoScan uses a greedy algorithm to find approximate camera locations and refines them later using a bundle-adjustment algorithm. This should have many things in common with Bundler, although we didn't compare our algorithm with Bundler thoroughly.*
>
> *Dense surface reconstruction.*
> *At this step several processing algorithms are available. Exact, Smooth and Height-field methods are based on pair-wise depth map computation, while Fast method utilizes a multi-view approach.*
>
> *Texture mapping.*
> *At this stage PhotoScan parametrizes a surface possibly cutting it in smaller pieces, and then blends source photos to form a texture atlas.*

The first step was to test out the capabilities of Agisoft PhotoScan and if the generated models were fit to give an immersive and believable experience inside the virtual reality environment.

To gather more test data to experiment with, we created a document called "Face Mesh Photography Manual" as a reference for the correct process and work-flow when taking photos for faces to be modeled later on in Agisoft PhotoScan. The document is based on the official advises given by Agisoft LLC in their manual for Agisoft PhotoScan [16].

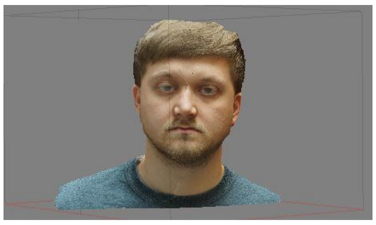
The following two pages show the original "Face Mesh Photography Manual" document:

Face Mesh Photography Manual



At a minimum level 12-16 pictures must be taken at eye level in a circle around the head. The best outcome can be achieved with 3 sets of this photos: One taken from eye level, one slightly above and one slightly below.

Example intermediate result:



Equipment:

- Digital single-lens reflex camera with at least 12 megapixels
- Ideally a 50mm fixed length focal lens or at least a lens from 20mm to 80mm. (When using a zoom lens, it is important that the zoom level is not changed while taking the photos)
- Tripod recommended but not necessary
- Seat for the photo model recommended

Lighting:

- The best lighting is like a bright cloudy day
- No projection shadows and only a small occlusion
- Reflection spots should be avoided
- The face should be evenly lit from all directions
- Don’t use a camera flash

Camera settings:

- Use manual mode
- Use the highest resolution possible
- Save the images in addition to JPEG/TIFF/etc. as well in RAW
- Use a high aperture value to get more sharpness and depth
- Ideally ISO 100, however can be adjusted higher depending on the light conditions and the noise
- Set the white balance according the light conditions to achieve the same colour temperature on all photos

Taking the photos:

- The individual photos taken next to each other should overlap by at least 60-80% (the more the better)
- At least 60% of the whole photo should be in focus
- The captured face must not be blurry

The quality of the produced face meshes depends very largely on the taken photos. Blurry images and dark shadows casted on the facial area should be avoided at all costs.

3.5.2 Results with Agisoft PhotoScan

Unfortunately even with much more effort applied we could not get much past the quality as seen in figure 3.5. The biggest problems were missing pieces of hair and holes at arbitrary positions inside the model which we would have to fill manually with the help of a modeling software application.

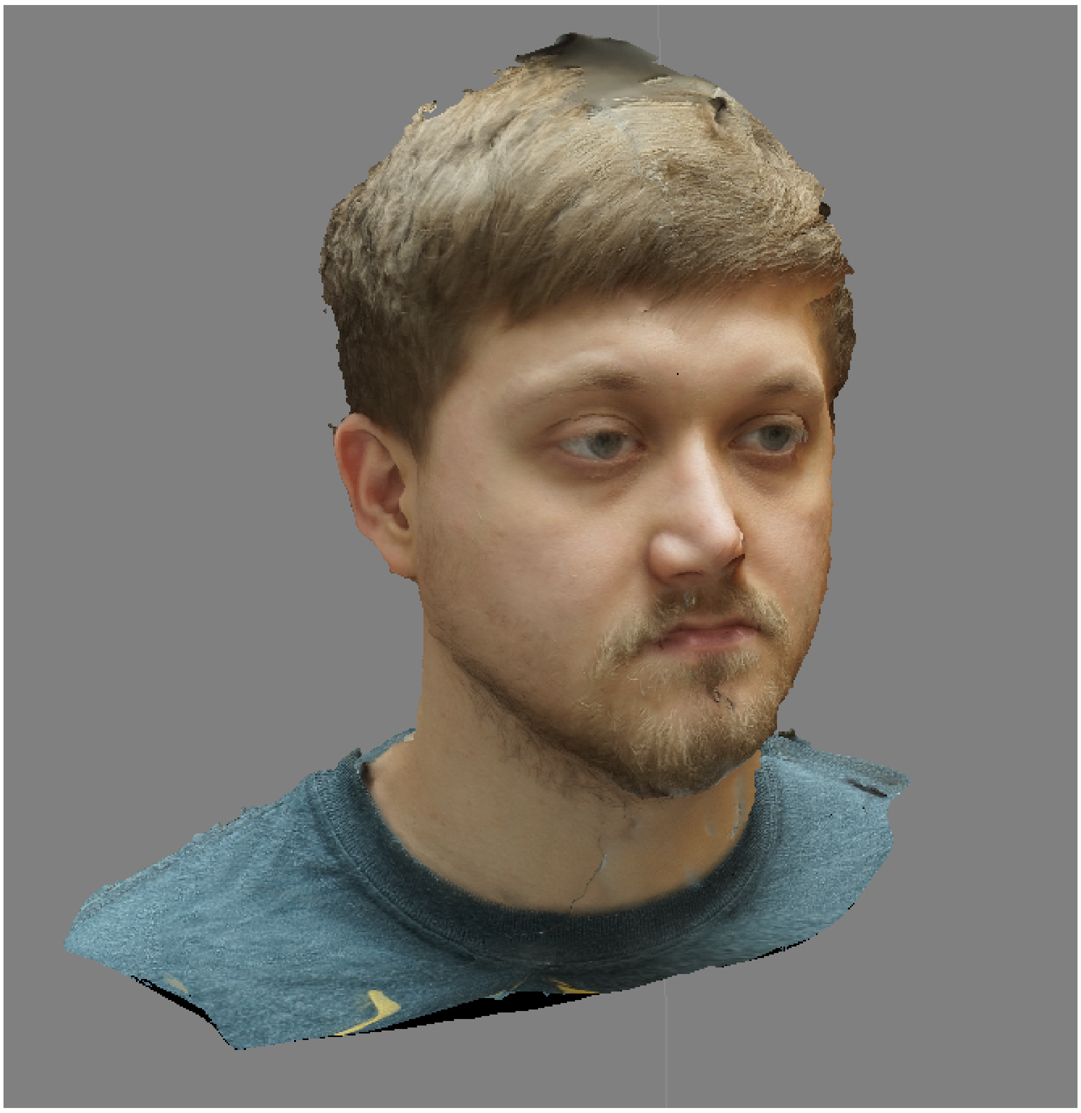


FIGURE 3.5: Screen-shot of reconstructed Face taken in the Agisoft PhotoScan Editor

Following reason the experimentation with Agisoft PhotoScan for this thesis stopped here and we went on with another application named FaceGen Modeller, described in the following section. With more time at hand and in regards to a future outlook, the approach taken with Agisoft PhotoScan promises a much more detailed result than with other methods and should be researched and evaluated further.

3.5.3 Morph-able Models with FaceGen

FaceGen Modeller is a software application by Singular Inversions and directly quoted described by them "to create realistic animatable 3D face and head meshes" [10].

This application offers a simple solution to quickly create usable avatar heads created from only three photos. In comparison to the experiments with Agisoft far less photos are needed. The output is a finished morphed rigged model ready to use with the animator in Unity® as shown in a test model from the FaceGen Modeller Demo Editor in figure 3.6.

Unfortunately Singular Inversions claims on their homepage in the FAQ section *FaceGen Frequently Asked Questions* that they have not published a paper detailing their methodology [9].

3.5.4 Character Models and Animation from Adobe Fuse

The main focus of the avatars was to capture the face of the user to represent her/him. For the models representing the body we were looking for a faster generic approach and found the solution in Adobe Fuse.

Adobe Fuse, originally developed by Mixamo[12], is a software application to create and animate 3D characters. With the help of this application we imported female and male character model variations with selected animations like sitting and talking gestures into the virtual reality conferencing prototype and now faced, in matters of character creation, the final missing part: putting it all together.

3.5.5 Combined result

Putting it all together demanded the character models heads to be replaced by the models from FaceGen and bones as well as the animations being synced and adapted.

[12]Homepage https://www.mixamo.com/

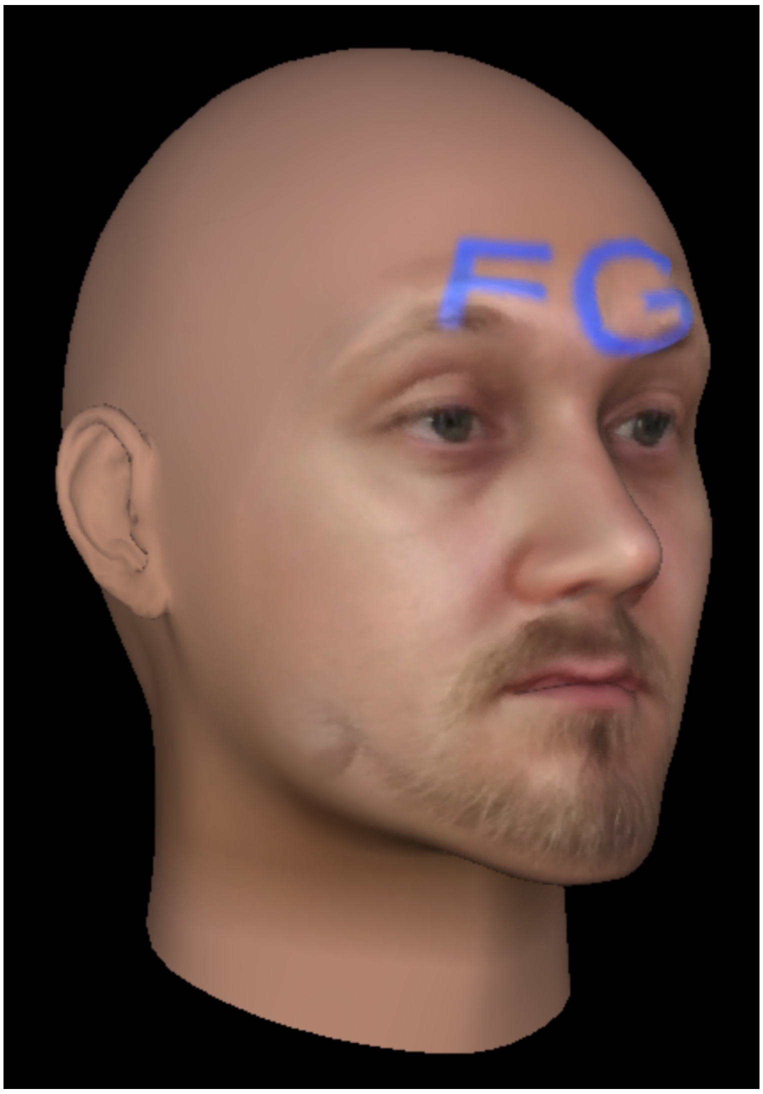
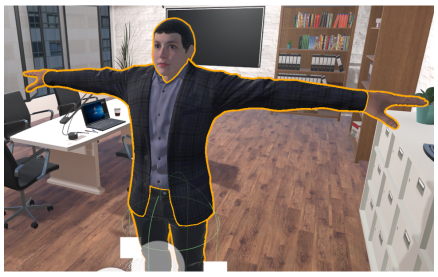


FIGURE 3.6: Screen-shot of reconstructed Face taken in the FaceGen Modeller Demo Editor

For this prototype solution we chose the brute force path and came up with the following helper class called HeadBones.cs inside the application for arranging the new head bones with the old ones:

```
public class HeadBones : MonoBehaviour {

  [SerializeField]
  public Transform[] bones = new Transform[2];
  [SerializeField]
  public Vector3 headTranslate;
  public float neckLimitStart = -1.2f;
  public float neckLimitEnd = -1f;
  public float neckZLimit = -1f;

  private SkinnedMeshRenderer rend;
  private Transform[] oldBones;
  private Matrix4x4[] oldBindPoses;
  private BoneWeight[] oldBoneWeights;

  // Use this for initialization
  void Start () {
    Mesh mesh =
      GetComponent<SkinnedMeshRenderer>().sharedMesh;
    BoneWeight[] weights =
      new BoneWeight[mesh.vertices.Length];
    float neckRange =
      neckLimitEnd - neckLimitStart;
    for (int i = 0; i < mesh.vertices.Length; i++)
    {
      float alpha = Mathf.Clamp01(
        (mesh.vertices[i].y - neckLimitStart)
        / neckRange
      );

      weights[i].boneIndex0 = 0;
      weights[i].boneIndex1 = 1;
      if (
        mesh.vertices[i].z < neckZLimit &&
        (1-alpha) > 0
        )
      {
        weights[i].weight0 = alpha;
        weights[i].weight1 = 1-alpha;
      } else
      {
        weights[i].weight0 = 1 - alpha;
        weights[i].weight1 = alpha;
      }
    }
    oldBoneWeights = mesh.boneWeights;
    mesh.boneWeights = weights;
    Matrix4x4[] bindPoses = new Matrix4x4[2];
    bindPoses[0] = bones[0].worldToLocalMatrix *
      transform.localToWorldMatrix *
      Matrix4x4.TRS(
        headTranslate,
        Quaternion.identity,
        new Vector3(
          0.099f,
          0.099f,
          0.099f
        )
      );
    bindPoses[1] = bones[1].worldToLocalMatrix *
      transform.localToWorldMatrix *
      Matrix4x4.TRS(
        headTranslate,
        Quaternion.identity,
        new Vector3(
          0.099f,
          0.099f,
          0.099f
        )
      );
    oldBindPoses = mesh.bindposes;
    mesh.bindposes = bindPoses;
    rend = GetComponent<SkinnedMeshRenderer>();
    oldBones = rend.bones;
```

The final avatars inside the application can be seen in figure 3.7a. The avatars will have their bones adjusted at the start of the application to display the animations correctly as shown in figure 3.7b.


(A) Final avatar example

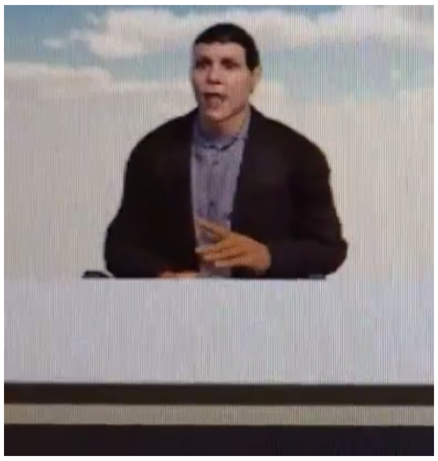

(B) Animation of final avatar example

FIGURE 3.7: Final avatar examples

3.6 Face Tracking

3.6.1 CSIRO Face Tracking

The CSIRO Face Analysis SDK [19] by M. Cox and Lucey is based on "Deformable Model Fitting by Regularized Landmark Mean-Shift" [24] by Saragih, Lucey, and Cohn and contains a collection of components to help capturing geometrical face data from video.

We have embedded the face tracker in the virtual reality conferencing application with a wrapper based on the master thesis "Capturing Performance Based Character Animation in Real Time for Unity" [37] of our colleague Hannes Wagner.

Once the face tracker is loaded the application gray-scales the camera input and sends the image every frame to the face tracker which answers with the calculated accuracy between 0, which signifies a lost tracking, and 10 for the optimum result as well as the number of tracked face feature points and their two-dimensional positions.

The first implementations in the early stadium of this masters thesis are shown in figure 3.8 with the representation on a avatar place holder and figure 3.9 with the live web cam picture included for reference.

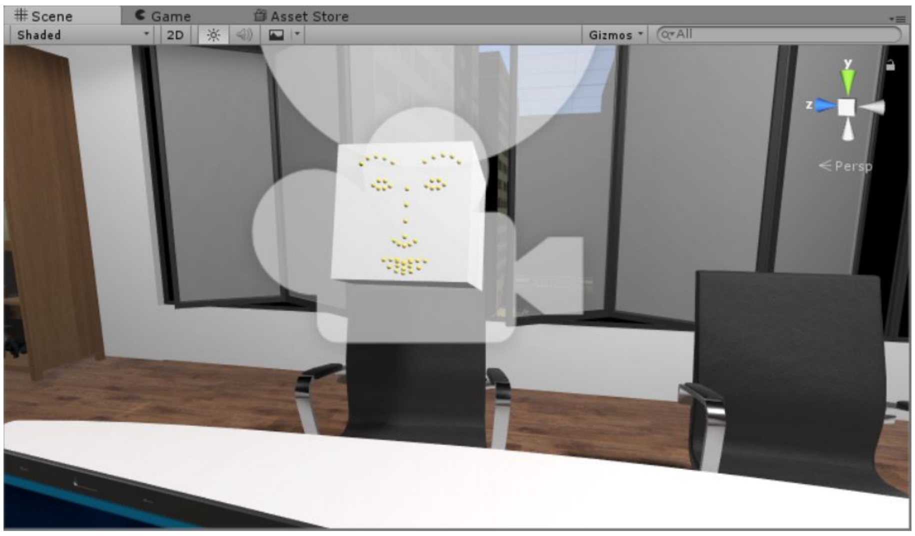


FIGURE 3.8: Screen-shot of first facial tracking implementation visualized on avatar placeholders taken in the Unity® Editor

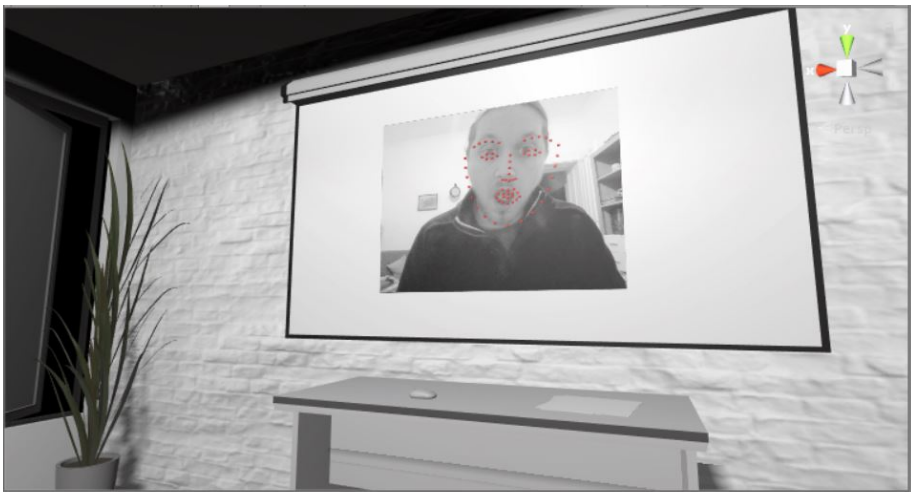


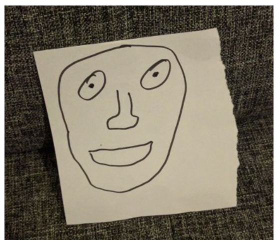
FIGURE 3.9: Screen-shot of first facial tracking implementation web cam reference taken in the Unity® Editor

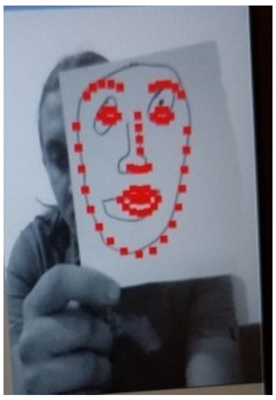
At the time the face tracking was implemented in the virtual reality conferencing application it was unclear if the absolute distance between the two dimensional data of the tracked points of the mouth will be used to shift the bones in the avatar directly or if a heuristic approach will be used to generate and blend between predicted expressions every few tracked frames.

Despite the quick success in getting relatively high accuracy two dimensional face feature positions from the used web cams two huge limitations arose:

The first limitation was that the user had to be be rather static in front of her/his screen for the tracker to perform well. The second much severe limitation was that even with some alterations and experimentation the tracker could not be altered easily to detect mouth features when the user was wearing a virtual reality headset.

Figures 3.10 and 3.11 illustrate the process to deal with the problem of mouth detection while wearing a virtual reality headset.


(A) First recognized face demo


(B) First recognized face demo test

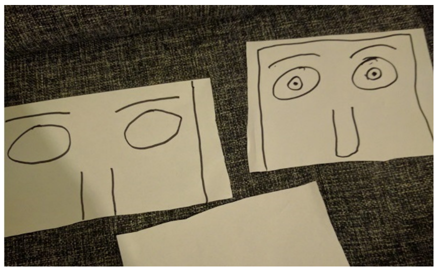
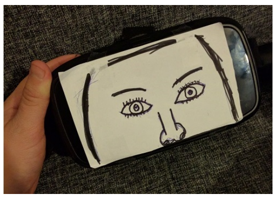
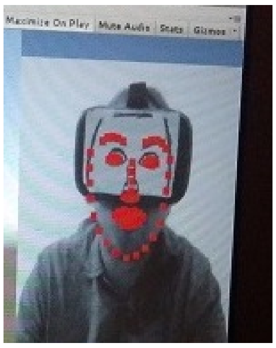

(C) Face demos for VR headset which were not recognized

FIGURE 3.10: Face tracker experiments for the VR headset

Despite some successful testing the accuracy of the detection turned out to be too low in order to work probably. Therefor new approaches were tested in order to track the mouth of the user while wearing a virtual reality headset without the need of additional hardware or headset mounted cameras targeted towards the mouth. One of the first ideas to experiment with was the ARToolkit, described in the following section.


(A) Creation of recognized VR headset face



(B) Recognized VR headset face

FIGURE 3.11: Detected face tracker experiments for the VR headset

3.6.2 ARToolkit Headset Tracking

ARTookit, originally developed by Hirokazu Kato is a software library for the development of augmented reality applications. It has been used for many academic works including some with adjacent topics to this thesis like:

- “Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System” [15] by Kato and Billinghurst
- “The Effect of Spatial Cues in Augmented Reality Video Conferencing” [14] by Kato and Tachibana
- “Collaborative Augmented Reality” [1] by Billinghurst and Kato

After adapting the ARToolkit plugin for Unity® to the system and networking architecture of the virtual reality conferencing application we experimented with markers placed on the headset as shown in figure 3.12.

Fortunately through previous work with augmented reality libraries in our bachelors thesis “Head Tracking with AR Toolkit” [23] the adaptations in Unity® did not take too much extra effort.

In the virtual scene stock images of a face are projected on the forehead of the user, giving him static eyes and a static nose while not obfuscating his mouth. This virtual output is than written in a render texture and post-processed to brighten up the mouth area for better tracking and negation of a possible dropped shadow by the virtual reality headset. This created image was then sent to the face tracker instead of the directly gray scaled output of the web cam.

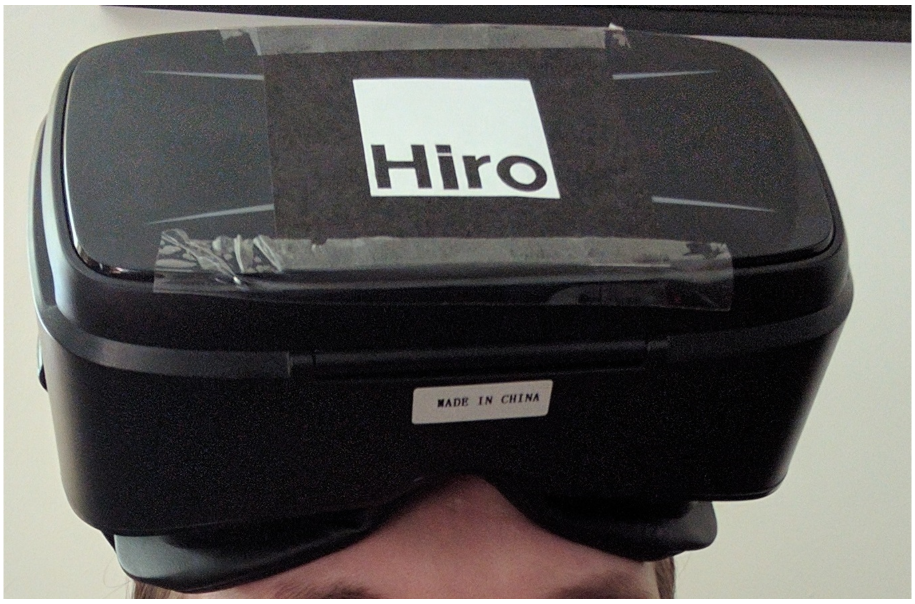


FIGURE 3.12: Photo of the test AR marker "Hiro" for VR headset tracking

Results varied very strong with the main factors being (i) positioning of the user and (ii) lighting conditions of the room.

3.6.3 Face Tracking Lessons Learned

Facial animation tracking is implemented, however currently optional. To fully support or enhance its capabilities a separate standalone thesis would be necessary. The default implementation of mouth animation inside the virtual reality conferencing application finally settled to go for an audio approach, which will be described in more detail in the subsections "Voice Chat" 3.7.1 and "Animation" 3.7.2.

3.7 Software Architecture

Figure 3.13 summarizes script lifetimes in Unity® for a better understanding of execution orders in Unity®. However for details about how Unity® basically works the reader is referred to the official manual [34].

The main parts of the VR conferencing application are its menu scene at the start of the application and the conference room scene. For login management a simple associative array with the user-name being the key and the value consisting of the password is used to load the according user avatar at the beginning.

After the login the user can upload a 3D object and upload a Power Point presentation before either hosting a conferencing room or joining one by IP-address in the current local network environment.

Depending on whether the user has a virtual reality headset device connected, the application renders to the headset and enables motion controls or falls back to the desktop mode with mouse control for interactions and rendering of the scene on the computers screen.

The virtual reality headset and motion controller interaction features are implemented with the use of the Steam VR library for Unity®, which is needed to run virtual reality applications in Unity® with OpenVR virtual reality headset devices like the HTC VIVE.

To enhance immersion and the feeling of presence inside the application a few small feature scripts were programmed including a way-point system for driving cars around the buildings outside the conferencing room and a wall-clock displaying real time.

3.7.1 Voice Chat

The voice chat was implemented upon the work of Holmström and his library described as *A simple VOIP solution for integration into unity* [8] found on GitHub[13]. Although the library had some errors in its communication flow between the audio network manager and the audio stream packages, these problems could be fixed with some minor adjustments and be adapted to the networking flow of the application.

Further improvements/adaptations include the audio proxy sources for each user inside the application to feature 3D locatable audio sources at the position of the user inside the virtual reality conferencing room.

[13]URL https://github.com/fholm/unityassets/tree/old/VoiceChat

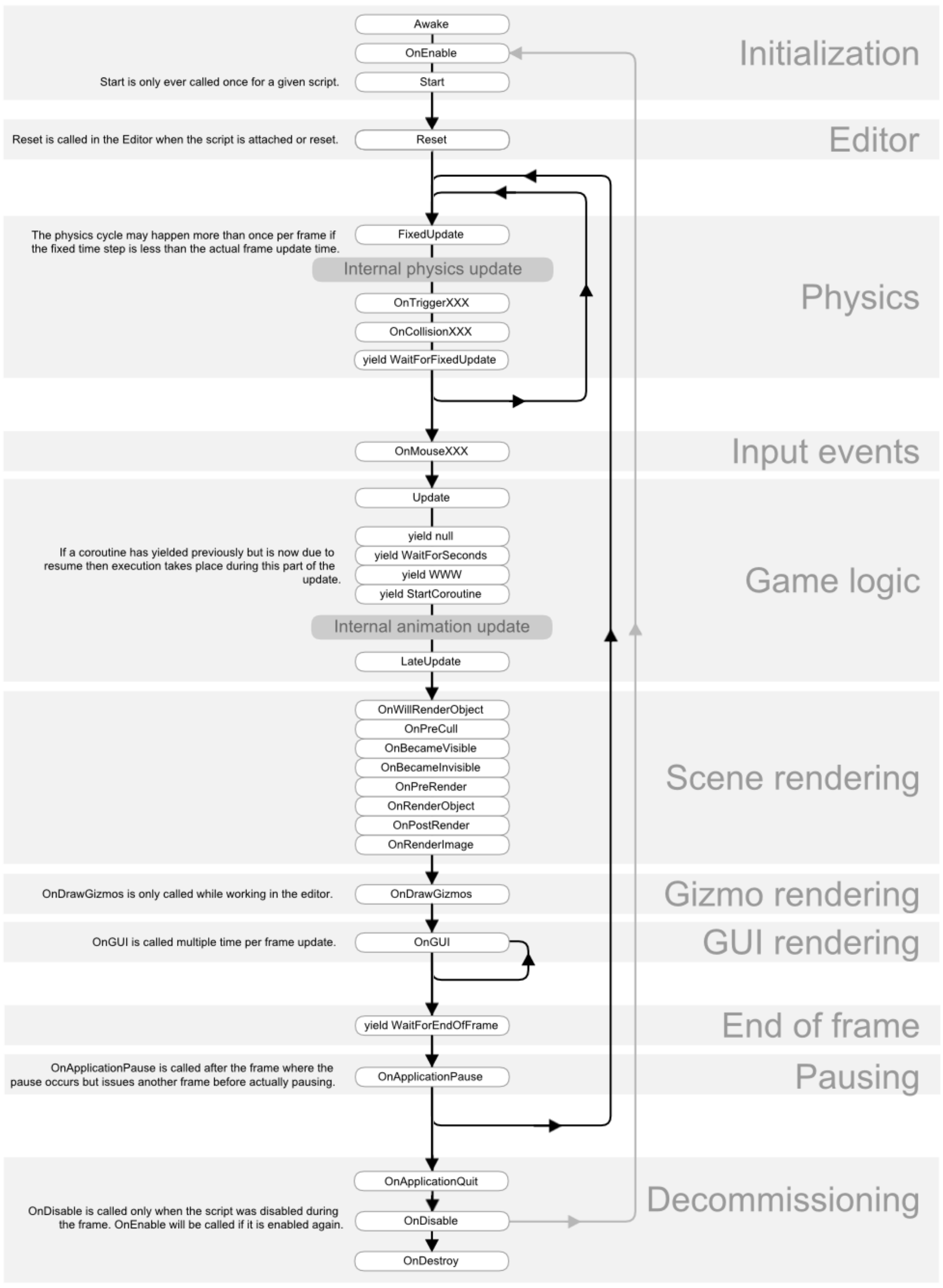

Initialization
Awake
OnEnable
Start is only ever called once for a given script.
Start
Editor
Reset is called in the Editor when the script is attached or reset.
Reset
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.
FixedUpdate
Internal physics update
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
LateUpdate
Scene rendering
OnWillRenderObject
OnPreCull
OnBecameVisible
OnBecameInvisible
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Gizmo rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos
GUI rendering
OnGUI is called multiple time per frame update.
OnGUI
End of frame
yield WaitForEndOfFrame
Pausing
OnApplicationPause is called after the frame where the pause occurs but issues another frame before actually pausing.
OnApplicationPause
Decommissioning
OnApplicationQuit
OnDisable is called only when the script was disabled during the frame. OnEnable will be called if it is enabled again.
OnDisable
OnDestroy

FIGURE 3.13: Ordering and repetition of event functions during a script's lifetime taken from Unity®'s online manual for version 5.6 [34]

3.7.2 Animation

The animation of the characters in the final prototype consists of:

- tilting the head and eye movements controlled by the viewing direction of the user's avatar,
- mouth animations supported by arm articulation gestures triggered by audio input of the users microphone and
- simulated blinking of the avatar's eyes.

Head and eye movement

The animation of the eyes and head movement of the avatar is calculated based on the offset of the camera looking at the user in comparison to the position of his avatars body. Everything below a certain threshold in front of the avatar moves solely the eyes. When the threshold is exceeded the head is rotated additionally. The animation can be interpolated in two dimensions (x and y).

Mouth animation

The mouth animations went through many experimentation cycles. From weighted moving certain muscles of the mouth and face based on visually tracked data to a concept of training a neural network to identify phonemes in the audio stream from the microphone and interpolate the blend shapes one after the other. The working implementation for testing in the questionnaire experiment was reduced to take the push to talk audio stream from the user and interpolate different randomly preselected blend shapes to animate a speaking mouth. This together with a fixed gesticulating arms and body animation turned out to be more convincing than an approach based on directly mimicking the tracked data of the user.

Eye blinking

To make the avatars seem more lively a script was written to make them blink between every two to six seconds based on J. Volpe's observations in "Adler's Physiology of the Eye: Clinical Application, Tenth Edition" [11].

The complete animation script files can be found in the appendix section of this thesis at Appendix A in the section Animation.

3.7.3 Interaction

The main interactions are limited to the host of the virtual reality conferencing session and consist of interacting with the uploaded presentation and 3D mesh object. Both objects can be painted and marked in real time to emphasize parts of them as shown in figures 3.14 and 3.15.

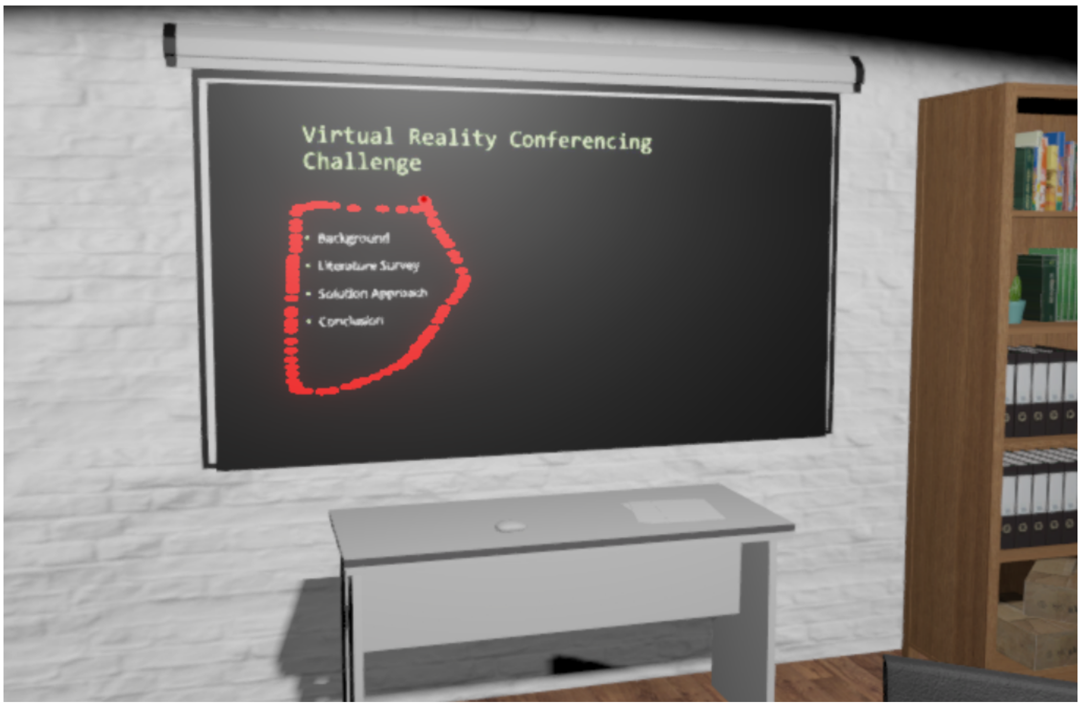


FIGURE 3.14: Presentation feature with painted markings in the VR conferencing application.

VR painting

To enable the painting on the 3D model and the presentation both were assigned a render texture for which an additional camera was placed below the gaming scene targeted at original texture of the objects. With every paint brush a ray-cast was made from the current viewpoint or motion controller to the targeted space on the object and where it hit the object the according uv position in the texture was calculated.

With this knowledge a 2D sprite was instantiated at this exact position in front of the additional rendering camera to paint atop the objects texture at this position.

The full source code of this script is included in Appendix A in the section VR-Painting.

Mesh importer

Another milestone in the development of this virtual reality conferencing application prototype was the dynamic importing of 3D meshes. Because of the voluntary use of syntax in wavefront files the loader works only with meshes with planar polygons and is likely to fail on most models found on the Internet.

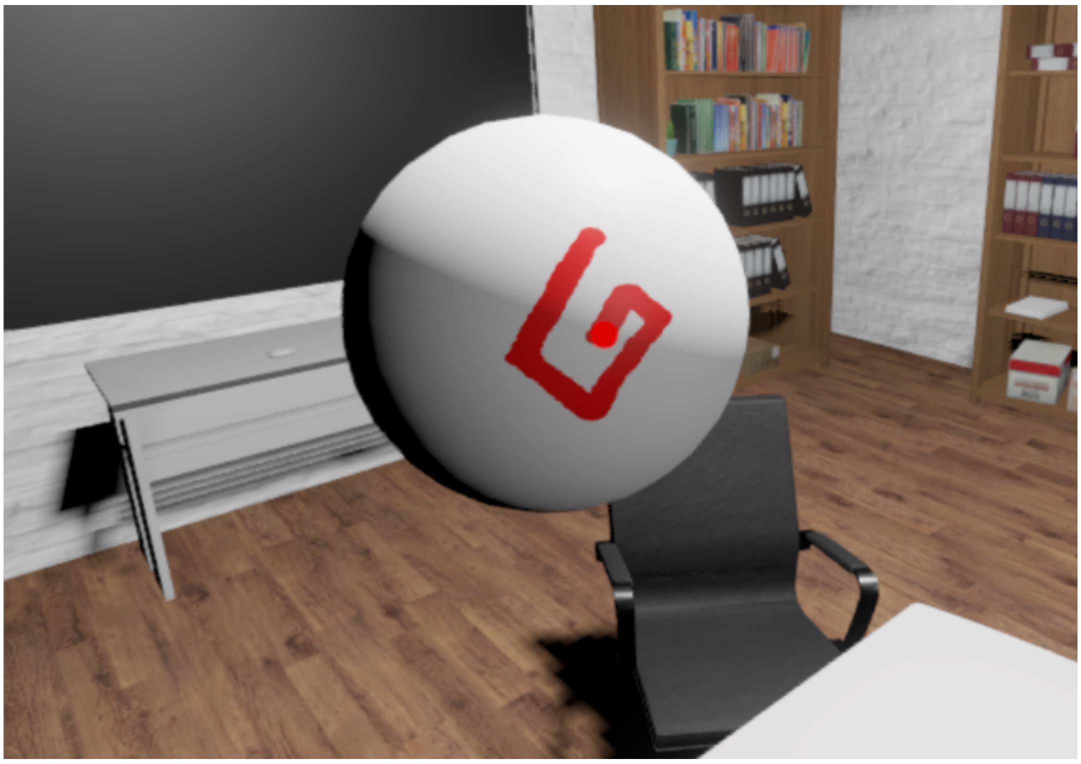


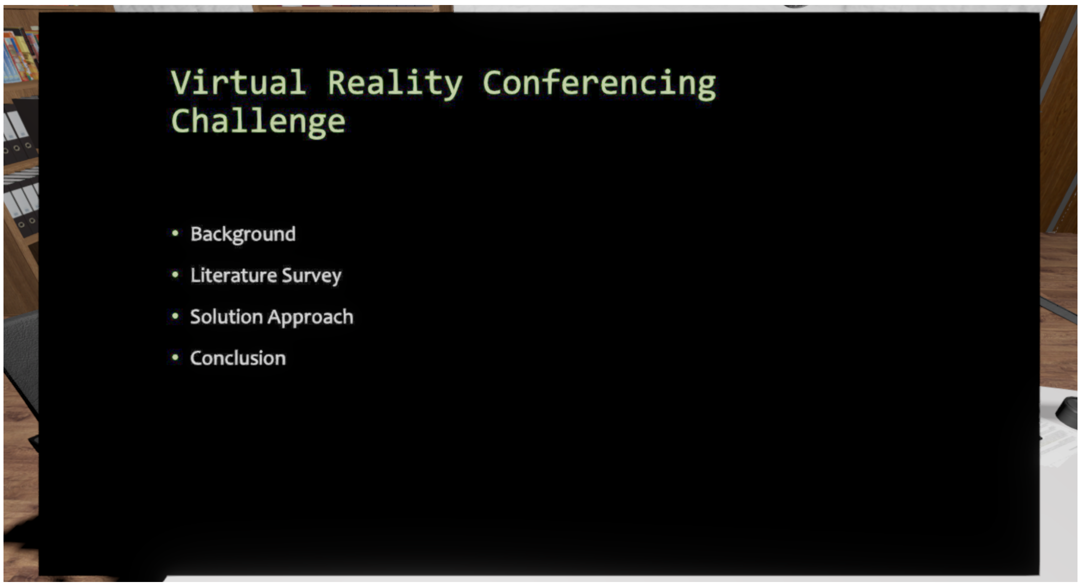
FIGURE 3.15: Manipulation and painting on 3D mesh object inside the VR conferencing room.

Presentation

The presentation to be used in the virtual reality conferencing room can be chosen in the menu via a file selector locally on the computer. The detailed process of the upload and server conversion of the presentation file is described in the following section.

The implementation of the slide-show like presentation feature was developed with a lightweight manager class which knows the URL and salt of the uploaded and converted slides and keeps track of the current one to be shown and how many there are to display. Figure 3.16 shows the presentation close up view inside the virtual reality conferencing application prototype.

To enable marking and painting on the slides for the host, a similar approach as for the 3D object meshes was used. While the presentation manger renders the downloaded slide images into a 2D plane this approach places an additional render camera pointed at this plane with the ability to place brush 2D sprites between. Afterward it renders the output into a render texture which is used on the presentation screen inside the virtual reality conferencing room.



Virtual Reality Conferencing Challenge

- Background
- Literature Survey
- Solution Approach
- Conclusion

FIGURE 3.16: Presentation close up feature in the VR conferencing application.

More information on the synchronization of the loaded mesh and presentation will be described in the following section.

3.8 Network Architecture

The network architecture is built with the Unity® Networking Manager implemented using their High-level API shown in figure 3.17, which works as a wrapper for game state management, spawning management, scene management, debugging information, matchmaking and customization. [34]

This networking manager was adapted and enhanced with custom calls on several connection and player ready states needed for the animation and interaction system of the virtual reality conferencing prototype and finally merged with the voice chat functionality and player based audio stream transmission.

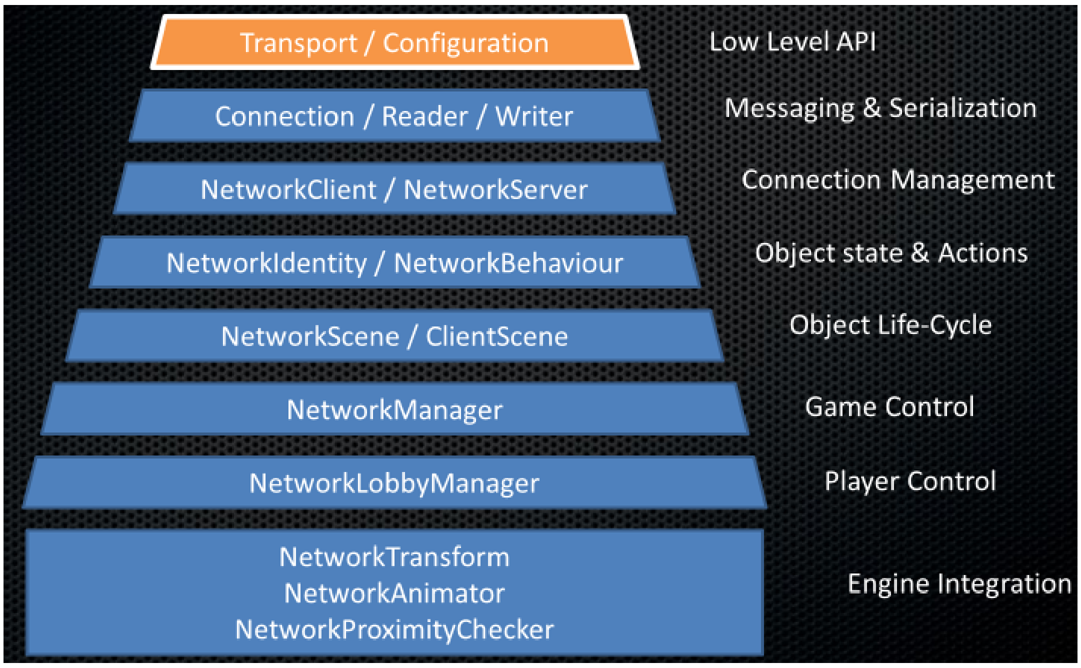


FIGURE 3.17: Screen-shot of the layers of Unity®'s networking High-level API taken from Unity®'s manual [34]

As we stated at the beginning of this chapter describing the controls of the virtual reality conferencing prototype we differentiate between hosts which start the conferencing room and optionally upload a presentation or 3D object, and users who can join a hosts conferencing room. In Unity®'s networking system a host takes over the role of the server when there is no dedicated server as shown in figure 3.18.

The avatars of the users are player objects inside of Unity® which have their own authority over changes on their game-object on the server. As shown

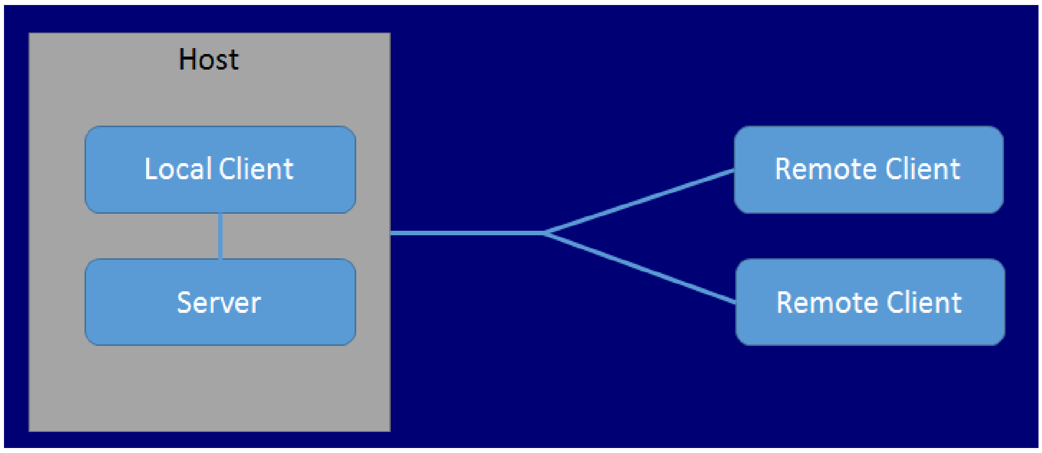


FIGURE 3.18: Screen-shot of Unity®'s networking host client illustration taken from Unity®'s manual [34]

in figure 3.19 these objects exist locally as well as on the server and can be modified independently.

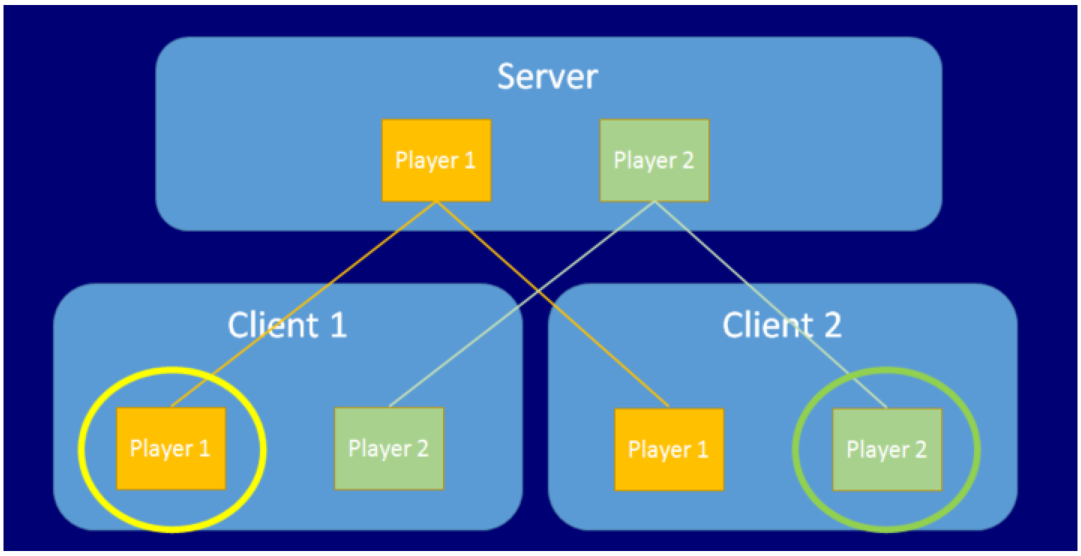


FIGURE 3.19: Screen-shot of Unity®'s networking player authority illustration taken from Unity®'s manual [34]

Network interactions

The menu scene offers options for uploading a power point presentation and a 3D wavefront mesh into the virtual reality conferencing room. The presentation is uploaded to a conversion server which converts the presentation to a PDF and splits the slides to image files. The address of these image files is known to the host and distributed to users on joining. Each user has an instance of the local presentation and the network presentation manager to notify them on slide changes. Updates to the 3D object are announced similarly to the users on change by the host.

The painting network manager receives calls when the host initiates a brush stroke to instantiate a sprite locally inside their own painting system to manage the render texture locally without the need to sync the whole texture on every change.

When a user presses a key to spawn an emote, a call is sent to the server to instantiate the object and distribute a local version the each of the connected clients with a preset time to live.

Voice chat and animation

When a user joins the virtual reality conferencing room a proxy voice chat object is instantiated and spawned directly at the users position at each connected client. On audio input via push to talk the user sends his audio stream packages to the server which distributes them to the registered local proxies for each user. The mouth and gesture animation will then be handled locally by each client on successfully delivered audio stream packages.

3.8.1 Server Plug-ins

For this prototype a simple web server was set up to host two main scripts:

- store a 3D mesh with the current unique session id
- store and convert a presentation with the current unique session id

For the conversion of the presentation *unoconv* [38][14] by Wieers was used inside a simple PHP form script to convert the presentation to a PDF and than split the presentation slides into separate images.

The image conversion of the mid-step PDF is done with the script *convert* [17][15] by LLC.

The source files of the two PHP server scripts can be found in Appendix B.

[14]unix manpage https://linux.die.net/man/1/unoconv
[15]unix manpage https://linux.die.net/man/1/convert

Chapter 4

Experiment Evaluation

4.1 igroup Presence Questionnaire

The Presence Questionnaire project started in 1997 with the goal to create a compact scale for measuring presence in evaluation studies in virtual environments and is according to their website [4] still an ongoing project. The igroup presence questionnaire scales are built on two phases of a Internet survey consisting of a large pool of question items based on earlier questionnaires and a theoretical model of their presence experiences. The scale consists of a 14-item scale measuring three sub-scales and one additional general item being divided into the following categories [4]:

- Spatial presence – the sense of being physically present in the virtual environment (VE)
- Involvement – measuring the attention devoted to the VE and the involvement experienced
- Sense of realness – measuring the subjective experience of realism in the VE

The additional general item has a high loading on all three factors and stands for the "sense of being there" inside the virtual environment.

The first presence scale consisting of the first three components was developed using structural equation modeling based on the first phase of the survey shown in figure 4.1.

While these first three scales identified by the factor analysis [27] were developed to be orthogonal and independent to each other, the resulting model received the final general item to load together with the sub-scale on the second-order factor general presence [4].

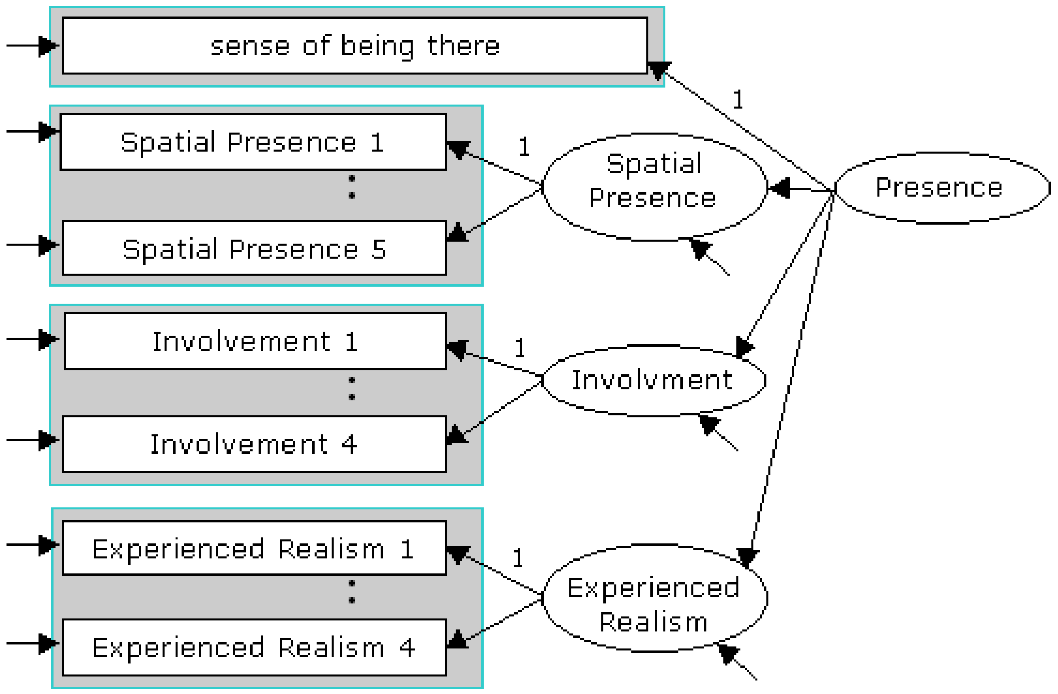


FIGURE 4.1: Illustration of the SEM model taken from http://www.igroup.org/pq/ipq/structure.gif

The Presence Questionnaire project distinguishes between immersion, which they see as an objectively described variable and presence, which is a subjectively variable based on the users experience. Therefor the questionnaire relies on self-reports to measure the users sense of presence. The figure 4.2 shows their illustrated flow of immersion, conception and experience of presence.

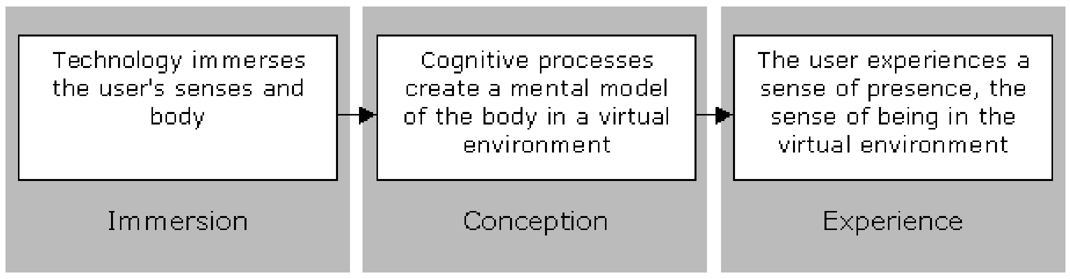

Technology immerses the user's senses and body
Cognitive processes create a mental model of the body in a virtual environment
The user experiences a sense of presence, the sense of being in the virtual environment
Immersion
Conception
Experience

FIGURE 4.2: Flow of immersion, conception and experience figure taken from http://www.igroup.org/pq/ipq/immersion.gif

Further information and references to the development of the igroup presence questionnaire can be found in:

- “The Experience of Presence: Factor Analytic Insights” [27] by Schubert, Friedmann, and Regenbrecht
- “Wer hat Angst vor virtueller Realität? Angst, Therapie und Präsenz in virtuellen Welten” [28] by Schubert and Regenbrecht
- “Embodied Presence in Virtual Environments” [26] by Schubert, Friedmann, and Regenbrecht
- “Real and Illusory Interaction Enhance Presence in Virtual Environments” [29] by Schubert et al.
- “The sense of presence in virtual environments: A three-component scale measuring spatial presence, involvement, and realness” [25] by Schubert

4.1.1 Questionnaire

The igroup presence questionnaire was originally only tested in its German version and the igroup.org – project consortium warns of the possibility that some terms in English might be lost in translation. Because of that the questionnaire was conducted with Austrian native speakers in German. The table 4.1 shows a list of the German ipq items taken from the igroup homepage [3].

Nr.	IPQ item name	German question	German anchors
1	G1	In der computererzeugten Welt hatte ich den Eindruck, dort gewesen zu sein...	überhaupt nicht – sehr stark
2	SP1	Ich hatte das Gefühl, dass die virtuelle Umgebung hinter mir weitergeht.	trifft gar nicht zu – trifft völlig zu
3	SP2	Ich hatte das Gefühl, nur Bilder zu sehen.	trifft gar nicht zu – trifft völlig zu
4	SP3	Ich hatte nicht das Gefühl, in dem virtuellen Raum zu sein.	hatte nicht das Gefühl – hatte das Gefühl

5	SP4	Ich hatte das Gefühl, in dem virtuellen Raum zu handeln statt etwas von außen zu bedienen.	trifft gar nicht zu – trifft völlig zu
6	SP5	Ich fühlte mich im virtuellen Raum anwesend.	trifft gar nicht zu – trifft völlig zu
7	INV1	Wie bewußt war Ihnen die reale Welt, während Sie sich durch die virtuelle Welt bewegten (z.B. Geräusche, Raumtemperatur, andere Personen etc.)?	extrem bewußt – mittelmäßig bewußt – unbewußt
8	INV2	Meine reale Umgebung war mir nicht mehr bewusst.	trifft gar nicht zu – trifft völlig zu
9	INV3	Ich achtete noch auf die reale Umgebung.	trifft gar nicht zu – trifft völlig zu
10	INV4	Meine Aufmerksamkeit war von der virtuellen Welt völlig in Bann gezogen.	trifft gar nicht zu – trifft völlig zu
11	REAL1	Wie real erschien Ihnen die virtuelle Umgebung?	vollkommen real – weder noch – gar nicht real
12	REAL2	Wie sehr glich Ihr Erleben der virtuellen Umgebung dem Erleben einer realen Umgebung?	überhaupt nicht – etwas – vollständig
13	REAL3	Wie real erschien Ihnen die virtuelle Welt?	wie eine vorgestellte Welt – nicht zu unterscheiden von der realen Welt
14	REAL4	Die virtuelle Welt erschien mir wirklicher als die reale Welt.	trifft gar nicht zu – trifft völlig zu

TABLE 4.1: igroup questionnaire items in German taken from http://www.igroup.org/pq/ipq/items.php

For an English reference we have added the table 4.2 with the English translations published on the homepage of the igroup.org – project consortium including the respective copyright holders of items which are taken from previously published scales in:

- "Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments" [32] by Slater and Usoh
- "Measuring Presence in Virtual Environments: A Presence Questionnaire" [6] by G. Witmer and J. Singer
- "Exploratory Studies on the Sense of Presence in Virtual Environments as a Function of Visual and Auditory Display Parameters" [20] by Mary Hendrix
- "Virtual reality and tactile augmentation in the treatment of spider phobia: A case report" [2] by Carlin, G. Hoffman, and Weghorst

Number	English question	English Anchors	Copyright (item source)
1	In the computer generated world I had a sense of "being there"	not at all–very much	Slater & Usoh (1994) [32]
2	Somehow I felt that the virtual world surrounded me.	fully disagree–fully agree	IPQ[1]
3	I felt like I was just perceiving pictures.	fully disagree–fully agree	IPQ
4	I did not feel present in the virtual space.	did not feel–felt present	???[2]
5	I had a sense of acting in the virtual space, rather than operating something from outside.	fully disagree–fully agree	IPQ
6	I felt present in the virtual space.	fully disagree–fully agree	IPQ

[1]Developed for the IPQ by Thomas Schubert, Holger Regenbrecht, and Frank Friedmann (for questions 2, 3, 5, 6, 8, 9, 10 and 14)

[2]no reference given on published scale on ipq homepage http://www.igroup.org/pq/ipq/download.php

7	How aware were you of the real world surrounding while navigating in the virtual world? (i.e. sounds, room temperature, other people, etc.)?	extremely aware-moderately aware-not aware at all	Witmer & Singer (1994) [6]
8	I was not aware of my real environment.	fully disagree–fully agree	IPQ
9	I still paid attention to the real environment.	fully disagree–fully agree	IPQ
10	I was completely captivated by the virtual world.	fully disagree–fully agree	IPQ
11	How real did the virtual world seem to you?	completely real–not real at all	Hendrix (1994) [20]
12	How much did your experience in the virtual environment seem consistent with your real world experience?	not consistent-moderately consistent-very consistent	Witmer & Singer (1994) [6]
13	How real did the virtual world seem to you?	about as real as an imagined world–indistinguishable from the real world	Carlin, Hoffman, & Weghorst (1997) [2]
14	The virtual world seemed more realistic than the real world.	fully disagree–fully agree	IPQ

TABLE 4.2: igroup questionnaire items in English taken from http://www.igroup.org/pq/ipq/items.php

4.2 Experiment Setup

The experiment was conducted with a group of ten people divided into two equal groups of five. The first group would be using the virtual reality conferencing application with a htc VIVE headset and the second group without a VR headset.

Each person was introduced to the experiment and controls outside the application by a supervisor beforehand for whom a 3D avatar with his reconstructed face was prepared in advance (shown in figure 4.3).

Inside the virtual reality conferencing application the test user was able to communicate remotely with the supervisor who joined the session in another room and his avatar.

The supervisor held a short presentation about the virtual reality conferencing application accompanied with slides shown in the virtual environment and marked specific spots on an simple 3D model to simulate a short business meeting.

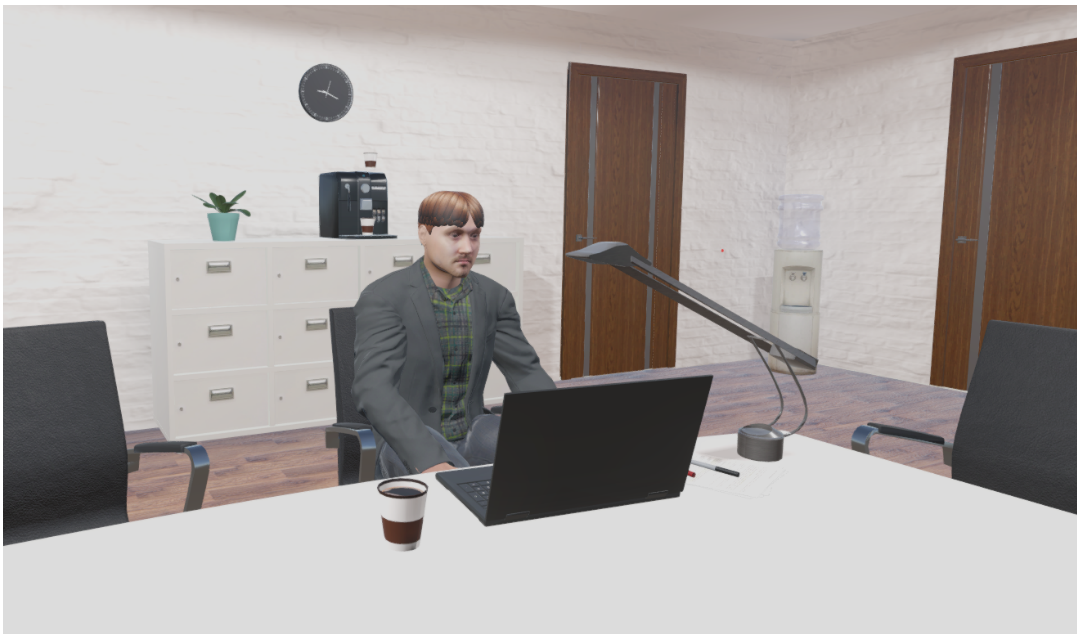


FIGURE 4.3: Virtual reality conferencing supervisor avatar

After the presentation the users were asked to quit the virtual reality conferencing application and to honestly answer the 14 igroup presence questionnaire questions.

4.3 Survey Results

The survey results show a significant increase of presence on all measured scales when wearing a virtual reality headset while using the virtual reality

IPQ items	mean with headset	mean without headset
General	4.80	3.20
Spatial presence	5.32	3.96
Experienced realism	3.50	2.75
Involvement	4.80	3.15

TABLE 4.3: Presence profile results for the test groups with and without a headset during the experiment

conferencing prototype application. Table 4.3 illustrates the calculated means of the general item, spatial presence, experienced realism and involvement for the questioned group using virtual reality headsets and the group without headsets.

All scales of the questionnaire range from zero being the lowest to six being the highest score. To visualize the results we plotted a diagram shown in figure 4.4 with three axis for spatial presence, experienced realism and involvement plus a bow for the general item all ranging from zero to seven.

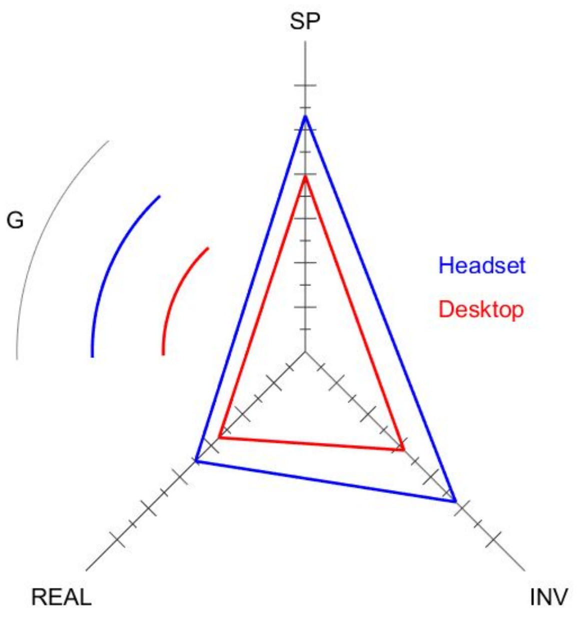


FIGURE 4.4: Presence profile results with G for general, SP for spatial presence, REAL for experienced realism and INV for involvement for the test groups with (Headset) and without a headset (Desktop) during the experiment.

ANOVA tests (with a chosen alpha of 0.05) of the means of each scale illustrate significant differences between the two groups with p-values far below 0.05 except for the experienced realism with a p-value of 0.082 where we do not have enough evidence to reject the null hypothesis that the population means are all equal.

Chapter 5

Summary and future Work

5.1 Summary

The development of the virtual reality conferencing application prototype was a very time consuming task with lots of experimentation and trials of different solution approaches. The most challenging part was to change the perspective of the current approach and find new technologies when stuck on a problem with a certain feature. The main drive, however, was influenced by the promising literature survey beforehand and the possibility to work with state of the art technology and modern devices.

The prototype developed and introduced in this thesis proves to be a very stable foundation for additional feature implementations to further increase the sense of collaborating inside a virtual reality environment.

The results of the conducted experiments questionnaire indicate a measurable advantage regarding the immersion of users using the latest technology in virtual reality headsets.

With the gathered data we believe that further computer scientific research on the feeling of immersion inside virtual reality environments using the current state of the art technology with additional technological advancements in the next few years will have the potential to close the final gap of feeling completely inside a virtual reality application.

5.2 Outlook and Possible Improvements

As stated above, many promising features led to dead ends and would have needed much more time consuming focus and considerations to overcome.

However, we hope that the promising results of the current state invite other researchers to pick up the work where we left and focus on the aspect of real time face-tracking and animation of the user avatars.

Interesting approaches for this matter could be to work with a neural network to classify the used phonemes in the audio stream to animate them with interpolated blend shapes on the mouth of the avatar, other approaches include the implementation of an improved face tracker with robust lightening and positioning which is trained and dependent only on the mouth of the user to animate separate bones of the mouth model directly with spatial influence on other areas of the face.

To improve future presence tests it would be beneficial to further separate the tested groups into users with prior virtual reality experience and users without. Having four groups compared to each other could reduce the possible biased impact of first time virtual reality device users and could illustrate interesting facts if there is a bias at the first place and if so how strong.

Zusammenfassung

Patrick David PAZOUR, BSc.

Virtual Reality Conferencing

Die Entwicklung und Verfügbarkeit immersiver Virtual-Reality-Geräte ist in den letzten Jahren deutlich gestiegen. Kombiniert mit den neuesten Meilensteinen in der Computergrafik und den aktuellen Motion-Tracking-Geräten lässt sich ein gewisses Gefühl von Präsenz in der virtuellen Welt erreichen, das auf lange Sicht von der realen Welt nicht mehr zu unterscheiden ist. Obwohl dieses Streben heute visionär sein mag, sind die Möglichkeiten kollaborativer menschlicher Interaktionen innerhalb eines Virtual-Reality-Systems bereits vielfältig.

Diese Masterarbeit konzentriert sich auf die Entwicklung und Evaluierung eines Prototyps für Virtual-Reality-Konferenzanwendungen, der personalisierte benutzerbasierte Avatare für bis zu vier Personen unterstützt, die sich remote in einem virtuellen Konferenzraum treffen können. Der Prototyp wurde im Hinblick auf das Gefühl der Präsenz in der virtuellen Umgebung der Anwendungen bewertet. Für die Bewertung waren zwei Gruppen von je fünf Personen einzeln an einer Präsentation in einem virtuellen Konferenzraum beteiligt, der von einem photogrammetrischen 3D-Avatar ihres Versuchsleiters gehostet wurde. Die erste Gruppe war mit Virtual-Reality Headsets ausgestattet, während die zweite Gruppe die Präsentation vor einem Computer-Monitor ohne Headset absolvierte. Anschließend wurden die beiden Gruppen gebeten, ein Formular auf der Grundlage der *igroup presence Fragebogenskala* auszufüllen, um die räumliche Präsenz, Beteiligung und den Realitätssinn im virtuellen Konferenzraum zu messen.

Die Ergebnisse der durchgeführten Experimente zeigen eine positive Auswirkung auf das Gefühl der "Anwesenheit" von Benutzern, die Virtual-Reality Headsets tragen, verglichen mit der Gruppe ohne Headsets.

Appendix A

VR Application Code Appendix

A.1 Animation

A.1.1 Eyes

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EyeBlink : MonoBehaviour {

  SkinnedMeshRenderer skinnedMeshRenderer;
  bool eyesClosed = false;
  bool blinkFinished = false;
  float blend = 0f;
  float blendSpeed = 1f;
  public bool showBlendShapes = false;
  public float blendStrength = 10f;
  public float blinkPause = 0.01f;
  public int leftBlink = 108;
  public int rightBlink = 109;

  void Awake()
  {
    skinnedMeshRenderer =
      GetComponent<SkinnedMeshRenderer>();
  }

  void Start()
  {
    StartCoroutine(blinkLoop());
  }

  IEnumerator blinkLoop()
  {
    for (;;)
    {
      while (!blinkFinished)
      {
        blink();
```

```
            yield return new WaitForSeconds(blinkPause);
          }
          blinkFinished = false;
          yield return new WaitForSeconds(
            Random.Range(
                2f,
                6f
              )
          );
          }
    }

    void blink()
    {
      if (!eyesClosed)
      {
        if (blend < 10f)
        {
          blend += blendSpeed;
          if (blend > 10f)
          {
            blend = 10f;
          }
          skinnedMeshRenderer.SetBlendShapeWeight(
              leftBlink,
              blend
          );
          skinnedMeshRenderer.SetBlendShapeWeight(
              rightBlink,
              blend
          );
        }
        else
        {
          eyesClosed = true;
        }
      }
      else
      {
        if (blend > 0f)
        {
          blend -= blendSpeed;
          if (blend < 0f)
          {
            blend = 0f;
          }
          skinnedMeshRenderer.SetBlendShapeWeight(
              leftBlink,
              blend
          );
          skinnedMeshRenderer.SetBlendShapeWeight(
              rightBlink,
              blend
          );
        }
        else
        {
         eyesClosed = false;
         blinkFinished = true;
       }
     }
   }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EyesFollow : MonoBehaviour {

 SkinnedMeshRenderer skinnedMeshRenderer;
 GameObject maincamera;

 public float thresh = 0.01f;
 public float blendSpeed = 100f;
 public float blendStrength = 20f;
 public int turnLeft = 43;
 public int turnRight = 44;
 public int eyesLeft = 112;
 public int eyesRight = 113;
 public int eyesup = 114;
 public int eyesdown = 115;
 public int headup = 45;
 public int headdown = 46;

 void Awake()
 {
   skinnedMeshRenderer =
     GetComponent<SkinnedMeshRenderer>();
 }

 private void Start()
 {
   maincamera =
     GameObject.FindGameObjectWithTag("MainCamera");
 }

 void Update()
 {
   //left right
   float diffX = transform.position.x -
    maincamera.transform.position.x;
   if (diffX < -thresh)
   {
    followLeft(Mathf.Abs(diffX));
   } else
   {
    skinnedMeshRenderer.SetBlendShapeWeight(
      turnLeft,
      0
    );
    skinnedMeshRenderer.SetBlendShapeWeight(
      eyesLeft,
      0
      );
    }
    if (diffX > thresh)
    {
     followRight(diffX);
    } else
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        turnRight,
        0
     );
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesRight,
        0
     );
    }
    //up down
    float diffY = transform.position.y −
     maincamera.transform.position.y;
    if (diffY < −thresh)
    {
     followUp(Mathf.Abs(diffY));
    }
    else
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        headup,
        0
     );
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesup, 0);
    }
    if (diffX > thresh)
    {
     followDown(diffY);
    }
    else
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        headdown,
        0
     );
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesdown,
        0
     );
    }
  }

  void followLeft(float diff)
  {
   float movement = diff ∗ blendSpeed;
   skinnedMeshRenderer.SetBlendShapeWeight(
     turnLeft,
     movement
   );
   if (
      skinnedMeshRenderer.GetBlendShapeWeight(turnLeft) >
      blendStrength
    )
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesLeft,
        movement - blendStrength
     );
    } else
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesLeft,
        0
     );
    }
  }

  void followRight(float diff)
  {
    float movement = diff * blendSpeed;
    skinnedMeshRenderer.SetBlendShapeWeight(
      turnRight,
      movement
    );
    if (
      skinnedMeshRenderer.GetBlendShapeWeight(turnRight) >
      blendStrength
    )
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesRight,
        movement - blendStrength
     );
    }
    else
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesRight,
        0
     );
    }
  }

  void followUp(float diff)
  {
    float movement = diff * blendSpeed;
    skinnedMeshRenderer.SetBlendShapeWeight(
      headup,
      movement
    );
    if (
      skinnedMeshRenderer.GetBlendShapeWeight(headup) >
      blendStrength
    )
    {
     skinnedMeshRenderer.SetBlendShapeWeight(
        eyesup,
      movement - blendStrength
    );
   }
   else
   {
    skinnedMeshRenderer.SetBlendShapeWeight(
      eyesup,
      0
    );
   }
  }

  void followDown(float diff)
  {
   float movement = diff * blendSpeed;
   skinnedMeshRenderer.SetBlendShapeWeight(
     headdown,
     movement
   );
   if (
     skinnedMeshRenderer.GetBlendShapeWeight(headdown) >
     blendStrength
   )
   {
    skinnedMeshRenderer.SetBlendShapeWeight(
      eyesdown,
      movement - blendStrength
    );
   }
   else
   {
    skinnedMeshRenderer.SetBlendShapeWeight(
      eyesdown,
      0
    );
   }
  }
}
```

A.1.2 Mouth

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MouthAnimation : MonoBehaviour {

  public bool debug = false;

  public int mouthOpenSpeed = 25;

  public int phonemeAAH = 82;
  public int phonemeBMP = 83;
  public int phonemeBigAAH = 84;
  public int phonemeCHJSH = 85;
  public int phonemeDST = 86;
public int phonemeEE = 87;
public int phonemeEH = 88;
public int phonemeF = 89;
public int phonemeI = 90;
public int phonemeK = 91;
public int phonemeN = 92;
public int phonemeOH = 93;
public int phonemeQ = 94;
public int phonemeR = 95;
public int phonemeTH = 96;
public int phonemeW = 97;

private int mouthopen = 28;

public int lookTilt = 45;

public float blendSpeed = 50f;

//private static float ANIMLIMIT = 10f;
public float animStrength = 10f;

[SerializeField]
public Animator anim;

public bool talking = false;

private bool animate = false;
private bool finished = true;
private bool midway = false;

[SerializeField]
public Transform headbone;
[SerializeField]
public Transform neckbone;

[SerializeField]
public SyncMovement lookFrom;

private int currentPhoneme = −1;
private int nextPhoneme = −1;

private int[] phonemes;
//private string[] phonemeNames;

private float lastX = 0;
private float lastY = 0;
private float lastZ = 0;

private float blend = 0f;
private bool mouthclosed = true;
private float currentMouth = 0f;

SkinnedMeshRenderer skinnedMeshRenderer;

void Awake()
{
  // phonemes
```

```
    this.phonemes = new int[] {
      phonemeAAH, phonemeBMP, phonemeBigAAH,
      phonemeCHJSH, phonemeDST, phonemeEE,
      phonemeEH, phonemeF, phonemeI, phonemeK,
      phonemeN, phonemeOH, phonemeQ, phonemeR,
      phonemeTH, phonemeW, phonemeAAH, phonemeAAH,
      phonemeAAH, phonemeAAH, phonemeAAH, phonemeAAH,
      phonemeAAH, phonemeAAH, phonemeAAH, phonemeAAH,
      phonemeAAH, phonemeAAH, phonemeAAH, phonemeAAH,
      phonemeAAH
    };
    skinnedMeshRenderer =
      GetComponent<SkinnedMeshRenderer>();
}

// Use this for initialization
void Start () {

        }

        // Update is called once per frame
        void Update () {

    openWide();
    //current animation still going
    if (!finished)
    {
      speakPhoneme();
    }
    //animation finished, new one called
    else if (animate)
    {
      anim.SetBool("talking", true);
      speakPhoneme();
      finished = false;
    }
        }

void LateUpdate()
{
    //DEBUG
    float x = lastX;
    float y = lastY;
    float z = lastZ;
    if (lookFrom)
    {

      if (
        Mathf.Abs(lookFrom.rotation.x) <
        lookTilt / 2
        ||
        Mathf.Abs(lookFrom.rotation.x) >
        360 - lookTilt / 2
      )
      {
        x = lookFrom.rotation.x;
        lastX = x;
      }
      if (
        Mathf.Abs(lookFrom.rotation.y) <
        lookTilt
        ||
        Mathf.Abs(lookFrom.rotation.y) >
        360 - lookTilt
      )
      {
        y = lookFrom.rotation.y;
        lastY = y;
      }
      if (
        Mathf.Abs(lookFrom.rotation.z) <
        lookTilt / 3
        ||
        Mathf.Abs(lookFrom.rotation.z) >
        360 - lookTilt / 3
      )
      {
        z = lookFrom.rotation.z;
        lastZ = z;
      }
    }

    //disabled Z rotation
    headbone.localEulerAngles =
      new Vector3(x, y, 0);
    neckbone.localEulerAngles =
      new Vector3(0, 0, 0);
  }

  private void speakPhoneme()
  {
    float currentBlend =
      skinnedMeshRenderer.GetBlendShapeWeight(
        currentPhoneme
      );
    //not midway
    if (!midway)
    {
      float blend =
        currentBlend + (blendSpeed * Time.deltaTime);
      //blend not max
      if (blend < animStrength)
      {
        skinnedMeshRenderer.SetBlendShapeWeight(
          currentPhoneme,
          blend
        );
      //blend at max
      } else
      {
        blend = animStrength;
        skinnedMeshRenderer.SetBlendShapeWeight(
          currentPhoneme,
          blend
        );
        midway = true;
      }
    //decrease from midway
    } else
    {
      float blend =
        currentBlend - (blendSpeed * Time.deltaTime);
      if (midway && blend > 0f)
      {
        skinnedMeshRenderer.SetBlendShapeWeight(
          currentPhoneme,
          blend
        );
        //blend at min and finished
      } else
      {
        blend = 0f;
        skinnedMeshRenderer.SetBlendShapeWeight(
          currentPhoneme,
          blend
        );
        midway = false;
        finished = true;
        currentPhoneme = nextPhoneme;
        if (currentPhoneme == -1)
        {
          animate = false;
          anim.SetBool("talking", false);
        }
        nextPhoneme = -1;
      }
    }
  }

  //set the next phoneme
  public void setSpeaking(int phoneme)
  {
    if (phoneme == 0)
    {
      phoneme =
        phonemes[Random.Range(0, phonemes.Length)];
    }
    if (currentPhoneme == -1)
    {
      currentPhoneme = phoneme;
    } else
    {
      nextPhoneme = phoneme;
    }
    animate = true;
  }

  public void setMouth(float amount)
  {
    blend = 0f;
    mouthclosed = true;
    if (amount == 0f)
    {
      blend = 10f;
      mouthclosed = false;
    }
    currentMouth = amount;
  }

  private void openWide()
  {
    if (currentMouth != blend)
    {
      if (mouthclosed)
      {
        blend =
          blend + Time.deltaTime * mouthOpenSpeed;
        if (blend > currentMouth)
        {
          blend = currentMouth;
        }
      }
      else
      {
        blend =
          blend - Time.deltaTime * mouthOpenSpeed;
        if (blend < currentMouth)
        {
          blend = currentMouth;
        }
      }
      skinnedMeshRenderer.SetBlendShapeWeight(
        mouthopen,
        blend
      );
    }
  }
}
```

A.2 VR Painting

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class PaintMaterialNetwork : MonoBehaviour
{

  [SerializeField]
  public GameObject brushPrefab;

  [SerializeField]
  public GameObject paintContainer;

  [SerializeField]
```

```
public Camera paintCamera;
public Camera playerCamera;

[SerializeField]
public GameObject presentationContainer;

[SerializeField]
public Camera presentationCamera;

[SerializeField]
public MeshCollider presentationCollider;

private GameObject hand;
private MeshCollider paintCollider;

private void Start()
{
  ResetColor();
}

private int Raycast(ref Vector3 uvWorldPosition, bool vr)
{
  RaycastHit hit;
  Ray cursorRay;
  if (!vr)
  {
    float position_x = Input.mousePosition.x;
    float position_y = Input.mousePosition.y;

    Vector3 paintPosition =
      new Vector3(position_x, position_y, 0.0f);
    cursorRay =
      playerCamera.ScreenPointToRay(paintPosition);
  } else
  {
    hand = GameObject.FindWithTag("Hand1");
    if (hand != null)
    {
      cursorRay =
        hand.GetComponent<MyLaserPointer>().getRay();
    } else
    {
      return 0;
    }
  }

  if (Physics.Raycast(cursorRay, out hit, 200))
  {
    MeshCollider meshCollider =
      hit.collider as MeshCollider;
    if (
      meshCollider == null ||
      meshCollider.sharedMesh == null
    )
    {
      return 0;
    }
```

```
      if (meshCollider == paintCollider)
      {
        Vector2 pixelUV =
          new Vector2(
            hit.textureCoord.x,
            hit.textureCoord.y
          );
        uvWorldPosition.x =
          pixelUV.x - paintCamera.orthographicSize;
        uvWorldPosition.y =
          pixelUV.y - paintCamera.orthographicSize;
        uvWorldPosition.z = 0.0f;
        return 1;
      } else if (meshCollider == presentationCollider)
      {
        Vector2 pixelUV =
          new Vector2(
            hit.textureCoord.x,
            hit.textureCoord.y
          );
        uvWorldPosition.x =
          pixelUV.x - presentationCamera.orthographicSize;
        uvWorldPosition.y =
          pixelUV.y - presentationCamera.orthographicSize;
        uvWorldPosition.z = 0.0f;
        return 2;
      }
      return 0;
    }
    else
    {
      return 0;
    }
  }

  public void PaintBrush(bool vr)
  {
    if (!vr && playerCamera == null)
    {
      Debug.Log("No␣player␣camera␣for␣painting");
      return;
    }
    if (
      paintCollider == null &&
      presentationCollider == null
    )
    {
      return;
    }
    Vector3 uvWorldPosition = Vector3.zero;
    int result = Raycast(ref uvWorldPosition, vr);
    if (result == 1)
    {
      Vector3 position =
        uvWorldPosition +
        paintContainer.transform.position;
      GameObject paintBrush =
        (GameObject)Instantiate(
          brushPrefab,position,
          Quaternion.identity
        );
      NetworkServer.Spawn(paintBrush);
    } else if (result == 2)
    {
      Vector3 position =
        uvWorldPosition +
        presentationContainer.transform.position;
      GameObject paintBrush =
        (GameObject)Instantiate(
          brushPrefab, position,
          Quaternion.identity
        );
      Vector3 zoom =
        presentationContainer.transform.localScale;
      Vector3 scaling =
        paintBrush.transform.localScale;
      paintBrush.transform.localScale =
        new Vector3(
          scaling.x * zoom.x,
          scaling.y * zoom.y,
          scaling.z
        );
      NetworkServer.Spawn(paintBrush);
    }
  }

  public void setPlayerCam(Camera playerCam)
  {
    this.playerCamera = playerCam;
  }

  public void setPresentationCollider(
    MeshCollider presentationCollider
  )
  {
    this.presentationCollider =
      presentationCollider;
  }

  public void setPaintCollider(MeshCollider paintCollider)
  {
    this.paintCollider = paintCollider;
  }

  public void ResetColor()
  {
    GameObject[] brushes =
      GameObject.FindGameObjectsWithTag("PaintBrush");
    foreach (GameObject brush in brushes)
    {
      NetworkServer.Destroy(brush);
    }
  }
}
```

Appendix B

Server Code Appendix

B.1 Presentation Conversion

```
<?php

$allowedSize = 5000000;
$targetDir = "uploads/";
$uploadOk = 1;
$formOk = 1;

if (!isset($_FILES["fileToUpload"])
  || basename($_FILES["fileToUpload"]["name"]) === "" ) {
  echo "ERROR:␣No␣file␣given.<br>";
  $formOk = 0;
}

if (!isset($_POST["sessionid"])
  || $_POST["sessionid"] === "" ) {
  echo "ERROR:␣No␣session␣ID␣given.<br>";
  $formOk = 0;
}

if ($formOk == 0) {
  return;
}

$sessionID = $_POST["sessionid"];

if (strlen($sessionID) != 13) {
  echo "ERROR:␣Wrong␣session␣ID.<br>";
  return;
}

$file = basename($_FILES["fileToUpload"]["name"]);
$fileSize = $_FILES["fileToUpload"]["size"];
$fileType = pathinfo($file ,PATHINFO_EXTENSION);

$targetFileName = $targetDir.$sessionID;
$targetFile = $targetFileName.".".$fileType;

if (file_exists($targetFile)) {
  echo "ERROR:␣File␣for␣this␣session␣ID␣already␣exists.<br>";
  $uploadOk = 0;
}

if ($fileSize > $allowedSize) {
  echo "ERROR:␣Your␣filesize␣("
    .$fileSize."␣bytes)␣exceeds␣allowed␣filesize:␣"
    .$allowedSize."␣bytes.<br>";
  $uploadOk = 0;
}

if($fileType != "pdf" && $fileType != "ppt" && $fileType != "PDF") {
  echo "ERROR:␣Only␣PPT␣or␣PDF␣files␣are␣allowed.<br>";
  $uploadOk = 0;
}


if ($uploadOk == 0) {
  echo "ERROR:␣Your␣file␣was␣not␣uploaded.<br>";

} else {

  if (move_uploaded_file(
    $_FILES["fileToUpload"]["tmp_name"],
    $targetFile
  )) {
    echo "SUCCESS:␣The␣file␣".$file. "␣has␣been␣uploaded.<br>";
    $convertOk = 1;
    if ($fileType == "ppt") {
      $res = shell_exec(
        "sudo␣/usr/local/bin/unoconv.sh␣"
        .$targetFile
      );
      if ($res != "") {
        $convertOk = 0;
        echo "ERROR:␣".$res."<br>";
      }
    }

    if ($convertOk == 0) {
      echo "ERROR:␣There␣was␣an␣error␣converting␣your␣file.<br>";
      return;
    }

    $res = shell_exec(
      "convert␣-density␣300␣"
      .$targetFileName."
␣␣␣␣␣␣.pdf␣-quality␣100␣"
      .$targetFileName.".jpg"
    );
    if ($res != "") {
      echo "ERROR:␣There␣was␣an␣error␣converting␣your␣file.<br>";
      return;
    }

    echo "SUCCESS:␣The␣file␣has␣been␣converted␣to␣JPG.<br>";
    $count = shell_exec(
            "ls␣"
            .$targetDir
            ."␣|␣grep␣\"".$sessionID.".*.*\.jpg\"␣|␣wc␣-l"
        );
        echo "SUCCESS:␣Created␣".$count."␣slide(s).<br>";
        if ($count == 1) {
            shell_exec(
                "mv␣".$targetFileName.".jpg␣"
                .$targetFileName."-0.jpg"
            );
        }
        echo "SUCCESS:␣Access␣the␣slides␣starting␣with␣"
            .$targetFileName."-0.jpg<br>";
    } else {
        echo "ERROR:␣There␣was␣an␣error␣uploading␣your␣file.<br>";
    }
}

?>
```

B.2 Mesh Upload

```
<?php

$allowedSize = 20000000;
$targetDir = "uploads/";
$uploadOk = 1;
$formOk = 1;

if (!isset($_FILES["fileToUpload"])
    || basename($_FILES["fileToUpload"]["name"]) === "" ) {
    echo "ERROR:␣No␣file␣given.<br>";
    $formOk = 0;
}

if (!isset($_POST["sessionid"])
    || $_POST["sessionid"] === "" ) {
    echo "ERROR:␣No␣session␣ID␣given.<br>";
    $formOk = 0;
}

if ($formOk == 0) {
    return;
}

$sessionID = $_POST["sessionid"];

if (strlen($sessionID) != 13) {
    echo "ERROR:␣Wrong␣session␣ID.<br>";
    return;
}

$file = basename($_FILES["fileToUpload"]["name"]);
$fileSize = $_FILES["fileToUpload"]["size"];
$fileType = pathinfo($file,PATHINFO_EXTENSION);
```

```
$targetFileName = $targetDir.$sessionID;
$targetFile = $targetFileName."."$fileType;

if (file_exists($targetFile)) {
  echo "ERROR:␣File␣for␣this␣session␣ID␣already␣exists.<br>";
  $uploadOk = 0;
}

if ($fileSize > $allowedSize) {
  echo "ERROR:␣Your␣filesize␣("
    .$fileSize."␣bytes)␣exceeds␣allowed␣filesize:␣"
    .$allowedSize."␣bytes.<br>";
  $uploadOk = 0;
}

if($fileType != "obj" && $fileType != "OBJ") {
  echo "ERROR:␣Only␣OBJ␣files␣are␣allowed.<br>";
  $uploadOk = 0;
}


if ($uploadOk == 0) {
  echo "ERROR:␣Your␣file␣was␣not␣uploaded.<br>";

} else {

  if (move_uploaded_file(
    $_FILES["fileToUpload"]["tmp_name"],
    $targetFile)
  ) {
    echo "SUCCESS:␣The␣file␣".$file. "␣has␣been␣uploaded.<br>";
    echo "SUCCESS:␣Access␣the␣file␣with␣".$targetFileName."obj<br>";
  } else {
    echo "ERROR:␣There␣was␣an␣error␣uploading␣your␣file.<br>";
  }
}

?>
```

Bibliography

[1] Mark Billinghurst and Hirokazu Kato. “Collaborative Augmented Reality”. In: *Commun. ACM* 45.7 (July 2002), pp. 64–70. ISSN: 0001-0782. DOI: `10.1145/514236.514265`. URL: `http://doi.acm.org/10.1145/514236.514265`.

[2] Albert Carlin, Hunter G. Hoffman, and Suzanne Weghorst. “Virtual reality and tactile augmentation in the treatment of spider phobia: A case report”. In: 35 (Mar. 1997), pp. 153–8.

[3] igroup.org – project consortium. *igroup presence questionnaire (IPQ) Download*. URL: `http://www.igroup.org/pq/ipq/download.php` (visited on 06/30/2018).

[4] igroup.org – project consortium. *Presence Questionnaire 1*. URL: `http://www.igroup.org/projects/ipq/` (visited on 06/30/2018).

[5] HTC Corporation. *What are the system requirements?* 2016. URL: `https://www.vive.com/us/support/vive/category_howto/what-are-the-system-requirements.html` (visited on 06/30/2018).

[6] Bob G. Witmer and Michael J. Singer. “Measuring Presence in Virtual Environments: A Presence Questionnaire”. In: 7 (July 1998).

[7] Andrei Niculescu Gisli Thorsteinsson Tom Page. “Using Virtual Reality for Developing Design Communication”. In: 19 (2010), pp. 93–106.

[8] Fredrik Holmström. *A simple VOIP solution for integration into unity*. 2014. URL: `https://github.com/fholm/unityassets/tree/old/VoiceChat` (visited on 06/30/2018).

[9] Singular Inversions. *FaceGen Frequently Asked Questions*. URL: `https://facegen.com/faq.htm` (visited on 06/30/2018).

[10] Singular Inversions. *FaceGen Modeller*. URL: `https://facegen.com/modeller.htm` (visited on 06/30/2018).

[11] Nicholas J. Volpe. “Adler's Physiology of the Eye: Clinical Application, Tenth Edition”. In: 24 (Dec. 2004), p. 348.

[12] Randolph L. Jackson and Eileen Fagan. “Collaboration and Learning Within Immersive Virtual Reality”. In: *Proceedings of the Third International Conference on Collaborative Virtual Environments*. CVE '00. San Francisco, California, USA: ACM, 2000, pp. 83–92. ISBN: 1-58113-303-0. DOI: `10.1145/351006.351018`. URL: `http://doi.acm.org/10.1145/351006.351018`.

[13] Randolph L. Jackson, Wayne Taylor, and William Winn. “Peer Collaboration and Virtual Environments: A Preliminary Investigation of Multi-participant Virtual Reality Applied in Science Education”. In: *Proceedings of the 1999 ACM Symposium on Applied Computing*. SAC '99. San Antonio, Texas, USA: ACM, 1999, pp. 121–125. ISBN: 1-58113-086-4. DOI: `10.1145/298151.298219`. URL: `http://doi.acm.org/10.1145/298151.298219`.

[14] Billinghurst M. Morinaga K. Kato H. and K. Tachibana. “The Effect of Spatial Cues in Augmented Reality Video Conferencing”. In: *Proceedings of the 9th International Conference on Human-Computer Interaction (HCI International 2001)*. New Orleans, LA, USA, 2001.

[15] Hirokazu Kato and Mark Billinghurst. “Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System”. In: *Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality*. IWAR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 85–. ISBN: 0-7695-0359-4. URL: `http://dl.acm.org/citation.cfm?id=857202.858134`.

[16] Agisoft LLC. *Agisoft PhotoScan User Manual*. URL: `http://www.agisoft.com/pdf/photoscan_1_4_en.pdf` (visited on 06/30/2018).

[17] ImageMagick Studio LLC. *Command-line Tools: Convert @ ImageMagick*. 1999. URL: `http://www.imagemagick.org/script/convert.php` (visited on 06/30/2018).

[18] Timothy Lottes. *FXAA*. Tech. rep. nvidia Corporation, Feb. 2009.

[19] J. Saragih M. Cox J. Nuevo and S. Lucey. “CSIRO Face Analysis SDK”. In: *AFGR 2013*. 2013.

[20] Claudia Mary Hendrix. "Exploratory Studies on the Sense of Presence in Virtual Environments as a Function of Visual and Auditory Display Parameters". In: (Feb. 1999).

[21] OED-Online. *virtual reality*. URL: `http://www.oed.com/view/Entry/328583?redirectedFrom=virtual+reality#eid` (visited on 12/20/2016).

[22] M. Pappas et al. "Development of a web-based collaboration platform for manufacturing product and process design evaluation using virtual reality techniques". In: *International Journal of Computer Integrated Manufacturing* 19.8 (2006), pp. 805–814. DOI: `10.1080/09511920600690426`. eprint: `https://doi.org/10.1080/09511920600690426`. URL: `https://doi.org/10.1080/09511920600690426`.

[23] Patrick David Pazour. "Head Tracking with AR Toolkit". In: (2015).

[24] Jason M. Saragih, Simon Lucey, and Jeffrey F. Cohn. "Deformable Model Fitting by Regularized Landmark Mean-Shift". In: *International Journal of Computer Vision* 91.2 (Jan. 2011), pp. 200–215. ISSN: 1573-1405. DOI: `10.1007/s11263-010-0380-4`. URL: `https://doi.org/10.1007/s11263-010-0380-4`.

[25] Thomas Schubert. "The sense of presence in virtual environments: A three-component scale measuring spatial presence, involvement, and realness". In: 15 (Apr. 2003), pp. 69–71.

[26] Thomas Schubert, Frank Friedmann, and Holger Regenbrecht. "Embodied Presence in Virtual Environments". In: (Dec. 1998).

[27] Thomas Schubert, Frank Friedmann, and Holger Regenbrecht. "The Experience of Presence: Factor Analytic Insights". In: 10 (June 2001), pp. 266–281.

[28] Thomas Schubert and Holger Regenbrecht. "Wer hat Angst vor virtueller Realität? Angst, Therapie und Präsenz in virtuellen Welten". In: (Jan. 2002).

[29] Thomas Schubert et al. "Real and Illusory Interaction Enhance Presence in Virtual Environments". In: (Jan. 2000).

[30] Dmitry Semyonov. *Re: Algorithms used in Photoscan*. URL: `http://www.agisoft.com/forum/index.php?topic=89.0` (visited on 06/30/2018).

[31] Geetika Sharma and Ralph Schroeder. “Mixing real and virtual conferencing: lessons learned”. In: *Virtual Reality* 17.3 (Sept. 2013), pp. 193–204. ISSN: 1434-9957. DOI: 10.1007/s10055-013-0225-x. URL: https://doi.org/10.1007/s10055-013-0225-x.

[32] Mel Slater and Martin Usoh. “Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments”. In: 2 (Jan. 1993), pp. 221–233.

[33] Noah Snavely, Steven M. Seitz, and Richard Szeliski. “Photo Tourism: Exploring Photo Collections in 3D”. In: *ACM Trans. Graph.* 25.3 (July 2006), pp. 835–846. ISSN: 0730-0301. DOI: 10.1145/1141911.1141964. URL: http://doi.acm.org/10.1145/1141911.1141964.

[34] Unity Technologies. *Unity User Manual (5.6)*. 2017. URL: https://docs.unity3d.com/560/Documentation/Manual/ (visited on 06/30/2018).

[35] Rhiannon Williams The Telegraph. *HTC Vive pre-orders to start on February 29*. 2016. URL: https://www.telegraph.co.uk/technology/news/12092607/HTC-Vive-pre-orders-to-start-on-February-29.html (visited on 06/30/2018).

[36] Oculus VR. *Oculus Touch Launches Today!* 2016. URL: https://www.oculus.com/blog/oculus-touch-launches-today/ (visited on 06/30/2018).

[37] Hannes Wagner. “Capturing Performance Based Character Animation in Real Time for Unity”. MA thesis. Austria: University of Vienna, 2015.

[38] Dag Wieers. *unoconv: convert any document from and to any LibreOffice supported format*. 2009. URL: http://dag.wiee.rs/home-made/unoconv/ (visited on 06/30/2018).