Phantom limb visualization in first-person perspective using a head-mounted display to treat Phantom Limb Pain (PLP)

Master's thesis in Software Engineering

Joel Cedric Lengeling

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Phantom limb visualization in first-person perspective using a head-mounted display to treat Phantom Limb Pain (PLP)

Joel Cedric Lengeling

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2016

Phantom limb visualization in first-person perspective using a head-mounted display to treat **P**hantom **L**imb **P**ain (PLP)

Joel Cedric Lengeling

Supervisor: Ulf Assarsson, Department of Computer Science and Engineering
Supervisor: Max Jair Ortiz Catalan, Integrum AB
Examiner: Miroslaw Staron, Department of Computer Science and Engineering

Master's Thesis 2016
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2016

Phantom limb visualization in first-person perspective using a head-mounted display to treat **P**hantom **L**imb **P**ain (PLP)

JOEL CEDRIC LENGELING
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

Phantom Limb Pain (PLP) is a seriously disabling condition faced by the majority of amputees. Ortiz-Catalan *et al.* introduced, in 2014 a novel medical treatment for PLP showing promising results in clinical studies. The Head Mounted Display (HMD) is an interesting interface to serve this novel treatment more efficiently and more broadly to a larger number of patients. HMDs have lately reached mass-market potential. This thesis looks into, and applies, HMD technology to the novel treatment for PLP. First, HMD technology and software libraries that allow the development of VR/AR experiences are analysed for their suitability for the novel treatment. Then, the implementation of the novel treatment for PLP utilizing HMD technology is described. Finally, computer graphical improvements are proposed to this initial implementation of the novel treatment for PLP utilizing HMD technology. To analyse HMD technology and the software libraries in the framework of this thesis, first, the requirements of the novel treatment utilizing HMD technology are elicited. This elicitation is performed by means of a review of the work by Ortiz-Catalan *et al.* as well as observing a treatment session. A number of hardware options are analysed in order to determine if they may be used in the medical field, if they bear additional requirements to that particular hardware option, and what software might be used to develop for that particular hardware option. A number of software options are analysed in order to determine if they may be used in the medical field, what might be achieved with them, and under which license they are published. Both hardware and software options are analysed by researching published material. The implementation is described by pointing out a number of interesting challenges faced during the implementation. The graphical improvements to the initial implementation of the novel treatment for PLP utilizing HMD technology are analysed by explaining how they may be implemented, how they may improve the embodiment illusion, and how they impact the performance. The outcome of the research shows that indeed HMD technology might be suitable to treat PLP by the novel treatment introduced by Ortiz-Catalan *et al.*

Keywords:PLP, HMD, VR, AR, GearVr, Google Cardboard, Vuforia

Acknowledgments

First of all, I would like to thank Integrum AB for allowing me to do this thesis at their office in Mölndal. I enjoyed my time at the office and I would like to thank everyone there for the welcoming atmosphere. Special thanks go towards Max Ortiz Catalan, my supervisor at Integrum AB. I would also like to thank Morten Kristoffersen, a previous employee at Integrum AB who contributed towards realizing the implementation. Special thanks go also towards Ulf Assarsson, my university supervisor. I would also like to thank Dan Dolonius, a member of Ulf Assarsson's research group, I really enjoyed sitting down with him and getting some general feedback on my work and really helpful suggestions with regards to some technical challenges. I also woulds like to thank my family and friends; for proofreading this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

PLP Phantom Limb Pain
VR Virtual Reality
AR Augmented Reality
HMD Head Mounted Display
PC Personal Computer
UI User Interface
TCP Transmission Control Protocol
OGRE Object-oriented Graphics Rendering Engine
3D Three Dimensional
APK Android Application Package
NDK Native Development Kit
API Application Program Interface
ARPA Advanced Research Projects Agency
USB Universal Serial Bus
TAC Target Achievement Control
SDK Software Development Kit
JNI Java Native Interface
IDE Integrated Development Environment
CES Consumer Electronics Show
MIT Massachusetts Institute of Technology
GPU Graphics Processing Unit
IEC International Electrotechnical Commission
EEC European Economic Community
MP3 MPEG (Motion Picture Experts Group) Layer-3 sound file
OPRA Osseointegrated Prostheses for the Rehabilitation of Amputees

1 Introduction

The majority of amputees suffer from a chronic, seriously disabling, and often intractable condition, called Phantom Limb Pain (PLP). That condition causes amputees to feel pain in a not actually existing (the phantom) limb. Ortiz-Catalan *et al.* published in 2014 a paper describing a novel medical treatment for PLP [30]. That treatment relies on either Virtual Reality (VR) or Augmented Reality (AR) in a mirror-like visualization in third-person perspective on a conventional computer screen. The technological progress seen lately allows to apply that treatment more efficiently and more broadly. A promising technology originating from that progress is Head Mounted Display (HMD) technology. The application of HMD technology supposedly enhances the success rate and addresses in addition some issues plaguing the novel treatment applied on conventional computer screen. This thesis describes the research undertaken towards the evaluation of HMD technology, the software for implementing VR and AR applications and the implementation of the novel treatment utilizing current HMD technology.

Within this introduction a more detailed view at the aim of this thesis is taken. In particular a more detailed description and a view at the numbers behind PLP is provided. Subsequently a number of research questions is stated. Specifying those research questions is followed up by defining a number of limits to the work performed in the framework of this thesis. In the next chapter the state of the art will be summarized. That chapter will begin to discuss HMD technology. After that discussion a view at the medical field - with regards to the rules and regulations to be applied in this field - will be provided. That section about the medical field will be followed up by a detailed description of the current implementation of the novel treatment as described by Ortiz-Catalan *et al.* [30]. In the third chapter HMD technology and software libraries which allow the implementation of the novel treatment will be evaluated. This will be done by first providing the requirements for the novel treatment and then by analyzing some available options. The forth chapter will address the implementation of the novel treatment for a current HMD, by pointing specific challenges encountered during development. The fifth chapter will address possible improvements to the implementation of the novel treatment. The sixth chapter will conclude this thesis and the seventh chapter will provide some suggestions for further research.

1.1 Purpose of this thesis

The purpose of this thesis might be summarized as the development of a more realistic first-person perspective visualization of phantom limbs in VR/AR on a HMD to treat PLP. Within this section some more detailed answers to a couple of general questions are given.

1.1.1 How does the current treatment for PLP work?

The current treatment visualizes motor movements of a phantom limb executed by the patient assuming that limb still does exist. During that treatment, the patient is supposed to perform certain exercises with that phantom limb. The corresponding movements are recognized and decoded by a method referred to as myoelectric pattern recognition [30]. A myoelectric signal is defined as "electrical activity produced by a contracting muscle" [25]. Following that definitions, myoelectric pattern recognition is a method that recognizes patterns in the electrical activity produced by contracting muscles of the patient. The decoded movements are then visualized for the patient. Currently, the visualization of those movements is done in a mirror-like setting applying third-person perspective. That mirror-like visualization is accomplished in AR by utilizing a conventional webcam displaying the result on a conventional computer screen and in VR by displaying the results also on a conventional computer screen. In figure 1.1 a picture of a patient subject to AR variant of the current novel treatment is displayed. A more detailed description of the treatment will be provided in chapter 2.3.1

Figure 1.1: Picture of a patient subject to the AR variant off the current novel treatment, provided by Integrum AB and used with permission by Integrum AB.

1.1.2 What may be improved upon?

Within the current treatment visualization of the patients movements is done in a mirror-like setting applying third-person perspective. Because of that perspective the immersion of the patient into the VR experience is limited. One idea to improve the embodiment illusion and, thus, gain back some realism is the visualization in first-person perspective.

The current approach raises also some more practical issues. The treatment requires a special, not casual setting. Exercise sessions require some time and they need to be performed in front of a workstation or a powerful PC. There might be some benefits in providing the patient with the freedom of undertaking the treatment at a place of his choosing, possibly away from his desk and workstation utilizing his smartphone and a HMD, e.g. in his favourite living room chair, and thus allows avoidance of workstation and webcam in an office setting. Another issue raised is, that the treatment of lower limb PLP is nearly unfeasible, as a lot of space is needed to recognize and display motor movements of a lower limb performed by the patient. There is also a visualization issue when applying lower limb treatment; not all end results of movements - the so-called poses - are easily distinguishable in third-person perspective. Multiple not easily distinguishable but different poses are displayed in figure 1.2. Within that figure the left pose is the visualization of the dorsiflexion pose, the middle pose is the visualization of the plantarflexion pose and the right pose is the visualization of the rest pose.

Figure 1.2: The visualization of not easily distinguishable lower limb poses.

1.1.3 Who might benefit from this research?

There are two major, quite different groups benefiting from this research. The most important group of beneficiaries is the group of PLP patients themselves as well as the medical community which is treating PLP and is performing research on PLP. Another major group is the software development community. But why are those groups beneficiaries; what is their gain from this research?

1.1.3.1 PLP patients and the medical community; what is PLP, and what are the numbers behind PLP?

To truly understand the possible benefits for the medical community and especially the PLP patients themselves a closer view at PLP needs to be taken.

After an amputation amputees might feel so-called phantom sensations. The American physician Mitchel described phantom sensations already in 1872. He wrote "Nearly every man who loses a limb carries about with him a constant or inconstant phantom of the missing member, a sensory ghost of that much of himself, and sometimes a most inconvenient presence, faintly felt at times, but ready to be called up to his perception by a blow, a touch, or a change of wind" [26]. Whenever the amputees feel, besides those sensations, also pain, than they are suffering from Phantom Limb Pain (PLP). Kooijman *et al.* presented a study in 2000 dealing with PLP, stump pain - another condition suffered by some of the amputees - and phantom sensations. Congenital (born without a limb) and non congenital (acquired) upper limb amputees had been interviewed about those conditions. With regards to the correlation between PLP and phantom sensations, Kooijman's study concluded that: "phantom pain was present in 36 out of the 37 subjects experiencing phantom sensations" and thus Kooijman concluded that: "A significant association was found between the prevalence of phantom pain and phantom sensations". In that study by Kooijman roughly half of the acquired upper limb amputees suffered from PLP [24]. Dijkstra *et al.* presented a study in 2002 in which upper and lower limb amputees participated. That study concluded, that roughly 41% of the upper limb amputees, and, 80% of the lower limb amputees, suffered from PLP [8].
As of today the mechanisms behind PLP are not yet fully understood. Nevertheless a number of different treatments are available. The results of those treatments differ from case to case [12]. The novel treatment developed by Ortiz-Catalan *et al.* showed some very promising results in the case of an upper limb amputee "The proposed set of technologies was administered to a chronic PLP patient who has shown resistance to a variety of treatments ... for 48 years."[30]. Generally speaking amputees, who struggle with PLP, might be able to reduce the intensity of the PLP they are facing on a daily base by applying this novel treatment. By further expanding this novel treatment, it might become easier and more comfortable for some patients (especially for lower limb amputees) to apply the novel treatment. It thus provides the medical community with another treatment option for their patients. The widespread application of the novel treatment could also contribute to the understanding of PLP, as well as help the medical professions to better understand how, and why, the novel treatment is working.

1.1.3.2 The software development community

The software development community might benefit from the work in the framework of this thesis in many different areas. Modern HMD technology is on the brink of mass-market acceptance. As a matter of fact, many devices which caused the surge of consumer enthusiasm into HMD technology have been released during the time of this thesis. There is still a lack of resources with regards to how to do development for HMD technology as that technology did not make into the mainstream yet. The software development community might benefit from any research, that makes the development more mainstream. This is especially true with regards to the limitations of the platform applied to the software development during this research. Another area in which the software development community might benefit from this research is, in the area of medical software development and VR/AR, as well as the area of computer graphics on the targeted platform.

1.2 Research questions

In order to achieve the goals of this thesis - the development of a more realistic first-person perspective visualization of phantom limbs in VR/AR on a HMD to treat PLP - a number of issues needs to be addressed. Those issues may be summarized by a number of research questions.

RQ 1 What are the requirements for the first-person perspective implementation of the novel treatment?
Especially:
A What are the special requirements by a PLP patient?
B What are the hardware requirements?
and with regards to individual HMDs:
- May that HMD be used in a medical environment?
- What additional requirements exist for that HMD to apply the novel treatment?
- What software and/or software libraries allow development for that HMD?

RQ 2 What software and/or software libraries may be used to implement the novel treatment for current HMDs?
and with regards to the individual software libraries:
- May that software library be used in a medical environment?
- What can be achieved with that software library?
  with regards to:
  - VR
  - AR
- Under which license is that software library published?

RQ 3 What are issues and points of interest that need to be addressed with regards of implementing the novel treatment?

RQ 4 What computer graphic algorithms allow to improve the embodiment illusion in the framework of the novel treatment?
and with regards to each proposed algorithm:
- How may that algorithm be implemented on the targeted platform?
- How does that algorithm improve the embodiment illusion?
- What is the performance impact of the implementation of that algorithm?
- Is the result of that implementation worth the performance gain?
May a clinical pilot be performed before finishing the thesis:
- How does the patient rate the improvement on the embodiment illusion gained by that algorithm?

1.3 The imposed restrictions to the research

To minimize scope creep a number of restrictions are needed. Those restrictions may be translated for the largest part to requirements within specific areas. For instance requirements that are put on the HMD hardware, will limit the number of HMDs that are interesting (and thus are analysed) in the framework of this thesis and the novel treatment. Those requirements, that are put on the HMD hardware, will be described in more detail in chapter 3.1. Another limitation to reduce scope creep is, that only a limited number of software libraries are analysed as part of chapter 3.3. Only software libraries that are at least supported by one of the HMDs described in chapter 3.2 are analysed. Another limitation to the work might be seen in chapter 4 and in chapter 5; only a limited number of development points of interests and graphical improvements are described.

Beside those requirements and limitations, there are also some requirements and limitations to the software developed as part of this thesis. Most importantly, the software needs to be compatible/fit-in with the current software ecosystem. The current software ecosystem is analysed in more detail in chapter 2.3.

2

State of the art

There are multiple different points of interests this chapter covers. First a view at HMD technology, the current HMD technology and the software development for HMD technology is taken. In the follow up HMD technology and software development in the medical field is analyzed with regards to laws and regulations for the medical field. Finally a detailed description of the current implementation of the novel treatment is provided.

2.1 HMD technology

HMD technology exists already for a surprisingly long time. Nevertheless the technology has not been able to become a mass market product until quite recently. Within this section an overview of HMD technology is given. And the field is categorized. Finally some points of interest for the development of software for current HMD technology is provided.

2.1.1 What is HMD technology?

One of the first, if not the first, HMDs was already developed by Sutherland et Al. at the University of Utah in 1968. The project of Sutherland was in part financed by ARPA[35]. Within the associated publication Sutherland emphasizes that: “if we can place suitable two-dimensional images on the observer’s retinas, we can create the illusion that he is seeing a three-dimensional object”, as well as that: “Although stereo presentation is important to the three-dimensional illusion, it is less important than the change that takes place in the image when the observer moves his head” [35]. Two pictures of that first HMD are displayed in figure 2.1.

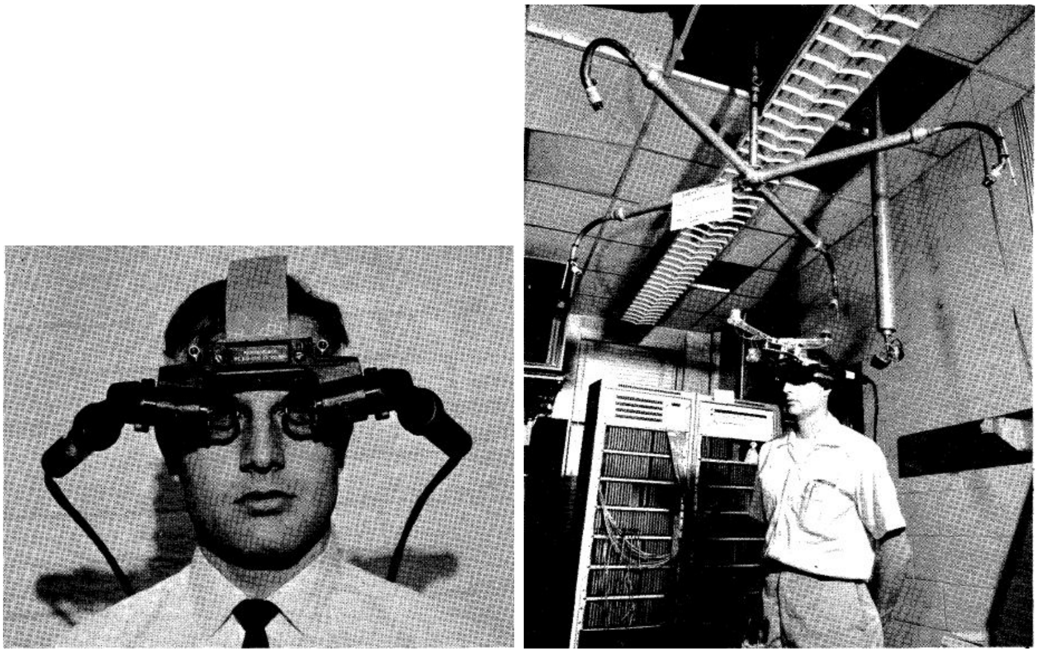

Figure 2.1: Pictures of one of the first HMDs (left) and the mechanical head position tracker (right) by Sutherland *et al.* [35]

The newest generation of HMDs still relies on the same technical concept. A HMD is a helmet, a visor, or even a pair of glasses that project a two dimensional image on each one of the observer's retinas to generate the illusion of a virtual three-dimensional object while at the same time tracking the movements of the observer's head to propagate the head movement transformations to the virtual three-dimensional objects. There is within HMD technology - especially because of AR - a differentiation made between optical and video see-through HMDs. Rolland et Al. did compare in 2000 optical and video see-through HMDs within the medical field. Rolland et Al. defined optical see-through HMDs as devices in which the real world is seen through half-transparent mirrors on that the virtual augmentation is projected. Rolland et Al. defined video see-through HMDs as devices that capture the real world using cameras and that then display the camera feed with virtual augmentation on internal screens.[31] Both, a video see-through HMD and an optical see-through HMD might be seen in figure 2.2.

Figure 2.2: Pictures of an optical see-through HMD (left; Innovega iOptik as presented during CES 2013 here cited from the verge [1]) and a video see-through HMD (right; HTC Vive as presented during CES 2016 here cited from the entertainment technology center [11]).

2.1.2 Current HMD technology

Since the early days of HMD technology many attempts to establish affordable consumer mass market HMDs have been made. One gets the impression that those attempts failed due to the lack of affordability of computational power, as well as sufficient enough screen technology, available to the general public. All that has changed quite recently. Cheaper computational power, modern screen technology and a surge of consumer enthusiasm towards VR and HMD technology, has heralded the area of modern mass market HMD devices. Within modern HMD technology it can be quite interesting to take a look at where the bulk of rendering computations are made and how the results of those computations are displayed. There are currently two families of devices available for purchase to the general public. The first family of devices are the devices which are relying on a powerful PC or workstation for the bulk of rendering computations (PC-HMD). The second family of devices is the family of devices that rely on a mobile phone for the bulk of rendering computations (Mobile-HMD). There are also indications towards an emerging third family of devices, those devices are not yet available to the general consumer market and thus those devices will be discussed as part of chapter 7. As previously already explained, it is important to have head movement tracking to generate an immersive three-dimensional illusion. Within modern HMD technology there exist two different methods of head movement tracking. The first method is the tracking of head movement using sensors within the HMD and thus sensors attached to the head like e.g. acceleration sensors, and the second method is the tracking of head movements using sensors within the environment and thus outside of the HMD e.g. by applying laser movement tracking. Most currently available consumer products use sensors within the HMD to accomplish the head movement tracking.

2.1.3 Development for current HMD technology

These days it seems to be good practice to use already available mainstream engines for the development of VR/AR experiences for HMD technology. Many mainstream engines provide ready to use interfaces towards different HMD hardware libraries. Those interfaces might thus be used to retrieve for instance head movement tracking data. Some of the mainstream engines are also easily expandable. A good example of this would be the Unity® engine and the Unity® AR extension from Vuforia. Should it be for whatever reason not an option to use a mainstream engine, a couple of additional steps are necessary. It is tremendously important to use and have access to head movement tracking data e.g. access to the specific HMD hardware libraries. It is not possible to create an immersive and convincing three-dimensional illusion without having that access [35]. Another important point is that within the main rendering loop two rendering calls - one for each display - need to be made. A well accepted approach is to split the main rendering loop into three different sub routines for stereoscopic rendering. The first sub routine takes care of calculations that are needed for the rendering of each of the two, left retina and right retina, displays. The second and third sub routines are taking care of the respectively left retina and right retina rendering calls.

2.2 Rules and regulations for medical information systems

The standard IEC 62304 “Medical device software – Software life cycle processes” is an international standard covering software products in medicine. Following the definitions set forward by that standard a classification of software in the medical field is made. According to the standard there are three classes of software in the medical field. Those classes are basically classified by the amount of damage they might inflict on the patient. The first class is defined as “No injury or damage to health is possible”, the second class is defined as “Non-SERIOUS INJURY is possible” and the third class is defined as “Death or SERIOUS INJURY is possible” [6]. The classification, by the amount of harm the product might inflict on the patient, is a recurring pattern in rules and regulations for technical devices in medicine. An example of this reacurrence might be seen in the COUNCIL DIRECTIVE 93/42/EEC of 14th of June 1993 concerning medical devices. That European directive defines three classes although one of those classes has been divided into two subclasses [9]. The publication “Medical Information Systems - guidance for qualification and classification of standalone software with a medical purpose” by Läkemedelsverket, the Swedisch medical products agency, heavily relies on that council directive 93/42/EEC [2]. Following those rules and regulations and after discussion with the experts at the company sponsoring this master thesis the conclusion was drawn that both, the software developed here and the hardware applied her might be classified in a way that it represents the low possible damage it might cause patients. Thus the software might be classified as the member of the first class “Class A: No injury or damage to health is possible.”

Following the procedures setup by the standard IEC 62304 and the internal rules of Integrum AB the novel treatment of PLP as introduced by Ortiz-Catalan *et al.* [30] was classified as class A product [29].

Rules and regulations not only in the medical field eventually lead to laws that bindingly have to be followed in individual countries. Thus before treating PLP by the novel treatment in foreign countries applying the technology described here, that technology needs to be reevaluated following the local laws of that country.

2.3 The novel treatment of PLP

The novel treatment introduced by Ortiz-Catalan *et al.* is based on the interaction of the real patient and his simulated phantom limb [30]. Within this section a detailed description of the novel treatment is provided. First a novel treatment exercise session is described. After this description a more detailed analysis of the fundamental technology of the novel treatment is provided. It is important to clarify that currently there are two software platforms that allow application of the novel treatment. The first platform is called BioPatRec and the second platform is called Neuromotus. BioPatRec is developed in Matlab and it is intended as a research platform. BioPatRec is also available to the public on github. The second platform, Neuromotus, is developed in C#. Neuromotus is intended for commercial application. For the research of this thesis it only matters indirectly which platform is applied. The initial implementation has been conducted with the research platform, the Matlab implementation, in mind. The definition of the requirements for the novel treatment as well as the description of novel treatment exercise sessions are made with the commercial platform, the C# implementation, in mind.

2.3.1 A novel treatment exercise session

The following description is the result of observing one novel treatment exercise session with an actual PLP patient in February 2016, as well as two exercise sessions undertaken personally utilizing the treatment as described by Ortiz-Catalan *et al.* [30]. The session with the patient took place under the supervision by Ortiz-Catalan.

The patient needs to undertake a couple of steps before the novel treatment exercise session might start. Initially the patient is asked to undertake a digital survey. That survey is used to monitor the long term results of the treatment sessions. After finishing the survey, the myoelectric pattern recognition has to be setup. Multiple electrodes have to be placed along the patients muscles and connected to, for instance, the Neuromotus easy setup box. The name of that easy setup box is as of today also Neuromotus, in order to not confuse the reader of this thesis, the box will be referred to as easy setup box for the rest of this thesis. That easy setup box might be connected to 17 electrodes, 16 muscle electrodes (taking 8 differential signals) and one reference/ground electrode. Figure 2.3 displays that easy setup box.

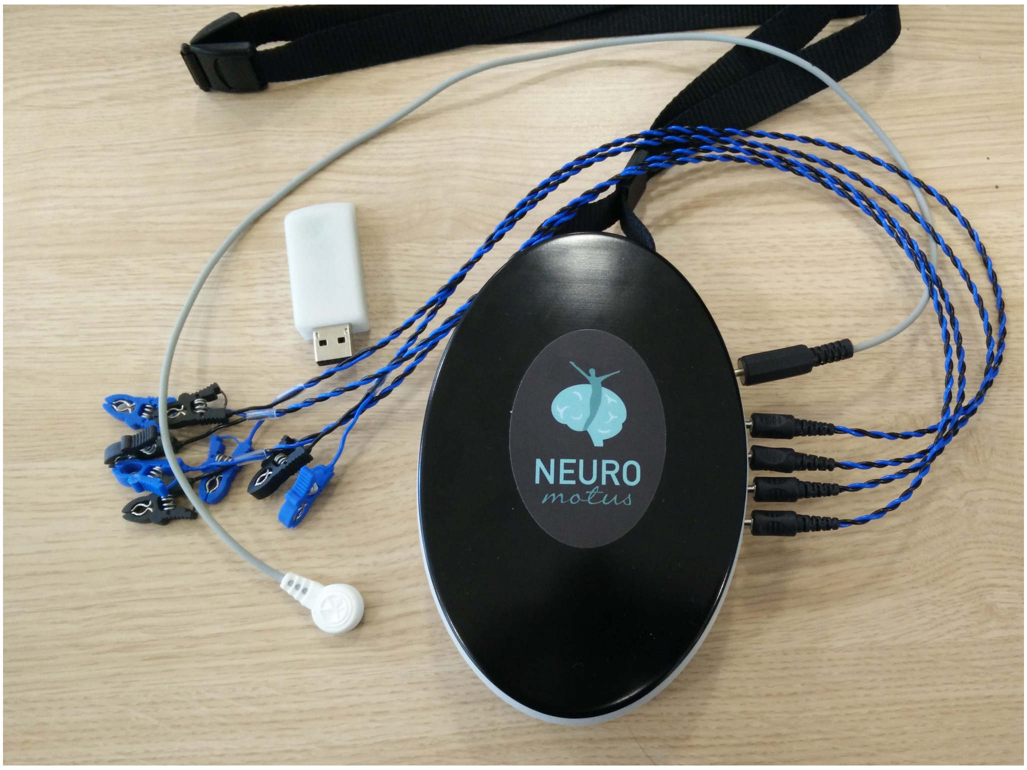

Figure 2.3: The easy setup box, provided by Integrum AB and used with permission by Integrum AB.

After placing the electrodes on the patients skin and connecting them to the easy setup box, a connection between setup box and workstation has to be established. The easy setup box utilizes an USB-stick that allows for wireless connectivity between setup box and workstation. The workstation has to execute Neuromotus to initiate the pattern recognition exercise session. In that exercise session, data for the myoelectric pattern recognition is collected and refined. After setting up the core exercise session, the Target Achievement Control (TAC)-test, is started. The TAC test displays simultaneously two rendered representations of the limb. The first representation is the representation of the limb showing the decoded motor movements by the patient. The second representation, the TAC-test limb, is a representation of the limb showing a target pose. The patient is asked to influence the representation of his own limb - to move the limb - to match the visualization of the representation of the TAC-test limb, and to hold that matched pose for a couple of seconds. After that timespan the TAC-test limb will reset - the patient is also asked to move his limb into a resting pose - before after another couple of seconds the TAC-test limb moves to another exercise pose and the whole ordeal restarts. The whole exercise session (survey, setup and exercises) can take quite some time in the order of up to one hour.

2.3.2 The technology behind the novel treatment and the TAC-test

The visualization of the TAC-test is separated from the core software platform. The VR/AR environments are both the same for BioPatRec and Neuromotus. Figure 2.4 displays the architecture of the existing software ecosystem.

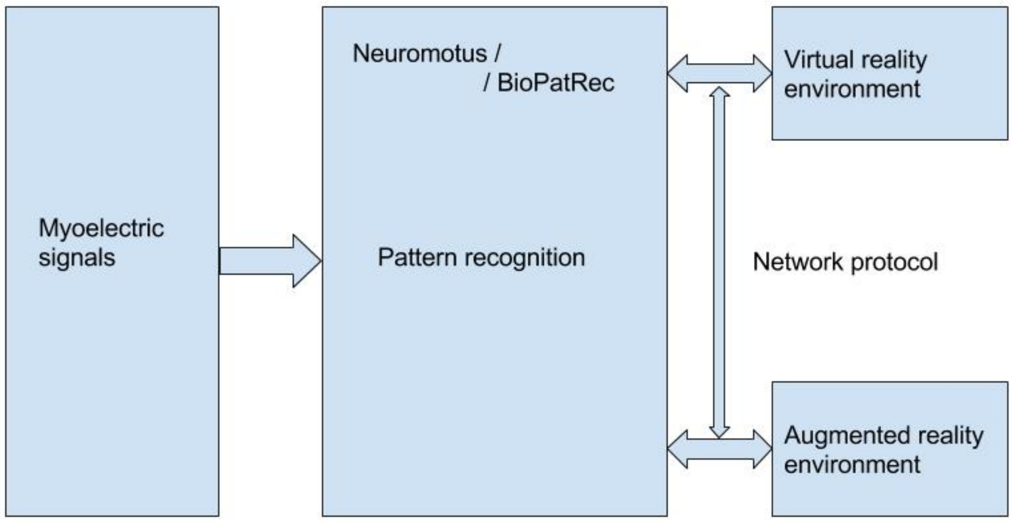

Figure 2.4: The architecture of the existing software ecosystem.

The VR and AR environments communicate with the core platforms using a TCP connection. Within the current implementation; the core platform will launch either the VR or the AR environment, when the corresponding button is clicked within the core platform UI. The core platform will then start to listen to the localhost port 23068. The previously launched software will then start to initialize a TCP connection between core platform and environment. The current versions of the VR/AR environment is implemented in C++ applying the OGRE engine. The AR part of the AR environment is made possible by utilizing OpenCV, ArUco and ArUcoOgre functionality on top of the VR environment functionality. Rather important is the use of the OGRE skeleton and animation system within both the environments. The communication over the TCP connection utilizes a custom communication protocol. The communication protocol might be used for three different purposes. It might be used to configure, reset and control the visualization within the VR/AR environments. The protocol contains up to five values of the type character. How those five values might be used is shown in more detail in table A.1 in the appendix. To a certain degree, one might conclude that the VR/AR environment is nothing but a visualization of an OGRE skeleton system that might be manipulated using the network protocol. Rather important for the prototype is thus, the support of the already established communication protocol, as well as a OGRE style implementation of the skeleton system and the animation of that skeleton system.

3

HMD and software technology for the novel treatment of PLP

Within this chapter an overview of HMD devices and software libraries that allow the development for the novel treatment utilizing HMD technology is given.

3.1 Hardware requirements

There are many quite different requirements put on the possible HMD devices. Most requirements have already been hinted at in the introduction and state of the art chapters. The requirements described here have been elicited as part of analysing the treatment description by Ortiz-Catalan *et al.* [30], by observing novel treatment exercise sessions under the supervision of Ortiz-Catalan and by discussing those sessions with Ortiz-Catalan [29].

Most importantly the device cannot interfere with the existing pattern recognition methods/setup e.g. the HMD should not negatively effect the previously described easy setup box. Because of this requirement it is preferred to use a HMD that does not add another cable connection to the general setup. There are already up to 9 cable connections (connecting up to 17 electrodes) in an exercise session utilizing the easy setup box as described in chapter 2.3.1. Another reason for choosing a HMD that does not add another cable connection is, that the exercise sessions might take quite some time and it is more comfortable for the patient if it is possible to conduct the exercise session away from his workstation. The easy setup box would also allow the patient to conduct his exercises away from his workstation, as that box is wirelessly connected to the workstation. Another important requirement is, that the HMD might be controlled by software implementing the network protocol as described in chapter 2.3.2. Generally speaking it is necessary that the application may be operated without direct user input; amputees might miss a limb which regular software rely on for operation. Since a treatment session might take up to one hour it is important that the HMD may be comfortably worn and used for such a long period of time. The device should also be capable of both VR and AR. Beside all those requirements another general requirement is, that the device is part of the current HMD technology spectrum and that the device is easily obtainable for the average patient and/or medical practitioner.

3.2 Reviewing HMDs

As of today there are only a limited number of HMDs on the market that meet all those requirements. First of all PC-HMDs are not an option due to the requirement, that the HMD does not interfere with the existing pattern recognition method/setup. A PC-HMD relies on the PC or workstation for the bulk of rendering computations and thus, the higher rendering computation demand might interfere with the already intensive pattern recognition computations. Another reason why PC-HMDs are not an option is the typically needed extra cable between PC or workstation and the HMD. The HMD should also be capable of both VR and AR. Because of this requirement only video see-through devices seem to be an option. Optical see-through HMD technology would as of today only allow AR and not VR. Mobile-HMDs are an option. A Mobile-HMD does not rely on a PC or workstation for the bulk of the rendering computations. The smartphones which are required for the bulk of rendering computations in a modern Mobile-HMDs do support network stacks. A Mobile-HMD would not add another cable connection. Modern smartphones also come with an integrated camera that possibly supports AR. Currently Mobile-HMDs are also the easiest to obtain and thus represent the cheapest available to the general public form of HMD technology.

Here we are dealing with two major players which provide Mobile-HMD technology. The first player is Google. Google provides with Google Cardboard™VR a platform that supports VR/AR experiences on Android- and IOS-smartphones. The second player is Samsung. Samsung is the company behind the GearVR platform. GearVR supports VR/AR experiences on a number of Android smartphones by Samsung. The Google Cardboard™VR platform comes in multiple quite different Mobile-HMD device incarnations. Here we restrict ourself to discuss the reference by Google, the Google Cardboard, with all its pros and cons. The GearVR platform only comes with one HMD, the GearVR.

3.2.1 Google Cardboard

Figure 3.1: Pictures of the Google Cardboard; as presented by Google on the official Cardboard store page [18]

The Google Cardboard is a HMD that needs to be held by hand in the proper position close to the eyes for proper operation. That obviously already provides a complication with regards to the novel treatment. Patients are most likely not able to hold the HMD for an entire exercise session; as a matter of fact some patients might miss the limb needed to hold the device. Thus an additional requirement is the purchase of a head strap, for Google Cardboard. Another requirement for the use of the Google Cardboard™VR platform is, since the Google Cardboard™VR platform is part of the Mobile-HMD family, a mobile phone that supports the use of the Google Cardboard™VR platform. It might become quite challenging for the average patient to figure out which mobile phone or smartphone he should purchase. On the official Google Cardboard website, Google points out that "Most Cardboard apps work with Android 4.1+ and the latest iOS smartphones."[19], otherwise there are no recommendations provided. The Google Cardboard™VR platform comes with a SDK which supports native (C++, C) Android development and IOS development; the Unity® engine is also supported by Cardboard [16]. Following up the discussion in chapter 2.2 none of the consulted sources claims any additional reason not to apply Google Cardboard in a medical environment. Thus Google Cardboard might be deployed for the application in question here. The Cardboard SDK offers an interface that grants developers access to head movement tracking data [17]. That head tracking data is reportedly calculated using internal phone sensors[40]. The normal Google app store distribution methods may be used for Cardboard apps.

3.2.2 GearVR

Figure 3.2: Pictures of the GearVR; as presented by Samsung on the official GearVR page [32]

The GearVR is a HMD that comes with a head strap, thus no extra head strap needs to be purchased. To use the GearVR, a supported Samsung smartphone is necessary. The GearVR is the result of a cooperation between Oculus and Samsung. The device is shipped with the Oculus mobile SDK. The Oculus mobile SDK allows for multiple different ways for native (C++/C) Android development. GearVR is supported by the Unity® and by the Unreal® engines [28]. The GearVR reportedly relies on internal tracking sensors within the HMD, reportedly those tracking sensors are quite similar to the head tracking technology used in the PC-HMD Oculus Rift and offers a lower latency compared to smartphone sensors [40]. It seems to be that the general public consent is, that the GearVR provides the most immersive Mobile-HMD technology as of today. Following up the discussion in chapter 2.2 none of the consulted sources claims any additional reason not to apply the GearVR in a medical environment. A remark is, that the platform is not an open platform. Only applications that have been approved by Oculus might be published. The approved applications might also only be distributed over the Oculus store front. Applications, that are not approved or that have not, by means of a developer keyfile, been unlocked, cannot be used with the device. The device will block the usage of not unlocked applications.

3.3 Analyzing software libraries

Within this section a number of software libraries and computer graphic engines that might be used to develop for GearVR and Google Cardboard are analyzed. Since there are many different engines and software libraries, that might be used to develop for GearVR and Google Cardboard available, only a limited number will be described in more detail. Those software libraries and engines are the Unity® engine, the Oculus mobile SDK and Vuforia (a software library which allows the development of AR experiences).

3.3.1 Unity®

Unity® is one of several so called game engines that might be deployed by software developers to develop computer games. Following the marketing information on the Unity Technologies website, the company claims to have a market share of plus 45% citing an unreleased McKinsey report [36]. As of today, the newest version of Unity® is version 5.3.5. Unity® is closed source and it ships with a variety of licensing models. There are licensing options for independent developers/development studios, enterprise solutions, educational licenses and industry solutions. One of those industry solutions is Unity for serious games. Unity Technologies employee Davey Jackson wrote about serious games in 2013 on the official Unity Technologies blog that they are games that are applied in the military field, the educational field or the medical field [23]. Following up the discussion in chapter 2.2 none of the consulted sources claims any additional reason not to apply the Unity® in a medical environment.

Unity® is a full-fledged game engine and offers thus many different game engine functionalities. The engine ships also with support for Google Cardboard and the GearVR. The engine allows for the inclusion of plugins and Vuforia (see chapter 3.3.3) offers such a plugin for Unity®.

3.3.2 Native development with the Oculus mobile SDK

As of today, the newest version of the Oculus mobile SDK is version 1.0.0.1. That SDK might be freely used as long as it is used on an Oculus device. The license also allows the modification of the SDK, although if the modification is part of a published product all modifications have to be shared with Oculus [27]. Following up the discussion in chapter 2.2 none of the consulted sources claims any additional reason not to apply the Oculus mobile SDK in a medical environment.

To fully understand what can be achieved with the Oculus mobile SDK it is important to understand how Android and Android Native code - the NDK - works. Android applications operate relying heavily on the Android lifecycle callbacks. Figure 3.3 displays a diagram from the offical Android development website, that visualizes the Android lifecycle callbacks quite well.

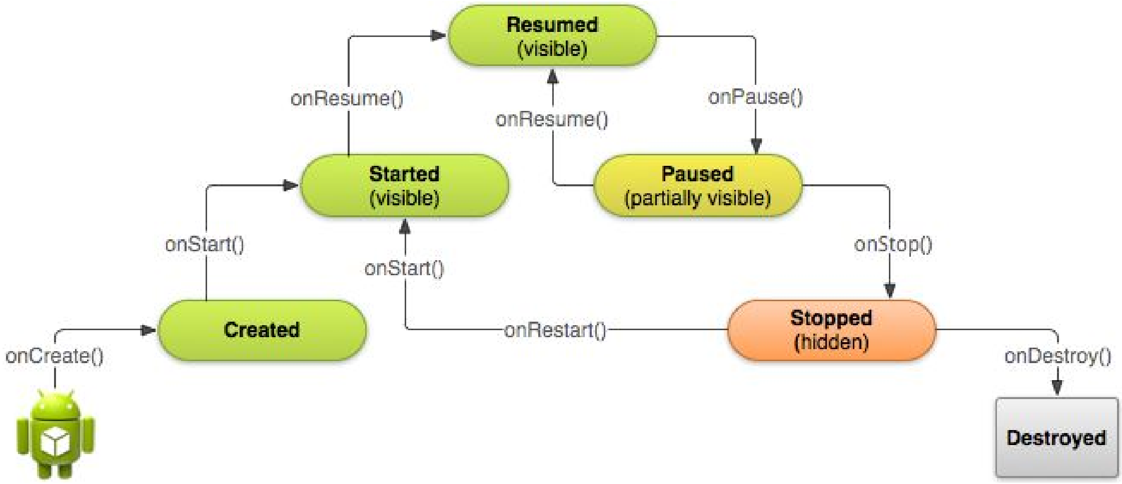

Figure 3.3: Diagram visualizing the android lifecycle callbacks; here taken from the offical Android development website [15]

The callbacks basically deal with the interaction between the operation system and the application. Android applications are created using the onCreate method and started using the onStart method. The applications might then be managed by the onPause and onResume methods. Those methods are in particular important for the multitasking functionalities that Android as a platform provides. Finally the application might be stopped and started again using the onRestart method or stoped and basically killed off, using the onDestroy method. Native Android development works in a way possibly best described with the terms wrapper, proxy or sandbox. Native C or C++ code is executed within a native core. That native core is surrounded by an Android/Java application. The Android application is able to call and execute native C or C++ functions by means of the JNI. It is also possible to create a purely native application for Android. There are some pros and cons to this full native approach. A pro is for instance that developers get access to some powerful native platform libraries. One example of those powerful native platform libraries is the NativeActivity class. A con of developing full natively is that developers are not able to access any default Android/Java libraries provided by the Android platform. Diagrams that visualize both, the full native and the JNI native development approach might be seen in figure 3.4.

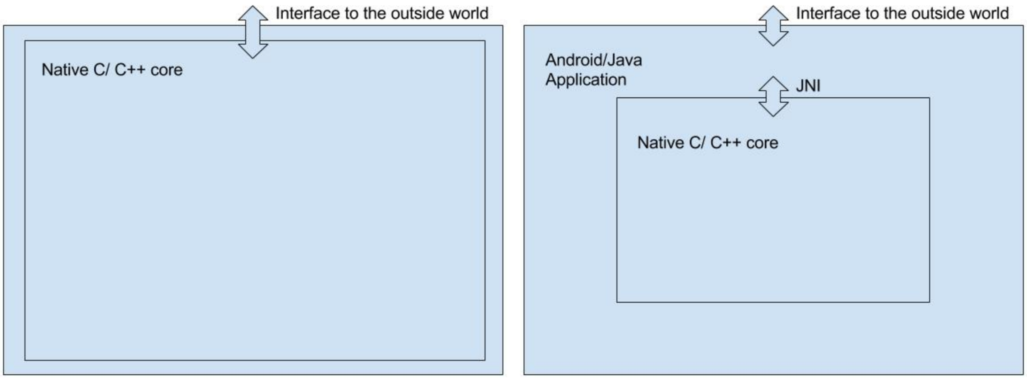

Figure 3.4: Diagrams that display the full native (left) and the JNI native (right) development approach.

There are three ways in which the Oculus mobile SDK might be used. The first way is the full native way. This approach allows developers to circumvent any Android/Java coding. The other two ways do rely on the JNI. The difference between those two ways is the Oculus mobile SDK framework. That framework is an optional framework that contains classes and interfaces developed and implemented by Oculus that take care of the Android lifecycle callbacks. The framework eases the development since developers do not need to worry about the Android lifecycle callbacks anylonger.

A topic necessary to be mentioned here is, the lack of resources for implementation of software applying the Oculus mobile SDK. Version 1.0.0.0 of the SDK was only released at the end of Oktober 2015; just shortly before the release of the mainstream GearVR consumer edition. Older versions - basically beta versions of the SDK - are not supported anymore and version 1.0.0.0 of the SDK is the first version that moved from the previously recommended Eclipse IDE to the now recommended official Android IDE. A general issue with developing for Oculus mobile SDK is, not only the lack of resources for the Oculus mobile SDK, but also the general lack of resources for native Android development. The official Android NDK website points out that "The NDK is not appropriate for most novice Android programmers, and has little value for many types of Android apps. It is often not worth the additional complexity it inevitably brings to the development process." [20]. The NDK support by the official Android IDE is as of today still of "preview quality" [14], as a matter of fact the recommended way for developing NDK applications utilizing the offical Android IDE changed at least onces quite drastically during the development period of the software for this thesis. It is thus not surprising that the native development using Oculus mobile SDK is lacking resources and documentation.

As a matter of fact and after developing a fully native application, it does not come as a surprise that the companies behind the GearVR strongly push developers towards the Unity® and the Unreal® engines.

3.3.3 Vuforia

The Vuforia SDK is as of today seemingly the goto solution for AR development on mobile, and thus also for Mobile-HMDs. The SDK is closed source and offers two licensing models, as well as a free starter package that ships with a Vuforia watermark. The SDK is licensed by a per application model and customers might either pay a lump sum as an one time fee, or pay on the base of a monthly subscription rate [38]. Following up the discussion in chapter 2.2 none of the consulted sources claims any additional reason not to apply Vuforia in a medical environment. As of today the newest version of Vuforia is version 5.5.9. Version 5.5.9 ships with Android (Native as well as non-Native), IOS and Unity support [37].

Vuforia offers a number of functionalities which allow for immersive AR. First of all Vuforia offers an interface to the camera feed of the smartphone. Vuforia also provides easy to use recognition of images, objects, cylinders, generally speaking user-defined targets, markers and even specified terrains. It is important though to point out that to use the recognition of all those different types first needs a reference scan to be created. The creation of that reference scan might be quite troublesome depending on the type and target. There is another important point which needs to be made with regards to native Android development. Vuforia when applied natively still needs to be initiated using the Vuforia non-native Android SDK. To successfully accomplish this initialization a network connection and a pre-generated license key need to be applied. As a result it is not possible to develop a fully native Android application with Vuforia support.

4

Development of a first-person perspective visualization of phantom limbs on a GearVR for application of the novel treatment

Before any program might be developed, the situation and all the available options need to be analyzed. As in chapter 3.2.2 already pointed out, there are three possible development options available for GearVR. Those options are the Unity® engine, the Unreal® engine and the native development approach. Since already at the start of the development the decision was made, that the AR part of the application shall be implemented using Vuforia, only two of those initial three options where still available options leaving out the Unreal® engine. As in chapter 3.3.3 already pointed out, Vuforia on Android ships only with support for the Unity engine and the native development approach. As in chapter 2.3.2 already mentioned, the existing VR/AR environments are basically just visualizations of an OGRE skeleton system that might be manipulated using the established network protocol. As of today OGRE is not a supported engine for GearVR. The next closes supported development approach is the native development approach. In order achieve the comparability to the network protocol/the OGRE skeleton system interface, that development approach was taken.

Initial efforts where made towards porting the whole of OGRE, applying the native development approach, to GearVR. This proved to be a quite tedious endeavor. The official android version - in particular the build system and the render system integration - of the OGRE engine turned out to be incompatible with the GearVR native development approach. Eventually the decision was made to re-implement the necessary OGRE parts in a custom engine that is compatible with the native development approach. The OGRE engine itself is distributed under a MIT license and the source code is available online.

Within this chapter a number of different elements that turned out to be issues and/or points of interest with regards to the implementation of the novel treatment for GearVR.

4.1 Dealing with application assets

The assets of a GearVR application are the 3D model-, the skeleton-, the material- and the texture-data, as well as the developer keyfile for the authorization to the closed GearVR platform as described in chapter 3.2.2. Loading some of those assets proved to be quite a challenge.

In particular loading 3D model- and corresponding skeleton- and material-data proved to be unreproducible. An Android application installation file is called an APK file and here APK files are compressed archive files. Those archived files contain besides assets, also the source code, manifest data, certificates and so on [13]. The compression of APK files is known to cause issues [10]. During the development of the application an issue, most likely caused by the compression, was encountered. It is important to understand that the APK is newly generated and compressed every time the application gets deployed to the smartphone [13]. At a certain point it turned out that the serialization results of the asset data were inconsistent. Without changing any source code, and just by redeploying the program (basically generating and installing a new compressed APK), different serialization outcomes where witnessed. A possible solution for solving APK compression issues is to modify the file extension of asset data to the MP3 extension [10]. The APK generator does not compress MP3 files. Changing the file extension to MP3 is not a solution in this case though. There are multiple different files that all belong to the same 3D model - the model-, the skeleton- and the material-data of a single model are distributed over a couple of files - and for all those files different serialization code is necessary. The program determines the right serialization code using the file extension, so if all file extensions are the MP3 file extension the serialization would not be able to commence. Thus it seems that the issues originate in the way, in which android packages data into the installation file. As there was no urgent need to follow up the way the operation system deals with the packaging and compressing process, that is not fully disclosed in the available literature, that step was avoided.

In order to circumvent the APK compression and the possibly related issues, the decision was made to write the 3D model as well as the corresponding material and skeleton data into a C++ source code file. To accomplish this a python parser, that utilizes the API of the 3D modeling tool Blender, has been developed. Utilizing that parser allowed the successful circumvention of the APK compression.

4.2 OGRE style skeleton and animation system

As mentioned already before the novel treatment described by Ortiz-Catalan *et al.* is based on the interaction of the patient and his simulated phantom limb [30]. The realistic movement of limbs is an old endeavor in computer graphics for e.g. computer games. As already a couple of times pointed out, the existing VR/AR environments are basically just visualizations of an OGRE style skeleton system. Those kind of skeleton systems are an established way of animating characters in video games.

While analyzing the OGRE skeleton system implementation, it became clear that the backbone of the system relies on a technique that had already been described in the GPUgems series by Nvidia earlier - the copyright in the here referenced online version dates back to 2004 [5]. As described in GPUgems three groups of elements are needed to pull off an OGRE style/general style skeleton system. Those three element groups are the joint transformation matrices, the joint assignment vectors and corresponding to those joint assignment vectors, the joint weight assignment vectors. The two vector groups are part of the 3D model- and skeleton data and are rarely during program execution. Only when the joint weight assignment vectors need to be normalized an alteration takes place. The joint transformation matrices are quite dynamic, and those matrices are usually recalculated whenever the scale, the rotation or the position of a bone within the skeleton system is modified. The rotation of a bone or joint is stored in form of a quaternion and the scale and position are stored using vectors. During runtime, the position and scale of the bones are, besides at the initialization, usually not modified. The novel treatment protocol, as described in chapter 2.3.2 and table A.1 of the appendix, only allows for the rotation of bones. The rotation values are used to create a new quaternion that might then be multiplied with the stored rotation data of the bone, to calculate a new quaternion result. That newly calculated quaternion result might then be applied together with the previously stored rotation data to calculate the interpolation curve between the two quaternions.There are multiple methods available that allow for the interpolation of quaternions. On of those methods is the linear interpolation, or Lerp function [7]. Source code displaying the Lerp function might be seen in code listing B.1.

4.3 Communication between treatment sub systems

One of the important requirements is, that the developed software implements the network protocol as described in section 2.3.2. That protocol needs to be extended though. The current implementation of the protocol does not allow for VR/AR switching, since the desktop VR and AR environments are totally separated. Another difference is, that perspective modifications might not be implemented for GearVR. All those changes might be easily implemented by e.g. extending the existing protocol.

The technical aspect of the server implementation might be of interest for some of the readers. The diagram 4.1 displays the technical aspects of the server implementation. In the current implementation the network stack has been implemented on the Java/Android side of the application. The server forwards all the incoming messages using the JNI to a native code function. The native code function forwards all messages by a public interface to the rendering thread. The requested internal operations are then executed and the rendering thread will (if not disabled) send an ACK - acknowledge/ an everything worked out fine - message back through the JNI function returning the ACK value to the Java/Android side.

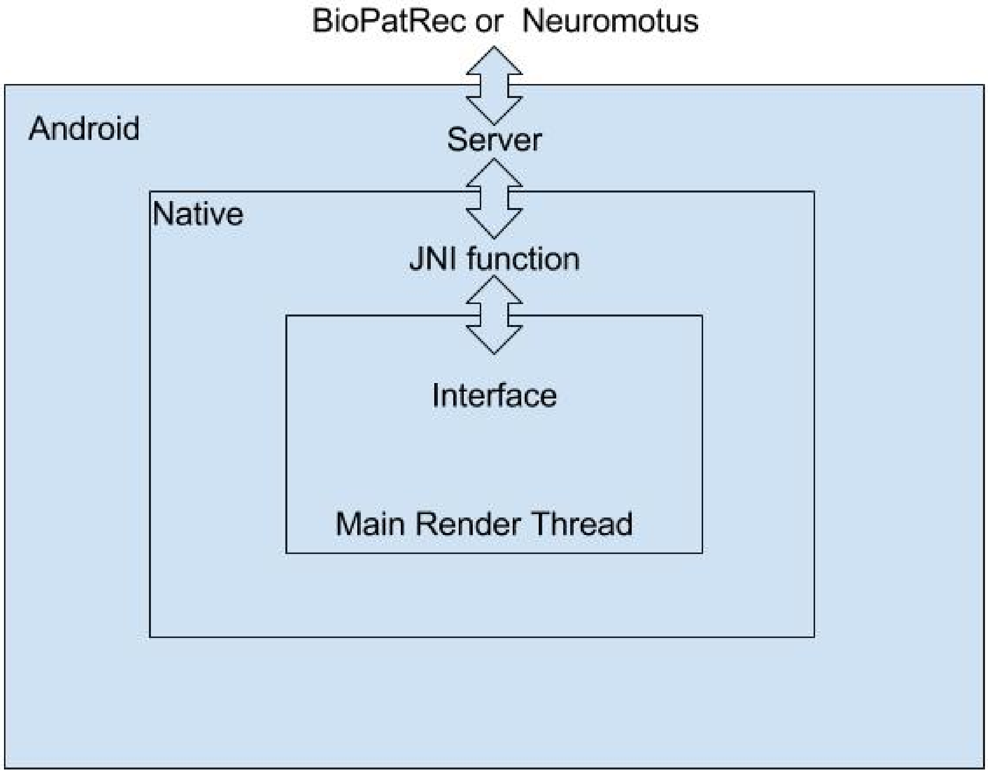

Figure 4.1: Visualization of the network interfaces.

4.4 AR applying Vuforia

As already described in chapter 3.3.3, Vuforia offers some useful functionality for AR. Important is that Vuforia needs to be activated utilizing a pre-generated key, before any of that functionality might be used. It is, as of today, not possible to perform that activation in a native code environment since the activation needs to be performed using a Java function defined in the provided Java library. For a successful activation a pre-generated Vuforia application key needs to be passed as a function argument to the Java function. If the activation has been successfully executed; access to the native side of the Vuforia library is granted.

In order to generate an AR experience first a couple of setup steps need to be performed. First the Vuforia tracker needs to be setup with tracking data. After initializing the tracker, and loading tracking data, the camera needs to be setup. To setup the camera, device specific data - screen resolution as well as camera resolution- needs to be specified. Afterwards first the camera and then the tracker might be started. As previously in chapter 3.3.2 already explained the Android life-cycle callbacks are important. When developing AR experiences applying Vuforia, developers should make sure that they turn off the camera and the tracker when calling the onPause method, as well as turn them both on again when calling the onResume method.

To turn the novel treatment VR experience into an AR experience, first the camera feed has to be rendered into a texture. This texture has to be drawn to a billboard that has been placed really close to the near plane of the view frustum. A visualization displaying the near plane of a view frustum might be seen in figure 4.2.

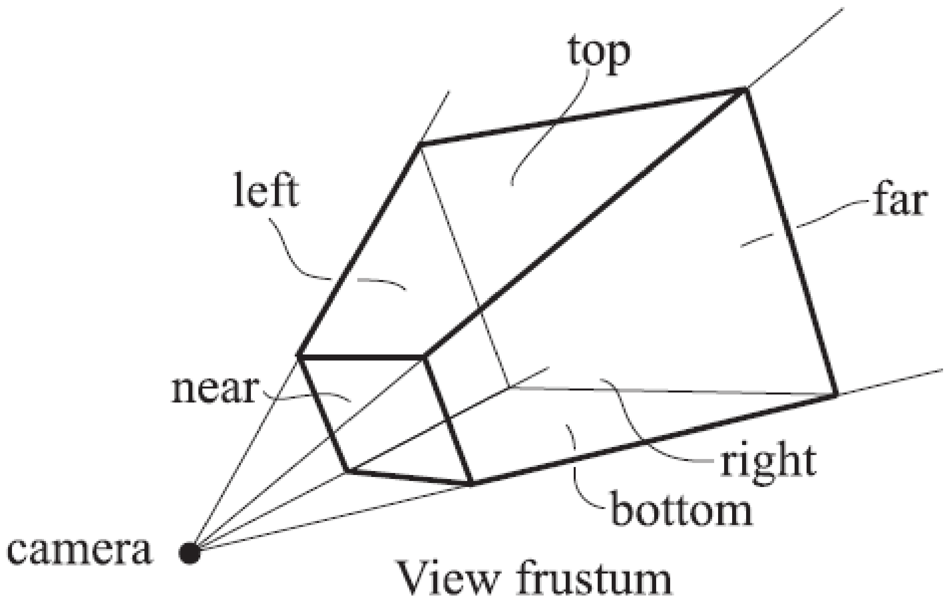

Figure 4.2: Visualization of the view frustum; here taken from the book Real-Time Rendering by Akenine-Möller *et al.*, there figure 16.22 page 771 [3].

In a second step a transformation matrix, that either contains by successful tracking generated values or zero values, has been forwarded to the vertex shader. That matrix might then be incorporated into the vertex matrix computations.

That described AR implementation does technically allow for the novel treatment. Practically it does not allow that step. Tracking turned out to be quite intensive and as a result the smartphone operating within the GearVR reaches it's thermal limits. The smartphone becomes literately to hot for further operation after a couple of minutes. The smartphone will, if this point has been reached, excuse itself from operation and it will switch into a low energy/standby mode.

5

Graphical improvements to the visualization

5.1 From Lerp to Slerp

To calculate the interpolation between two quaternions the Lerp function has been applied in the initial implementation. There are more function available that allow for interpolating quaternions. One of those functions is the spherical linear interpolation function (Slerp) [7]. In an effort to improve the embodiment illusion of the initial implementation the Slerp function has been implemented. The Slerp compared to the Lerp allows for more natural looking interpolation. A lerp animation shows a constant speed of movement of the phantom limb; a Slerp varies the speed of movement of the phantom limb from slow speed over faster speed back again to lower speed reassembling typical human motions. Figure 5.1 displays interpolation points of a Lerp function and a Slerp function.

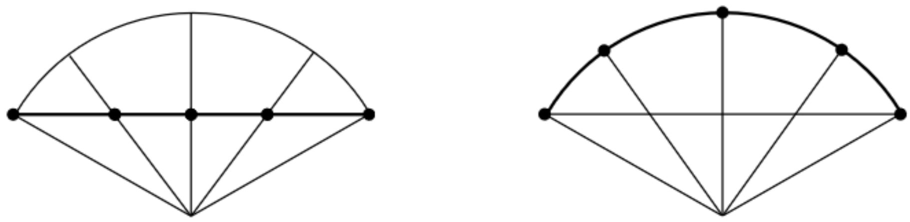

Figure 5.1: Visualization of the interpolation results between two quaternions. On the left using the Lerp function and on the right using the Slerp function. Pictures are cited from a presentation by Armstrong [4].

The mathematics behind an interpolation applying a Slerp calculation is way more complicated compared to the same procedure applying a Lerp calculation. This may also be seen in the source code of listing B.2. By comparing the implementation of the Lerp function(B.1) and the implementation of the Slerp function(B.2), it is easy to see that the Slerp function is heavier computation wise. Even though the Slerp function is more heavier computation wise, there is no measurable impact on the performance. Nevertheless the difference between the Slerp and the Lerp interpolation proofed to be hardly noticeable, as the rotations applied within in the novel treatment are rather small.

6

Conclusions and discussion

Within chapters 3, 4 and 5, a number of different aspects of developing a more realistic first-person perspective visualization of phantom limbs in VR/AR on a HMD to treat PLP have been analysed. Topic of chapter 3 has been the different HMDs and software libraries. Chapter 4 targets the development of a first-person perspective visualization of phantom limbs on a GearVR for application of the novel treatment. Graphical improvements to the initial implementation have been discussed in chapter 5. But how do these aspects compare to the research questions stated in chapter 1.2? Furthermore, what are the general results of the research committed within the framework of this thesis?

6.1 Revisiting the research questions

Within chapter 1.2 of the introduction a number of research questions have been stated. Those research questions might be categorized into two categories. The first category deals with the prelude of the development. The second category deals with the development itself.

With regards to the prelude of the development many different research questions have been answered by chapter 3. It has been found that there are many different requirements put on the possible HMD devices. Those requirements are best met by Mobile-HMDs. Important is ...

- that the HMD should be compatible with the existing software ecosystem
- that the HMD must not add extra cables
- that the HMD should allow operability by a network connection
- that the HMD should be comfortably usable for up to one hour
- that the HMD should allow VR and AR within one device
- that the HMD should be part of the current HMD technology spectrum
- that the HMD should be easily obtainable for the average patient and/or medical practitioner

As part of this thesis two HMD devices have been analysed (the GearVR and the Google Cardboard) in more detail. For both HMDs none of the consulted sources claims any additional reason not to apply the devices in a medical environment. Both HMDs require an extra mobile phone besides the HMD itself for operation. The Google Cardboard requires an extra head strap. For both HMDs Unity® and native development might be applied. The Unreal® Engine allows within this framework only development for the GearVR.

The prelude of the development also deals with three software libraries (Unity®, Oculus mobile SDK and Vuforia). For all three libraries none of the consulted sources claims any additional reason not to apply those libraries in a medical environment for an class A device. Unity® was found to be the goto solution for most VR and AR - applying the Vuforia Unity® plugin - experiences. There are multiple different licenses available for Unity® (see chapter 3.3.2 for more details). The Oculus mobile SDK was found to meet all the required VR aspects of the novel treatment. The SDK is compatible with Vuforia and thus allows AR by Vuforia. The SDK might be used freely if a number of requirements (see chapter 3.3.2) are met. Vuforia allows to apply AR for HMD technology (although some compatibility issues with the capabilities of the relied on platform were found later on). The Vuforia library offers multiple different licensing options (see chapter 3.3.3).

During the development itself many interesting challenges have been met. Dealing with the loading of the assets has been solved by parsing the required data into a C header file. Implementing the skeleton and animation system has been solved by applying technologies published by Nvidia and by implementing the Lerp function. The challenges of the network communication of the treatment sub system have been met by expending the current protocol and by implementing a Java server stack. The AR has been technically achieved by implementing Vuforia; but the in chapter 4.4 described heat issues definitely pose a practical obstacle here.

As part of the later stages, one possible graphical improvement has been implemented. The Slerp compared to the Lerp allows for more natural looking interpolation. The implementation of the Slerp might be seen in code listing B.2 of the appendix. The algorithm did not provide a visual nor had a measurable impact on the performance of the implemented novel treatment. Sadly, no clinical pilot has been performed as of today. It remains to be seen, how patients react to the difference between Lerp and Slerp animation.

6.2 Discussion of the results

The introduction states that "The purpose of this thesis could be summarized as the development of a more realistic first-person perspective visualization of phantom limbs in VR/AR on a HMD to treat PLP". The introduction also points out a number of issues which might be improved upon. Looking back on the work performed throughout this thesis, the purpose has been mostly achieved. The work performed here has created a first-person perspective visualization of phantom limbs in VR on a HMD that potentially allows the treatment of PLP. The work performed here did not successfully manage to create an first-person perspective visualization of phantom limbs in AR on a HMD for treatment of PLP; due to heat issues of the platform. It remains to be seen how the patients of the treatment respond to the implementation of the treatment on HMD technology; especially with regards to the possible improvements stated in the introduction. Although technically managing to implement a working AR version of the treatment, practically it might not be used for the treatment of PLP due to thermal limitations. Sadly, the current version of the novel treatment will cause overheating when applying AR utilizing Vuforia on the provided hardware. Figure 6.1 displays a screenshot of a warning patients would face if the heat issue occurs.

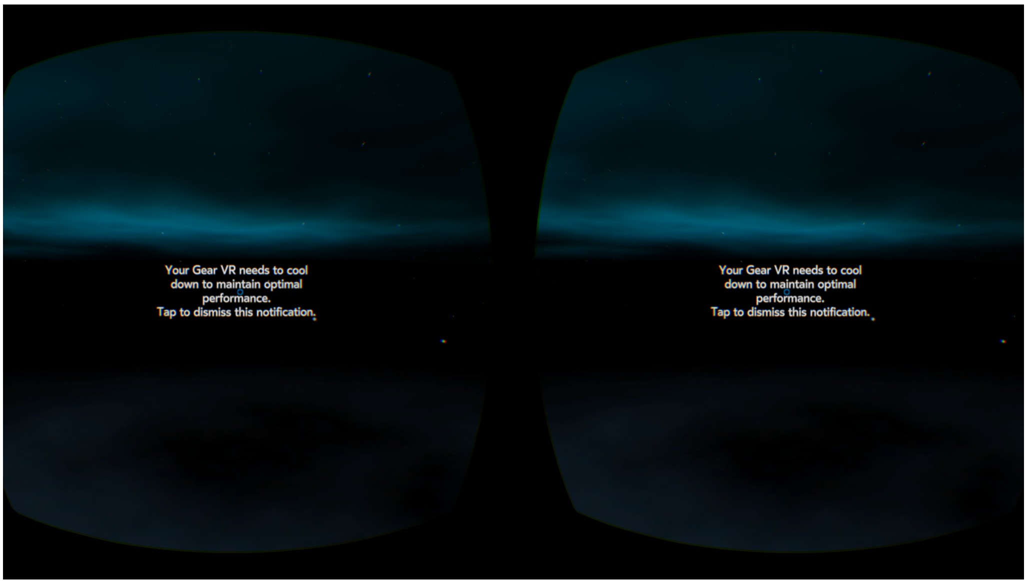

Figure 6.1: Screenshot taken in developer mode, of the warning patients would potentially face when applying AR on the provided hardware due to heat issues.

6.3 Ethical aspects

The work performed in the framework of this thesis has shown that HMD technology might be used to treat PLP. It is interesting to see that, technology originating in research, partly financed by ARPA, is today available to the general public, as well as that, this technology might be used to treat pain, amputees might face. Most likely the work performed in the framework of this thesis will be soon implemented to help pain suffering amputees.

7

Further Research

Within this chapter a number of options that might be elaborated upon during further research are suggested. Those suggested options deal mostly with software and hardware related topics.

7.1 Future HMD technology

As already hinted at in chapter 2.1.2, a new family of HMD devices has already been announced. That family of devices does not rely on a PC nor a mobile phone for the bulk of computations; all rendering and processing hardware has been build into the HMD. A first example of this is Sulon Q™. An intresting fact about SulonQ™is, that the rendering and processing hardware within the device might even rival the hardware of a standalone PC [34].

There have also been some interesting developments on the mobile-HMD market towards the end of this thesis time. During the Google IO in May 2016 Google announced the Daydream platform. Daydream might be seen as the successor of Google Cardboard. Google promises with Daydream " a platform for high quality, mobile virtual reality. Coming in Fall 2016, Daydream provides rich, responsive, and immersive experiences" [21], it remains to be seen how Daydream performance compared to GearVR.

7.2 Vuforia

Two further research suggestions are made with Vuforia in mind. First of all as in chapter 4.4 pointed out the current hardware faces thermal issues when enabling Vuforia. Halfway trough the research time of this thesis a new smartphone by Samsung was unveiled. One of the marketing key points of this new device is the internal cooling system, based on liquid cooling [33]. It might be interesting to see how the new generation of smartphone performs with regards to the thermal issues of the AR implementation.

Another interesting research field with regards to Vuforia and the novel treatment might be the Vuforia object tracker. The object recognition works best with objects that are " opaque, rigid and contain few moving parts." [39]. Further research might potentially look into the possibility to recognize stumps or the by Integrum developed Osseointegrated Prostheses for the Rehabilitation of Amputees (OPRA)

Implant System. Figure 7.1 displays a visualization of that implant system. The Abutment might turn out to be a good tracking target for the Vuforia object tracker.

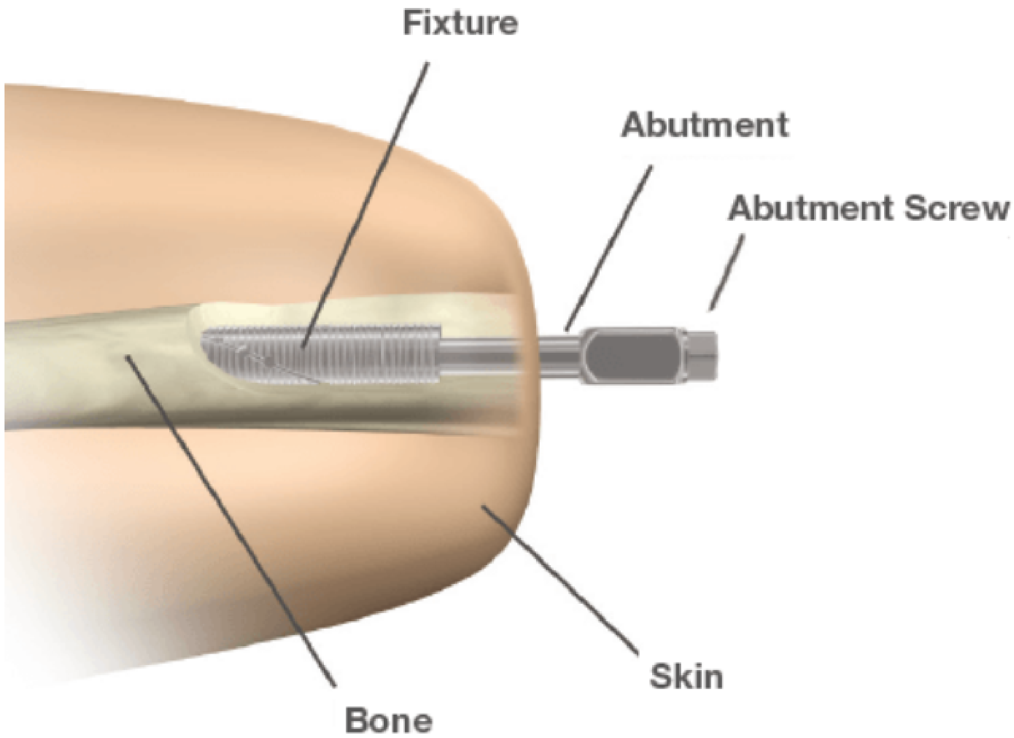

Figure 7.1: The OPRA Implant System; as presented by Integrum AB on the Integrum website [22], used with permission from Integrum AB.

7.3 The server implementation

The server implemented in the way described by chapter 4.3 might be easily redeployed for other applications. E.g. further research might envisage possible applications that might rely on this server implementation. One possible example would be the implementation of a video game.

7.4 The novel treatment implementation

There are some further research options with regards to the implementation of the novel treatment. Research might be done towards other possible graphical improvements. There is also research possible towards the reactions of the patients to the HMD implementation e.g. a clinical study.

Bibliography

[1] Robertson a. *Picture of an optical see-through HMD*. Website (theverge.com). accessed: 29.05.2016. 2013. URL: http://www.theverge.com/2013/1/10/3863550/innovega-augmented-reality-glasses-contacts-hands-on.

[2] Swedish Medical Products Agency. *Medical Information Systems - guidance for qualification and classification of standalone software with a medical purpose*. accessed: 16.06.2016. 2012. URL: https://lakemedelsverket.se/upload/eng-mpa-se/vagledningar_eng/medical-information-system-guideline.pdf.

[3] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering*. Figure 16.22. CRC Press, 2008, p. 771. ISBN: 978-1-4398-6529-3.

[4] Arthur Armstrong. *Presentation by Arthur Armstrong*. Website (slideplayer.com). accessed: 01.06.2016. 2016. URL: http://slideplayer.com/slide/8419872.

[5] Beeson C. *Animation/skeleton system*. Website (nvidia.com). accessed: 29.05.2016. 2004. URL: http://http.developer.nvidia.com/GPUGems/gpugems_ch04.html.

[6] International Electrotechnical Commission et al. *Medical device software: software life cycle processes*. IEC 62304:2006. IEC, 2006.

[7] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*. Technical Report DIKU-TR-98/5. Datalogisk Institut, Københavns Universitet, 1998.

[8] Pieter U Dijkstra, Jan HB Geertzen, Roy Stewart, and Cees P van der Schans. "Phantom pain and risk factors: a multivariate analysis". In: *Journal of pain and symptom management* 24.6 (2002), pp. 578–585.

[9] Council Directive. *93/42/EEC of 14 June 1993 concerning medical devices*. Legal binding version published by EC in OJ L 169, 12.7.1993, p. 1. 1993.

[10] egoflux. *Anecdote describing APK compression issues*. Website (eondev.blogspot.se). accessed: 29.05.2016. 2012. URL: http://eondev.blogspot.se/2012/03/after-following-nice-code-sample-from.html.

[11] ETC-USC. *Picture of a video see-through HMD*. Website (flickr.com). accessed: 29.05.2016. 2016. URL: https://www.flickr.com/photos/92587836@N04/24177102722/.

[12] Herta Flor. "Phantom-limb pain: characteristics, causes, and treatment". In: *The Lancet Neurology* 1.3 (2002), pp. 182–189.

[13] Google. *Google Android glossary*. Website (android.com). accessed: 29.05.2016. 2016. URL: https://developer.android.com/guide/appendix/glossary.html.

[14] Google. *Google Android IDE NDK*. Website (android.com). accessed: 28.05.2016. 2016. URL: http://tools.android.com/tech-docs/android-ndk-preview/.

[15] Google. *Google Android lifecycle callbacks*. Website (android.com). accessed: 28.05.2016. 2016. URL: https://developer.android.com/training/basics/activity-lifecycle/starting.html/.

[16] Google. *Google cardboard SDK*. Website (google.com). accessed: 01.06.2016. 2016. URL: https://developers.google.com/vr/overview#sdks.

[17] Google. *Google cardboard SDK head tracking*. Website (google.com). accessed: 01.06.2016. 2016. URL: https://developers.google.com/vr.

[18] Google. *Google cardboard store*. Website (google.com). accessed: 28.05.2016. 2016. URL: https://store.google.com/product/google_cardboard.

[19] Google. *Google get cardboard website*. Website (google.com). accessed: 28.05.2016. 2016. URL: https://vr.google.com/cardboard/get-cardboard.

[20] Google. *Google NDK*. Website (android.com). accessed: 28.05.2016. 2016. URL: https://developer.android.com/ndk/guides/index.html/.

[21] Google. *Google VR start page*. Website (google.com). accessed: 02.06.2016. 2016. URL: https://developers.google.com/vr/android/.

[22] Integrum. *The Opra implant system*. Website (integrum.se). accessed: 01.06.2016. 2016. URL: http://integrum.se/our-solutions/opra-implant-systems.

[23] Davey Jackson. *Blog post Serious Games*. Website (unity3d.com). accessed: 30.05.2016. 2013. URL: http://blogs.unity3d.com/2013/03/05/unitys-serious-business-with-serious-games/.

[24] Carolien M Kooijman, Pieter U Dijkstra, Jan HB Geertzen, Albert Elzinga, and Cees P van der Schans. "Phantom pain and phantom sensations in upper limb amputees: an epidemiological study". In: *Pain* 87.1 (2000), pp. 33–41.

[25] D. F. Lovely. "Signals and Signal Processing for Myoelectric Control". In: *Powered Upper Limb Prostheses: Control, Implementation and Clinical Application*. Ed. by Ashok Muzumdar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. Chap. 2, pp. 17–33. ISBN: 978-3-642-62302-8.

[26] Silas Weir Mitchell. *Injuries of nerves and their consequences*. JB Lippincott, 1872, p. 348.

[27] Oculus. *Oculus mobile SDK*. Website (oculus.com). accessed: 29.05.2016. 2016. URL: https://developer.oculus.com/licenses/mobile-3.2.1.

[28] Oculus. *Oculus mobile SDK getting started*. Website (oculus.com). accessed: 29.05.2016. 2016. URL: https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-getting-started.

[29] Max Ortiz-Catalan. private communication. Feb. 12, 2016.

[30] Max Ortiz-Catalan, Nichlas Sander, Morten B. Kristoffersen, Bo Håkansson, and Rickard Brånemark. "Treatment of phantom limb pain (PLP) based on augmented reality and gaming controlled by myoelectric pattern recognition: a case study of a chronic PLP patient". In: *Frontiers in Neuroscience* 8.24 (2014). ISSN: 1662-453X. DOI: 10.3389/fnins.2014.00024. URL: http://www.frontiersin.org/neuroprosthetics/10.3389/fnins.2014.00024/abstract.

[31] Jannick P Rolland and Henry Fuchs. "Optical versus video see-through head-mounted displays in medical visualization". In: *Presence* 9.3 (2000), pp. 287–309.

[32] Samsung. *Gear VR*. Website (samsung.com). accessed: 01.06.2016. 2016. URL: http://www.samsung.com/global/galaxy/wearables/gear-vr.

[33] Samsung. *Samsung Galaxy S7*. Website (samsung.com). accessed: 30.05.2016. 2016. URL: https://news.samsung.com/global/samsung-electronics-brings-galaxy-s7-and-galaxy-s7-edge-to-the-global-market.

[34] Sulon. *The Sulon HMD*. Website (sulon.com). accessed: 30.05.2016. 2016. URL: http://sulon.com/release/sulon-unveils.

[35] Ivan E Sutherland. "A head-mounted three dimensional display". In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM. 1968, pp. 757–764.

[36] Unity. *Unity marketing material*. Website (unity3d.com). accessed: 30.05.2016. 2013. URL: https://unity3d.com/public-relation.

[37] Vuforia. *Vuforia download page*. Website (vuforia.com). accessed: 28.05.2016. 2016. URL: https://developer.vuforia.com/downloads/sdk/.

[38] Vuforia. *Vuforia licensing information*. Website (vuforia.com). accessed: 28.05.2016. 2016. URL: https://developer.vuforia.com/pricing/.

[39] Vuforia. *Vuforia objeckt tracker*. Website (vuforia.com). accessed: 30.05.2016. 2016. URL: https://developer.vuforia.com/library/articles/training/object-recognition.

[40] Wikipedia. *Gear VR Wikipedia page*. Website (wikipedia.org). accessed: 01.06.2016. 2016. URL: https://en.wikipedia.org/wiki/Samsung_Gear_VR.

[41] XnaGeometry. *Quaternion library source code*. Website (technologicalutopia.com). accessed: 01.06.2016. 2013. URL: http://www.technologicalutopia.com/sourcecode/xnageometry/quaternion.cs.htm.

A

Tables

A.1 The existing communication protocol

[Byte] - Operation	Value #1	Value #2	Value #3	Value #4
[1] - Movement of limb	Joint	Direction	Distance	Fraction
[2] - Movement of TAC-limb	Joint	Direction	Distance	Fraction
[r] - Reset	[t] ->TAC else limb	-	-	-
[c] - Configure	see	sub	table	below

Value #1	Value #2	Value #3	Value #4	Description
1	byte ([1]/[2])	-	-	[1]-> upper limb [2]-> lower limb
2	-	-	-	switch TAC on / off
3	degrees	-	-	modify precision margin TAC
4	camer id	-	-	modify perspective
5	-	-	-	switch right / left limb
6	-	-	-	switch above / below, elbow / knee
7	-	-	-	turn TCP-ACK response on / off

Table A.1: The communication protocol table

B

Source code listings

B.1 Lerp

```
//The Quatf type is the quaternion implementation provided by the Oculus mobile SDK
Quatf lerp(Quatf q1, Quatf q2, h){
    double hInverse = 1-h;
    Quatf interpolated;
    //The helper value is used to detmine in which direction the interpolation should take place
    double helper = ((q1.x*q2.x)+(q1.y*q2.y)+(q1.z*q2.z)+(q1.w*q2.w));
    if (helper >= 0){
        interpolated.x = (hInverse * q1.x) + (h * q2.x);
        interpolated.y = (hInverse * q1.y) + (h * q2.y);
        interpolated.z = (hInverse * q1.z) + (h * q2.z);
        interpolated.w = (hInverse * q1.w) + (h * q2.w);
    }else{
        interpolated.x = (hInverse * q1.x) - (h * q2.x);
        interpolated.y = (hInverse * q1.y) - (h * q2.y);
        interpolated.z = (hInverse * q1.z) - (h * q2.z);
        interpolated.w = (hInverse * q1.w) - (h * q2.w);
    }
    //it is good practise to always normalize quaternions
    return interpolated.Normalize();
}
```

Listing B.1: Source code displaying the Lerp implementation in C++ for the novel treatment. Custom implementation of code presented by the XnaGeometry library, modified for use with the Oculus mobile SDK quaternion type [41].

B.2 Slerp

```
//The Quatf type is the quaternion implementation provided by the ↩
    Oculus mobile SDK
Quatf slerp(Quatf q1, Quatf q2, h){
   double hInverse;
   Quatf interpolated;
   //The helper value is used to detmine in which direction the ↩
      interpolation should take place
   double helper = ((q1.x*q2.x)+(q1.y*q2.y)+(q1.z*q2.z)+(q1.w*q2.w));
   double helper2;
   bool flag = false;
   if (helper < 0){
      flag = true;
      helper = -helper;
   }
   if (helper > 0.999999 ){
      hInverse = 1 - h;
      helper2 = flag ? -h : h;
   } else {
      double tmpA = acos(helper);
      double tmpB = 1.0 / sin(tmpA);
      hInverse = sin((1 - h) * tmpA) * tmpB;
      helper2 = flag ?  (-sin(h * tmpA) * tmpB) :
                        ( sin(h * tmpA) * tmpB);
   }

   interpolated.x = (hInverse * q1.x) + (helper2 * q2.x);
   interpolated.y = (hInverse * q1.y) + (helper2 * q2.y);
   interpolated.z = (hInverse * q1.z) + (helper2 * q2.z);
   interpolated.w = (hInverse * q1.w) + (helper2 * q2.w);

   //it is good practise to always normalize quaternions
   return interpolated.Normalize();
}
```

Listing B.2: Source code displaying the Slerp implementation in C++ for the novel treatment. Custom implementation of code presented by the XnaGeometry library, modified for use with the Oculus mobile SDK quaternion type [41].