DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Virtual Embodiment of Human Feet in the Neurorobotics Platform

Jason Janse van Rensburg

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Virtual Embodiment of Human Feet in the Neurorobotics Platform

Virtuelle Verkörperung von menschlichen Füßen in der Neurorobotics Platform

Author:	Jason Janse van Rensburg
Supervisor:	Prof. Gudrun Klinker
Advisor:	M.Sc. Sandro Weber
Submission Date:	15.8.2019

I confirm that this Bachelor's Thesis in Informatics: Games Engineering is my own work and I have documented all sources and material used.

Munich, Jason Janse van Rensburg

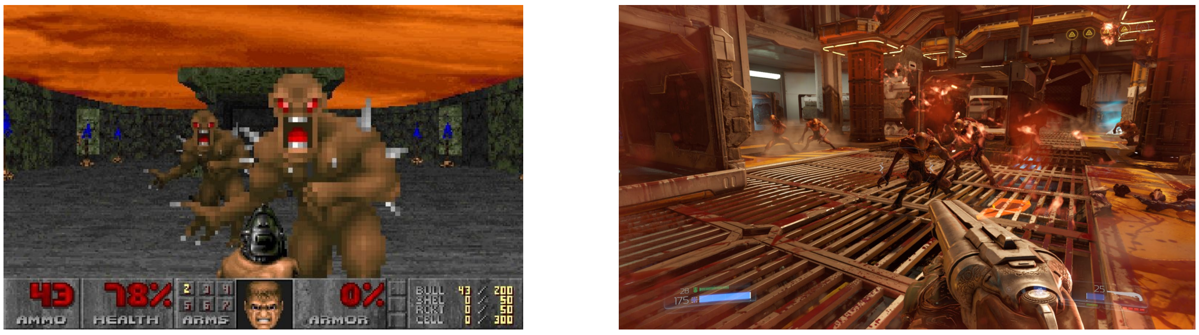
Abstract

As virtual reality increasingly finds more use cases, methods of improving the experience beyond graphical fidelity have to be investigated. One such improvement is giving the user a full virtual body, to make him feel part of the virtual world. Problems, however, are created when the user tries to interact with the virtual world with his virtual body, but but without the interaction having any effect on his real body. There has been research into preventing, or encouraging the user to resolve, situations in which his real and virtual body are not in sync, but the focus has largely been on the head and hands. Recently, hardware has become widely available that also allows tracking of any other object, such as the user's feet.
In this thesis, techniques of alerting the user to, and encouraging him to resolve, such discrepancies between his real and virtual body will be investigated, and implemented in the *Neurorobotics Unity3D Client*, with a focus on the feet.

Contents

1 Introduction

Ever since the dawn of video games, the focus has been concentrated on making virtual worlds look as realistic as possible. Advances in computer graphics (CG) techniques as well as continuously increasing processing power have allowed even real time CG to become almost photo-realistic [Kar17].
An example for this evolution in graphical fidelity is the comparison of *id Software*'s *Doom* video games from 1993 and 2016 in Figure 1.1. The basic way the game is played has stayed essentially the same: buttons on the keyboard trigger actions displayed on the screen, which acts as a window into the virtual world. The largest difference is the drastically more believable presentation of the game world, which is a comparatively easily quantifiable improvement of player immersion [BA04]. This thesis, however, will explore a different avenue to increased player immersion.



(a) Doom 1993 [Gam17] (b) Doom 2016 [New16]

Figure 1.1: Evolution of CG exemplified by *Doom* games

The rise of high quality virtual reality (VR) has allowed users to not only look into a virtual world, but to see it as if they were actually part of it. There are various factors which influence how effective the illusion of being part of a virtual environment is. This thesis will focus on one: giving the user a fully animated body, as "a virtual body in the context of a head-mounted display based virtual reality is a critical contributor to the sense of being in the virtual location" [KGS12, p. 374]. This is commonly referred to as *virtual embodiment*, of which a more in depth discussion will follow in chapter 3.

A problem that arises when giving the user a body in VR is that the virtual world can influence the virtual body, exerting forces on it, but not on the real body of the user. This can result in situations where the real and virtual body become mismatched due to, for example,

the tracked position of a real hand being inside a virtual object, such as a wall, and the virtual hand not following into the object.
One effect of such discrepancies being handled inadequately, is motion sickness. According to James Lackner, “The sensory conflict theory of motion sickness proposed by Reason [. . .] is the most widely accepted theory of motion sickness” [Lac14, p. 11].Here it is stated that a discrepancy between visually perceived motion and bodily felt motion, is the main cause of motion sickness.
Another problem that arises when such discrepancies are ignored, is that the user can, for example, stick his head through a wall, and is thus able to see parts of the environment he should normally not be able to, or even to walk through obstacles to interact with objects he is not meant to be able to access. From the environment designer’s point of view, this is problematic since the designer is not in full control of where the player can be at all times. From the user’s perspective, these abilities, depending on the context, can impair his immersion, since he is able to do impossible things.
Assuming, that the user’s virtual limbs will not follow his tracked, real limbs into (in the virtual world) impossible states, one could assume that it would be simple for the user to correct his pose. However the Rubber Hand Illusion experiment[1] shows that the human body is not adept at locating unseen parts of itself [BC98], so we cannot rely on the user’s sense of kinesthesia to tell him where his real body parts are.

In an effort to mitigate these problems, methods for alerting the user to discrepancies between his real and his virtual body which will allow him to easily adjust his pose to match that of his virtual body, especially with respect to his feet, will be investigated.

[1]In experiments, stimulating a test subject’s hidden hand as well as a rubber hand placed in view before them simultaneously in the same manner, caused the test subject to mislocate his real, hidden hand.

2 Related Work

2.1 Virtual Embodiment: Dealing with Discrepancies between the Virtual and the Real Body

In his thesis *Virtual Embodiment: Dealing with Discrepancies between the Virtual and the Real Body* [Hau18] Jonathan Haudenschild proposed and implemented various feedback mechanisms to deal with such discrepancies in room-scale VR. He focused on ideas that rely only on the basic *HTC Vive* hardware, namely the two controllers and the head mounted display (HMD). His work forms the basis of this thesis, with the addition of a fully tracked body, utilising some new hardware, *Vive Trackers*, discussed in chapter 4. The focus is shifted from discrepancies only at hands and head to also include the user's tracked feet, as well as implementing all feedback mechanisms into a VR client for the *Neurorobotics Platform*, which will be presented in chapter 5.

Haudenschild proposes various feedback mechanisms to notify the user of, and encourage him to correct, situations in which he has a pose or is in a position his virtual body should physically not be, which results in discrepancies.
For discrepancies at the head, such as when the user sticks his head into a wall, he implemented multiple visual effects [Hau18, p. 20]:

- Blurring the user's view
- Distorting the user's view
- Changing the colours of everything in the user's view
- Fading the user's view to black
- Rendering the insides of objects with highlighted edges.

For discrepancies at the hand, for instance when the user places his hand into an object, he implemented visual, haptic, and auditory effects [Hau18, p. 24-25]:

- A ghost image of the user's hand, which follows his real, tracked hand into objects and is visible inside them
- Lines connecting the tracked and physically simulated hand
- Vibrating the controller

- Playing a sound at the position of the user's hand.

In his user study, it was found that, for discrepancies at the hand, the haptic feedback provided by vibrating the controller, as well as seeing the ghost image of the user's real hands were most helpful in resolving or avoiding discrepancies, as well as increasing the participants' immersion. The sound feedback was found to be not only barely helpful, but also proved negatively impacting on immersion [Hau18, p. 29-34].
Concerning discrepancies at the head, the most obvious finding was that the distortion effect was not well received by participants, the blur effect was received best, and the colour change effect, as well as the fade to black effect, had mixed reception. Rendering the interior of objects was mainly effective at preventing users from seeing through walls, and fulfils a different role than the other effects, which also encourage the user to resolve the discrepancy. It was suggested to combine the blur and fade to black effects at lower intensity, as well as using the colour change effect more subtly, merely to notify the user about discrepancies [Hau18, p. 35-39].

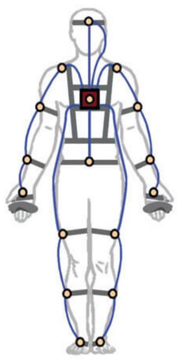
2.2 Real-Time Body Tracking in Virtual Reality using a Vive Tracker

Polona Caserman et al. in their article *Real-Time Body Tracking in Virtual Reality using a Vive Tracker* [Cas+18] evaluated currently used methods of full body tracking in VR, namely *Microsoft Kinect*[1], systems using multiple cameras with a motion capture suit such as *OptiTrack*[2], and systems using inertial measurment units (IMUs) attached to limbs, such as *PrioVR*[3] (Figure 2.1).

They found that while "the Kinect suffers from occlusion, provides noise in skeleton tracking and has a high latency" [Cas+18, p. 3] it is used in many applications for which it provides adequate accuracy due to its low cost and ease of use.
On the other hand, "a wearable motion capture suit is capable of a very accurate body tracking. However, it is very expensive and complicated to use" and as such "not applicable for home-based usage" [Cas+18, p. 3].
The authors also favourably mention systems using IMUs, without analysing latency or cost. For their research they decided to use *Vive Trackers* attached to hands and feet, with the pose of the body calculated based on the position and rotation of the trackers using inverse kinematics (IK).

[1]Microsoft Kinect: `https://developer.microsoft.com/en-us/windows/kinect`
[2]OptiTrack: `https://optitrack.com/`
[3]PrioVR: `https://yostlabs.com/priovr/`

(a) Microsoft Kinect 2 [Amo14]


(b) PrioVR diagram [Yos]

(c) OptiTrack diagram [Nat]

Figure 2.1: Full Body Tracking Systems

2.3 You Shall Not Pass: Non-Intrusive Feedback for Virtual Walls in VR Environments with Room-Scale Mapping

In their thesis *You Shall Not Pass: Non-Intrusive Feedback for Virtual Walls in VR Environments with Room-Scale Mapping* [Bol+18], a team of researchers around M. Boldt investigated methods of making the user respect virtual boundaries, such as walls in room-scale VR.
They began by analysing if and how commercial VR applications handle situations in which the user could interact with walls. Only one application, namely *Nvidia VR Funhouse*, "provides vibrotactile feedback when interacting with big game elements. However, it features neither auditory nor visual feedback." [Bol+18, p. 1]. None of the other tested games provided any sensory feedback in such situations.
Another observation was that only one tested game (*The Lab*) had walls which had two-side rendered planes, meaning that in all other games, when the camera moved through a wall, the backside of it would be invisible, allowing the user to see through it.
It was found that "most state of the art VR games either (1) stop the game progress and limit rewards or otherwise punish the player using game mechanics when crossing walls, (2) are designed in a way so the players cannot get close to the walls at all, or (3) simply avoid walls completely within the play area" [Bol+18, p. 2]. The conclusion was that most VR applications are designed to simply avoid situations in which the user can interact with walls.

The researchers continue with presenting other research into collision feedback in VR, some of which will be discussed further on.
The majority of the systems only provides haptic feedback, without visual or auditory components. They also rely on proprietary hardware, requiring "precise setup and calibration for every use" [Bol+18, p. 2] which did not suit their use case. An exception to this is the *simulated surface constraints technique*, which stops the virtual hand from penetrating the

object, introducing a discrepancy between the virtual and real hand. It is claimed that "users are more sensitive to noticing hand-object penetration" [Bol+18, p. 3] than such discrepancies.

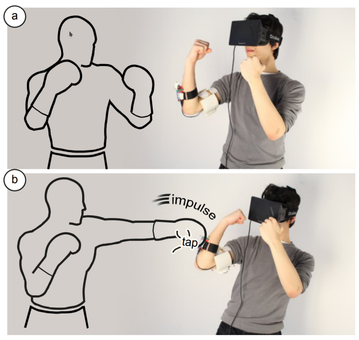
The feedback mechanisms implemented in their experiment are:

- for HMD-wall collisions: black vision as visual feedback, and muffled background music as auditory feedback;
- for controller-wall collisions: a knocking sound as auditory feedback, and vibration as tactile feedback.

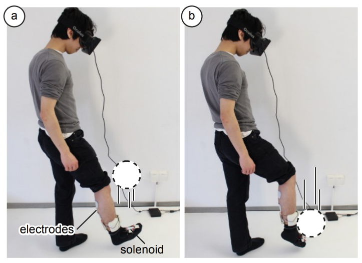
In the experiment it was found that in the control group, significantly more participants walked through walls, compared to the participants which had the feedback mechanisms enabled. It was also found that having the auditory and tactile feedback on controller-wall collisions deterred many participants from even attempting to walk through walls, as well as the visual feedback on HMD-wall collisions causing many, who did attempt to move through walls anyway, to step back and decide not to pass through [Bol+18, p. 6,7].

2.4 Impacto: Simulating Physical Impact by Combining Tactile Stimulation with Electrical Muscle Stimulation

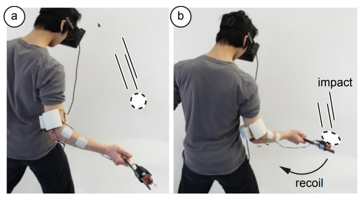
Pedro Lopes, Alexandra Ion, and Patrick Baudisch present a device they call *impacto*, "designed to render the haptic sensation of hitting and being hit in virtual reality" [LIB15, p. 1] in their paper *Impacto: Simulating Physical Impact by Combining Tactile Stimulation with Electrical Muscle Stimulation* [LIB15].



(a) Attached to arm, simulating being hit by a boxer



(b) Attached to leg, simulating kicking a ball



(c) Attached to arm and bat, simulating hitting a ball

Figure 2.2: *Impacto* in various configurations [LIB15]

The *impacto* device is wireless and can be attached to a user's arms to create the sensation of being hit by creating a tactile stimulus using a solenoid, as well as simulating impulse by causing the arm to thrust backwards due to electrical muscle stimulation (EMS) (Figure 2.2a).

Use on legs is also possible, enhancing the experience of kicking (Figure 2.2b), as well as use in combination with props, such as a bat (Figure 2.2c). Multiple *impacto* units can also be combined into a simple haptic suit.
A user study was conducted, in which it was found that each effect, EMS and the solenoid hit, yielded increased perceived realism with increasing intensity of the effect individually, but the "highest score, however, was achieved by combining both stimuli" [LIB15, p. 8].

2.5 Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation

A similar approach to that discussed under section 2.4 is taken in another paper by Pedro Lopes et al., *Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation* [Lop+17], in which EMS is used to create the sensation of heavy or immovable objects exerting force on a user's arms.
The method employed works by actuating opposing muscles in order to create a force in the direction an object would be exerting force on the user's arm, when interacting with the object. For example, picking up a heavy box requires the user's biceps to contract, so, depending on the object's weight, the user's triceps is actuated using EMS with an appropriate intensity. This same principle is used for stationary objects, such as walls.
It was however found that preventing the user's hand from penetrating an object by applying a continuous opposing force "would draw the user's attention to the EMS-actuated muscles" [Lop+17, p. 2] due to it becoming arbitrarily strong the harder the user pushed against the object. In response to this, two other approaches were implemented. The soft object design (Figure 2.3b) allows users to penetrate objects with increasing resistance, creating the impression of soft objects. The other approach is the repulsion object design, "where the EMS propels the user's hand backwards, removing it from the virtual object it is trying to touch" [Lop+17, p. 3] (Figure 2.3c).

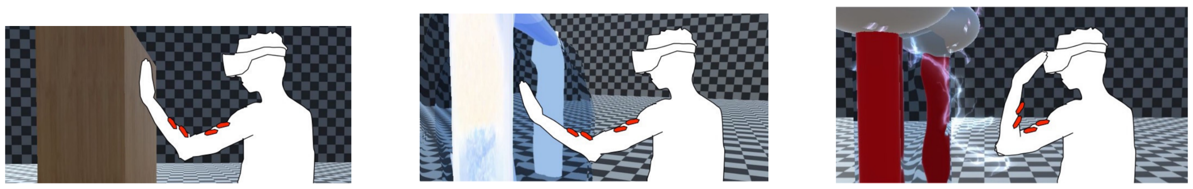

(a) Hard object design makes EMS too obvious (b) Soft object design allows users to penetrate object with increasing resistance (c) Repulsion object design repels user's hand with a short impulse

Figure 2.3: Various approaches to stop object penetration [Lop+17]

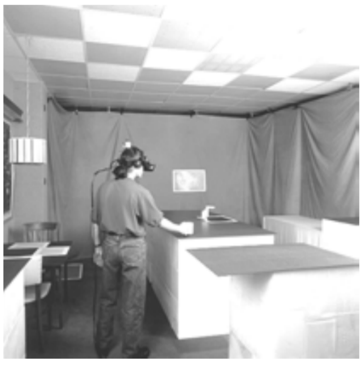
In the user study, the repulsion object design was implemented with the so called *electro* visuals. "This design complements the EMS pulse with a strong white flash which turns the screen white for 100 ms and then fades it back in in 100 ms. At the same time, users hear a

loud electrical "bang"" [Lop+17, p. 3]. In addition, a vibration motor suggests an electric shock to the user's hand. This object design was favoured by most participants, and was rated as most realistic.
The soft object design was implemented in two ways, one with visuals suggesting a magnetic field, another depicting a solid wooden wall. Of the two, the wooden version was preferred. The wall with the magnetic field however was rated as more realistic [Lop+17, p. 4]. This suggests that the visual representation matching the type of force feedback is important in regard to perceived realism.
Both the soft and the repulsion object designs were considered significantly more impermeable than control objects, which only actuated the vibration motors on user's hands, whereby impermeability "shows participants' assessment of 'this wall was able to prevent me from passing through'" [Lop+17, p. 5].

2.6 Passive Haptics

In the context of virtual reality, passive haptics (PH) is a "technique that incorporates passive physical objects into virtual environments to physically simulate the virtual objects" [Ins01, p. 9]. An example would be a sphere roughly the size of a virtual football. The user can pick up and manipulate the sphere, and the virtual football will move accordingly. In his PhD thesis, *Passive haptics significantly enhances virtual environments*, Brent Edward Insko investigates the increase in *presence* (a component of immersion) of the user in a virtual environment, if said environment is recreated to mimic the virtual one with simple props in the real world.

(a) Virtual environment


(b) Real environment with low detail props

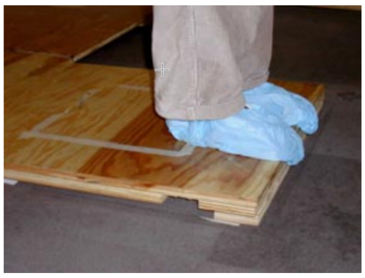

(c) Small ledge for simulating standing at a precipice

Figure 2.4: Passive haptics examples

The complexity and amount of needed props varies. At one end of the spectrum, an entire room with most furniture was recreated using cheaply made low detail props (Figures 2.4a and 2.4b), on the other end, a simple wooden ledge was used to simulate standing on a precipice of a much deeper pit (Figure 2.4c).

This kind of setup is most likely the way to achieve maximum immersion for the user, as the virtual world is essentially replicated in enough detail for adequate haptic feedback. It is however impractical for most VR applications, as it needs to be set up and registered precisely (Insko claims a misalignment of up to three inches could be tolerated by the user [Ins01, p. 63]) for each virtual environment. PH also limits the virtual environment to something that even can be replicated well enough in reality.

A similar approach is proposed in *TurkDeck: Physical Virtual Reality Based on People* [Che+15] by Lung-Pan Cheng et al., where a team of humans are constantly instructed to construct and operate props on the fly around the user, depending on where in the virtual environment he is. It is clear that such an involved solution to PH is highly impractical for most VR applications.

3 Virtual Embodiment

The term *embodiment* has been used in many different contexts, so that its meaning largely is dependent on the point of view from which the issue is approached.
In their paper *The Sense of Embodiment in Virtual Reality* [KGS12] Kilteni, Groten and Slater identified multiple differing definitions of *embodiment*. In philosophical terms, it refers to "how one defines and experiences one's self" [KGS12, p. 2]. In psychology and cognitive neuroscience, it refers to the way the body is represented in the brain, as well as how certain neurological conditions may alter this representation. In robotics, *embodiment* distinguishes how forms of artificial intelligence are represented, to contrast "virtual agents and robots that have a real physical representation compared to those that do not" [KGS12, p. 2], meaning agents which are *embodied* or not. It has also been used in the context of users' presence in virtual environments, where giving the user a virtual body in HMD based VR was identified to be "a critical contributor to the sense of being in the virtual location" [KGS12, p. 2].
In this thesis, following Haudenschild [Hau18], the term *sense of embodiment (SoE)* will be used to evaluate the various feedback mechanisms which will be implemented. Kilteni et al. use the following definition of *SoE* [KGS12, p. 2-3]:

> SoE toward a body B is the sense that emerges when B's properties are processed as if they were the properties of one's own biological body.

SoE is described as having three components:

- Sense of self-location
- Sense of agency
- Sense of body ownership

The sense of self-location refers to one's spatial experience of being inside a body. In contrast to this, presence refers to the sensation of being in an environment. An illustration of this distinction is the fact that a person can feel present in an environment, such as a room, without having a body, but will not feel self-located.
One large factor of a person's sense of self-location is visuospatial perspective, which is normally egocentric. Studies have shown "that physiological responses to a threat given to an artificial body were greater for first-person perspective than for third-person perspective" [KGS12, p. 4]. This implies that an egocentric first-person perspective increases a person's feeling of being in a body.
Another factor is matching vestibular signals between virtual and real body, meaning that the virtual body should be in a similar pose to the real body, as well as having a similar

orientation in respect to gravity [KGS12]. Trying to self-locate while standing, for example, in a virtual body lying horizontally with it's face to the ground, will be less effective than in a virtual body standing upright and facing the same direction as the real body.
Another study "revealed that the position of seen tactile stimulation when accompanied by congruent physical stimulation can dominate the visual perspective and thus determine our self-location" [KGS12, p. 4].

The sense of agency refers to the sense of having "global motor control, including the subjective experience of action, control, intention, motor selection and the conscious experience of will" [(Blanke & Metzinger, 2009, p. 7) as cited in [KGS12, p. 4]]. It is a result of predicted consequences of an action done with the real body matching the perceived results of this action on the virtual body. A simple example is performing an action for which the performer expects his arm to move up, and seeing the virtual arm move up in sync with this action. This leads to a feeling of being the agent of this action. It was found in a study that a latency of more than 150 ms between action and perceived action lead to reduced agency [KGS12, p. 5].

The sense of body ownership "refers to one's self-attribution of a body", "has a possessive character", and "implies that the body is the source of the experienced sensations" [KGS12, p. 5].
Early research showed that visual, tactile and kinesthetic stimuli should be in sync between the real and artificial body part, as well as the artificial body part having a sufficient degree of morphological similarity to the person's real body part, in order for ownership of the artificial body part to be felt [KGS12]. In newer research however, it has also been shown that "body ownership is not exclusive to artificial body parts but can also be felt for artificial whole bodies; for example, avatars or mannequins" [KGS12, p. 5].

The degree to which one experiences SoE towards a body is defined as minimal, when at least one of either the sense of self-location, agency or ownership is felt at least in minimal intensity. Full SoE is only experienced, when all three senses are felt at maximum intensity [KGS12]. It is assumed that directing full SoE towards a body which is not one's own, is not possible with current technology, however this structure allows developing and evaluating new solutions in order to increase SoE in VR applications.

The distinction between *SoE* and *embodiment* is given by Vignemont as follows: "Embodiment corresponds to a specific type of information processing, whereas the sense of embodiment corresponds to the associated phenomenology, which includes feelings of body ownership" [Vig10, p. 3]. The goal of this thesis is to find ways of increasing SoE for users in an application which *embodies* artificially intelligent actors and real humans in a virtual environment, which will be explained in detail in chapter 5.

4 Hardware

This chapter gives an overview of hardware used to implement this thesis. In addition to the basic set of *HTC Vive* hardware consisting of two base stations, a headset and two controllers, three *Vive Trackers* are used to track the user's feet and his upper torso. It is a VR system jointly developed by *HTC* and *Valve Corporation* [DS15]. The consumer version was released on the 7th of June 2016 at a price of $799 US [HTC16].

4.1 Base Stations

The base stations, also called Lighthouses, are placed diagonally apart and ideally above head height around the designated tracked area, which can be up to 3.5m by 3.5m [HTCa]. They are connected wirelessly to the computer and emit timed infrared light pulses 60 times per second, which are registered by tracked objects in order to calculate their position and pose in the tracked area [Buc15].

Figure 4.1: Base Stations, also called Lighthouses [HTCa]

4.2 Headset

The headset is a HMD with two AMOLED 3.6" panels with a resolution of 1080 by 1200 pixels each per eye, running at a refresh rate of 90Hz. It allows users to see a field of view of about 110 degrees.
There are many infrared sensors dotted around the casing, which detect infrared light pulses sent out from the base stations [Buc15]. It also includes an accelerometer and a gyroscope, as well as a front facing camera [HTCa]. Data from these sensors is combined to calculate at high precision the headset's current position within the tracked area.

The headset is tethered to the computer by a collection of cables, carrying power as well as image, audio and other data. It offers a 3.5mm headphone jack to plug headphones into, as well as a USB port to attach other peripherals.

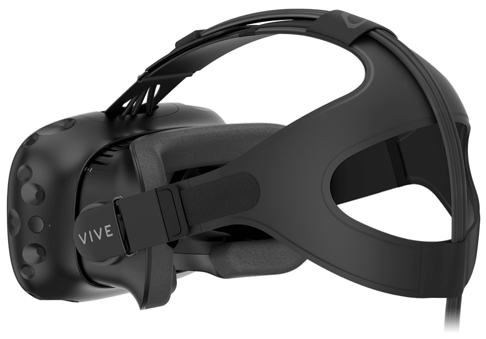


Figure 4.2: HTC Vive Headset [HTCa]

4.3 Controllers

The controllers use the same mechanisms as the headset to track their position. They are battery powered and connect wirelessly to the computer. Various input modalities are present on the controllers which can be accessed by the VR application [HTCa]:

- Multifunctional trackpad
- Grip buttons
- Dual-stage trigger
- Menu button

A vibration motor is included, which can also be controlled by the VR application, in order to, for example, provide haptic feedback.

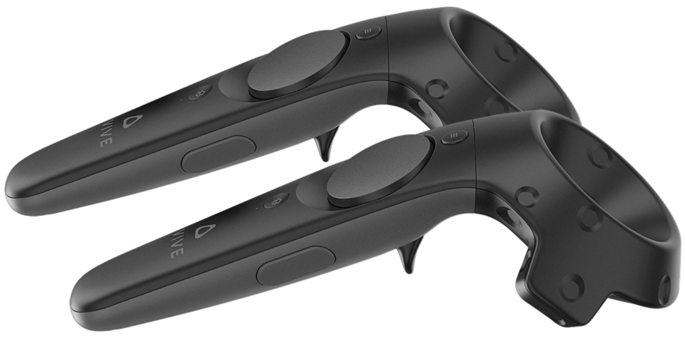
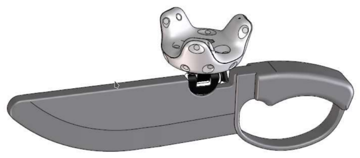


Figure 4.3: HTC Vive Controllers [HTCa]

4.4 Trackers

The *Vive Trackers* are a separately sold motion tracking accessory, intended to be attached to other physical accessories, such as guns or swords, or strapped to body parts for full body tracking. They became widely available for consumers by the end of 2017 [Ong17].
Their position is tracked the same way as the controllers and headset. They also connect wirelessly to the computer and are battery powered. No input modalities intended to be accessed by a VR application are included, however on the underside is a connector which offers 6 electrical pins, called *Pogo pins*. These can be used to communicate with the attached accessory. In the centre on the underside is a standard camera mount, which is used to fasten the tracker to physical accessories or straps [HTC17].
The *Vive Tracker* unfortunately does not include a vibration motor, relying instead on potential accessories to provide haptic feedback, however it can receive the instruction to vibrate like the controller, upon which "the POGO OUT pin sets an output of HIGH with the duration in 'ms'" [HTC17, p. 29], so as long as the accessory supports it, the *Vive Tracker* can be treated like the controller for haptic events.

(a) HTC Vive Tracker [HTCb]


(b) Mounted on Sword accessory [HTC17]

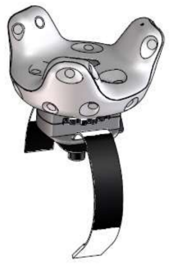

(c) Attached to straps [HTC17]

Figure 4.4: HTC Vive Tracker

5 Software

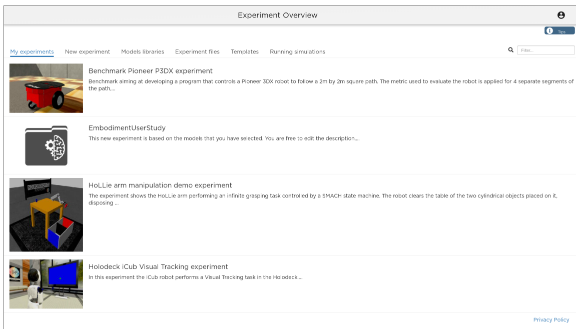
This chapter gives an overview of software used to implement this thesis.

5.1 Neurorobotics Platform

The *Neurorobotics Platform (NRP)* is a subproject of the *Human Brain Project*[1] which “allows researchers to give any simulated brain model its own 'body' — virtual or even real — and explore how it controls movement, reacts to stimulus and learns in a virtual environment” [Humb].
One application of this is to assess the fidelity of a certain brain simulation, under the assumption that a combination of body and simulated brain behaving similarly to its real counterpart would validate the brain simulation [Humb]. The project is creating a complete virtual mouse, which will have all the same sensory organs as well as bones and muscles made to mimic their counterparts in a real mouse. The aim is to be able to use this mouse in virtual experiments, with the added benefit of monitoring the simulated brain in real-time for analysis, as well as to perform experiments which are impractical in the real world due to their scope, since practically all data of the simulation can be recorded and the simulation sped up by orders of magnitude [Humc]. Another intended use case is for the development of robots. The *NRP* allows robots to be quickly and reliably created and tested in a virtual environment, prior to an actual prototype being been built [Humb].

The backend of the *NRP* is intended to be run independently from the frontend, for example on a high performance server. The user connects to it via a web frontend, which is called the *web cockpit* (see 5.1a), with a supported browser, where he can choose from a list of premade experiments, or create his own. Once a simulation is started, the user is presented with the *simulation view* (see 5.1b) in which he can pause, inspect and manipulate the experiment. The camera can be freely moved, objects can be teleported elsewhere or have forces applied to them interactively, and the state of the experiment, for example visual input the robot recieves, can be viewed in real time.
Only the graphics are rendered locally on the user's machine, the heavy computations of the rest of the simulation are all done by the backend, potentially on another machine.

[1]The Human Brain Project is a large 10 year scientific project funded by the European Union “building a research infrastructure to help advance neuroscience, medicine and computing” [Huma]


Experiment Overview

Tips

My experiments | New experiment | Models libraries | Experiment files | Templates | Running simulations

Filter...

Benchmark Pioneer P3DX experiment
Benchmark aiming at developing a program that controls a Pioneer 3DX robot to follow a 2m by 2m square path. The metric used to evaluate the robot is applied for 4 separate segments of the path,...

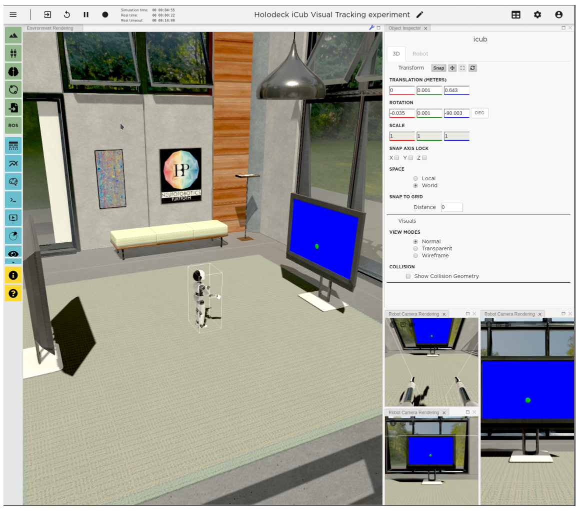
EmbodimentUserStudy
This new experiment is based on the models that you have selected. You are free to edit the description....

HoLLie arm manipulation demo experiment
The experiment shows the HoLLie arm performing an infinite grasping task controlled by a SMACH state machine. The robot clears the table of the two cylindrical objects placed on it, disposing ...

Holodeck iCub Visual Tracking experiment
In this experiment the iCub robot performs a Visual Tracking task in the Holodeck....

Privacy Policy

(a) NRP Web Cockpit


Simulation time: 00 00:04:55
Real time: 00 00:09:22
Real timeout: 00 00:14:00

Holodeck iCub Visual Tracking experiment

Environment Rendering

Object Inspector

icub

3D | Robot

Transform: Snap

TRANSLATION (METERS): 0 | 0.001 | 0.643

ROTATION: -0.035 | 0.001 | -90.003 DEG

SCALE: 1 | 1 | 1

SNAP AXIS LOCK: X Y Z

SPACE: Local [✓ World]

SNAP TO GRID: Distance 0

Visuals

VIEW MODES: [✓ Normal] Transparent Wireframe

COLLISION: Show Collision Geometry

Robot Camera Rendering

Robot Camera Rendering

Robot Camera Rendering

(b) NRP Simulation View

Figure 5.1: NRP Frontend

5.2 Unity3D with SteamVR

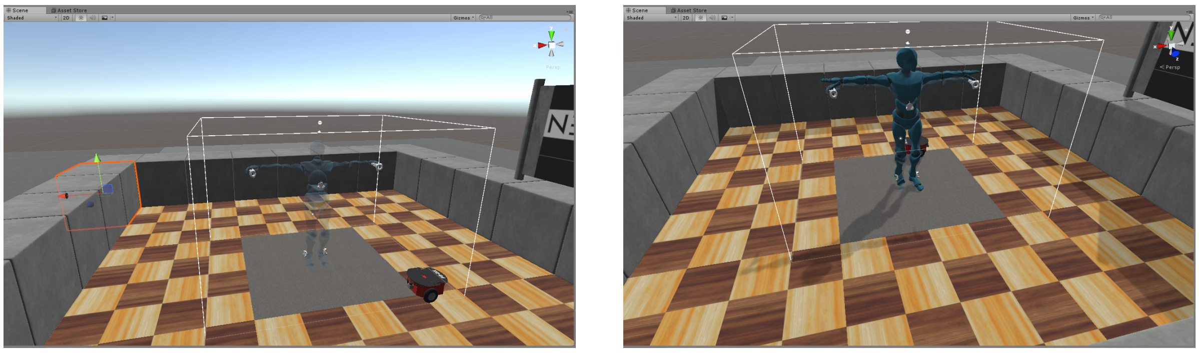
Unity3D is a cross-platform game engine that was originally launched exclusively for *Mac OS-X* in 2005 [Axo16], but has since been expanded to support most desktop operating systems, consoles, mobile devices, web browsers, and most VR and augmented reality (AR) platforms, a total of more than 25 platforms. It is continually being developed by *Unity Technologies*, and is available gratis for personal use [Uni].
More than half of all mobile games, as well as 60% of VR and AR content is created with *Unity3D* [Bon18].

SteamVR is a VR platform developed by *Valve Inc.* originally for the *HTC Vive* (see chapter 4) that implements the *OpenVR* application programming interface (API) "that acts as the interface between VR hardware and software", which makes it compatible with any headset that supports *OpenVR* [Lan19].
SteamVR is available as a plugin for *Unity3D*, allowing for easy integration of *HTC Vive* hardware into any project.

5.3 Neurorobotics Unity3D Client

The *Neurorobotics Unity3D Client* (referred to as client from here on) is a VR client for the NRP. At the time of writing, it connects to a running simulation on the NRP, from which it receives the entire scene information, meaning the position of all objects in the simulation. The client does not handle any of the physics calculations, the client only gets updated scene information from the NRP instance to which it is connected.

In addition to displaying the simulation, the client can spawn a humanoid body for the user (referred to as avatar from here on).
This initially takes the shape of a translucent avatar, which follows the tracked position of the user without colliding with any objects of the simulation on the NRP, and is referred to as the *local* avatar. The local avatar is only present in the client. It is implemented using *Unity3D*'s animation system, which includes special features for humanoid characters. Target positions for hands, feet, head and body can be set, after which the animation system will calculate a joint configuration to match these target positions using IK, and animate the avatar according to changing targets.
A corresponding avatar is spawned on the NRP instance, appearing as an opaque copy of the local avatar. This *remote* avatar is a robot, which will try to achieve the same pose as the local avatar, but while doing so will be constrained by the physics of the NRP.



(a) Translucent local avatar only

(b) Opaque remote avatar spawned in

Figure 5.2: Simulation as seen in Unity3D client, with local and remote avatars

The base state of the client was easily extendable to include full body tracking by adding three *Vive Trackers* (see chapter 4) to the scene, and assigning them to the corresponding animation targets of the local avatar.
With this modification, the user is able to navigate the play space (shown as a box with white borders) in the virtual environment on the NRP with a fully tracked body.

6 Proposed Work

In the course of this thesis, feedback mechanisms to alert the user of discrepancies between his tracked body, the *local* avatar, and his physically simulated body, the *remote* avatar, will be implemented into the *Neurorobotics Unity3D Client* (see section 5.3). The aim is to encourage the user to prevent, and resolve, such discrepancies, to increase his SoE (see chapter 3).
Although the focus of this thesis is on the addition of feet, feedback mechanisms for hands and head will also be implemented. These mechanisms will be discussed further in chapter 7, and will include visual, auditory and haptic feedback.
Due to the *Neurorobotics Unity3D Client* already having a seperate local and remote avatar implemented, the *simulated surface constraints technique* mentioned in section 2.3 is already present. The user can place his hand inside an object, but the remote, physically simulated avatar's hand will not follow into the object.
This will be used as the baseline experience to which the newly added feedback mechanisms will be compared in a user study and evaluation, discussed in chapter 8.

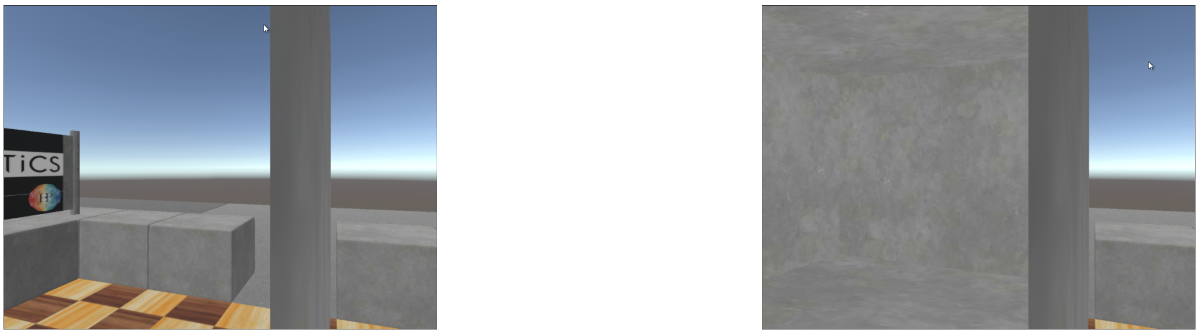
7 Implementation

This chapter will discuss, in detail, the added feedback mechanisms and how they were implemented into the *Neurorobotics Unity3D Client*. For the more technical explanations it is assumed the reader has a basic grasp of CG and the workings of *Unity3D*.

7.1 List of Implemented Feedback Mechanisms

All listed effects can be disabled or enabled individually, most even at runtime.

- **Interior rendering for objects:** (see Figure 7.1)
  Usually in 3D graphics, only the front facing sides of an object are rendered for performance reasons. The result is that when the camera is inside an object, it will only see the back facing sides of an object, which are not rendered, making them invisible. To correct this, all back facing sides of an object will show the same texture as the front facing sides. The effect of this is that a camera within an object will simply see it inside out, so that a user will be unable to see through walls as easily, and thereby keep his orientation within the object. It is hoped that this will provide a more immersive representation of what being inside of a solid object would look like.
  This feedback mechanism is only useful for discrepancies at the head.



(a) Interior not rendered (b) Interior rendered

Figure 7.1: Camera half inside an object. When interior is not rendered, all back facing polygons of the object become invisible

- **Avatar silhouette visible inside of objects:** (see Figure 7.2)
  When the user, for example, steps into an object, the local avatar will follow his tracked foot, however the remote avatar will try to follow, but be stopped from penetrating the

object. This will result in the user not being able to see where his real foot is, as he will only see the remote foot in front of the object. Making the silhouette of the local avatar visible, when inside of objects, will allow the user to see exactly where his real body is, while simultaneously seeing where it should be (where the simulated body is).
This should allow the user to keep his orientation as he will never be confused as to where his real limbs are. Due to the way this feature is implemented, the silhouette is only visible as a solid colour. This makes it easy to see when one is inside of an object, but it most likely will have a negative effect on immersion.
This effect is applied to the entire body and is not dependent on any discrepancies being detected.

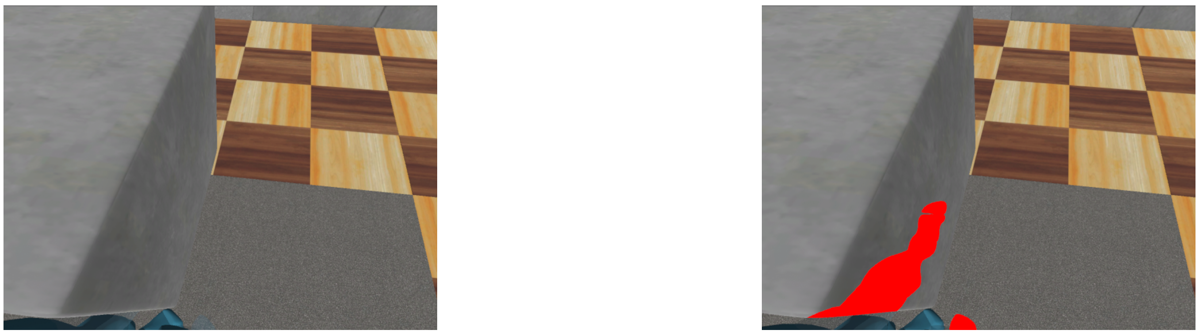
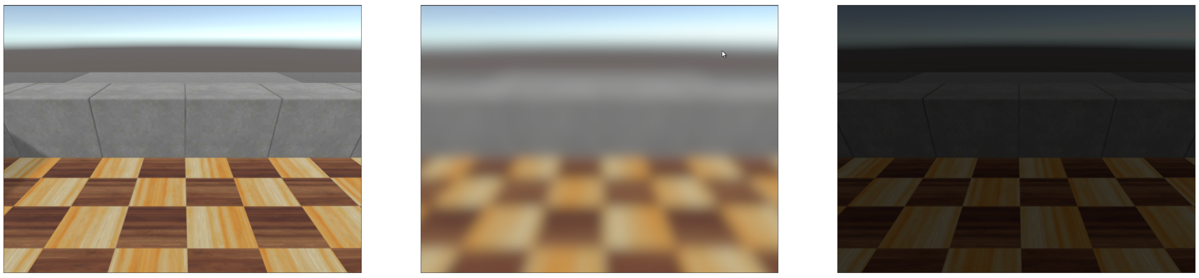

(a) Avatar silhouette not visible in objects (b) Avatar silhouette visible in objects

Figure 7.2: Local avatar's left leg partially inside of an object

- **Blur effect:** (see Figure 7.3b)
  When the head of the local avatar moves away from that of the remote avatar, the user's view will progressively be blurred, for example if his head is inside a low ceiling. This allows the user to keep his orientation, while restricting his view, to alert him to the discrepancy. It should make the user uncomfortable enough to wish to avoid such discrepancies. As blurry vision is something that humans experience in real life, it is anticipated that this effect will have a low impact on immersion.
  This feedback mechanism could also be used for discrepancies at hands and feet, however it was deemed unhelpful during early testing for these purposes, as it restricts the user's view, impeding him from realising where the discrepancy is. It is only used for discrepancies at the head.

- **Fade to black effect:** (see Figure 7.3c)
  When the head of the local avatar moves away from that of the remote avatar, the user's view will progressively fade to black. This is another way of preventing the user from seeing through walls, as he will simply be unable to see anything once the discrepancy is large enough. It has a negative impact on the user's orientation, and should make the user uncomfortable to encourage him to avoid such discrepancies. Vision fading can be likened to humans closing their eyes, and thus this should be a fairly immersive effect.

This mechanism is also possible to use for other discrepancies, but is also deemed unhelpful, like the blur effect. It is only used for discrepancies at the head.



(a) Normal view (b) Blur effect (c) Fade to black effect

Figure 7.3: Blur and Fade to black effects

- **Line effect:** (see Figure 7.4)
  When a local and remote body part (hand or foot) are too far apart, a line is drawn between them, whereby the thickness of the line increases with the distance. The line is thicker on the local end, allowing the user to see which side is which. The main effect of the lines is to show the user which limbs belong together, increasing his sense of orientation towards the remote body. The line effect will probably have a negative impact on immersion, due to it looking unnatural.
  This mechanism also could be used for discrepancies at the head, but there it would not be helpful, since the user cannot see the line there anyway. Therefore it is implemented for hands and feet only.

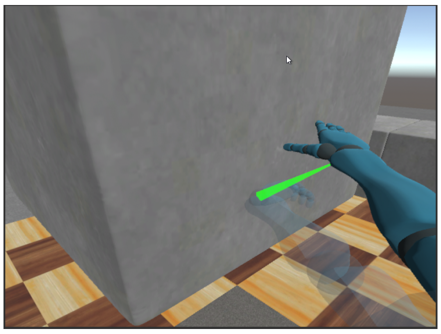


Figure 7.4: Line effect

- **Haptic effect:**
  When a discrepancy occurs at a hand, depending on the severity and duration of the discrepancy, the corresponding controller will vibrate in pulses. The duration of one pulse is longer, the larger the discrepancy, and the frequency of pulses increases with increasing duration of the discrepancy. This should give the user an increasing sense of urgency to deal with the discrepancy the larger and longer it is, while being subtle for

small and short discrepancies.
Although the *Vive Trackers*, attached to the user's feet, do not have vibration motors of their own, the same vibration instruction can be sent to them, where an external device could be configured to vibrate like the controllers (see chapter 4). This has been implemented, but was not tested due to there currently being no such devices commercially available yet.
Currently this effect only works on discrepancies at the hand.

- **Geiger sound effect:**
For discrepancies at hands and feet, the Geiger sound effect mimics a Geiger counter, whereby an increase in severity of the discrepancy increases the frequency of clicks. These clicks can be configured to either play from the position of the local or the remote body part. This allows the user to locate the discrepancy without having to see it. The reason for making this effect sound like a Geiger counter is that most people are familiar with its sound as an indicator of danger from video games and movies. It also is a sound based in reality, in order to have a lesser negative effect on immersion, but is not a pleasant auditory tone, which instils a sense of urgency in the user to deal with this discrepancy.
This effect was not implemented for discrepancies at the head, as the biggest advantage, being able to notice the discrepancy without seeing it, does not apply there.

- **Noise sound effect:**
Implemented similarly to the Geiger sound effect, this effect simply plays looping static noise from the affected limb when a discrepancy is detected. The volume of this static noise increases with the severity of the discrepancy. As this sound is constant, it is intended to be less distracting than the irregular clicking of the Geiger sound effect.

- **Discrepancy Indicators:** (see Figure 7.5)
The discrepancy indicators are little red arrows which rotate around the centre of the user's view to point at a detected discrepancy. It is configurable if these indicators are only visible for discrepancies that are not in the user's view, as well as whether they point at the local or remote limb.
This allows the user to be notified visually of discrepancies not in his view, while also indicating where they are occurring. People not used to head-up displays (HUDs) will probably find these indicators distracting, but people acquainted with HUDs from video games will probably not experience a negative impact on immersion. This effect was not implemented for discrepancies at the head, as the way the indicators rotate to point to the discrepancies, leads to confusing results, if the discrepancy is very close to the camera, and it was deemed unnecessary to add another visual feedback mechanism for head discrepancies.

- **Colour shift effect:** (see Figure 7.6b)
When a discrepancy is detected, depending on its severity, the hue of all pixels in the user's view is shifted. The effect this has is that colours of all objects change, whereby

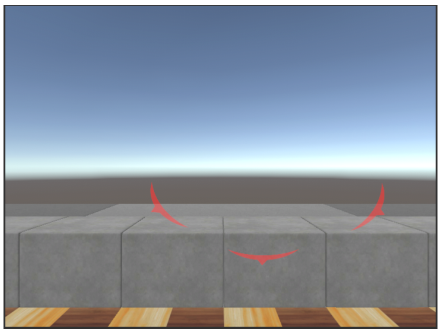
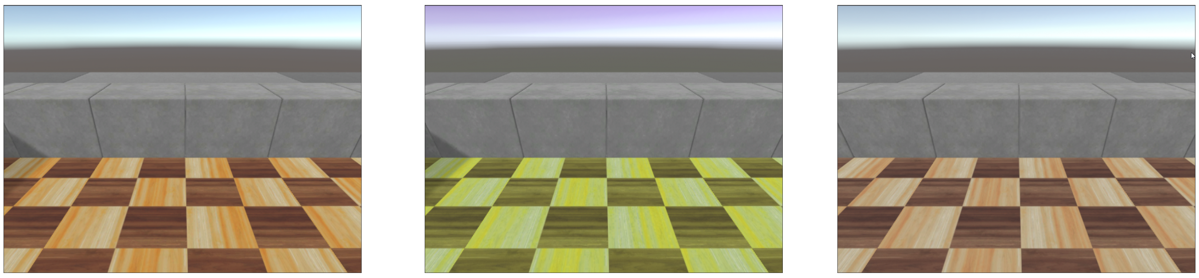


Figure 7.5: Discrepancy indicators pointing at left hand, left foot and right hand

the change is more drastic, the larger the discrepancy is. It is intended to give the user an increasing sense of there being something wrong, while not inhibiting his vision. Unfortunately, this effect does not indicate where a discrepancy occurs.
This effect is implemented for discrepancies at head, hands and feet, however is probably not as useful, when used universally.

- **Desaturation effect:** (see Figure 7.6c)
  Similar to the colour shift effect, the desaturation effect lowers the saturation of all pixels in the user's view, depending on the severity of the detected discrepancy. It is more subtle and is used frequently in the context of video games, where it often indicates low health of the player's character.
  This effect is implemented for discrepancies at head, hands and feet, however is probably not as useful when used universally.



(a) Normal view (b) Colour shift effect (c) Desaturation effect

Figure 7.6: Normal view, colour shift and desaturation effects

7.2 Implementation Details for Feedback Mechanisms

During the implementation phase, the aim was to create a largely self contained package that could be used in another project with little modification, as well as leaving most existing scripts and functionality of the *Neurorobotics Unity3D Client* unchanged. All code can be found in the *HBPNeurorobotics/Unity3D-Client* repository in the *VirtualEmbodiment* branch [1] in the `Assets/EmbodimentDiscrepancy` directory.

The `DiscrepancyTracker` takes references to the `Transform` components of all tracked limbs (head, left hand, right hand, left foot, right foot) of the local avatar. Once the remote avatar has been spawned, the corresponding objects to each tracked limb are searched for on the remote avatar.
Then, in each frame, the distance between corresponding body parts of the local and remote avatar is computed, and, if greater than a configurable global tolerance, a `Discrepancy` object is created, which contains the tracked limb, its local position, its remote position, the distance, and the duration of the discrepancy. This object is sent to the `DiscrepancyHandler`.

The `DiscrepancyHandler` contains references to all objects which are responsible for the feedback mechanisms. It also stores the configuration for which effect is enabled for which tracked limb, as well as configurable tolerance timers, before a discrepancy is handled, for each tracked limb.
For each `Discrepancy` object sent to it by the `DiscrepancyTracker` every frame, it is decided which feedback mechanism will be applied for it, based on the configuration. The `Discrepancy` object is sent on to the designated handler.
The way these feedback mechanisms are implemented in their handlers will be explained in the following. Apart from the rendering of object interiors and the avatar silhouette inside objects, whose components need to be enabled or disabled before runtime, all effects can be toggled at runtime in the `DiscrepancyHandler`.

7.2.1 Interior rendering for objects

In order to render the insides of objects, a shader posted on the *Unity* forums by user Tom163[2], which simply displays a texture without any lighting calculations, was adapted to only show the texture on the back facing sides of an object. A material using this shader is added to every object in the NRP scene, except the remote avatar.
As all NRP objects are created at runtime, the material has to be added to them at runtime as well. In order to do this, the `modelsParent` variable of the `GazeboSceneManager` was exposed as a read-only property. It is the parent object of all objects synced from the NRP simulation. For all its child objects with a `MeshRenderer` component, an instance of the interior material using the same texture as the existing material, is added to the renderer.
This is implemented in the `InteriorRenderingHandler` script.

[1] `https://github.com/HBPNeurorobotics/Unity3D-Client/tree/VirtualEmbodiment`,
mirrored here: `https://github.com/VoodaGod/Neurorobotics_Unity3D-Client/tree/VirtualEmbodiment`
[2] `https://forum.unity.com/threads/what-simple-shader-only-shows-diffuse-without-lighting.41054/#post-262044`

7.2.2 Avatar silhouette visible inside of objects

In order to make the local avatar visible inside of objects, a shader shown in a *YouTube* video by user N3K EN[3] was adapted to only draw a configurable solid colour when an object is between it and the camera. A material using this silhouette shader is added to the local avatar's `SkinnedMeshRenderers`.
A problem that arises, is that now the local avatar also constantly shines through the remote avatar. To solve this problem, a stencil test is added to the silhouette shader, whereby no colour is drawn when the configured stencil is set. The remote avatar receives a material using a shader which sets the stencil buffer to the configured value.
The result is, that if any object is between the camera and the local avatar, the avatar's silhouette will shine through, but not through the remote avatar. This allows the user to still clearly see the remote avatar.
This is implemented in the `UserAvatarSilhouetteHandler` script.

7.2.3 Blur effect

The blur effect is implemented in the `DiscrepancyHeadEffects` script using SuperBlur by user PavelDoGreat[4]. A `Discrepancy` object is received from the `DiscrepancyHandler` in every frame the effect should be active. The intensity of blurring is dependent on the distance of the discrepancy, and the configured `maxDistToFullBlur`. The amount of blur is smoothly increased when the distance jumps suddenly up from zero, for example when the tolerance timer was reached.

7.2.4 Fade to black effect

This effect is implemented similarly to the blur effect, also in the `DiscrepancyHeadEffects` script. It uses *SteamVR*'s `SteamVR_Fade` functionality, by setting it to black with an increasing alpha value. The amount of darkening is dependent on the configured `maxDistToBlack` and the distance of the discrepancy. The darkening is also smoothed like in the blur effect.

7.2.5 Line effect

The `DiscrepancyLineHandler` receives a `Discrepancy` object from the `DiscrepancyHandler` in every frame for every tracked limb for which a discrepancy was detected.
If not existing, a `DiscrepancyLine` object is created for the tracked limb, and this object draws a line between the local and remote body part, with the thickness of the line depending on the distance of the discrepancy. The thickness can be modified by changing the `thicknessMultiplier`.
The appearance of the lines can be changed by adjusting the `DiscrepancyLinePrefab`.

[3]`https://www.youtube.com/watch?v=EthjeNeNTsM`
[4]`https://github.com/PavelDoGreat/Super-Blur`

7.2.6 Haptic effect

The haptic effect is implemented in the `DiscrepancyHapticHandler` script, which has references to `SteamVR_TrackedObjects` (the controllers and *Vive Trackers*), which need to be set correctly at run time.
Every time a discrepancy needs to be handled (also received every frame from the `DiscrepancyHandler`), the length of the next haptic pulse, as well as the time until the next pulse, is computed for the former by the ratio of distance of the discrepancy to `maxRumblePulseDistance` and `maxRumblePulseDuration`, and for the latter by the ratio of duration of the discrepancy to `maxRumblesTime` and `maxRumblesPerSecond`. These factors are configurable. To allow for longer vibrations, a coroutine posted by user mptp on the *Steam* forums[5] was adapted.
The result is longer pulses, the more severe a discrepancy is, and more frequent pulses, the longer a discrepancy has endured.

7.2.7 Geiger sound effect

This sound effect is implemented in the `DiscrepancySoundHandler` script. It has a collection of sound clips of Geiger counter clicks, which were taken from a *YouTube* video by user Ryhindor[6]. `Discrepancy` objects are received from the `DiscrepancyHandler` every frame a discrepancy should be handled.
A `GeigerAudioSourcePrefab` is positioned, as configured, either at the local avatar's limb or the remote avatar's. The intensity of the clicking is calculated with the ratio of distance of the discrepancy to `geigerDistanceToMaxIntensity` and `geigerMaxClicksPerMinute`. A random sound clip is played by the corresponding audiosource, as well as the time between clicks being influenced by `geigerVariance`.
The result is more frequent clicking, the larger the discrepancy is. The frequency has some random variance added to make it sound more like a Geiger counter. As the sound is played from either the local or remote limb, the user can identify the position of the discrepancy without seeing it.

7.2.8 Noise sound effect

The noise sound effect is also implemented in the `DiscrepancySoundHandler`, similarly to the Geiger sound effect. The main difference is that the noise sound, taken from a *YouTube* video by user dalesnale[7], is continous. Only the volume of the sound is changed, depending on the distance of the discrepancy and `noiseDistanceToMaxVolume`.
The result is louder noise coming from either the local or remote limb, the larger the discrepancy.

[5] `https://steamcommunity.com/app/358720/discussions/0/405693392914144440/#c357284767229628161`
[6] `https://youtu.be/upPiJ9vOYiY`
[7] `https://www.youtube.com/watch?v=8SHf6wmX5MU`

7.2.9 Discrepancy indicators

The `DiscrepancyIndicatorHandler` instantiates a `Canvas` as a child of the main camera. As screen-space canvases are discouraged in VR application due to them being hard for the user to focus on, it is a world-space canvas, placed fairly close to the near plane of the main camera.
On this canvas, for each tracked limb for which a discrepancy needs to be handled, a `DiscrepancyIndicator` is instantiated. If, in a given frame, a `Discrepancy` object for a tracked limb is received, the corresponding `DiscrepancyIndicator` is enabled, and it's rotation set to point, depending on the `pointsAt` setting, at the local or remote body part.
When the `hideIndicatorWhenTargetInView` setting is enabled, the indicator will only be visible, if the target it is pointing to is not in view.

7.2.10 Colour shift effect

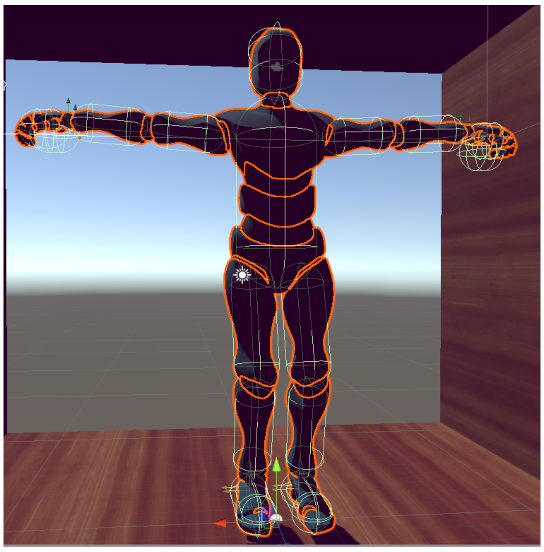
This effect is implemented in the `DiscrepancyPostProcessingEffects` script. It uses *Unity3D*'s built in post processing stack. The necessary objects are instantiated at run-time. For all `Discrepancy` objects received in a frame, the one with the largest distance is used for calculating the amount of colour shift. The amount of shift is also influenced by `distToFullColorShift` and `maxShiftDegrees`, whereby shift degrees refers to how far around the colour wheel a colour should be rotated. The calculated value is set in the `ColorGrading/hueShift` settings of the `PostProcessProfile`.
The amount of colour shift is also smoothed for sudden jumps up from zero, for example when the tolerance timer is reached.

7.2.11 Desaturation effect

The Desaturation effect is implemented like the colour shift effect, it just uses the `ColorGrading/saturation` setting of the `PostProcessProfile`.

8 Evaluation

In order to evaluate the benefit of using the aforementioned feedback mechanisms in regards to the embodiment of feet, a user study was performed. For this user study, a suitable test environment had to be built to showcase these feedback mechanisms.
During early testing it was found that, in its current state, the remote avatar in the NRP simulation did not manage to follow the local avatar at an acceptable speed. The result was that, even for any reasonable tolerance timer setting, discrepancies would constantly be detected for any limb that moved, due to the remote avatar taking too long to move to the target position, and often overshooting it once it caught up. To get a semblance of intended behaviour, it was necessary to move in slow motion.
As the NRP and the *Neurorobotics Unity3D Client* are undergoing constant development, it is assumed that this should work in real time in the future.
As an alternative, a similarly behaving second avatar with colliders was implemented, taking over the role of the remote avatar, which is connected to the local avatar with spring joints (see Figure 8.1a). This resulted in behaviour which closely mimics the intended behaviour of the avatar in the NRP simulation, so the findings in this evaluation should be applicable to using the *Neurorobotics Unity3D Client* with a NRP instance. Following this, the test environment could be built fully in *Unity*, utilising the built in physics system.


(a) Second avatar with colliders attached

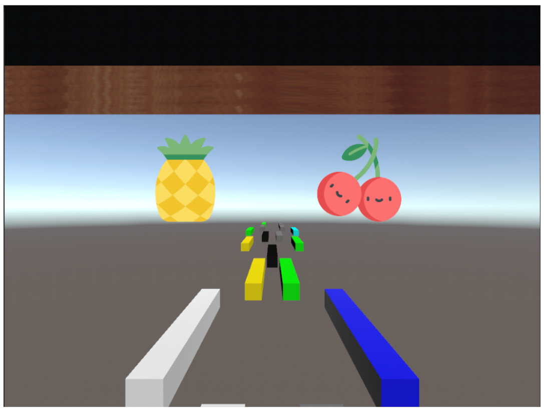

(b) Plank test environment

Figure 8.1: Avatar setup and initial plank test environment

Initially, as a test environment, a little game was implemented, which placed the user in a small hut, with random constellations of planks, endlessly spawned, moving towards him (see Figure 8.1b). The idea was that, due to these planks always being in different positions, this would invariably lead to discrepancies at the user's feet, when they collided with him. In order to distract the user from dodging the planks, two canvases displaying changing, random images floated in front of the hut. If a watermelon appeared on the left canvas, the user was to squeeze the controller in his left hand, if it appeared on the right, vice versa. Successfully squeezing the correct controller was rewarded with a little success icon in between the two random images.
During testing, however, it was found that, when the planks moved fast enough for the user to be hit by them, the discrepancies did not last long enough for any of the feedback mechanisms to be noticed sufficiently, especially when things got hectic. Slowing the planks down, made it too easy to dodge them, which again provided no insight into the usefulness of the feedback mechanisms.

Another test environment was then created, placing the user in a small room with a protruding wall, behind which is a hole (see Figure 8.2). The user is asked to face the yellow wall, and try to move a ball, which is spawned in front of that wall, out of the room, through the hole. When the user walks over to the ball, a platform slides out of the wall and knocks the user's (virtual) feet out from under them, creating a discrepancy at his feet. If the user resolves this discrepancy by moving away from the yellow wall towards his remote feet, the wall will be slid back, and the user can attempt to move the ball out of the room by kicking and shoving it with his feet.
The toggling of the various effects, as well as the sliding platform and ball spawning is done manually at runtime using the *Unity* editor.

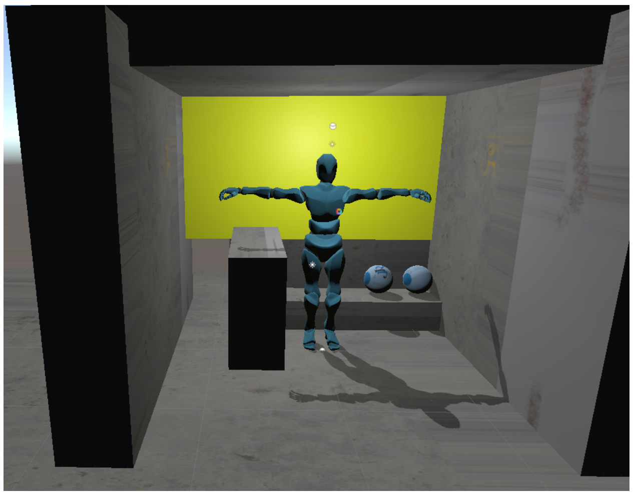


Figure 8.2: Final user study test environment

8.1 User Study

For the user study, eight test subjects were recruited during the course of a day with the promise of snacks and drinks for taking part. Of these test subjects, all but one were in the age bracket 18-30. More than half of the test subjects were female, and 75 percent had no prior experience with VR.
The user study had the following procedure for each tested feedback mechanism:

- Set up currently tested feedback mechanism **X**
- Ask subject to face the yellow wall
- Spawn a ball
- Ask subject to move a ball out of the room
- When close enough, slide in the platform to cause a discrepancy
- Let subject resolve the discrepancy
- Remove platform
- Let subject move a ball out of the room with his feet
- Ask subject questions and fill out questionnaire with the answers:
  - "How connected to your virtual body did you feel with **X**?" on a scale from 1 (low connection) to 5 (high connection)
  - "How much did **X** help you notice and correct discrepancies?" on a scale from 1 (hardly) to 5 (a lot)
  - "How distracting was **X**?" on a scale from 1 (hardly) to 5 (very)
- Discuss uncertainties about the questions and additional comments by the subject

The tested feedback mechanisms for discrepancies at the feet were:

- Ghost body only as a baseline
- Avatar silhouette visible inside of objects
- Line effect
- Geiger sound effect
- Noise sound effect
- Discrepancy indicators
- Colour shift effect
- Desaturation effect

See section 7.1 how these work.

8.2 Results

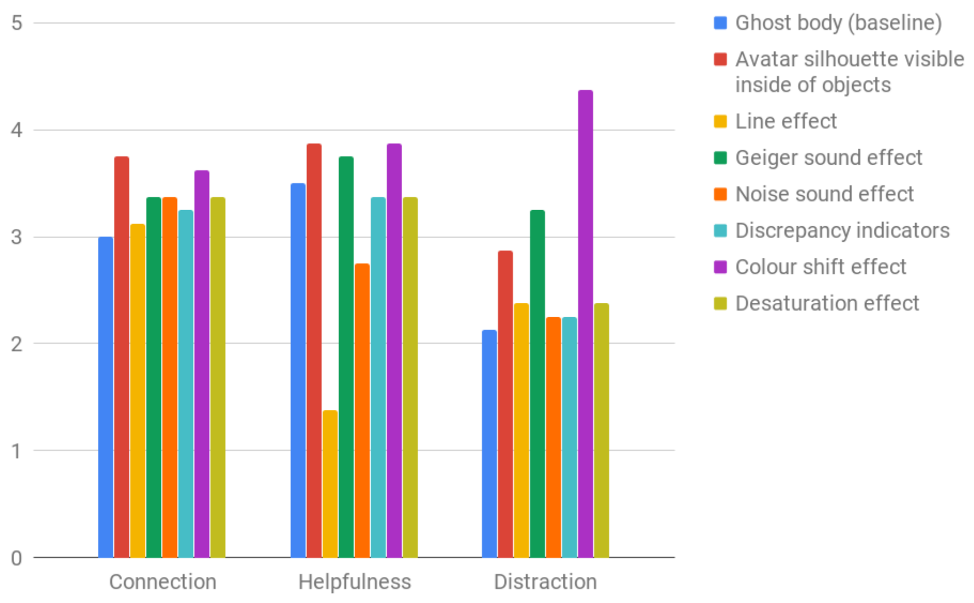


Figure 8.3: Averaged answers for all tested feedback mechanisms

As can be seen in Figure 8.3, the amount of connection to the virtual body felt by the test subjects (indicating their SoE) did not differ by a large margin from the baseline of just having a ghost body in a similar pose to their real body. The reason for this is most likely that the ghost body is active in all effects, and already provides a large benefit to SoE. A notable exception is having the silhouette of this ghost body visible inside of objects, which almost increased the connection by a full point. The least impactful addition was the line effect, with barely any increase over the baseline.
Looking at the helpfulness of these effects in noticing and correcting discrepancies, a much larger variance between effects can be seen. The ghost body baseline already has a fairly good rating, which is increased again by seeing it inside of objects. Of the two auditory feedback mechanisms, the Geiger sound effect is rated significantly more helpful. The colour shift effect is also rated comparatively high. The least helpful by a large margin is the line effect, which is most likely because the lines are hard to see down by the feet, and often are occluded by the user's body.
When considering how distracting (as in having a negative impact on immersion) these feedback mechanisms are, it can be seen that the most helpful effects are simultaneously the most distracting. This is unfortunate, but understandable, as the noise sound effect, desaturation effect, and line effect can be too subtle to prompt the user to take action, whereas the colour shift effect, Geiger sound effect and the avatar silhouette visible inside of objects are a lot more obvious. An exception to this pattern is the discrepancy indicators, which rate

fairly high on helpfulness, but low on distraction. This makes sense when only looking at discrepancies at the feet, since this effect is good for alerting the user to discrepancies which are out of view, as feet usually are.

It was suggested by a test subject that the colour shift effect could be modified to tint the entire view red as opposed to shifting all colours, under the assumption that most people would understand red as a warning sign. Another suggestion was to somehow make the appearance of the avatar silhouette in objects less jarring, in order to make it feel more part of the world. The noise sound effect was felt likely to blend in to background noise. Most subjects suggested combining multiple effects.

The conclusion drawn is that most feedback mechanisms worked as intended for discrepancies at the feet, apart from the line effect, which in Haudenschild's work also did not offer much benefit for discrepancies at the hands [Hau18, p. 32].

9 Conclusion

In conclusion it can be said that using the feedback mechanisms for discrepancies at the feet, outlined and evaluated in this thesis, have a positive impact on the quality of embodiment in the NRP. However there is still much work to be done in this field. This thesis focused on methods easily implementable with software and hardware that is widely available to anyone with access to a VR setup. In chapter 2, some other approaches were discussed, which did not have this limitation, however most of them are fairly inflexible and difficult to set up. It is to be hoped that more research into this field will lead to more generally applicable commercially available hardware to supplement non-intrusive feedback mechanisms such as those proposed in this thesis.

9.1 Future Work

In order to find out which feedback mechanisms should be implemented in any project, a more in-depth user study should be carried out. It should not only focus on the embodiment of feet, but the entire body. Many potential combinations of effects should be evaluated.
Ways of adding haptic feedback to the *Vive Trackers* should be investigated, as haptic feedback is generally agreed to be very effective and immersive.
Integrating hardware such as *Impacto* (see section 2.4) using EMS also is an approach that should be investigated further.

9.2 Final Words

As the *Neurorobotücts Unity3D Client* matures, I hope some of the feedback mechanisms implemented during the course of this thesis will prove useful. Adapting the solution to only contain the wanted mechanisms should be trivial.
The user study was very bare-bones, due to the focus on discrepancies at the feet. It was hard to find ways to encourage the user to do things he should not be doing, such as step into objects, for long enough that any feedback mechanisms would apply before the user stepped out again. Another problem turned out to be that the test subjects had differing understandings of the questions, requiring verbal rephrasing and discussion to make sure that the question had its intended effect.
Due to the time constraints of a Bachelor's Thesis it also was not possible to do a more in-depth user study to evaluate sensible combinations of the implemented effects.

All in all, I am happy with the way this thesis turned out, as almost all implemented feedback mechanisms worked as intended and were received well.

List of Figures

Glossary

controller in the context of VR and gaming refers to the input device. 3, 6, 12, 13

immersion In video games, immersion refers to ”the degree of involvement with a game” [BA04, p. 2]. 1, 2, 4, 8, 9, 20–22, 31

inverse kinematics Inverse kinematics (IK) makes it possible, just from knowing the target position of, for example, a hand attached to an arm with multiple joints, to calculate the required joint angles to place the hand at its target position. This is used in 3D animation to create realistic movement of game characters [Cas+18, p. 3f]. 4, 37

kinesthesia Also referred to as proprioception, “the sensation of body position and movement”[TA18]. 2

Neurorobotics Platform see section 5.1. 15, 37

passive haptics see section 2.6. 8, 37

sense of embodiment see chapter 3. 10, 37

Acronyms

API application programming interface. 16

AR augmented reality. 16

CG computer graphics. 1, 19, 35

EMS electrical muscle stimulation. 6, 7, 33

HMD head mounted display. 3, 6, 10, 12

HUD head-up display. 22

IK inverse kinematics. 4, 17

IMU inertial measurment unit. 4

NRP Neurorobotics Platform. 15–17, 24, 28, 33, 35

PH passive haptics. 8, 9

SoE sense of embodiment. 10, 11, 18, 31

VR virtual reality. 1, 3–5, 9–14, 16, 27, 30, 33, 36

Bibliography

[Kar17] Karen Moltenbrey. “Photorealism in Real Time”. In: *Computer Graphics World* (Volume 40 Issue 2: (Mar/Apr 2017) 2017). [Online; accessed 03/07/2019]. URL: http://www.cgw.com/Publications/CGW/2017/Volume-40-Issue-2-Mar-Apr-2017-/Photorealism-in-Real-Time.aspx.

[BA04] E. Brown and P. A. Cairns. “A grounded investigation of game immersion”. In: *Conference on Human Factors in Computing Systems - Proceedings* (Jan. 2004), pp. 1297–1300. DOI: 10.1145/985921.986048. URL: https://www-users.cs.york.ac.uk/~pcairns/pubs/Immersion.pdf.

[Gam17] Gamestar. *Doom - Screenshots aus dem PC-Original von 1993*. [Online; accessed 03/07/2019]. 2017. URL: https://www.gamestar.de/galerien/doom,132803.html.

[New16] New Game Network. *Doom screenshots*. [Online; accessed 03/07/2019]. 2016. URL: https://www.newgamenetwork.com/media/18721/doom/.

[KGS12] K. Kilteni, R. Groten, and M. Slater. “The Sense of Embodiment in Virtual Reality”. In: *Presence Teleoperators & Virtual Environments* 21 (Nov. 2012), pp. 373–387. DOI: 10.1162/PRES_a_00124. URL: https://www.researchgate.net/publication/243056510_The_Sense_of_Embodiment_in_Virtual_Reality.

[Lac14] J. Lackner. “Motion sickness: More than nausea and vomiting”. In: *Experimental brain research* 232 (June 2014). DOI: 10.1007/s00221-014-4008-8. URL: https://www.researchgate.net/publication/263432912_Motion_sickness_More_than_nausea_and_vomiting.

[BC98] M. Botvinick and J. Cohen. “Rubber hands ‘feel’ touch that eyes see”. In: *Nature* (391(6669):756 1998). DOI: 10.1038/35784. URL: http://www.nyu.edu/gsas/dept/philo/courses/consciousness/papers/Botvinick.pdf.

[Hau18] J. Haudenschild. “Virtual Embodiment: Dealing with Discrepancies between the Virtual and Real Body”. Bachelor Thesis. Technische Universität München, 2018. URL: https://wiki.tum.de/download/attachments/145457363/ba-haudenschild-vr_embodiment_discrepancies.pdf.

[Cas+18] P. Caserman, A. Garcia-Agundez, R. Konrad, S. Göbel, and R. Steinmetz. “Real-time body tracking in virtual reality using a Vive tracker”. In: *Virtual Reality* 22 (Nov. 2018). full text requested and received on 21.06.19. DOI: 10.1007/s10055-018-0374-z. URL: https://www.researchgate.net/publication/329150144_Real-time_body_tracking_in_virtual_reality_using_a_Vive_tracker.

[Amo14] E. Amos. *Kinect 2 picture*. [Online; accessed 22/07/2019]. 2014. URL: https://commons.wikimedia.org/wiki/File:Xbox-One-Kinect.jpg.

[Yos] Yost Labs Inc. *PrioVR diagram*. [Online; accessed 22/07/2019]. URL: https://yostlabs.com/priovr/.

[Nat] NaturalPoint Inc. *OptiTrack Diagram*. [Online; accessed 22/07/2019]. URL: https://optitrack.com/motion-capture-animation/#wall-to-wall-tracking.

[Bol+18] M. Boldt, M. Bonfert, I. Lehne, M. Cahnbley, K. Korschinq, L. Bikas, S. Finke, M. Hanci, V. Kraft, B. Liu, T. Nguyen, A. Panova, R. Singh, A. Steenbergen, R. Malaka, and J. Smeddinck. "You Shall Not Pass: Non-Intrusive Feedback for Virtual Walls in VR Environments with Room-Scale Mapping". In: Mar. 2018, pp. 143–150. DOI: 10.1109/VR.2018.8446177. URL: https://www.smeddinck.com/files/papers/Boldt_etal_You-Shall-Not-Pass_IEEE-VR_Preprint.pdf.

[LIB15] P. Lopes, A. Ion, and P. Baudisch. "Impacto: Simulating Physical Impact by Combining Tactile Stimulation with Electrical Muscle Stimulation". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology*. UIST '15. Charlotte, NC, USA: ACM, 2015, pp. 11–19. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807443. URL: http://plopes.org/wp-content/uploads/papers/2015-UIST-Impacto.pdf.

[Lop+17] P. Lopes, S. You, L.-P. Cheng, S. Marwecki, and P. Baudisch. "Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, 2017, pp. 1471–1482. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025600. URL: https://hpi.de/fileadmin/user_upload/fachgebiete/baudisch/projects/mobile_force_feedback/2017-CHI-VRwalls.pdf.

[Ins01] B. E. Insko. "Passive Haptics Significantly Enhances Virtual Environments". AAI3007820. PhD thesis. 2001. ISBN: 0-493-17286-6. URL: http://wwwx.cs.unc.edu/Research/eve/dissertations/2001-Insko.pdf.

[Che+15] L.-P. Cheng, T. Roumen, H. Rantzsch, S. Köhler, P. Schmidt, R. Kovacs, J. Jaspers, J. Kemper, and P. Baudisch. "TurkDeck: Physical Virtual Reality Based on People". In: Nov. 2015. DOI: 10.1145/2807442.2807463. URL: https://www.researchgate.net/publication/285056109_TurkDeck_Physical_Virtual_Reality_Based_on_People_Prop-based_virtual_reality_passive_virtual_reality.

[Vig10] F. de Vignemont. "Embodiment, ownership and disownership". In: *Consciousness and Cognition* (2010), pp. 1–12. DOI: 10.1016/j.concog.2010.09.004. URL: https://jeannicod.ccsd.cnrs.fr/ijn_00526983/document.

[DS15] D. D'Orazio and V. Savov. *Valve's VR headset is called the Vive and it's made by HTC*. Ed. by T. Verge. [Online; accessed 24/7/2019]. Mar. 2015. URL: https://www.theverge.com/2015/3/1/8127445/htc-vive-valve-vr-headset.

[HTC16] HTC. *VIVE NOW SHIPPING IMMEDIATELY FROM HTC, RETAIL PARTNERS EXPAND DEMO LOCATIONS*. [Online; accessed 24/7/2019]. June 2016. URL: `https://www.htc.com/us/newsroom/2016-06-07/`.

[HTCa] HTC. *HTC Vive Product Page*. [Online; accessed 24/7/2019]. URL: `https://www.vive.com/uk/product/`.

[Buc15] S. Buckley. *This Is How Valve's Amazing Lighthouse Tracking Technology Works*. Ed. by Gizmodo. [Online; accessed 24/7/2019]. May 2015. URL: `https://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768`.

[Ong17] T. Ong. *HTC launches Vive tracker bundles*. Ed. by T. Verge. [Online; accessed 25/7/2019]. Nov. 2017. URL: `https://www.theverge.com/circuitbreaker/2017/11/16/16664916/htc-vive-tracker-bundles-date-price`.

[HTC17] HTC. *HTC Vive Tracker Developer Guidelines*. Version 1.5. HTC. July 2017.

[HTCb] HTC. *HTC Vive Accessory Page*. [Online; accessed 24/7/2019]. URL: `https://www.vive.com/eu/accessory/`.

[Huma] Human Brain Project. *Short Overview of the Human Brain Project*. [Online; accessed 25/7/2019]. URL: `https://www.humanbrainproject.eu/en/about/overview/`.

[Humb] Human Brain Project. *Bodies for brains*. [Online; accessed 25/7/2019]. URL: `https://www.humanbrainproject.eu/en/robots/`.

[Humc] Human Brain Project. *Mid-Term Vision for HBP*. [Online; accessed 29/7/2019. URL: `https://neurorobotics.net/vision.html`.

[Axo16] S. Axon. *Unity at 10: For better—or worse—game development has never been easier*. Ed. by A. Technica. [Online; accessed 30/7/2019]. Sept. 2016. URL: `https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/`.

[Uni] Unity Technologies. *Explore Unity*. [Online; accessed 30/7/2019]. URL: `https://unity3d.com/unity`.

[Bon18] N. Bonfiglio. *DeepMind partners with gaming company for AI research*. Ed. by T. D. Dot. [Online; accessed 30/7/2019]. Oct. 2018. URL: `https://www.dailydot.com/debug/unity-deempind-ai/`.

[Lan19] B. Lang. *HTC Confirms Vive Cosmos Will Support OpenVR/SteamVR*. Ed. by R. T. VR. [Online; accessed 30/7/2019]. Jan. 2019. URL: `https://www.roadtovr.com/htc-vive-cosmos-steamvr-openvr-ces-2019/`.

[TA18] J. C. Tuthill and E. Azim. "Proprioception". In: *Current Biology* 28 (5 Mar. 2018). DOI: `10.1016/j.cub.2018.01.064`. URL: `https://doi.org/10.1016/j.cub.2018.01.064`.