Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Felix Baumgartner

Steuerung einer Wellenfeldsynthese-Anlage mittels eines Virtual-Reality-Interfaces

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Felix Baumgartner

Steuerung einer Wellenfeldsynthese-Anlage mittels eines Virtual-Reality-Interfaces

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Wolfgang Fohl
Zweitgutachter: Prof. Dr. Andreas Meisel

Eingereicht am: 24. August 2017

Felix Baumgartner

Thema der Arbeit

Steuerung einer Wellenfeldsynthese-Anlage mittels eines Virtual-Reality-Interfaces

Stichworte

Wellenfeldsynthese, Virtual Reality, WFS, VR, OSC, Unity, HTC Vive, SuperCollider

Kurzzusammenfassung

Dieses Arbeit enthält die Dokumentation der Entwicklung eines Virtual Reality Interfaces, welches zur Steuerung einer Wellenfeldsynthese (WFS)-Anlage genutzt werden kann. Über das entwickelte Interface kann zum einen die Position und Ausrichtung von Audioquellen manipuliert werden und zum anderen die Belegung der Quellen mit unterschiedlichen Signalen bestimmt werden. Der große Vorteil einer über Virtual Reality gestützten Steuerung der WFS-Anlage gegenüber einer Gestensteuerung liegt darin, dass eine Feinjustierung durch die zusätzlichen optischen Informationen deutlich vereinfacht wird.

Felix Baumgartner

Title of the paper

Controlling of a Wave-Field-Synthesis system through a Virtual Reality interface

Keywords

Wave-Field-Synthesis, Virtual Reality, WFS, VR, OSC, Unity, HTC Vive, SuperCollider

Abstract

This paper contains a documentary about the development of a Virtual Reality interface which can be used to control a Wave-Field-Synthesis system. Through the developed interface the user is able to manipulate the position as well as the orientation of the audiosources. It is also possible to controll the routing of sound to these audiosources. The biggest advantage of the Virtual Reality interface over a gesture based controll mechanism lays in the additional optical informations which really help at fine adjusting.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

1. Einleitung

Diese Bachelorarbeit befasst sich mit der Entwicklung eines VR-Interfaces zur Steuerung der WFS-Anlage an der HAW Hamburg. Das VR-Interface wird mit der GameEngine Unity implementiert, da Unity eine gute Unterstützung für die HTC Vive bietet.

1.1. Motivation

Eine Steuerung einer WFS-Anlage, siehe Kapitel 2, über ein VR-Interface bringt verschiedene Vorteile mit sich.

Zum einen ist es möglich, die Szene, die von der WFS-Anlage akustisch dargestellt wird, zusätzlich optisch wahrzunehmen. Hierdurch wird die Szene deutlich greifbarer. Durch die erhöhte Greifbarkeit der Szene ist es außerdem möglich, feinere Anpassungen der Szene vorzunehmen.

Zum anderen wird durch das VR-Interface die Steuerung durch mehrere Programme in ein Programm gebündelt. Es werden im Falle der Steuerung, ohne das hier entwickelte VR-Interface, mehrere Programme benötigt, um einerseits WFS-Quellen zu erzeugen, zu löschen, zu positionieren und zu rotieren und andererseits auf den Quellen Audiosignale abzuspielen. Diese Programme müssen zur selben Zeit bedient werden, was ein häufiges Wechseln der Programmumgebung zur Folge hat. Mit dem hier entwickelten VR-Interface ist es möglich all diese Manipulationen der Szene in der WFS-Anlage über ein Programm zu steuern. Das hat zur Folge, dass zusätzlich dazu, dass die Szene greifbarer wird, die Anpassung der Szene stark vereinfacht werden kann.

1.2. Eingrenzung des Themenbereichs

Diese Bachelorarbeit begrenzt sich auf die Ausführung von Steuerungskommandos zum Zeitpunkt des Aufrufens der jeweiligen Aktion. Das bedeutet, dass die Szene innerhalb des VR-Interfaces nicht zulässt, den Zeitpunkt der Ausführung einer Aktion festzulegen und auch die gespeicherten Szenen keinerlei Zeitkomponente aufweisen. Des Weiteren ist es nicht möglich, Quellen über eine gewisse Zeit zu verschieben, wie es von der WFS-Anlage unterstützt

wird, da hierfür ebenfalls eine zeitliche Regulierung innerhalb des VR-Interfaces notwendig wäre, welche mit der WFS-Anlage synchronisiert werden müsste, um eine genaue Steuerung gewährleisten zu können.

1.3. Weitere Lösungsansätze

Mir ist kein Lösungsansatz bekannt, der sich mit einer Steuerung einer WFS-Anlage über ein VR-Interface befasst, jedoch gibt es bereits mehrere andere Ansätze zur Steuerung der WFS-Anlage an der HAW Hamburg, die nicht das Medium VR nutzen.

Ein Steuerungsansatz beruht auf einer Gestenerkennung und ist in der Bachelorarbeit von Malte Nogalski [Nog12] näher beschrieben. Diese Gestensteuerung erlaubt es dem Nutzer WFS-Quellen neu zu positionieren. Hierfür werden Marker für das Trackingsystem, siehe Abschnitt 2.4.2.2, an der Hand getragen und die Bewegungen des Nutzers nachverfolgt.

Ein weiterer Steuerungsansatz besteht in Form eines Touchtables, welcher es dem Nutzer unter anderem ermöglicht, die Positionierung der Quellen über einen großen Touchscreen zu beeinflussen.

Es gibt Lösungsansätze in denen die VR genutzt wird, um Inhalte zu veranschaulichen. Die Öl- und Gasindustrie beispielsweise nutzen die VR um die Zusammensetzung der Erdkruste nach einer Messung zu visualisieren um eventuell geeignete Förderstandorte zu finden (vgl. [DBGJ13] Seite 300–303).

1.4. Abgrenzung

Diese Bachelorarbeit grenzt sich zu der Gestensteuerung von Malte Nogalski insofern ab, als dass zum einen die Steuerung in der VR stattfindet und die Szene auf vielfältigere Weise manipuliert werden kann. Zum anderen ist bei der Gestensteuerung die Nutzung eines weiteren Programmes zur Belegung der WFS-Quellen mit Audiosignalen notwendig, wohingegen dies in dem für diese Bachelorarbeit entwickelten VR-Interface gebündelt wird, sodass kein Programmwechsel notwendig ist.

Die Abgrenzung zur Touchtable-Steuerung, liegt zum einen in der Nutzung der VR und zum anderen in den vielfältigeren Manipulationsmöglichkeiten, die das VR-Interface bietet.

Zur VR-Nutzung der Öl- und Gasindustrie grenzt sich dieses VR-Interface insofern ab, als dass nicht nur der aktuelle Stand visualisiert wird, sondern auch manipulierend eingegriffen werden kann, somit der aktuelle Stand verändert werden kann.

Heutzutage wird VR häufig dazu genutzt dem Nutzer ein Spiel zu präsentieren, sodass der Nutzer sehr tief in das Spiel eintauchen kann. Ebenso gibt es Ansätze Filme[1] über die VR immersiver zu gestalten. Dieses VR-Interface grenzt sich hiergegen insofern ab, als dass hierüber kein Spiel oder Film präsentiert wird, sondern die Vorzüge, des tief Eintauchens in das Geschehen, der VR genutzt werden, um die Steuerung einer WFS-Anlage zu vereinfachen.

Außerdem gibt es Programme die sich über die VR benutzen lassen und somit noch immersiver wirken, einen ganz neuen Blickwinkel eröffnen und neue Möglichkeiten bieten. Ein Programm dieser Art ist z.B. das Zeichenprogramm Google Tilt Brush [2]. Dieses ermöglicht es dem Nutzer direkt in den dreidimensionalen Raum zu Malen und sich um und durch das gemalte zu bewegen. Hierdurch werden neue Arten des Ansehens von Malereien möglich, es werden aber auch die Schwierigkeiten des Malens, ohne eine feste Leinwand, deutlich. Hiergegen grenzt sich das VR-Interface insofern ab, als dass dieses eine Steuerungssoftware darstellt, welche die VR nutzt um die Steuerung zu vereinfachen. Die anderen Programme hingegen stellen in sich geschlossene Einheiten dar, die nicht auf andere Softwareeinheiten einwirken.

1.5. Aufbau der Arbeit

Diese Arbeit gliedert sich nach der Einleitung in sechs Einheiten. Zunächst werden die für diese Arbeit relevanten Begriffe, wie Wellenfeldsynthese (s. Kapitel 2) und Virtual Reality (s. Kapitel 3) erläutert. Anschließend wird auf die Methodik eingegangen, welche für die Entwicklung des VR-Interfaces genutzt wurde. Nachdem die Methodik vorgestellt wurde, wird die Implementierung, der für das VR-Interface benötigten Komponenten, beleuchtet. Im Anschluss daran, wird auf die Auswertung einer Benutzbarkeitsstudie eingegangen. Abschließend wird ein Fazit gezogen und ein Ausblick auf mögliche Erweiterungen gegeben. Es folgen der Anhang, das Glossar und das Quellenverzeichnis.

[1] Der Kurzfilm „The Rose and I“ (http://store.steampowered.com/app/396060/The_Rose_and_I/) nutzt die VR um den Zuschauer noch tiefer ins Geschehen eintauchen zu lassen.
[2] https://www.tiltbrush.com/

2. Wellenfeldsynthese und Laborumgebung

Dieses Kapitel befasst sich mit dem Thema Wellenfeldsynthese, sowie der Laborumgebung des Wellenfeldsynthese-Labors an der HAW Hamburg. Zuerst wird erläutert, weshalb man Wellenfeldsynthese betreibt und wie das technische Prinzip dahinter lautet. Anschließend wird darauf eingegangen, wie das Labor aufgebaut ist und welche Software zum Einsatz kommt.

2.1. Weshalb Wellenfeldsynthese?

Wellenfeldsynthese bedeutet, dass man das Wellenfeld, welches entstehen würde, wenn eine Audioquelle an einem bestimmten Ort Schall freisetzt, künstlich nachbildet. Dies geschieht indem man sich das Huygen's Prinzip zunutze macht und eine Summe künstlich erzeugter Schallwellen zu einer Schallwelle, welcher einer natürlichen Schallwelle sehr ähnelt, zusammenfügt (s. [EB16]). Dies ist nicht nur auf eine Audioquelle beschränkt, sondern kann auch für viele Quellen zur selben Zeit geschehen. Hierdurch kann ein sehr realistisch wirkendes Klangbild erzeugt werden.

Um dieses realitätsnahe Klangbild zu erzeugen, wird ein anderer Ansatz gegenüber eines klassischen Audiosystems, wie z.B. eines 2.1 Systems, verfolgt. Bei einem klassischen Audiosystem handelt es sich um ein kanalbasiertes System. Dies bedeutet, dass die Quelldatei bereits so abgemischt wird, dass klar definiert ist, welche Signale zu einer bestimmten Zeit von einem bestimmten Lautsprecher wiedergegeben werden. Die Wellenfeldsynthese nutzt eine objektbasiertes System. Hierbei wird nicht bereits im Audiosignal codiert, von welchem Lautsprecher das Signal wiedergegeben werden soll, sondern es kann zur Laufzeit definiert werden, wo sich im Raum die Schallquelle zu einem bestimmten Zeitpunkt befinden soll. Aus dieser Information heraus wird dann zum Zeitpunkt der Wiedergabe errechnet, von welchem Lautsprecher zu welcher Zeit das Signal wiedergegeben werden muss, um ein Wellenfeld in der Form zu synthetisieren, dass es für den Hörer so klingt, als wäre der Punkt der Schallerzeugung an dem Ort, wo sich die Audioquelle befindet.

Der große Vorteil dieses synthetisch erzeugten Wellenfeldes gegenüber eines Schallfeldes eines klassischen kanalbasierten Audiosystems besteht darin, dass man sich innerhalb des Raumes bewegen kann, ohne dass das Hörerlebnis darunter leidet.
Bei einem kanalbasierten System verhält es sich derart, dass es einen sogenannten „Sweet Spot“ gibt. An diesem Punkt und leider auch nur an diesem Punkt, ist das Hörerlebnis wie beim Abmischen der Quelldatei beabsichtigt. Sobald man sich etwas links oder rechts dieses Punktes befindet, leidet das Hörerlebnis, da sich der virtuelle Punkt der Schallerzeugung stark verschiebt. Dies ist der Fall, da jegliche Positionierung der virtuellen Schallquelle lediglich durch Unterschiede in der Lautstärke bei der Signalwiedergabe auf den unterschiedlichen Audiokanälen geschieht. Das wiederum hat zur Folge, dass der Effekt der Lautstärkedifferenz noch gravierender zum Tragen kommt, falls der Abstand zu den Lautsprechern nicht gleichmäßig ist.
Sitzt man direkt vor einem Lautsprecher wird man alles aus diesem Lautsprecher viel lauter wahrnehmen als die Signale aus den anderen Lautsprechern. Somit werden alle virtuellen Schallquellen aus Richtung dieses Lautsprechers wahrgenommen.

Ein weiteres Konzept, welches mit der WFS-Technik konkurriert, ist das der Higher-Order-Ambisonics. Hierbei werden die unterschiedlichen Audiosignale anhand verschiedener Gleichungen in das sogenannte B-Format gebracht. Dieses ist unabhängig davon, welche Lautsprecher verwendet werden, so dass dieses Format sich zur Übertragung der Signale eignet. Die Audiosignale im B-Format können dann manipuliert werden. Zur Wiedergabe der Audiosignale wird dann das B-Format decodiert und auf die Lautsprecher angewendet.

Higher-Order-Ambisonics schaffen es den „Sweet Spot“ auszudehnen und somit das Hörerlebenis gegenüber einem klassischen Audiosystem zu verbessern (vgl. [MDB06]). Es werden jedoch je nach Höhe der Ordnung mehr Lautsprecher benötigt, denn mit der Höhe der Ordnung steigt die Anzahl an Ambisonics-Kanälen. Die Anzahl der Lautsprecher muss mindestens der Anzahl an Ambisonics-Kanälen entsprechen um die gewünschte Wiedergabe zu erhalten. Der Vorteil der WFS-Technologie gegenüber den Higher-Order-Ambisonics besteht darin, dass der „Sweet Spot“ nicht ausgedehnt, sondern vermieden wird.

2.2. Technisches Prinzip

Das der WFS-Anlage zugrundeliegende Prinzip ist das von dem niederländischen Mathematiker, Physiker und Astronom Christiaan Huygens entdeckte und nach diesem benannte Huygens Prinzip, vergleiche [Her17]. Dieses besagt, dass jede Stelle einer Wellenfront der Ausgangspunkt einer neuen sogenannten Elementarwelle ist. Das Prinzip bezieht sich ursprünglich auf die

Optik, ist aber auch auf Schallwellen anwendbar. Weitere Informationen sind in dem Artikel zum Thema „Huygens' priciple" der Encyclopædia Britannica [EB16] zu finden.

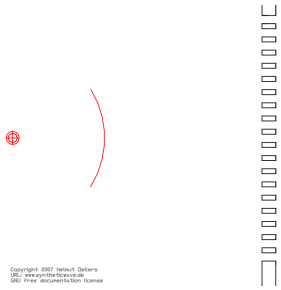
Auf die WFS-Anlage bezogen ist Huygens' Prinzip wie folgt anwendbar: Man stelle sich vor, eine Schallwelle breite sich von einer Schallquelle kugelförmig aus. Dies ist die natürliche Ausbreitung von Schall. Diese Schallwelle trifft nun auf eine Wand mit vielen Schlitzen. Die Schallwelle wird sich durch diese Schlitze hinweg ausbreiten. Am Ende dieser Schlitze, also am anderen Ende der Wand, tritt die Schallwelle nun wieder aus den Schlitzen aus. Hier kommt nun das Huygens' Prinzip zum Tragen. Jeder Teil der ursprünglichen Schallwelle, welche nun aus einem der Schlitze austritt, ist der Ursprungspunkt einer neuen Elementarwelle mit derselben Schwingung und Amplitude. Das bedeutet, dass sich von jedem dieser Schlitze eine neue Elementarwelle kugelförmig ausbreitet. Die unterschiedlichen Elementarwellen wiederum überlagern sich nach kurzer Zeit und verbinden sich zu einer einzigen Wellenfront, welche sehr stark der Wellenfront entspricht, die an diesem Ort wäre, wäre kein Hindernis in Form einer Wand dort gewesen.

Die WFS-Anlage macht sich dieses Prinzip zunutze, indem die Schlitze in der Wand von Lautsprechern simuliert werden. D.h. jeder Lautsprecher stellt einen Schlitz dar. Es wird nun errechnet, wie die Wellenfront beim Aufschlag auf die Wand aussehen würde und die Lautsprecher werden entsprechend zeitversetzt mit dem wiederzugebenden Signal angesteuert, um eine entsprechende Elementarwelle zu erzeugen. Die so erzeugten Elementarwellen wiederum vereinigen sich wieder zu einer gemeinsamen Wellenfront, welche einer Wellenfront sehr gleicht, die von einer Schallquelle an Ort x ausgehen würde, obwohl sich an besagtem Ort keinerlei schallemittierendes Objekt befindet (vgl. [FN13]). Die resultierende Wellenfront entspricht für Schallwellen ab einer gewissen Frequenz nicht komplett der ursprünglichen Wellenfront, da die Ausgangspunkte der Elementarwellen nicht direkt aneinander grenzen, sondern in einem gewissen Abstand zueinander stehen. Dies ist der Fall, da die Lautsprecher eine gewisse Größe haben, die Mittelpunkt der Lausprecher also einen gewissen Abstand zueinander haben, je kleiner dieser Abstand ist, umso größer können die Frequenzen sein, ohne dass der Effekt des Spatial-Aliasings auftritt (vgl. [Ver97]).

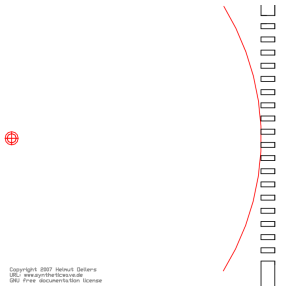
2.3. Aufbau der Laborumgebung

Das WFS-Labor umfasst eine WFS-Anlage der Firma Four Audio[1], welche aus insgesamt 26 WFS-Modulen besteht, die jeweils 8 Kanäle besitzen und jeweils 26 Lautsprecher beinhalten. Die Lautsprechermodule sind auf ca. 2 m Höhe angebracht. Somit stehen der Anlage insgesamt

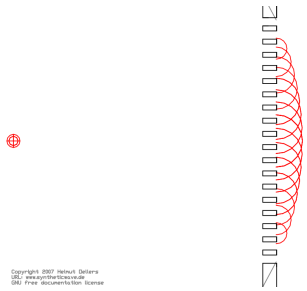
[1] http://fouraudio.com/



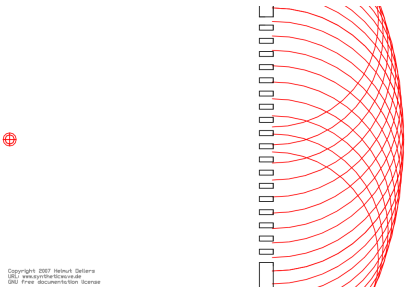
(a) Zunächst wird errechnet, wie sich der Schall ausbreitet.



(b) Der Moment, in dem die Schallwelle auf die Lautsprecher treffen würde



(c) Jeder Lautsprecher erzeugt eine Elementarwelle



(d) Die Elementarwellen vereinigen sich zu einer gemeinsamen Wellenfront

Abbildung 2.1.: Schritte beim Erzeugen einer synthethischen Schallwelle (vgl. [Oel])

208 Kanäle und 676 Lautsprecher zur Verfügung. Die 26 Module sind in einem Rechteck mit den Seitenlängen 5 m und 6 m angeordnet. Außerdem besteht die WFS-Anlage aus drei Rendering Servern, welche als ein gemeinsames Cluster agieren und über Ethernet miteinander kommunizieren. Des Weiteren ist noch ein Mac Pro vorhanden, über den die Anlage von den Nutzern gesteuert werden kann (vgl. [Bad13] Seite 247).

Für viele Nutzungsszenarien ist es notwendig, die Position des Hörers, bzw. eines oder mehrerer Objekte, nachzuverfolgen. Dies ist z.B. der Fall, wenn man auf ein nachverfolgtes (getracktes) Objekt eine Audioquelle legen möchte. Außerdem ist es notwendig die Position des Hörers zu wissen, um das Rendering der Audiosignale von Quellen innerhalb des Raumes richtig steuern zu können. Ohne die Position des Hörers in die Berechnung einfließen zu lassen können Quellen nicht richtig gerendert werden, wenn sich der Hörer zwischen der Quelle und den Lautsprechern befindet, aus dem die Signale wiedergegeben werden.

Befindet sich eine Quelle innerhalb des Rechtecks der WFS-Module werden immer die Lautsprecher zur Wiedergabe des Signales benutzt, die sich auf der Seite des Raumes von der Mitte aus gesehen befinden, auf der sich auch die Quelle befindet. Steht nun der Hörer

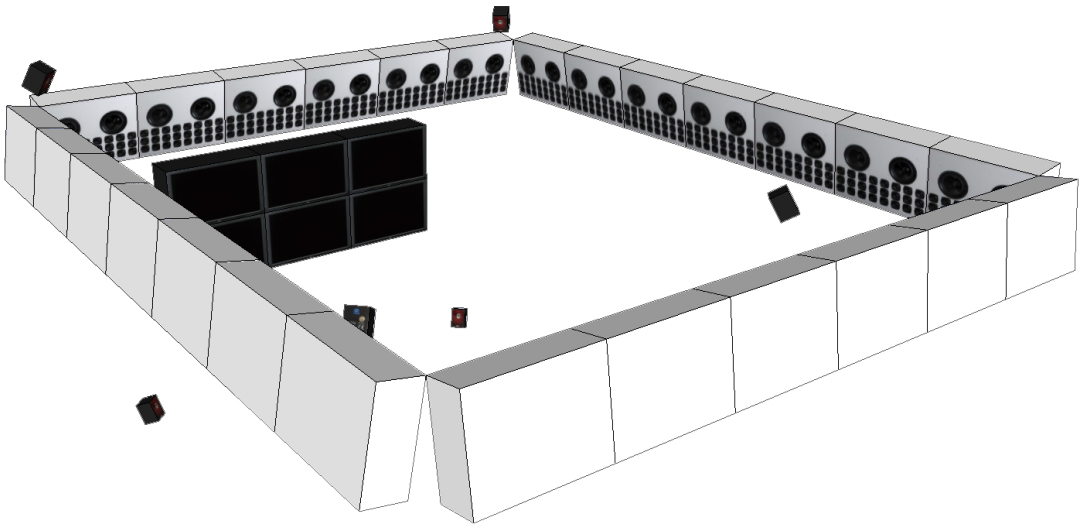


Abbildung 2.2.: Aufbau des WFS-Labors an der HAW Hamburg mit WFS-Modulen, Powerwall und ART-Trackingsystem [Nog12]

zwischen den Lautsprechern und der Quelle nimmt der Hörer die Quelle aus Richtung der Lautsprecher war, obwohl sie eigentlich auf der anderen Seite des Hörers ist. Ist die Position des Hörers bekannt, kann die virtuelle Mitte des Raumes, die wie eben beschrieben in die Berechnung der Wiedergabe mit einfließt, auf die Position des Hörers gesetzt werden. Somit sind dann die Lautsprecher, welche das Signal wiedergeben, immer auf der richtigen Seite des Hörers. Leider ist das nur für einen Hörer anwendbar, da der Raum nicht mehrere Mitten besitzen kann (vgl. [FW15]). Um den Hörer oder andere Objekte nachverfolgen zu können, ist ein ART-Trackingsystem verbaut, welches von der Firma Advanced Realtime Tracking[2] hergestellt wurde. Dieses ist ein auf Infrarot basierendes Trackingsystem, bei dem die Targets passiv sind, diese also lediglich das Infrarotlicht reflektieren und selber keine Signale aussenden (vgl. [PD15] Seite 293).

Außerdem befindet sich eine sogenannte Powerwall im Labor. Dies ist eine aus sechs Monitoren bestehende Monitorwand. Die Monitore sind in zwei übereinander liegenden Reihen angebracht. Sie können als 6 Einzelmonitore betrieben werden oder auch als ein großer virtueller Monitor. Die Powerwall kann genutzt werden, um zusätzlich zu den durch die WFS-Anlage bereitgestellten akustischen Reizen, noch optische Reize zu stimulieren.

[2] http://www.ar-tracking.com/home/

Der Gesamtaufbau der Anlage ist in Abbildung 2.2 dargestellt. Zusätzlich zu diesem Aufbau ist seit einiger Zeit noch die HTC Vive inklusive der Lighthouses im Labor vorhanden. Somit sind alle Voraussetzung für diese Bachelorarbeit im WFS-Labor der HAW Hamburg gegeben.

2.4. Systeme innerhalb dieser Wellenfeldsynthese-Anlage

Innerhalb der WFS-Anlage sind verschiedene Systeme im Einsatz. In diesem Abschnitt werden einige wichtige Einheiten näher beleuchtet.

2.4.1. Software

Ein wichtiger Teil der WFS-Anlage stellt die Software zur Steuerung der Anlage dar. Es gibt mehrere Ebenen der Software, welche zur Steuerung benutzt werden. Hier wird allerdings nur auf die vom Benutzer der Anlage direkt benutzten Einheiten eingegangen.

2.4.1.1. Open-Sound-Control

„Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology.“ [Opea]

OSC ist ein Protokoll, welches für moderne Netzwerktechnologien optimiert wurde und zur Kommunikation zwischen Computern, Sound Synthesizern und anderen Multimediageräten genutzt werden kann.

Es handelt sich hierbei um ein recht einfach gestricktes Protokoll. Eine über das OSC-Protokoll übertragene Einheit ist ein OSC-Paket, welches neben einigen Metadaten als Inhalt entweder eine OSC-Nachricht oder ein OSC-Bundle enthält. Eine OSC-Nachricht besteht immer aus einer Adresse gefolgt von einem String, welcher die Typen der Parameter angibt. Anschließend können noch beliebig viele Parameter folgen. Eine beispielhafte OSC-Nachricht könnte wie folgt aussehen:

```
1 /testMessage, // Die Adresse
2 iis, // Parametertypen Angabe, Integer, Integer, String
3 1, // Erster Parameter
4 2, // Zweiter Parameter
5 test // Dritter Parameter
```

Listing 2.1: Beispielhafte OSC-Nachricht

Ein OSC-Bundle hingegen besteht zum einen aus dem Tag #bundle gefolgt von einem TimeTag und zum anderen aus dem Inhalt. Dieser Inhalt kann aus beliebig vielen OSC-Nachrichten sowie OSC-Bundles bestehen (vgl. [Opeb]).

2.4.1.2. SuperCollider

„SuperCollider is a platform for audio synthesis and algorithmic composition, used by musicians, artists, and researchers working with sound." [Supa]

SuperCollider ist eine Plattform zur Audiosynthese und algorithmischen Komposition, die von Musikern, Künstlern und Forschern, welche mit Audio arbeiten, benutzt wird. SuperCollider besteht grob aus drei Elementen:

1. Dem Audioserver scsynth, welcher zur Synthese von Audiosignalen benutzt werden kann.
2. Der Programmiersprache sclang welche genutzt werden kann um den Server zu steuern.
3. Dem Editor scide welcher zum Schreiben des Programmcodes benutzt wird.

SuperCollider ist eine OpenSource Software, die sowohl für MacOS, als auch für Linux und Windows verfügbar ist (vgl. [Supa]).

Da es sich bei dem Server um eine sehr flexibel einstellbare Einheit handelt, mit der es ohne Weiteres möglich ist sehr viele verschiedene Audiokanäle gleichzeitig zu steuern, eignet sich SuperCollider außerordentlich gut für die Entwicklung des OSCVRServers (s. Abschnitt 5.2 auf Seite 40).

2.4.1.3. xWonder

Die Software xWonder stellt die graphische Nutzerschnittstelle der Software Wonder dar, welche zur Berechnung der Wiedergabe der Signale durch die WFS-Anlage benutzt wird (für weitere Informationen siehe [BP04]).

Über xWonder wird es dem Nutzer ermöglicht, die Positionierung der Audioquellen zu sehen und zu manipulieren. Es können hierüber sowohl die Position, die Rotation, die ID, der Quelltyp, die Farbe, als auch der Name der Quellen geändert werden. Alle Änderungen, die über xWonder vorgenommen werden, werden an die zugrundeliegende Wonder-Software weiter gegeben, sodass jegliche Anpassung unmittelbar in der Audiowiedergabe widergespiegelt wird. Die Kommunikation funktioniert hierbei bidirektional. Das bedeutet, dass auch alle Änderungen, die außerhalb von xWonder vorgenommen werden, sei es zum Beispiel über das

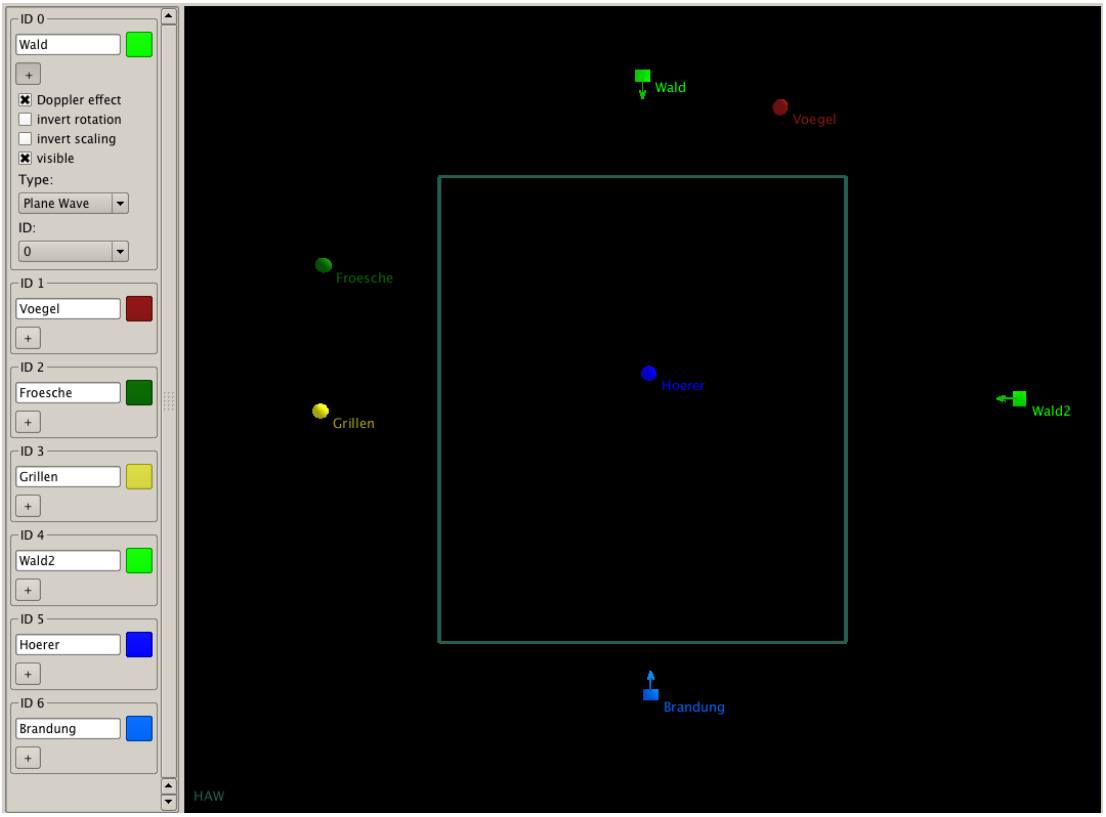


Abbildung 2.3.: Ansicht einer Beispielszene in xWonder. Links: eine Auflistung aller aktiven Quellen. Rechts: eine zweidimensionale Ansicht der Positionen der Quellen. Das Rechteck stellt den Raum innerhalb der WFS-Module dar.

hier entwickelte VR-Interface oder über eine andere Schnittstelle, direkt in xWonder visualisiert werden.

2.4.2. Trackingsystem

Innerhalb des WFS-Labors sind zwei verschiedene Trackingsysteme installiert. Zum einen ist das Trackingsystem der Firma Advanced Realtime Tracking verbaut (s. Abschnitt 2.4.2.2 auf Seite 12), zum anderen das Trackingsystem, welches im Lieferumfang der HTC Vive enthalten ist (s. Abschnitt 2.4.2.3 auf Seite 13). Diese werden zum einen dafür benötigt, die Position des Hörers für die Berechnung der Wiedergabe des Signals auf den Lautsprechern bereit zu stellen, was notwendig ist, sobald sich die virtuelle Audioquelle innerhalb des von den WFS-Modulen eingefassten Raumes befindet, wie in Abschnitt 2.3 auf Seite 6 beschrieben. Zum anderen kann

man die Trackingsysteme für weitere Zwecke benutzen, z.B. um eine Gestensteuerung zu implementieren, wie es Malte Nogalski für seine Bachelorarbeit [Nog12] getan hat.

2.4.2.1. Klassifizierung von Trackingsystemen

Trackingsysteme lassen sich nach mehreren Merkmalen klassifizieren. Es gibt z.B. markerbasierte und markerlose Trackingsysteme. Hierbei handelt es sich um ein Unterscheidungsmerkmal, welches sich darauf bezieht, ob zum Identifizieren eines Objektes ein Marker benutzt wird oder, ob die Objekte anhand anderer Merkmale, wie beispielsweise Form, Farbe, Oberflächenstruktur oder ähnlichem, erkannt werden können.
Die markerbasierten Trackingsysteme lassen sich wiederum an der Art der verwendeten Marker unterscheiden. Hierbei werden aktive und passive Marker unterschieden. Aktive Marker zeichnen sich dadurch aus, dass sie aktiv Signale, wie beispielsweise Infrarotlicht, aussenden. Diese Signale müssen somit nicht außerhalb des Markers erzeugt werden. Passive Marker hingegen sind, wie der Name schon sagt, passiv, senden also keine Signale aus, sondern sind beispielsweise ein Muster mit hohem Kontrast oder stark reflektierend. Es ist aber ebenso möglich Trackingsysteme anhand der verwendeten Signale zu klassifizieren. Man kann dabei zum Beispiel optisches, magnetisches aber auch akustisches Tracking unterscheiden (vgl. [PD15] Seite 293). Des Weiteren lässt sich bei den optischen Verfahren zwischen solchen unterscheiden, bei denen die Kameras/Sensoren außerhalb des Interaktionsbereiches angebracht sind, sogenannte Outside-In-Verfahren, und welche bei den die Kameras/Sensoren mit dem Objekt dessen Bewegung aufgenommen wird verbunden sind (vgl. [DBGJ13] Seiten 107–109). Letztere heißen Inside-Out-Verfahren. Bei dem Trackingsystem der Firma Advanced Realtime Tracking (s. Abschnitt 2.4.2.2) handelt es sich um Outside-In-Tracking mithilfe passiver Marker. Das Trackingsystem der HTC Vive hingegen ist ein System nach dem Inside-Out-Verfahren aber ebenfalls mit passiven Markern (s. Abschnitt 2.4.2.3).

2.4.2.2. Trackingsystem der Firma Advanced Realtime Tracking

Das Trackingsystem, welches im WFS-Labor der HAW Hamburg verbaut ist, besteht aus 6 Kameras, die Infrarotblitze aussenden und die Reflexionen dieser Blitze aufnehmen. Aus der Menge der Reflexionen werden bestimmte Muster erkannt, welche sich aus den Reflexionen von guten Reflektoren, die in einer bestimmten Position zueinander stehen, zusammensetzen. Diese Konstellationen von Reflektoren müssen einmalig eingemessen werden und unter all den anderen eingemessenen Konstellationen einzigartig sein. Es ist dann möglich aus den Reflexionen dieser Konstellationen zu bestimmen, um welches Objekt es sich handelt und wo sich dieses im Raum befindet. Außerdem kann die Orientierung des Objektes erkannt werden.

Voraussetzung für die Positions- und Rotationserfassung ist allerdings, dass die Reflexionen von mehreren Kameras aufgefangen werden, da dann über einfache Peilung der Ort der Reflexion bestimmt werden kann.

Das Trackingsystem stellt die erfassten Positions- und Rotationsdaten dann per Multicast zur Verfügung. Hierbei wird zwischen Objekten unterschieden, die in drei Dimensionen erfasst wurden (3DOF = three dimensions of freedom), also lediglich die Position im dreidimensionalen Raum und den Objekten, die in sechs Dimensionen erfasst wurden (6DOF = six dimensions of freedom). Bei den Objekten, die mit 6DOF erfasst wurden, wurde sowohl die Position im dreidimensionalen Raum, als auch die Rotation erfasst. Bei der Rotation wird der Gier-, der Nick- und der Rollwinkel erfasst.

2.4.2.3. Trackingsystem der HTC Vive

Das Trackingsystem der HTC Vive funktioniert, wie oben schon erwähnt, nach dem Inside-Out-Verfahren mit passiven Markern. Diese Marker werden entweder durch die Controller oder durch das Head Mounted Display dargestellt. An diesen befinden sich Sensoren, welche die von den sogenannten Lighthouses ausgehenden Infrarotsignale registrieren und daraus errechnen, an welcher Stelle und in welcher Orientierung sie sich befinden. Von den Lighthouses gehen unterschiedliche Signale aus. Zuerst kommt ein Infrarot-Blitz, dann läuft ein Infrarot-Fächer von links nach rechts durch den Raum. Anschließend kommt wieder ein Infrarot-Blitz, gefolgt von einem Fächer, der nun von oben nach unten durch den Raum geht. Diese Fächer kommen alle 8.333 ms was einer Frequenz von 120 Hz entspricht. Durch die Zeitunterschiede, mit denen die Fächer die einzelnen Sensoren auf dem Marker treffen, lässt sich dann die Position und Orientierung ermitteln (vgl. [ligb]).

Des Weiteren sind in den Controllern und in dem HMD Gyroskope verbaut, welche zur Berechnung der wahrscheinlichen Position des verfolgten Objektes benutzt werden. Die Signale der Lighthouses korrigieren dann den Drift dieser Berechnung. Durch die gemeinsame Nutzung beider Technologien ist es möglich die Position der Controller mit einer Frequenz von 366Hz und die Position des HMDs mit 1006Hz zu erneuern. Die Position beider Elemente, Controller und HMD sind dabei ungefähr auf 2 mm genau [liga].

3. Virtual Reality

Dieses Kapitel befasst sich mit dem Begriff der Virtual Reality, zu Deutsch „virtuelle Realität“. Es werden zunächst einige wichtige Begriffe definiert. Anschließend wird sich mit dem Thema User Interfaces, also Benutzerschnittstellen innerhalb einer VR-Umgebung, befasst. Abschließend werden noch die für dieses Projekt genutzten Werkzeuge Unity, insbesondere SteamVR und HTC Vive, vorgestellt.

3.1. Begriffserklärungen

Um eine bessere Vorstellung des Themas VR zu haben werden zunächst die wichtigen Begriffe Virtual Reality, Immersion sowie Suspension of Disbelief erläutert.

3.1.1. Virtual Reality

„Virtual Reality bezeichnet realistische 3D-Umgebungen, in denen eine virtuelle Welt in Echtzeit interaktiv exploriert und manipuliert werden kann. Dazu werden Ausgabegeräte benötigt, die möglichst das gesamte Sichtfeld umfassen und einen Stereoeindruck vermitteln.“ ([PD15] Seite 246).

Durch die Virtual Reality wird es dem Nutzer ermöglicht in eine virtuelle Welt einzutauchen und diese zu erkunden. Hierfür ist es notwendig ein dreidimensionales Modell der virtuellen Welt zu haben, um die Interaktion mit dieser Welt simulieren zu können. Diese virtuellen Welten können ganz unterschiedlicher Natur sein. Es ist möglich in eine virtuelle Welt, welche der realen Welt in vielen Dingen sehr nachempfunden ist, einzutauchen. Ebenso ist es aber auch möglich in eine sich stark von der Realität abhebende virtuelle Welt einzutauchen. Es kann also auch als natürlich empfunden werden, wenn sich die Physik in der virtuellen Welt in Teilen oder auch komplett anders verhält, als in der Realität. Eine Person, die eine virtuelle Welt erkundet, ist bereit bestimmte realitätsferne Aspekte zu akzeptieren, dies ist als Suspension of Disbelief bekannt (s. Abschnitt 3.1.3 auf Seite 16).

Virtuelle Realität zeichnet sich gegenüber traditioneller Computergrafik dadurch aus, dass man sich hierbei auf dreidimensionale Inhalte fokussiert und diese in Echtzeit auf die In-

teraktion des Nutzers reagieren müssen. Um diese Inhalte wirkungsvoll darzustellen wird hier auf geeignete dreidimensionale Displays zurückgegriffen. Außerdem beschränkt sich VR nicht ausschließlich auf die graphische Darstellung. Das Maß der Immersion kann durch das Ansprechen weiterer Sinne wie dem Hör- und Tastsinn noch gesteigert werden (vgl. [DBGJ13] Seiten 2–14).

3.1.2. Immersion

„Immersion beschreibt den Grad des Eintauchens bedingt durch objektive, quantifizierbare Stimuli, d.h. multimodale Stimulationen der menschlichen Wahrnehmung“ ([DBGJ13] Seite 46).

„Immersion is a description of a technology, and describes the extent to which the computer displays are capable of delivering an inclusive, extensive, surrounding and vivid illusion of reality to the senses of a human participant. Inclusive (I) indicates the extent to which physical reality is shut out. Extensive (E) indicates the range of sensory modalities accommodated. Surrounding (S) indicates the extent to which this virtual reality is panoramic rather than limited to a narrow field. Vivid (V) indicates the resolution, fidelity, and variety of energy simulated within a particular modality (for example, the visual and colour resolution). Vividness is concerned with the richness, information content, resolution and quality of the displays.“ [SW97]

Immersion ist also ein Maß dafür, wie sehr die Displays es ermöglichen, ein isoliertes, auf breiter Fläche lebendig wirkendes und viele Sinne ansprechendes virtuelles Erlebnis zu liefern. Mit anderen Worten, es ist ein Maß dafür, wie tief ein Nutzer einer VR in die virtuelle Welt eintauchen kann.

Hierbei kommt es laut Slater und Wilbur [SW97] auf vier Aspekte an:

1. *Inclusive* beschreibt das Maß, in dem der Nutzer von der Realität isoliert ist, also jegliche Sinneseindrücke, die der Nutzer wahrnimmt, vom Computer generiert werden.

2. *Extensive* beschreibt wie viele Sinne des Nutzers angesprochen werden.

3. *Surroundings* beschreibt, ob dem Nutzer nur ein kleines Sichtfeld geboten wird oder, ob große Teile oder gar das ganze Sichtfeld abgedeckt wird.

4. *Vivid* beschreibt die Lebendigkeit des gebotenen Inhaltes. Dieses Maß wird stark von der Farbtreue und Auflösung des Bildes beeinflusst.

3.1.3. Suspension of Disbelief

Suspension of Disbelief bezeichnet die Fähigkeit des Menschen die Widersprüche der virtuellen Welt zur Realität auszublenden und dies auch zu wollen. Hierdurch wird es möglich, auch in klar als virtuelle Welt erkennbare Simulationen tief einzutauchen und hier ein großes Maß an Immersion zu verspüren (vgl. [DBGJ13] Seite 8).

3.2. User Interfaces

Ein User Interface (UI) bezeichnet die Schnittstelle zwischen Nutzer und Programm. Hierfür wird häufig eine graphische Oberfläche genutzt, in diesem Falle heißt die UI dann Graphical User Interface (GUI). Es gibt aber auch rein textuelle UIs welche häufig bei Konsolenanwendungen eingesetzt werden. Für jede Form des UIs, ist es wichtig, dass der Umgang mit ihm leicht zu erlernen, wenn nicht sogar intuitiv, sein sollte. Auch ist es relevant, dass eine effiziente Nutzung des Programms über das UI ermöglicht wird. Für diese Bachelorarbeit ist es notwendig ein GUI in der virtuellen Welt zu erschaffen. Hierbei kommt es vor allem darauf an, dass die Inhalte des GUI relevant für die Benutzung des Programms, in diesem Falle für die Benutzung des Interfaces zur Steuerung der WFS-Anlage, sind und plausibel dargestellt werden (s. [PD15] Seite 250 + 251). Das bedeutet unter anderem, dass die Funktionen, die über das GUI verfügbar sind in einer sinnvollen Anordnung zueinander stehen und man sich auf die Funktionen beschränkt, die zur Steuerung der WFS-Anlage sinnvoll sind.

3.3. Unity

Um einem Nutzer ein VR-Erlebnis zur Verfügung zu stellen ist es notwendig, die virtuelle Welt als dreidimensionales Modell zu erstellen und die Interaktion mit dieser zu ermöglichen. Hierfür eignen sich Gameengines wie die Unreal-Engine[1] oder Unity[2] sehr gut. Da für dieses Projekt die Gameengine Unity benutzt wurde, wird sich hier auf diese beschränkt.

Unity bietet eine Plattform um Spiele zu entwickeln, die sowohl für Einsteiger als auch für fortgeschrittene Nutzer geeignet ist. Dem Nutzer wird die Möglichkeit gegeben zweidimensionale Spiele, wie z.B. Pong, zu erstellen. Aber auch dreidimensionale Spiele können erzeugt werden. Bei diesen handelt es sich nicht zwangsläufig um Spiele die über 3D-Displays gespielt werden müssen, sondern um alle Spiele, in denen sich der Spieler im Spiel im dreidimensionalen Raum bewegt. Für viele VR-Geräte, wie z.B. die HTC Vive oder die Oculus Rift, gibt es einen

[1] https://www.unrealengine.com
[2] https://unity3d.com

guten Support für Unity, sodass es mit Unity ermöglicht wird, Spiele für die VR zu erstellen. Bei dem für diese Bachelorarbeit entwickelten VR-Interface handelt es sich zwar nicht um ein Spiel, es kann aber trotzdem gut über die Plattform Unity erstellt werden.

3.3.1. SteamVR

SteamVR[3] ist ein Softwarepaket der Firma VALVE, welches einem ermöglicht, VR-Inhalte auf einem geeigneten VR-Gerät, wie beispielsweise der HTC Vive, wiederzugeben. Es ist möglich VR-Inhalte, die über die Plattform Steam[4] bezogen werden können, abzuspielen. Des Weiteren besteht auch die Möglichkeit die eigenen, z.B. in Unity erstellten VR-Inhalte, direkt wiederzugeben.

3.4. HTC Vive

Die HTC Vive ist ein von der Firma HTC, in Kooperation mit der Firma VALVE, entwickeltes VR-Gerät, welches aus einem HMD, zwei Controllern und dem Trackingsystem, bestehend aus zwei Lighthouses, besteht. Die HTC Vive bietet ein Sichfeld von 110°, bei einer Auflösung von 2160 x 1200 Pixel und 90 Hz Wiederholungsrate. Dies ermöglicht es dem Nutzer ein großes Maß an Immersion zu erleben.

Das HMD bietet zusätzlich zu dem Display und dem Trackingsystem (s. Abschnitt 2.4.2.3 auf Seite 13) noch eine Frontkamera, welche genutzt werden kann, um die virtuelle Welt durch einblenden realer Elemente zu erweitern (vgl. [vivb]).

Die Controller sind baugleich, es ist also nicht festgelegt, bei welchem es sich um den linken und bei welchem es sich um den rechten Controller handelt. Ein Controller bietet folgende Knöpfe bzw. Interaktionsmöglichkeiten:

- Menüknopf (1): Dieser Knopf befindet sich oberhalb des Trackpads und wird i.d.R. genutzt, um das Spielemenü aufzurufen.
- Trackpad (2): Das Trackpad ist ein recht großes berührungssensitives Feld, welches zwischen Berührung und Druck unterscheiden kann. Des Weiteren ist der Ort der Berührung bzw. des Drucks auf dem Trackpad bekannt. Somit können hierüber z.B. Wisch- oder Rotationsgesten benutzt werden.
- Systemknopf (3): Dieser Knopf kann benutzt werden, um auf das SteamVR-Menü zuzugreifen. Es ist nicht möglich den Knopf über Unity anders zu belegen.

[3] https://steamcommunity.com/steamvr
[4] http://store.steampowered.com/

- Trigger (7): Der Trigger Knopf ist ein Knopf, welcher auf der Unterseite des Controllers angebracht ist und frei belegt werden kann.
- Grip button (8): Der Grip button ist ein auf beiden Seiten des Stiels angebrachter Knopf, der gut durch das feste Greifen des Controllers gedrückt werden kann. Auch dieser ist frei belegbar.

Die Zahlen hinter dem Namen der Interaktionsmöglichkeit beziehen sich auf die Zahlen in der Abbildung 3.1, welche den Aufbau eines Vive Controllers zeigt.

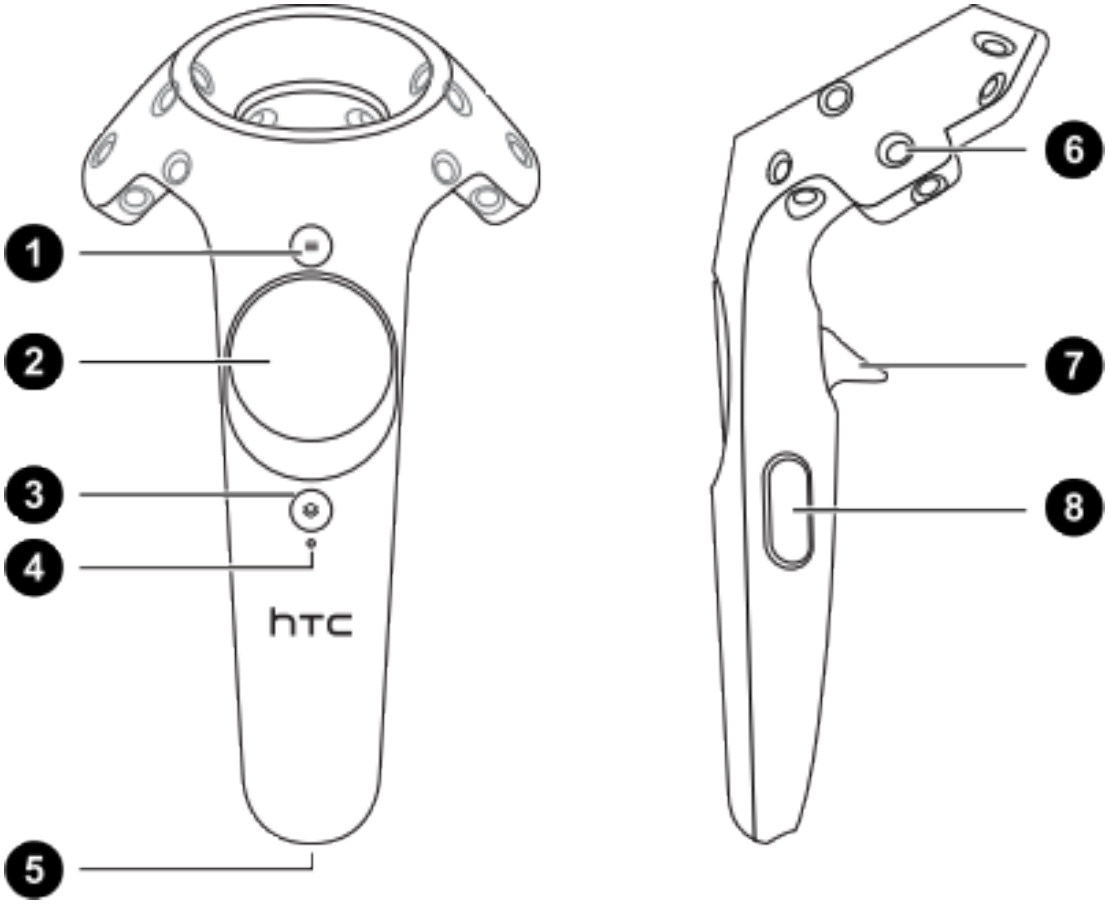


Abbildung 3.1.: Der Aufbau eines Vive Controllers ([viva] Seite 15).

4. Methodik

Dieses Kapitel befasst sich mit der Methodik und dem gewünschten Ergebnis dieser Bachelorarbeit. Es wird zunächst das Konzept erläutert, welches hinter diesem Projekt steht, anschließend wird die verwendete Entwicklungsmethodik näher beleuchtet. Folgend werden einige Use Cases und die nicht-funktionalen Anforderungen beschrieben.

4.1. Konzept

Das Konzept dieser Bachelorarbeit besteht darin, die Steuerung der WFS an der HAW Hamburg (s. Kapitel 2 ab Seite 4) zu vereinfachen, sodass man ohne selber Befehle eingeben zu müssen, die Quellen der Anlage manipulieren kann. Der Prozess des Manipulierens der Quellen soll hierbei zusätzlich durch die Visualisierung innerhalb einer virtuellen Welt (s. Kapitel 3 ab Seite 14) erleichtert werden. Es soll über das VR-Interface unter anderem möglich sein, Quellen zu erzeugen, zu löschen, zu verschieben, mit Sounds zu belegen und diese abzuspielen, sowie Szenen zu speichern und zu laden. Diese Steuerung teilt sich in verschiedene Elemente, auf wie sie in Abbildung 4.1 zu sehen sind. Die einzelnen Komponenten werden in Kapitel 5 ab Seite 29 genauer beschrieben.

4.2. Entwicklungsmethodik

Für die Umsetzung dieses Projektes müssen zunächst einige Use Cases, also Szenarien bei denen der Nutzer von der Software unterstützt werden soll, definiert werden, um diese dann entsprechend durch die Implementation abzudecken. Zuerst muss sich in die Thematik des Erstellens von Spielen mit Unity eingearbeitet werden. Dann muss die Grundlage für das VR-Interface geschaffen werden, indem die HTC Vive über OSC-Nachrichten mit der WFS-Anlage verbunden wurde. Anschließend muss der OSCVRServer (s. Abschnitt 5.2 ab Seite 40) entwickelt werden, da festgestellt wurde, dass keine der im WFS-Labor der HAW Hamburg vorhandene Software die notwendige Steuerungsmöglichkeit über Netzwerknachrichten unterstützt. Darauf folgend müssen die Funktionen zur Steuerung der Anlage auf Seiten des

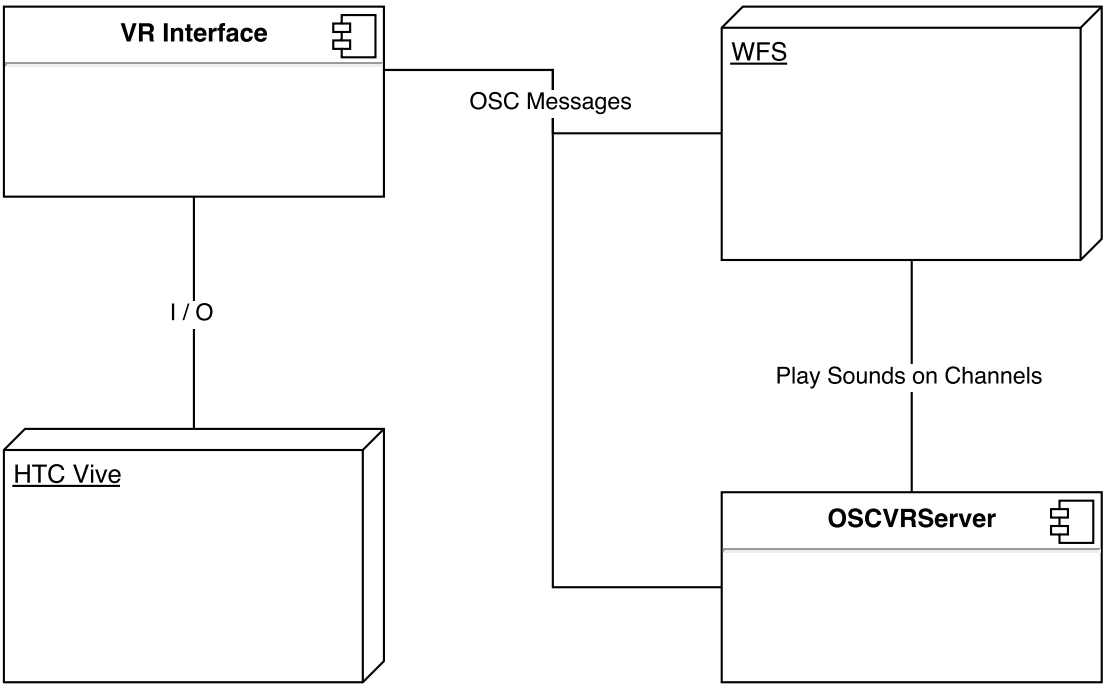


Abbildung 4.1.: Die Aufteilung der Steuerung in verschiedene Elemente und deren Kommunikationswege

VR-Interfaces implementiert und diese dann über ein User Interface dem Nutzer zur Verfügung gestellt werden.

4.3. Use Cases

In diesem Abschnitt werden die für diese Software wichtigen Use Cases erläutert. Bei Use Cases handelt es sich um Szenarien, bei denen der Nutzer von der Software unterstützt werden soll. Diese können dann verwendet werden, um die Software entsprechend zu entwerfen. Die Use Cases definieren entsprechende Ziele, Vor- und Nachbedingungen, sowie das Erfolgsszenario, mögliche Fehlerfälle und die Anforderungen an die Software.

4.3.1. Laden und manipulieren einer bestehenden WFS-Szene

Dieser Use Case beschreibt ein Szenario, in dem bereits eine Szene in der WFS-Anlage existiert und diese über das VR-Interface visualisiert werden soll, um die Szene dann manipulieren zu können. Die genaue Beschreibung des Use Cases ist in Tabelle 4.1 zu sehen.

Titel	Laden und manipulieren einer bestehenden WFS-Szene
Akteur	WFS-Nutzer
Ziel	Es soll eine derzeit auf der WFS-Anlage laufende Szene visualisiert werden, um diese manipulieren zu können.
Auslöser	Ein WFS-Nutzer beschließt, dass eine momentan laufende WFS-Szene über das VR-Interface visualisiert und eventuell auch manipuliert werden soll.
Vorbedingungen	In der WFS-Anlage existiert eine Szene. Außerdem ist der OSCVR-Server (s. Abschnitt 5.2 auf Seite 40) in Betrieb.
Nachbedingungen	In dem Falle, dass nur visualisiert wurde: Die WFS-Szene wurde nicht verändert. In dem Falle, das auch Manipulationen vorgenommen wurden: Die Manipulationen der WFS-Szene wurden an die WFS-Anlage übertragen und umgesetzt.
Erfolgsszenario	Der Nutzer startet das VR-Interface und steigt in die virtuelle Welt ein. Die Informationen über den aktuellen Zustand der Szene werden abgerufen. Hierzu zählen: • Alle Informationen über die Quellen, welche von der WFS-Anlage geliefert werden können. Dies sind unter anderem: Position, Rotation, Typ, Farbe und ID. • Alle Informationen vom OSCVRServer. Dies sind alle bisher registrierten Sounds sowie alle verfügbaren Sounds (s. Abschnitt 5.2.3.2.1 auf Seite 42). Anschließend kann der WFS-Nutzer die Szene manipulieren. Für weitere Informationen zum Manipulieren siehe Use Case: Manipulation einer WFS Szene in Abschnitt 4.3.4.
Erweiterungen	Der Use Case: Speichern einer Szene, siehe Abschnitt 4.3.5, stellt eine mögliche Erweiterung zu diesem Use Case dar.

Fehlerfälle	Es ist möglich, dass die Daten von der WFS-Anlage und/oder dem OSCVRServers nicht beim VR-Interface ankommen. Somit ist nicht gewährleistet, dass die Manipulation der Szene auf den richtigen Daten stattfindet. Des Weiteren ist es möglich, dass die Manipulationen der Szene nicht oder nicht komplett bei der WFS-Anlage ankommen.
Häufigkeit	Bei jedem Start des VR-Interfaces
Anforderungen	Die Informationen über die Sounds und WFS-Quellen müssen direkt beim Start des VR-Interfaces geladen werden, um von Anfang an das Arbeiten auf aktuellen Daten gewährleisten zu können. Des Weiteren müssen Änderungen durch das VR-Interface direkt an die WFS-Anlage bzw. den OSCVRServer weitergegeben werden. Außerdem müssen auch Änderungen auf Seiten der WFS-Anlage durch andere Manipulatoren an das VR-Interface propagiert werden, so dass dieses immer auf aktuellen Daten agiert.

Tabelle 4.1.: Use Case: Laden und manipulieren einer bestehenden WFS-Szene

4.3.2. Laden einer Szene

Dieser Use Case beschreibt ein Szenario, in dem eine zu einem früheren Zeitpunkt gespeicherte Szene wieder in das VR-Interface geladen werden soll.

Titel	**Laden einer Szene**
Akteur	WFS-Nutzer
Ziel	Eine vorher gespeicherte Szene soll geladen werden, so dass die WFS-Anlage und der OSCVRServer sich in dem der geladenen Szene entsprechenden Zustand befinden.
Auslöser	Der VR-Interface-Nutzer beschließt eine Szene zu laden, um diese dann entweder wiederzugeben, oder zu manipulieren.
Vorbedingungen	Das VR-Interface, sowie die WFS-Anlage und der OSCVRServer sind in Betrieb.

Nachbedingungen	Die gespeicherte Szene wurde geladen und die WFS-Anlage, sowie der OSCVRServer befinden sich in dem der Szene entsprechenden Zustand.
Erfolgsszenario	Die WFS-Anlage und der OSCVRServer werden auf den Zustand zurückgesetzt, dass keine aktiven Quellen vorhanden und keine Sounds registriert sind. Anschließend wird die Szene aus einer Datei eingelesen, entsprechende WFS-Quellen erzeugt und die Sounds auf ihnen registriert.
Erweiterungen	Der Use Case: Speichern einer Szene (s. Abschnitt 4.3.5) stellt eine mögliche Erweiterung zu diesem Use Case dar, da eine Szene erst gespeichert werden muss, um sie zu einem späteren Zeitpunkt laden zu können. Außerdem kann eine geladene Szene nach der Manipulation durch den Nutzer wieder gespeichert werden. Zu dem stellt der Use Case: Manipulation einer WFS-Szene eine Erweiterung dar (s. Abschnitt 4.3.4).
Fehlerfälle	Die WFS-Anlage und der OSCVRServer könnten nicht richtig zurückgesetzt werden, die Datei nicht richtig eingelesen werden oder die Datei nicht im richtigen Format sein. Des Weiteren könnte die Erzeugung von WFS-Quellen und die Registrierung von Sounds fehlerhaft sein.
Häufigkeit	Jedes Mal, wenn nicht auf eine bestehende WFS-Szene (s. Abschnitt 4.3.1) oder eine neue Szene (s. Abschnitt 4.3.3) aufgebaut werden soll.
Anforderungen	Die WFS-Anlage, sowie der OSCVRServer sollen sich nach der Ausführung in exakt dem Zustand der geladenen Szene befinden, unabhängig davon, in welchem Zustand sie vorher waren.

Tabelle 4.2.: Use Case: Laden einer Szene

4.3.3. Neue Szene erstellen

Dieser Use Case beschreibt ein Szenario, in dem eine neue Szene erzeugt werden soll, um diese dann mit neuen WFS-Quellen und registrierten Sounds anzureichern.

Titel	Neue Szene erstellen
Akteur	WFS-Nutzer
Ziel	Eine neue, leere Szene soll erstellt werden
Auslöser	Der WFS-Nutzer möchte eine neue, leere Szene haben um hier neue WFS-Quellen erzeugen zu können.
Vorbedingungen	Das VR-Interface ist in Betrieb.
Nachbedingungen	In der WFS-Anlage sind keine aktiven Quellen vorhanden und im OSCVRServer sind keine Sounds auf Kanälen registriert. Das VR-Interface ist also im Zustand einer leeren Szene.
Erfolgsszenario	Der Nutzer entscheidet sich dazu eine neue Szene zu erstellen. Die WFS-Anlage und der OSCVRServer werden zurückgesetzt, indem alle WFS-Quellen gelöscht werden und alle registrierten Sounds entfernt werden.
Erweiterungen	Die Use Cases: Speichern einer Szene (s. Abschnitt 4.3.5) und Manipulation einer WFS-Szene (s. Abschnitt 4.3.4) stellen mögliche Erweiterungen zu diesem Use Case dar.
Fehlerfälle	Es ist möglich, dass das Zurücksetzen der WFS-Anlage und des OSCVRServers nicht funktioniert, da beispielsweise OSC-Nachrichten verloren gehen.
Häufigkeit	Jedes Mal, wenn nicht eine bestehende Szene von der WFS-Anlage (s. Abschnitt 4.3.1) oder aus einer Datei (s. Abschnitt 4.3.2) geladen und bearbeitet werden soll, sondern wenn eine neue Szene erstellt werden soll.
Anforderungen	Die WFS-Anlage und der OSCVRServer müssen zurückgesetzt werden.

Tabelle 4.3.: Use Case: Neue Szene erstellen

4.3.4. Manipulation einer WFS-Szene

Dieser Use Case beschreibt ein Szenario, in dem eine Szene geladen wurde und diese nun manipuliert werden soll.

Titel	Manipulation einer WFS-Szene
Akteur	WFS-Nutzer
Ziel	Die Quellen einer WFS-Anlage sollen in Position, Rotation, Namen und Typ manipuliert werden können. Außerdem sollen Quellen erstellt und gelöscht werden können. Des Weiteren sollen Sounds auf bestimmten Quellen in bestimmter Lautstärke abgespielt werden können.
Auslöser	Ein WFS-Nutzer hat das VR-Interface gestartet und ist entweder in eine bestehende WFS-Szene eingestiegen, wie in Use Case 4.3.1 beschrieben, hat eine neue Szene erstellt, wie in Use Case 4.3.3 beschrieben, oder hat eine Szene geladen, wie es in Use Case 4.3.2 der Fall ist. Die dem WFS-Nutzer visualisierte Szene soll nun manipuliert werden.
Vorbedingungen	Der VR-Interface-Nutzer befindet sich in einer Szene.
Nachbedingungen	Die Manipulationen an den WFS-Quellen, sowie deren Erzeugung oder Entfernung aus der Szene wurden an die WFS-Anlage und den OSCVRServer propagiert.

Erfolgsszenario	Der VR-Interface-Nutzer… • …ändert die Position einer WFS-Quelle. • …ändert die Rotation einer WFS-Quelle. • …ändert den Typ einer WFS-Quelle. • …ändert den Namen einer WFS-Quelle. • …erzeugt eine neue WFS-Quelle. • …löscht eine WFS-Quelle. • …registriert einen Sound auf einer WFS-Quelle. • …entfernt einen Sound von einer WFS-Quelle. • …startet die Wiedergabe eines auf einer WFS-Quelle registrierten Sounds auf einer Quelle. • …stoppt die Wiedergabe eines auf einer WFS-Quelle spielenden Sounds. Jede dieser Manipulationen an der Szene wird entweder an die WFS-Anlage oder aber an den OSCVRServer gesendet, sodass diese sich entsprechend verhalten können.
Erweiterungen	Der Use Case: Speichern einer Szene (s. Abschnitt 4.3.5) stellt eine mögliche Erweiterung zu diesem Use Case dar, da die manipulierte Szene gespeichert werden kann.
Fehlerfälle	Die Manipulationen der Szene werden nicht richtig an der WFS-Anlage bzw. an dem OSCVRServer empfangen, sodass diese nicht entsprechend handeln können.
Häufigkeit	Viele Male pro Nutzung des VR-Interfaces, da dies der Hauptgrund zur Nutzung des VR-Interfaces sein wird.
Anforderungen	Alle Manipulationen an der Szene müssen umgehend an die WFS-Anlage bzw. an den OSCVRServer gesendet werden, sodass sich die Änderungen sofort auswirken.

Tabelle 4.4.: Use Case: Manipulation einer WFS-Szene

4.3.5. Speichern einer Szene

Dieser Use Case beschreibt ein Szenario, in dem eine im VR-Interface bestehende Szene gespeichert werden soll, sodass diese zu einem späteren Zeitpunkt wieder geladen werden kann.

Titel	Speichern einer Szene
Akteur	WFS-Nutzer
Ziel	Die derzeitige Szene soll inklusive aller registrierten Sounds und aktiven WFS-Quellen und deren Eigenschaften gespeichert werden, sodass zu einem späteren Zeitpunkt genau diese Szene wiederhergestellt werden kann.
Auslöser	Der VR-Interface-Nutzer beschließt die Szene abzuspeichern.
Vorbedingungen	Es ist eine Szene in das VR-Interface geladen.
Nachbedingungen	Es gibt eine permanente textuelle Repräsentation der Szene, beispielsweise im JSON oder XML Format.
Erfolgsszenario	Der VR-Interface-Nutzer möchte die aktuelle Szene abspeichern. Es wird eine textuelle Repräsentation der aktuellen Szene im gewünschten Format erstellt, in der für jede aktive Quelle einige Metadaten sowie die auf ihr registrierten Sounds, ebenfalls mit entsprechenden Metadaten, enthalten sind. Zu den Metadaten der Quellen gehören z.B. die ID, Position, Rotation, Name und Farbe. Die Metadaten eines registrierten Sounds umfassen den Namen der Sound-Datei, die Lautstärke und ob dieser Sound in einer Wiederholungsschleife wiedergegeben werden soll. Diese textuelle Repräsentation wird dann in einer Datei gespeichert.
Erweiterungen	Die Use Cases: Laden einer Szene (s. Abschnitt 4.3.2), Neue Szene erstellen (s. Abschnitt 4.3.3), Laden und manipulieren einer bestehenden WFS-Szene (s. Abschnitt 4.3.1), sowie Manipulation einer WFS-Szene (s. Abschnitt 4.3.4) stellen mögliche Erweiterungen zu diesem Use Case dar, denn jede Szene, die geladen oder neu erstellt wurde, kann gespeichert werden.
Fehlerfälle	Die Erzeugung der textuellen Repräsentation kann fehlerhaft sein und die Speicherung der Repräsentation in einer Datei kann fehlschlagen.

Häufigkeit	Oft, wenn nicht sogar immer am Ende der Nutzung des VR-Interfaces.
Anforderungen	Die textuelle Repräsentation muss alle relevanten Daten umfassen, die zur Reproduktion der Szene benötigt werden und muss permanent gespeichert sein.

Tabelle 4.5.: Use Case: Speichern einer Szene

4.4. Nicht-funktionale Anforderungen

An das VR-Interface werden folgende nicht-funktionale Anforderungen gestellt:

- Die derzeitige Szene sollte beim Starten des VR-Interfaces unmittelbar in das VR-Interface geladen werden.
- Die Manipulation der Quellen und registrierten Sounds soll mit geringem zeitlichem Verzug an die WFS-Anlage bzw. an den OSCVRServer propagiert werden, sodass diese direkt umgesetzt werden können.
- Das User Interface soll leicht bedienbar sein.

5. Steuerung der WFS-Anlage mittels VR

Dieses Kapitel handelt von der Umsetzung des VR-Interfaces. Zuerst wird auf die Implementierung in Unity eingegangen. Hier findet jegliche Steuerung, die mit der VR zu tun hat, statt. Anschließend wird auf die Implementierung des OSCVRServers eingegangen, welcher zur Steuerung der Wiedergabe von Audiodateien auf der WFS-Anlage benutzt wird.

5.1. Implementierung in Unity

Die Implementierung in Unity umfasst mehrere Komponenten, welche zur Steuerung der WFS-Anlage und zur Steuerung des OSCVRServers (s. Abschnitt 5.2) benötigt werden. Diese sind unter anderem der SourceManager, welcher die WFS-Quellen verwaltet, der OSCManager, welcher jeglichen OSC-Nachrichten-Verkehr verwaltet, das User Interface und der ConfigReader, welcher die Konfigurationsdatei ausliest und die Werte den anderen Elementen zur Verfügung stellt. Wie diese ganzen Komponenten zusammenspielen ist der Abbildung 5.1 auf Seite 30 zu entnehmen.

5.1.1. SourceManager

Der SourceManager dient zur Verwaltung der WFS-Sound-Quellen (s. Abschnitt 5.1.1.1), den Szenen und der Zuordnung der Sounds zu den WFS-Audio-Quellen. Um Probleme, die durch Benutzung mehrerer SourceManager entstehen können, zu vermeiden, handelt es sich beim SourceManager um ein Singleton. Das bedeutet, dass eine Klasse nur maximal ein Mal zur Zeit instanziiert sein kann. Hierfür ist es notwendig, dass jedes Mal, wenn ein neuer Verweis auf eine Instanz der Klasse erzeugt werden soll, geprüft wird, ob die Klasse schon einmal instanziiert wurde. Falls die Klasse bereits instanziiert wurde, ist auf diese Instanz zu verweisen, andernfalls muss eine neue Instanz erzeugt werden. Für weitere Informationen zum Singleton Design Pattern siehe [Gam95]. Die Instanz des SourceManagers erhält man über die statische Methode *getSourceManager()*.

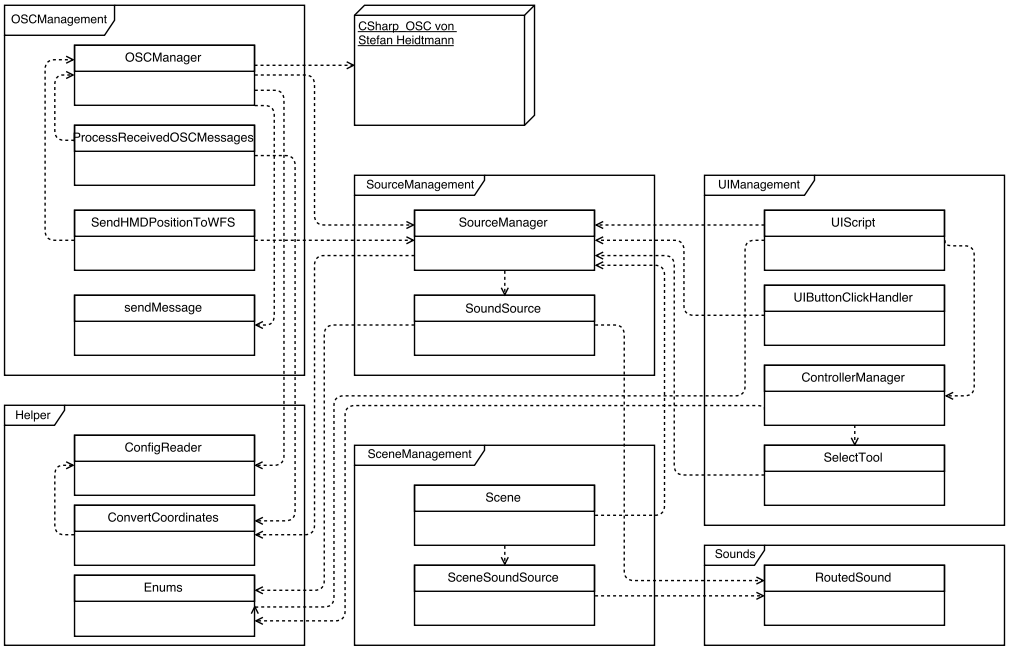


Abbildung 5.1.: UML-Klassendiagramm des VR-Interfaces

Mit dem SourceManager besteht die Möglichkeit …

- … neue Quellen hinzuzufügen.
- … vorhandene Quellen zu entfernen.
- … sich eine vorhandene Quelle ausgeben zu lassen.
- … die derzeitig vorhandenen WFS-Quellen abzufragen.
- … die derzeitig selektierte WFS-Quelle abzufragen oder zu ändern.
- … die nächste verfügbare ID für eine Quelle zu erfragen.
- … die derzeitige Szene zu speichern.
- … eine Szene zu laden.

- ...die derzeitige Szene auf den Zustand keiner WFS-Sound-Quellen zurückzusetzen.
- ...die derzeitige Szene wiederzugeben.
- ...die Wiedergabe der derzeitigen Szene zu stoppen.
- ...den Pfad zum Ordner, in dem die Szenen gespeichert werden, abzufragen.
- ...eine Liste der auf dem OSCVRServer verfügbaren Sounds zu erhalten.
- ...die Liste der verfügbaren Sounds um einen Eintrag zu erweitern.

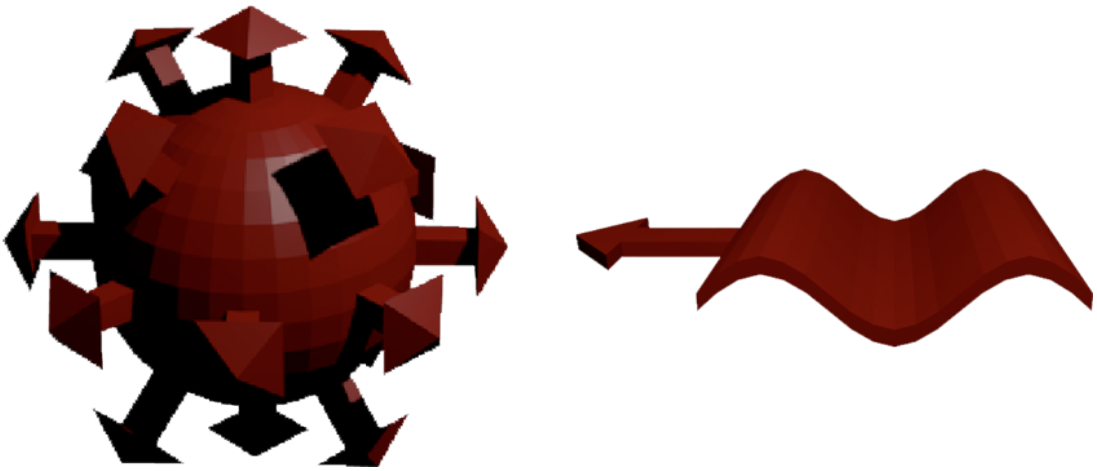
5.1.1.1. SoundSource

Die WFS-Sound-Quellen werden durch die Klasse SoundSource, welche von der Klasse MonoBehaviour[1] erbt, repräsentiert. Diese Klasse bietet die Möglichkeit folgende Daten zu speichern:

- *bool:* Active
- *string:* SourceName
- *int:* IdAtWFS
- *Enums.SoundSourceType:* Type
- *Vector3:* Position
- *float:* Angle
- *List<RoutedSound>:* RoutedSounds
- *Color:* Color
- *int:* Groupid
- *bool:* Doppler

Je nachdem, welchen Wert die Variable *Type* (Plane oder Point) hält, stellt sich die SoundSource in der virtuellen Welt entsprechend dar (s. Abbildung 5.2). Die Quelle wird in derselben Farbe, wie sie in der WFS-Anlage bekannt ist, angezeigt. Abbildung 5.2a zeigt die Darstellung einer Punktquelle, Abbildung 5.2b zeigt die Darstellung einer Planar-Quelle. Bei beiden Quellarten wird der Name der Quelle in der virtuellen Welt oberhalb der Quelle angezeigt. Dieser Text zeigt immer zum Nutzer, sodass der Name der Quelle immer gut lesbar ist.

[1] Die Standard-Klasse von der jedes Unity C# Skript erbt. Für mehr Informationen siehe: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html



(a) Darstellung einer WFS-Quelle vom Typ: Point (b) Darstellung einer WFS-Quelle vom Typ: Plane

Abbildung 5.2.: Darstellungen einer Planar und einer Punktquelle im VR-Interface.

5.1.1.2. Koordinaten Konvertierung

Um die Quellen und den Nutzer in beiden Systemen richtig positionieren zu können, ist es notwendig die Koordinaten zu konvertieren, denn das Koordinatensystem der Vive und das der WFS-Anlage unterscheiden sich sowohl in ihrer Achsenausrichtung, als auch im Ort ihres Ursprungs. Wie die Achsen zueinander stehen ist der Tabelle 5.1 zu entnehmen. Hierbei ist eine Zeile wie folgt zu lesen: Die x-Achse im Koordinatensystem der HTC Vive entspricht der Y-Achse des WFS-Anlagen-Koordinatensystems. Die Koordinaten der Ursprünge der Koordinatensysteme im jeweils anderen Koordinatensystem ausgedrückt ist in Tabelle 5.2 nachzulesen.

HTC Vive	WFS-Anlage
X-Achse	Y-Achse
Z-Achse	X-Achse
Y-Achse	Z-Achse

Tabelle 5.1.: Die Zuordnung der Achsen beider Koordinatensysteme zueinander.

Achse	HTC Vive	WFS-Anlage
X	-2,5	0
Y	0	2,5
Z	0	0

Tabelle 5.2.: Die Koordinaten des Ursprungs im jeweils anderen Koordinatensystem

Um den Umgang mit diesen Konvertierungen zu erleichtern wurde eine Helferklasse entworfen, ConvertCoordinates, welche alle anfallenden Koordinatenkonvertierungen übernimmt. Über diese Klasse ist es möglich, sich die Koordinaten des einen Systems in die Koordinaten des anderen Systems überführen zu lassen. Des Weiteren können für den Fall, dass bei einer even-

tuellen Erweiterung des Projektes Rotationsmatrizen benötigt werden, auch diese hier erstellt werden. Um eventuelle Änderungen in der Zuordnung der Achsen, sowie der Koordinaten der Ursprünge leicht handhabbar zu machen, wurde es so eingerichtet, dass die zugrundeliegenden Daten in der Konfigurationsdatei geändert werden können.

5.1.1.3. Registrierte Sounds

Um die Registrierung von Sounds auf den WFS-Quellen zu speichern, hält jede SoundSource eine Liste von RoutedSounds. Beim RoutedSound handelt es sich um eine Klasse, welche zur Datenspeicherung dient. Sie beinhaltet die Variablen:

- *string:* buffername
- *bool:* loop
- *float:* volume

Es ist somit möglich zu speichern, welche Sounds auf welcher Quelle registriert sind und ob diese sich wiederholen. Außerdem ist es möglich, die Lautstärke zu speichern, damit ist alles gegeben, um die Umgebung permanent speichern und später genau wieder herstellen zu können.

5.1.2. Speichern und Laden von Szenen

Das Speichern der aktuellen Szene wird auch über den SourceManager angestoßen, da hier alle Informationen zum Speichern der Szene vorhanden sind. Von hieraus wird dann eine JSON-Repräsentation der Szene erstellt. Diese enthält die Daten zu jeder aktiven Quelle und allen auf diesen registrierten Sounds.

Über den SourceManager ist es auch möglich, Szenen zu laden. Hierfür wird der SourceManager auf den Zustand keiner aktiven Quellen und keiner registrierten Sounds zurückgesetzt. Anschließend wird die entsprechende Szene-Datei eingelesen und die entsprechenden SoundSources erstellt. Zum Parsen der Szene-Datei im JSON Format wird die Bibliothek LitJSON[2] verwendet. Diese ermöglicht es leicht über die eingelesenen Daten zu iterieren und diese zu verarbeiten.

5.1.3. OSCManager

Analog zum Entwurf des SourceManagers (s. Abschnitt 5.1.1) ist der OSCManager zur Verwaltung des OSC-Nachrichtenverkehrs zuständig. Ebenso wie beim SourceManager handelt es

[2] https://lbv.github.io/litjson/

sich beim OSCManager um ein Singleton, um Nebeneffekte durch Mehrfachinstanziierung zu vermeiden. Der OSCManager stellt die Anbindung an die benutzte OSC-Bibliothek (s. Abschnitt 5.1.3.2) her. Zum Senden der Nachrichten werden entsprechende Methoden angeboten. Es ist möglich OSC-Nachrichten zu senden um …

- … sich mit dem visuellen oder dem Rendering Stream zu verbinden oder die Verbindung zu trennen
- … Quellen zu aktivieren oder zu deaktivieren.
- … den Quelltyp zu ändern.
- … die Position und Ausrichtung von Quellen zu ändern.
- … den Namen einer Quelle zu ändern.
- … die Position des Hörers zu ändern.
- … die Wiedergabe von Sounds zu starten.
- … die Wiedergabe von Sounds zu stoppen.
- … Sounds auf Quellen zu registrieren oder aus der Registrierung zu entfernen.
- … verfügbare Sounds zu erhalten.
- … registrierte Sounds zu erhalten.

Alle für das Senden und Empfangen von OSC-Nachrichten verwendeten OSC-Adressen sind in der Konfigurationsdatei im Bereich „oscAdresses“ änderbar. Hierdurch wird eine leichte Anpassungsfähigkeit an eine sich ändernde Systemlandschaft gewährleistet.

5.1.3.1. Verarbeitung eingegangener OSC Nachrichten

Das Empfangen der Nachrichten wird durch eine weitere Klasse, die ProcessReceivedOSCMessages-Klasse, ausgelöst, da der OSCManager nicht vom MonoBehaviour erben soll. Klassen, die von MonoBehaviour erben, müssen an einem oder mehreren GameObjects aufgehängt sein. Dies widerspricht mit dem Konzept des Singletons. Das Auslösen des Empfangsprozesses kann sehr gut über ein MonoBehaviour geregelt werden, weshalb die Klasse ProcessReceivedOSCMessages vom MonoBehaviour erbt. Von hieraus wird dann die bei jedem Update, sprich bei jeder Erzeugung eines neuen Bildes, die Receive-Methode des OSCManagers aufgerufen. Diese prüft, ob neue Nachrichten eingegangen sind und verarbeitet eingetroffene Nachrichten.

Für das Verarbeiten der eingehenden OSC-Nachrichten greift der OSCManager auf den SourceManager zu. Hier werden dann entsprechend der eingehenden Nachrichten die Daten an den SoundSources gesetzt, Sounds als zur Verfügung stehend eingetragen, sowie Sounds an den Quellen registriert.

5.1.3.2. Benutzte OSC-Bibliotheken

5.1.3.2.1. UnityOSC von Jorge Garcia Für Unity gibt es bereits eine Bibliothek namens UnityOSC[3] von Jorge Garcia, welche benutzt werden kann, um sowohl OSC-Nachrichten zu empfangen, als auch zu senden. Diese Bibliothek wurde zunächst für dieses Projekt benutzt. UnityOSC wird eingebunden, indem man die Klasse OSCHelper, sowie den Ordner Editor in den Assets Ordner des Unity-Projektes kopiert. Die Verbindungen, sogenannte Clients und Server, werden dann in der OSCHelper Klasse erstellt und können von hier angesprochen werden.

Die Bibliothek wurde später durch eine andere Bibliothek ausgetauscht, da UnityOSC nach jedem Abholen eines UDP-Paketes den Thread 10 ms schlafen gelegt hat. Dies entspricht in etwa dem Zeitraum den Unity benötigt um einen neuen Frame zu rendern. Das bedeutet aber, dass falls häufiger als alle 10 ms Nachrichten empfangen werden, sich eine immer länger werdende Warteschlange an UDP-Paketen bildet. Dies führt dann dazu, dass die Latenz vom Absenden von Nachrichten bei cWonder bis zum Verarbeiten der Nachrichten im VR-Interface stetig zunimmt. Da es aber recht leicht passieren kann, dass häufiger als alle 10 ms Nachrichten eintreffen, da cWonder alle eintreffenden Nachrichten, also auch die vom VR-Interface ausgesendeten, an alle im Stream registrierten Empfänger, also auch an das VR-Interface, weiter sendet, ist dies ein Problem.

Naheliegend wäre es den Zeitraum des Schlafens des Threads zu verkürzen, da die Bibliothek aber immer nur ein Paket abholt und dieses als das zuletzt empfangene Paket zugänglich macht und die Referenz auf dieses Paket beim nächsten Empfangen überschreibt, ist nicht mehr gewährleistet, dass alle Nachrichten die empfangen werden, auch verarbeitet werden.

5.1.3.2.2. CSharp_OSC von Stefan Heidtmann Stefan Heidtmann, ein Master-Student im AudioLab an der HAW Hamburg, hat ebenfalls eine OSC-Bibliothek entwickelt, welche in Unity einsetzbar ist. Diese löst das oben beschriebene Problem, indem die OSC-Nachrichten asynchron empfangen werden. Des Weiteren können alle neuen Nachrichten abgefragt werden, nicht nur die letzte. Hierdurch werden alle eintreffenden Nachrichten umgehend im VR-Interface bearbeitet [Hei].

[3] https://github.com/jorgegarcia/UnityOSC

Hier ergibt sich bislang allerdings noch ein Problem beim Empfangen eines sehr großen xmlDumps. Der xmlDump ist eine XML-Repräsentation aller WFS-Quellen, welche von cWonder versendet wird, wenn man sich beim visualStream anmeldet. Ab ca. 40 WFS-Quellen wird dieser xmlDump so groß, dass das UDPSocket beim Empfangen der Nachricht eine Exception wirft. Dies wird momentan so gehandhabt, dass dann das Empfangen neu gestartet wird, um wenigstens noch die anderen Nachrichten empfangen zu können.

5.1.4. User Interface

Das User Interface besteht aus mehreren Teilen. Grundsätzlich ist es so aufgeteilt, dass um den linken Controller herum alle Arten von Interaktionsmöglichkeiten eingeblendet werden, wohingegen der rechte Controller zur Interaktion mit ebendiesen genutzt wird. Alle Knöpfe werden somit mit dem rechten Controller bedient, der rechte Controller wird benutzt um Quellen zu rotieren oder zu positionieren. Er dient ebenfalls zur Auswahl von Quellen.

5.1.4.1. Layout

Das User Interface ist um den linken Controller herum angeordnet und setzt sich grundsätzlich aus drei verschiedenen Seiten zusammen. Die Seiten können mit Hilfe des Trackpads auf dem linken Controller um den Controller herum rotiert werden, sodass alle Knöpfe jederzeit gut erreichbar sind. Alle Seiten des User Interfaces nebeneinander gelegt sind in Abbildung 5.3 zu sehen.

Abbildung 5.3.: Alle drei Seiten des User Interfaces nebeneinander gelegt.

Es ist für manche Funktionalitäten, wie z.B. das Registrieren von Sounds, notwendig, dass eine Auswahlliste angezeigt wird. Diese befindet sich immer auf der rechten Seite des Standard

User Interfaces und wird je nach Situation ein- bzw. ausgeblendet. Eine Auswahlliste wird für folgende Funktionalitäten eingeblendet:

- Registrieren eines Sounds auf einer Quelle (Seite „Soundbelegung“ oberste Reihe mittiger Knopf)
- Entfernen eines registrierten Sounds auf einer Quelle (Seite „Soundbelegung“ oberste Reihe rechter Knopf)
- Einen bestimmten Sound auf einer Quelle abspielen (Seite „Soundbelegung“ mittige Reihe rechter Knopf)
- Die Wiedergabe eines bestimmten Sounds auf einer Quelle beenden (Seite „Soundbelegung“ untere Reihe rechter Knopf)
- Den Typ der Quelle ändern (Seite „Positionierung und Transformierung“ untere Reihe linker Knopf)
- Laden einer Szene (Seite „Projektsteuerung“ obere Reihe mittlerer Knopf)

Für die Funktion der Lautstärkeeinstellung (Seite „Soundbelegung“ obere Reihe linker Knopf) hingegen wird ein Schieberegler eingeblendet, welcher über das Trackpad des rechten Controllers bewegt werden kann.

Für das Umbenennen einer Quelle (Seite „Positionierung und Transformierung“ untere Reihe mittlerer bzw. rechter Knopf), sowie für das Speichern einer Szene (Seite „Projektsteuerung“ obere Reihe rechter Knopf) wird neben dem Standard User Interface eine Tastatur eingeblendet, welche aus dem Unity Assetstore stammt und mittels des rechten Controllers bedient werden kann.

Das Ein- und Ausblenden der entsprechenden Teile des User-Interfaces wird durch das sogenannte UIScript geregelt.

5.1.4.2. Verarbeitung des User-Inputs

Der User-Input kann je nach Situation sehr unterschiedlich ausfallen. Alle Aktionen die mit Klicken eines Knopfes auf dem User Interface zu tun haben, werden durch die Klasse UIButtonClickHandler gehandhabt. Von hieraus werden dann die notwendigen Aktionen angestoßen. Einige wichtige Aktionen sind beispielsweise das Verhalten des rechten Controllers zu ändern, die Erstellung einer neuen Quelle beim SoundManager anzustoßen, oder das Selektierwerkzeug des rechten Controllers einzuschalten.

Das Selektieren von Quellen (s. Abbildung 5.4), ist eine sehr wichtige Funktion, welche wie folgt benutzt wird: Der Nutzer wählt auf dem User Interface das Selektierwerkzeug (Seite „Positionierung und Transformierung“ obere Reihe mittlerer Knopf) aus. Aus dem rechten Controller kommt nun ein blauer Strahl. Mit diesem Strahl wird auf eine Quelle gezielt. Ist die Quelle getroffen, so wird sie mit einem dünnen grünen Streifen um die Kontur herum hervorgehoben. Ist eine Quelle hervorgehoben kann sie mithilfe des Triggers des rechten Controllers selektiert werden. Falls mehrere Quellen von dem Strahl getroffen werden, werden diese alle hervorgehoben. Jedoch sind sie bis auf eine mit einem halb so dicken grünen Streifen hervorgehoben. Nur die Quelle mit dem dickeren Streifen kann selektiert werden. Möchte man ändern, welche Quelle mit einem dickeren Streifen hervorgehoben wird, so ist das mit Hilfe des Trackpads des rechten Controllers möglich. Wird hier auf die obere Hälfte des Trackpads geklickt, wird die Quelle, die weiter vom Benutzer entfernt ist, dicker hervorgehoben. Wird auf die untere Hälfte geklickt, verhält es sich umgekehrt.

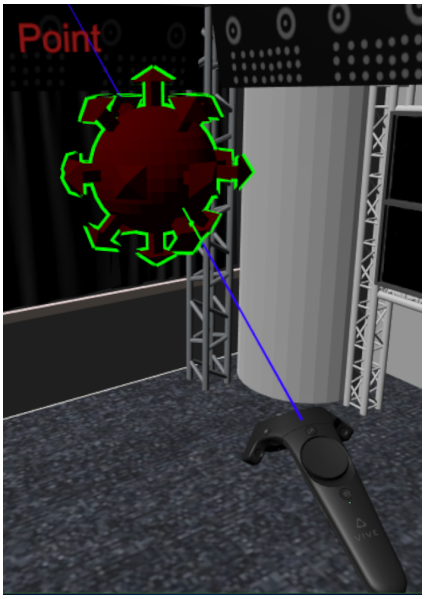


Abbildung 5.4.: Der Vorgang des Selektierens einer Quelle im Zustand des Hervorhebens

Ist eine Quelle selektiert, kann diese z.B. in ihrer Rotation oder Position manipuliert werden. Dies geschieht, indem die entsprechende Funktion auf dem User Interface gewählt wird. Dabei wird dem Controller eine neue Funktionsweise zugewiesen. Wird nun mit dem Finger auf dem Trackpad des rechten Controllers rotiert, so wird je nach gewählter Funktion entweder

- die Quelle um sich selbst rotiert,
- die Quelle um den Nutzer rotiert oder
- die Distanz vom Nutzer zur Quelle verändert.

Beim Wählen einer anderen Funktion, oder dem Rotieren des User Interfaces, wird die Funktionalität auf dem rechten Controller wieder zurückgesetzt, um versehentliche Manipulationen der Quellen zu vermeiden.

Das Verarbeiten der Eingabe aus einer Auswahlliste erfolgt so, dass beim Erstellen der Liste über ein Enum festgelegt wird, welche Aktion beim Wählen eines Eintrages aus der

Auswahlliste ausgeführt werden soll. Die Kombination des Textes des gewählten Eintrags mit der vorher festgelegten Aktion gibt genau an, was zu geschehen hat. Ist beispielsweise die gewählte Aktion PicklistAction.Route, so wird nach der Wahl des Eintrags der Sound mit dem entsprechenden Namen des Eintrags auf der momentan selektierten Quelle registriert.

Analog zur Verarbeitung der Eingaben aus einer Auswahlliste heraus funktioniert die Eingabe durch die Tastatur. Auch hier wird über ein Enum festgelegt, welche Aktion ausgeführt werden soll.

5.1.5. User Tracking

Um die Position des Nutzers, also des Hörers, der WFS-Anlage bekannt machen zu können, muss die Position des Nutzers ermittelt, in das Koordinatensystem der WFS-Anlage konvertiert und über OSC-Nachrichten übermittelt werden. Diese Aufgabe übernimmt die Klasse SendHMDPositionToWFS. Diese Klasse erbt von der Klasse MonoBehaviour und ist an dem GameObject, welches das HMD darstellt, aufgehängt.

5.1.6. ConfigReader

Um das VR-Interface leichter an eine sich ändernde Systemlandschaft anpassbar zu gestalten, wurde eine Konfigurationsdatei erstellt. Diese befindet sich im Ordner Assets/Resources/Config und heißt Labor_in_Unity.conf. Bei der Konfigurationsdatei handelt es sich um eine Textdatei im JSON-Format.

In der Konfigurationsdatei sind folgende Daten enthalten:

- Alle für das Projekt relevanten OSC-Adressen
- Die Verbindungsdetails zur WFS-Anlage sowie zum OSCVRServer
- Der Port auf dem Unity auf OSC-Nachrichten hören soll
- Die Einheitsmatrizen für die Umrechnung der Koordinatensysteme
- Die Koordinaten der Ursprünge des jeweils anderen Koordinatensystems

Die gesamte ursprüngliche Konfigurationsdatei ist im Anhang unter A.1 zu finden.

5.2. OSCVRServer

Beim OSCVRServer handelt es sich um einen eigens entwickelten auf OSC-Nachrichten basierenden Service, welcher dazu dient, WAV- und AIFF-Dateien auf verschiedenen Audiokanälen auszugeben.

5.2.1. Motivation

Verschiedene Audiodateien auf unterschiedlichen Audiokanälen auszugeben ist über viele Programme möglich, davon sind im WFS-Labor unter anderem Ableton Live 9 und Cubase 9 vertreten. Diese Programme lassen sich zwar über Netzwerknachrichten steuern, jedoch ist es nicht möglich, die Ausgabe auf die Audiokanäle über Netzwerknachrichten zu verändern. Da dies aber zwingend notwendig für die Umsetzung dieses Projektes ist, kam die Benutzung dieser Programme hier nicht in Frage.

5.2.2. Umsetzung

Der OSCVRServer ist in einem SuperCollider Skript implementiert. Dieses startet den in SuperCollider integrierten Server und setzt die Anzahl an gewünschten Audioausgangskanälen auf dem Server um. Hierfür ist ein Neustart des Servers nötig. Nach diesem Neustart werden dann die WAV- und AIFF-Dateien in einem vom Nutzer angebbaren Ordner eingelesen und in Buffern gespeichert. Anschließend werden die OSC-Nachrichten definiert, auf die der Server reagieren wird.

5.2.3. OSC Nachrichten

Im OSCVRServer sind einige OSC-Nachrichten definiert. Wie diese definiert werden können, ist in dem Abschnitt 5.2.3.1 beschrieben. Eine Übersicht, sowie eine kurze Erklärung darüber, was diese OSC-Nachrichten bewirken, ist im Abschnitt 5.2.3.2 zu finden.

5.2.3.1. Definition von OSC-Nachrichten in SuperCollider

In SuperCollider können OSC-Nachrichten, auf die reagiert werden soll, wie folgt definiert werden:

```
1 OSCdef.new(
2   \bezeichner,
3   {
4     Funktion welche beim Eintreffen
```

```
    der Nachricht ausgeführt wird
  },
  '/OSCAdresse'
);
```

Listing 5.1: Grundlegende Definition von OSC-Nachrichten in SuperCollider

Weitere Informationen sind in der SuperCollider Dokumentation[4] zu finden.

```
OSCdef.new(
  \testMsg,
  {
    arg msg, time, addr, port;
    msg.postln;
  },
  '/testMsg'
);
```

Listing 5.2: Beispiel einer Definition einer OSC-Nachricht in SuperCollider

Die OSC-Nachrichtendefinition in Listing 5.2 bewirkt, dass die gesendete Nachricht inklusive aller übergebenen Parameter auf der Konsole der SuperCollider-Instanz ausgedruckt wird.

5.2.3.2. Definierte OSC-Nachrichten

Um mit dem OSCVRServer zu interagieren werden OSC-Nachrichten verwendet. Der OSCVR-Server empfängt diese Nachrichten und antwortet im Falle der */OSCVR/getAvailableSoundnames* Nachricht sowie der */OSCVR/getRoutedSounds* Nachricht ebenfalls mit einer OSC-Nachricht. Die im OSCVRServer definierten OSC-Nachrichten sind in Listing 5.3 aufgeführt. Diese gliedern sich grob in vier Gruppen:

1. Das Abfragen der momentanen Situation (siehe Absatz 5.2.3.2.1)
2. Das Routen und Unrouten von Sounds auf Ausgangskanäle (siehe Absatz 5.2.3.2.2)
3. Sounds abspielen (siehe Absatz 5.2.3.2.3)
4. Das Abspielen von Sounds stoppen (siehe Absatz 5.2.3.2.4)

```
/OSCVR/getRoutedSounds
/OSCVR/getAvailableSoundnames
```

[4] http://doc.sccode.org/Classes/OSCdef.html

```
/OSCVR/routeToChannel
/OSCVR/unroute
/OSCVR/playBufferAtChannel
/OSCVR/playChannel
/OSCVR/playAll
/OSCVR/stopBufferAtChannel
/OSCVR/stopChannel
/OSCVR/stopAll
```

Listing 5.3: Die im OSCVRServer definierten OSC-Nachrichten

5.2.3.2.1. Abfragen der momentanen Situation

Um die momentane Situation erfassen zu können ist es notwendig, die momentan registrierten Sounds sowie die verfügbaren Sounds, abfragen zu können.

/OSCVR/getRoutedSounds

Ersteres kann mit der OSC-Nachricht */OSCVR/getRoutedSounds* geschehen. Die Nachricht nimmt zwei Parameter entgegen:

- *string*: IP-Adresse, an die die Antwort gesendet werden soll
- *int*: Port, an den die Antwort gesendet werden soll

Hieraus wird eine Antwort-Nachricht generiert, welche als einzigen Parameter einen String enthält, der eine XML Repräsentation der registrierten Sounds beinhaltet. Ein Beispiel ist in Listing 5.4 zu sehen. Die Adresse der Antwort-Nachricht lautet: */OSCVR/routedSounds* und wird an die IP-Adresse und den Port gesendet, welche der Nachricht */OSCVR/getRoutedSounds* mitgegeben wurden.

```
<routedSounds>
  <routedSound buffername="piano.wav" channelid="20" loop="1" />
  <routedSound buffername="rain.wav" channelid="21" loop="1" />
  <routedSound buffername="birds.wav" channelid="22" loop="1" />
  <routedSound buffername="brandung.wav" channelid="23" loop="1" />
  <routedSound buffername="crickets.wav" channelid="24" loop="1" />
</routedSounds>
```

Listing 5.4: Eine beispielhafte XML Repräsentation registrierter Sounds, welche als Antwort auf die OSC-Nachricht */OSCVR/getRoutedSounds* zu erwarten ist

/OSCVR/getAvailableSoundnames

Für das Abfragen verfügbarer Sounds gibt es die OSC-Nachricht */OSCVR/getAvailableSoundnames*, welche folgende zwei Parameter entgegen nimmt:

- *string*: IP-Adresse, an die die Antwort gesendet werden soll
- *int*: Port, an den die Antwort gesendet werden soll

Als Antwort darauf wird eine OSC-Nachricht an die angegebene IP-Adresse und den angegeben Port mit der OSC-Adresse */OSCVR/availableSounds* gesendet. Diese enthält als Parameter:

- *string*: Eine XML Repräsentation verfügbarer Sounds, wie sie in Listing 5.5 zu sehen ist. Ein Sound gilt als available, insofern er eine WAV- oder AIFF-Datei ist und diese in dem Verzeichnis liegt, welches beim Starten des OSCVRServers eingelesen wird.

```
<sounds>
  <sound name="birds.wav"/>
  <sound name="brandung.wav"/>
  <sound name="crickets.wav"/>
  <sound name="forest.wav"/>
  <sound name="frogs.wav"/>
  <sound name="frogs2.wav"/>
  <sound name="gimme-verse.wav"/>
  <sound name="piano.wav"/>
  <sound name="rain.wav"/>
  <sound name="sparrows.wav"/>
  <sound name="view-island.wav"/>
</sounds>
```

Listing 5.5: Eine beispielhafte XML Repräsentation verfügbarer Sounds, welche als Antwort auf die OSC-Nachricht */OSCVR/getAvailableSoundnames* zu erwarten ist

5.2.3.2.2. Routen und Unrouten von Sounds

Um einen Sound auf einem Ausgangskanal abspielen zu können, muss dieser auf dem entsprechenden Kanal geroutet werden. Dies ist über die OSC-Nachricht */OSCVR/routeToChannel* möglich. Möchte man einen Sound nicht mehr auf einem Ausgangskanal ausgeben, so kann dies über die Nachricht */OSCVR/unroute* erreicht werden.

/OSCVR/routeToChannel

Die OSC-Nachricht */OSCVR/routeToChannel* nimmt als Parameter:

- *string*: Name der Sound-Datei
- *int*: ID des Ausgangskanals
- *int*: Wiederholung (0 = keine Wiederholung, 1 = mit Wiederholung)

Anschließend wird der entsprechende Buffer gesucht, welcher mit dem Inhalt der entsprechenden Datei gefüllt wurde und dieser auf dem Ausgangskanal mit der angegebenen ID mit der entsprechenden Wiederholungs-Einstellung registriert. Wie das Routing genau funktioniert ist im Abschnitt 5.2.4.0.1 nachzulesen.

/OSCVR/unroute

Die OSC-Nachricht */OSCVR/unroute* nimmt zwei Parameter an:

- *string*: Name der Sound-Datei
- *int*: ID des Ausgangskanals

Nach dem Eintreffen dieser OSC-Nachricht, wird der entsprechende Buffer gesucht und dieser auf dem Ausgangskanal entfernt. Die genaue Umsetzung ist im Abschnitt 5.2.4.0.1 erläutert.

5.2.3.2.3. Sounds abspielen

Um Sounds abzuspielen gibt es drei Möglichkeiten.:

1. Einen bestimmten Sound auf einem Ausgangskanal abspielen
2. Alle auf einem bestimmten Ausgangskanal registrierten Sounds abspielen
3. Alle auf allen Ausgangskanälen registrierten Sounds abspielen

Für alle Nachrichten gilt, dass vorher der entsprechende Buffer auf dem Kanal registriert werden muss, sonst wird dieser nicht wiedergegeben. Des Weiteren gilt: sollte der Buffer bereits auf dem Kanal abgespielt werden, wird diese Wiedergabe beendet, bevor mit der neuen Wiedergabe begonnen wird.

/OSCVR/playBufferAtChannel

Möchte man einen bestimmten Sound auf einem Ausgangskanal abspielen, ist dies die OSC-Nachricht, welche benutzt werden sollte. Es können drei Parameter übergeben werden, wobei es auch möglich ist, nur die ersten beiden Parameter zu übergeben, dann wird für den dritten Parameter der Default-Wert verwendet. Die Parameter sind:

- *string*: Name der Sound-Datei, welche im Buffer eingelesen wurde der abgespielt werden soll.
- *int*: ID des Ausgangskanals, auf dem die Wiedergabe stattfinden soll.
- *float*: Lautstärke in prozentualer Schreibweise. (1 entspricht 100%) (Default: 1)

/OSCVR/playChannel

Soll ein ganzer Ausgangskanal mit allen auf ihm registrierten Sounds abgespielt werden, ist diese Nachricht zu verwenden. Es können zwei Parameter übergeben werden, wobei es sich bei dem Lautstärke Parameter wie oben um eine optionale Angabe handelt. Die Parameter sind:

- *int*: ID des Ausgangskanals, auf dem die Wiedergabe stattfinden soll.
- *float*: Lautstärke in prozentualer Schreibweise. (1 entspricht 100%) (Default: 1)

/OSCVR/playAll

Wenn alle Ausgangskanäle abgespielt werden sollen, ist die OSC-Nachricht */OSCVR/playAll* benutzbar. Diese nimmt einen optionalen Parameter, bei dem es sich wie bei den oben aufgeführten OSC-Nachrichten um die Lautstärke handelt.

5.2.3.2.4. Abspielen von Sounds stoppen

Im obigen Abschnitt wurde erklärt, wie Sounds abgespielt werden können. Analog dazu verhält sich das Beenden von Wiedergaben.

/OSCVR/stopBufferAtChannel

Diese Nachricht kann verwendet werden, um den in den Parametern spezifizierten Sound nicht mehr auf dem ebenfalls in den Parametern definierten Ausgangskanal wiederzugeben. Die Parameter sind hierbei:

- *string*: Name der Sound-Datei, welche im Buffer eingelesen wurde der gestoppt werden soll.

- *int*: ID des Ausgangskanals, auf dem die Wiedergabe beendet werden soll.

/OSCVR/stopChannel

Möchte man erreichen, dass ein ganzer Ausgangskanal nicht mehr abgespielt wird, kann diese Nachricht benutzt werden. Hierbei wird ein Parameter vom Typ *int* übergeben, welcher angibt, um welchen Ausgangskanal es sich handelt.

/OSCVR/stopAll

Um alle Wiedergaben auf allen Ausgangskanälen zu beenden, kann diese parameterlose OSC-Nachricht genutzt werden.

5.2.4. Channel Routing

Um das Routing von Sounds auf die Ausgangskanäle, sowie die Wiedergabe von ebendiesen zu steuern, wurde SuperCollider um eine weitere Klasse erweitert. Diese heißt SoundChannelRouting, wird pro Ausgangskanal einmal instanziiert und übernimmt die Kontrolle über diesen Ausgangskanal.

5.2.4.0.1. Routing

Die Klasse SoundChannelRouting bietet die Möglichkeit, Buffer hinzuzufügen, sowie zu entfernen.

Beim Hinzufügen eines Buffers ist mit anzugeben, ob der Buffer, wenn er wiedergegeben wird, in einer Wiederholungsschleife wiedergegeben werden soll. Falls sowohl ein Buffer, als auch ein Wert für die Wiederholungsschleife angegeben sind, wird der Buffer entsprechend in die Liste der registrierten Buffer übernommen. Von nun an kann er auf dem Kanal abgespielt werden.

Das Entfernen von Buffern sorgt dafür, dass eventuell laufende Wiedergaben mit diesem Buffer angehalten werden und der Buffer aus der Liste der verfügbaren Buffer entfernt wird. Sollte der Buffer vorher gar nicht auf dem Kanal registriert gewesen sein, so geschieht nichts.

5.2.4.0.2. Wiedergabe von Sounds

Um die Wiedergabe der registrierten Sounds zu kontrollieren gibt es vier Methoden, die aufgerufen werden können. Davon sind jeweils zwei zum Starten bzw. zum Stoppen der Wiedergabe.

Um eine Wiedergabe zu starten bestehen zwei Möglichkeiten:

1. Einen bestimmten Buffer starten

2. Alle registrierten Buffer starten

Für beide Varianten ist es möglich, eine relative Lautstärke mitzugeben. Sollte eine Wiedergabe des wiederzugebenden Buffers bereits laufen, so wird diese abgebrochen und neu begonnen.

Für das Stoppen einer Wiedergabe besteht ebenfalls die Möglichkeit die Wiedergabe eines bestimmten Buffers oder aller Buffer zu beenden.

6. Benutzbarkeit

Dieses Kapitel beschäftigt sich mit der Benutzbarkeit des entwickelten VR-Interfaces. Um die Benutzbarkeit besser beurteilen zu können, wurde eine kleine Studie durchgeführt.

An dieser Studie haben acht Personen teilgenommen, welche einen Altersdurchschnitt von ca. 34,5 Jahren hatten. Die jüngste Person war 25 Jahre alt, die älteste 59 Jahre. Von den Teilnehmern waren sechs männlich und zwei weiblich. Im Durchschnitt hatten die Teilnehmer 19,5 Jahre Erfahrung mit Computern. Von den Teilnehmern hatten zwei schon viel Erfahrung mit VR sammeln können, zwei hatten etwas Erfahrung mit VR und vier hatten durch dieses VR-Interface ihre erste Berührung mit dem Thema VR.

Zur Durchführung der Studie wurde ein Fragebogen entworfen, welcher im Anhang einzusehen ist.

6.1. User-Interface

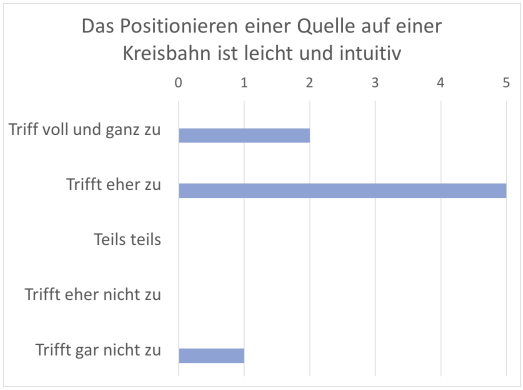
Der Fragebogen zur Benutzbarkeit enthält hauptsächlich Fragen zum User Interface, da dies für die Bewertung der Benutzbarkeit sehr wichtig ist. Die Antworten der Teilnehmer wurden ausgewertet. Die ausgewerteten Daten werden in den nächsten Abschnitten vorgestellt und interpretiert. Hierfür werden einige Auswertungsgrafiken gezeigt. Alle Auswertungsgrafiken, auch die hier nicht aufgeführten, sind im Anhang in der Abbildung A.2 auf Seite 61 aufgeführt.

6.1.1. Positionierung der Quellen

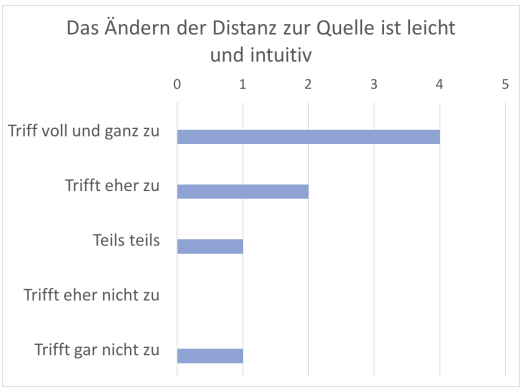
Dieser Abschnitt befasst sich mit der Positionierung von WFS-Quellen auf einer Kreisbahn um den Nutzer herum, sowie dem Ändern der Distanz zwischen dem Nutzer und der WFS-Quelle.

6.1.1.1. Positionierung auf einer Kreisbahn

Abbildung 6.1a zeigt die Auswertung zur Positionierung einer Quelle auf einer Kreisbahn um den VR-Interface-Nutzer. Die Mehrheit der Nutzer empfand das Positionieren dieser Art als eher leicht und intuitiv. Lediglich ein Proband beschrieb die Interaktion als nicht leicht und



(a) Grafische Darstellung der Auswertungsergebnisse zur Positionierung auf einer Kreisbahn



(b) Grafische Darstellung der Auswertungsergebnisse zur Änderung der Distanz zwischen dem Nutzer und der WFS-Quelle

Abbildung 6.1.: Grafische Darstellung der Auswertungsergebnisse zur Positionierung von WFS-Quellen

intuitiv. Dies war allerdings auf einen Fehler zurückzuführen, der den Nutzer dazu zwang, die Trackpads beider Controller zu verwenden. Dieser Fehler war bis dato noch nicht aufgetreten und konnte nicht rekonstruiert werden.

6.1.1.2. Positionierung in der Distanz

Das Ändern der Distanz zwischen einer WFS-Quelle und dem Nutzer empfanden 75% der Probanden als eher leicht und intuitiv oder besser. Eine Person empfand es als teils leicht und intuitiv und teils nicht. Lediglich ein Proband empfand es als nicht leicht und intuitiv. Dies ist ebenfalls, wie oben beschrieben, auf einen nicht rekonstruierbaren Fehler zurückzuführen.

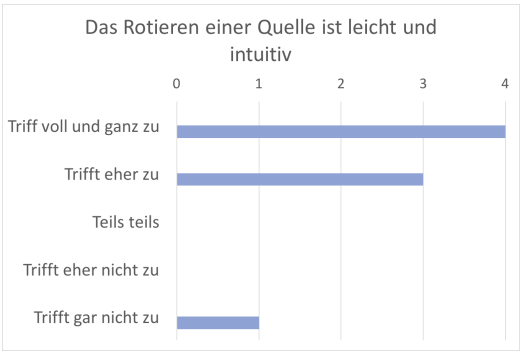


Abbildung 6.2.: Grafische Darstellung der Auswertungsergebnisse zum Rotieren von Quellen

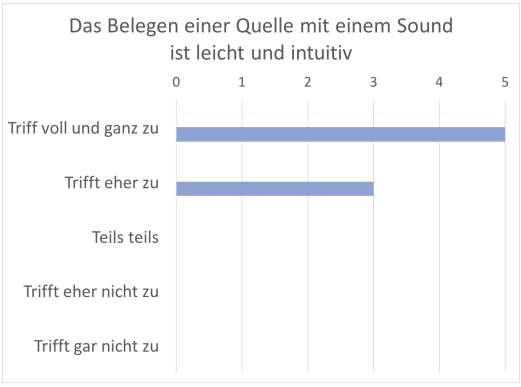
6.1.2. Rotation der Quellen

Das Rotieren der WFS-Quellen wurde von sieben Personen als eher leicht und intuitiv oder besser empfunden, wie Abbildung 6.2

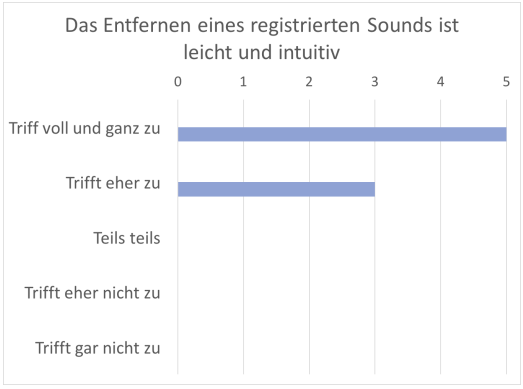
zu entnehmen ist. Auch hier ist die eine Einschätzung als nicht leicht und intuitiv auf den nicht rekonstruierbaren Fehler zurückzuführen.

6.1.3. Belegung der Quellen mit Sound

Das Belegen der Quellen mit Sound und das Entfernen von registrierten Sounds an WFS-Quellen erfolgt jeweils über eine Auswahlliste in der entweder alle verfügbaren Sound oder alle bisher registrieren Sounds aufgeführt werden.



(a) Grafische Darstellung der Auswertungsergebnisse zum Registrieren von Sounds



(b) Grafische Darstellung der Auswertungsergebnisse zum Entfernen registrierter Sounds

Abbildung 6.3.: Grafische Darstellung der Auswertungsergebnisse zum Belegen der Quellen mit Sound

6.1.3.1. Sound auf Kanal registrieren

Das Registrieren von Sounds auf einer Quelle wurde von drei Personen als eher leicht und intuitiv wahrgenommen. Die weiteren fünf Probanden haben es voll und ganz als leicht und intuitiv empfunden, wie der Abbildung 6.3a zu entnehmen ist.

6.1.3.2. Sound auf Kanal entfernen

Ebenso wie das Registrieren von Sounds, wurde das Entfernen bereits registrierter Sounds von allen Probanden als eher leicht und intuitiv oder besser empfunden, wie in Abbildung 6.3b zu sehen ist.

6.1.4. Selektieren von WFS-Quellen

Das Selektieren von WFS-Quellen wurde von allen Probanden als leicht und intuitiv eingestuft. Dies kann der Auswertungsgrafik A.2a im Anhang entnommen werden.

6.1.5. Weitere Anmerkungen

Auf dem Fragebogen zur Benutzbarkeit wurde den Probanden die Möglichkeit geboten Anmerkungen abzugeben.

Hier wurden folgende Anmerkungen abgegeben, wobei ähnliche Anmerkungen zusammen gefasst wurden:

- „Laser zur Auswahl bei Tastaturen benutzen, um Buchstaben leichter anvisieren zu können."
- „Das Schriftbild der Menüs ist nicht sehr scharf."
- „Bei der Auswahl des Sounds etc. aus längeren Listen, hat man das Gefühl, dass man den Controller höher halten muss um genau zu treffen."
- „Häufiger trifft man die Funktion, die auf der Rückseite der ausgewählten Funktion liegt."
- „Es wäre gut, wenn man die Lautstärke beim Abspielen komplett oder selektiv ändern könnte."
- „Ich vergaß häufig bei der Werkzeugauswahl den „Trigger" zu verwenden."
- „Das Highlighting ob ein Knopf gedrückt werden kann ist nicht konsequent."
- „Icons für Soundquelle rotieren und Sound registrieren/Sound abmelden überarbeiten."
- „Minimale Distanz zum User beim Positionieren."
- „Kontextmenüs sollten kleiner sein und ggf. im Winkel zu Menüs stehen."
- „UI-Buttons brauchen teilweise Tooltips."
- „Selektionsmodus sollte auch in Soundmenü wählbar sein."
- „Touchpad sollte nicht beide Pads brauchen."
- „Touchpad-Interaktionen: eher Richtung als Rotation (intuitiver)."
- „Menüs sollten anzeigen, ob eine Interaktion das Touchpad oder eine andere Funktion der Wands braucht."

6.1.5.1. Umgesetzte Anmerkungen

Folgende Punkte aus den Anmerkungen wurden umgesetzt bzw. verbessert.

- „Laser zur Auswahl bei Tastaturen benutzen, um Buchstaben leichter anvisieren zu können."
  - Es wurde ein Laserpointer hinzugefügt, der angezeigt wird, sobald auch die Tastatur angezeigt wird. Mit diesem fällt das anvisieren der Knöpfe auf der Tastatur deutlich leichter.
  - Mit dem Laserpointer ist es außerdem möglich, den OK-Knopf zu drücken. Bei diesem ist es aber auch weiterhin möglich, ihn durch das Berühren mit dem rechten Controller und gleichzeitigem betätigen des Triggers zu drücken.
- „Das Schriftbild der Menüs ist nicht sehr scharf."
  - Das Schriftbild wurde so verändert, dass eine größere Schrift gewählt wurde und diese dann auf die gewünschte Größe runter skaliert wurde. Dies hat deutlich schärfere Kanten beim Schriftbild zur Folge.
- „Bei der Auswahl des Sounds etc. aus längeren Listen, hat man das Gefühl, dass man den Controller höher halten muss um genau zu treffen."
  - Das Auswählen von Funktionen und Auswahllistenelementen wurde insofern verändert, als dass vor dem rechten Controller jetzt ein Kegel existiert. Mit diesem muss man den Knopf berühren. Hierdurch konnte die Fläche, mit der man eine Auswahl trifft, stark verringert werden. Dies bewirkt, dass die Auswahl deutlich präziser getroffen werden kann. Außerdem ist nun klar ersichtlich, mit welchem Teil des Controllers man die Funktion treffen muss.
- „Häufiger trifft man die Funktion, die auf der Rückseite der ausgewählten Funktion liegt."
  - Das User-Interface wurde so angepasst, dass die Knöpfe, welche nicht auf der Oberseite des User-Interfaces liegen, nicht mehr ausgewählt werden können.
- „Das Highlighting ob ein Knopf gedrückt werden kann ist nicht konsequent."
  - Das Hervorheben der Knöpfe wurde auch in den Auswahllisten eingefügt.
- „Icons für Soundquelle rotieren und Sound registrieren/Sound abmelden überarbeiten."
  - Das Icon für die Rotation der Quelle um den Nutzer wurde insofern angepasst, als dass noch eine stilisierte Person in die Mitte des Rotationskreises eingefügt wurde.

  - Die Icons für das Registrieren von Sounds und das Entfernen der Registrierung wurden ersetzt. Vorher handelte es sich um einen Lautsprecher mit einem „+“ und einem „-“ daneben. Dies war leicht zu verwechseln mit einer Lautstärke-Einstellung. Die Icons wurden durch eine stilisierte Datei, in der eine Musiknote enthalten ist, ersetzt. Neben der stilisierten Datei ist jeweils ein „+“ oder ein „-“, je nachdem, ob es sich um den Registrieren oder den Registrierung entfernen Knopf handelt.

- „Kontextmenüs sollten kleiner sein und ggf. im Winkel zu Menüs stehen.“
  - Die Auswahlliste wurde verkleinert, näher an der restlichen UI platziert und um weitere 15° angewinkelt, sodass diese jetzt in einem 30° Winkel zur UI Oberfläche steht.

- „UI-Buttons brauchen teilweise Tooltips.“
  - Unterhalb des Titels jeder User-Interface-Seite wurde ein Textfeld eingefügt. In dieses wird, wenn man einen Knopf berührt, sodass dieser farblich hervorgehoben wird, ein kleiner Erklärungstext geschrieben.

- „Menüs sollten anzeigen, ob eine Interaktion das Touchpad oder eine andere Funktion der Wands braucht.“
  - Innerhalb des Tooltips wird bei den Knöpfen, die keine Auswahlliste öffnen, Tastatur anzeigen oder unmittelbare Aktionen, wie z.B. das Erstellen einer neuen Quelle, ausführen angezeigt, welche Interaktion vorzunehmen ist.

- „Selektionsmodus sollte auch in Soundmenü wählbar sein.“
  - Der Selektionsmodus kann nun zusätzlich über den Grip Button des rechten Controllers jederzeit aktiviert und deaktiviert werden.

6.1.6. Zusammenfassung

Insgesamt wurden alle gemessenen Funktionen zu 88,75% als eher leicht und intuitiv oder besser wahr genommen. Die 88,75% setzen sich aus 56,25% „leicht und intuitiv“ und 32,5% „eher leicht und intuitiv“ zusammen. Die restlichen 12,25% teilen sich in 7,5% „teils teil“ und 3,75% „nicht leicht und intuitiv“ auf.

Die 3,75% Einschätzungen als „nicht leicht und intuitiv“ wurden alle von einer Person abgegeben, welche aufgrund eines nicht reproduzierbaren Fehlers beide Trackpads anstatt nur eines nutzen musste, um die Funktionen auszuführen.

7. Fazit

Für diese Bachelorarbeit wurde ein VR-Interface entwickelt, welches es einem Nutzer ermöglicht, die WFS-Anlage an der HAW Hamburg zu steuern. Das VR-Interface wurde in Unity entwickelt und wird durch ein SuperCollider-Skript namens OSCVRServer unterstützt. Der OSCVRServer bietet dem VR-Interface die Möglichkeit, Sounds auf bestimmten Ausgangskanälen zu registrieren, diese Registrierung wieder aufzuheben, Kanäle oder einzelne Sounds auf bestimmten Kanälen mit einer bestimmten Lautstärke wiederzugeben und diese Wiedergaben zu beenden.

Dem Nutzer des VR-Interfaces wird die Möglichkeit geboten, sich WFS-Szenen anzusehen und diese in vielerlei Hinsicht zu manipulieren. Des Weiteren können Szenen gespeichert und geladen werden, sodass man Szenen über mehrere Sessions hinweg bearbeiten und verfeinern kann. Über die Steuerung lassen sich unter anderem die Position, die Rotation, der Name der WFS-Quelle manipulieren, indem das VR-Interface die Nutzereingaben in eine Kommunikation mit der Steuerungssoftware cWonder übersetzt. Analog dazu werden auch die Nutzereingaben, die das Registrieren, das Entfernen einer Registrierung, das Starten von Soundwiedergaben, sowie das Stoppen dieser betreffen, in eine Kommunikation mit dem OSCVRServer übersetzt.

Außerdem wurde noch eine Benutzbarkeitsstudie durchgeführt. Diese ergab, dass das hier entwickelte VR-Interface von der großen Mehrheit (88,75% der Probanden) als recht leicht und intuitiv zu benutzen eingeschätzt wurde.

Im Allgemeinen lässt sich sagen, dass die Entwicklung des VR-Interfaces gezeigt hat, dass sich eine WFS-Anlage gut über ein VR-Interface steuern lässt, da so zusätzlich zu der akustischen Wahrnehmung, der visuelle Sinn des Nutzers angesprochen wird. Hierdurch gelingt es leicht die WFS-Quellen an die gewünschten Positionen zu bringen. Des Weiteren wäre eine Belegung der Quellen über eine Gestensteuerung recht kompliziert und musste daher durch ein externes Programm vollzogen werden. Da dies über das VR-Interface ermöglicht wird, bietet das den hohen Komfort nicht zwischen verschiedenen Programmen und/oder Arbeitsplätzen wechseln zu müssen.

8. Ausblick

Es gibt mehrere mögliche Erweiterungen zu dem hier entwickelten VR-Interface. Zum einen wäre es möglich, eine Zeitkomponente in die Szene einzuarbeiten, sodass der Nutzer bestimmen kann, dass eine Aktion zu einer bestimmten Zeit in der Szene stattfinden soll und wie lange die Durchführung der Aktion dauern soll. Dies wäre für die Positionierung der Quellen eine interessante Erweiterung, da man so automatisierte Bewegungsabläufe in die Szene bringen könnte. Eine weitere Möglichkeit der Nutzung, die sich hierüber ergeben würde, wäre das Erstellen einer Szene, welche ein Hörspiel mit mehreren Personen, die sich bewegen und zu bestimmten Zeiten sprechen, darstellt.

Die Erweiterung der Zeitkomponente müsste dann auch in die Speicherung von Szenen mit einfließen. Außerdem müsste eine Kompatibilität mit älteren Szenen, ohne eine solche Zeitkomponente, hergestellt werden, sodass die Szenen die der Nutzer vor der Umstellung gespeichert hat, nicht verloren sind.

Zum anderen wäre es eine mögliche Erweiterung zu visualisieren, wenn ein Sound auf einer WFS-Quelle abgespielt wird. Dies könnte zum Beispiel darüber passieren, dass sich die Welle der Planarquellen-Repräsentation bewegt, oder indem man eine sich bewegende Welle, welche das Signal des abgespielten Audiosignals repräsentiert, einblendet. Hierfür müsste eine Synchronisation zwischen VR-Interface und dem OSCVRServer erfolgen.

Des Weiteren könnte der OSCVRServer entsprechend so umgebaut werden, dass nicht nur die Dateien in einem bestimmten Ordner auf dem Rechner auf dem der OSCVRServer läuft zur Wiedergabe zur Verfügung stehen. Es wäre eine gute Funktionalität, wenn man auch Audiodateien über das Netzwerk übertragen könnte, um diese dann wiedergeben zu können. Dies würde die Erweiterbarkeit der Szenen verbessern, da dann neue Audiodateien leicht hinzugefügt werden könnten. Die Audiodateien müssten dann allerdings über ein anderes Netzwerkprotokoll als OSC übertragen werden, da dies nicht dafür ausgelegt ist, Dateien zu übertragen.

Außerdem wäre es eine schöne Erweiterung des Projektes dem Nutzer noch mehr verschiedene VR-Umgebungen zur Verfügung zu stellen, sodass der Nutzer, falls es sich bei der WFS-Szene um eine Naturszene handelt, in eine VR-Wald-Umgebung eintauchen kann. Hier

können beliebig viele verschiedene Umgebungen denkbar sein, sodass es für jede denkbare WFS-Szene eine entsprechende VR-Umgebung gibt.

A. Anhang

A.1. Zu Kapitel 5 Abschnitt 5.1.6

```
{
  "oscAdresses":{
    "visualConnectAdress": "/WONDER/stream/visual/connect",
    "rendererConnectAdress": "/WONDER/stream/render/connect",
    "visualDisconnectAdress": "/WONDER/stream/visual/disconnect",
    "rendererDisconnectAdress": "/WONDER/stream/render/disconnect",
    "activateAdress": "/WONDER/source/activate",
    "deactivateAdress": "/WONDER/source/deactivate",
    "typeAdress": "/WONDER/source/type",
    "angleAdress": "/WONDER/source/angle",
    "positionAdress": "/WONDER/source/position",
    "colorAdress": "/WONDER/source/color",
    "nameAdress": "/WONDER/source/name",
    "listenerAdress": "/WONDER/listener/position",
    "xmlDumpAdress": "/WONDER/project/xmlDump",
    "maxNoSourcesAdress": "/WONDER/global/maxNoSources",
    "groupIdAdress": "/WONDER/source/groupId",
    "dopplerEffektAdress": "/WONDER/source/dopplerEffect",
    "streamConnectedAdress": "/WONDER/stream/connected",
    "playChannelAdress": "/OSCVR/playChannel",
    "playBufferAtChannelAdress": "/OSCVR/playBufferAtChannel",
    "playAllAdress": "/OSCVR/playAll",
    "stopChannelAdress": "/OSCVR/stopChannel",
    "stopBufferAtChannelAdress": "/OSCVR/stopBufferAtChannel",
    "stopAllAdress": "/OSCVR/stopAll",
    "routeAdress": "/OSCVR/routeToChannel",
    "unrouteAdress": "/OSCVR/unroute",
    "getAvailableSoundNamesAdress": "/OSCVR/getAvailableSoundnames",
    "availableSoundsAdress": "/OSCVR/availableSounds",
    "getRoutedSoundsAdress": "/OSCVR/getRoutedSounds",
```

```
    "routedSoundsAdress": "/OSCVR/routedSounds"
  },
  "oscconnections":{
    "wfs":{
      "ip":"192.168.14.100",
      "port":58100
    },
    "oscvr":{
      "ip":"192.168.14.100",
      "port": 57120
    }
  },
  "oscserver":{
    "vive":{
      "port":58585
    }
  },
  "unitmatrices":{
    "towfs":[[0,1,0],[0,0,1],[1,0,0]],
    "fromwfs":[[0,0,1],[1,0,0],[0,1,0]]
  },
  "origins":{
    "wfs":[0,2.5,0],
    "vive":[-2.5,0,0]
  }
}
```

Listing A.1: Inhalt der ursprünglichen Konfigurationsdatei

A.2. Zu Kapitel 6

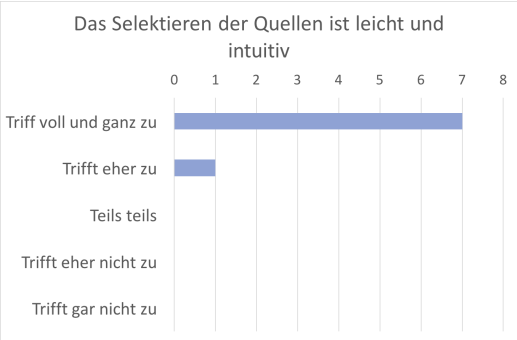
Fragebogen Benutzbarkeit

1) Fragen zum User Interface:	Triff voll und ganz zu	Trifft eher zu	Teils teils	Trifft eher nicht zu	Trifft gar nicht zu
1a) Positionierung und Transformierung					
Das Selektieren der Quellen ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Rotieren einer Quelle ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Positionieren einer Quelle auf einer Kreisbahn ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Ändern der Distanz zur Quelle ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Ändern des Quelltyps ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Umbenennen einer Quelle ist leicht und intuitiv	☐	☐	☐	☐	☐
1b) Soundbelegung					
Das Belegen einer Quelle mit einem Sound ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Entfernen eines registrierten Sounds ist leicht und intuitiv	☐	☐	☐	☐	☐
1c) Projektsteuerung					
Das Speichern einer Szene ist leicht und intuitiv	☐	☐	☐	☐	☐
Das Laden einer gespeicherten Szene ist leicht und intuitiv	☐	☐	☐	☐	☐

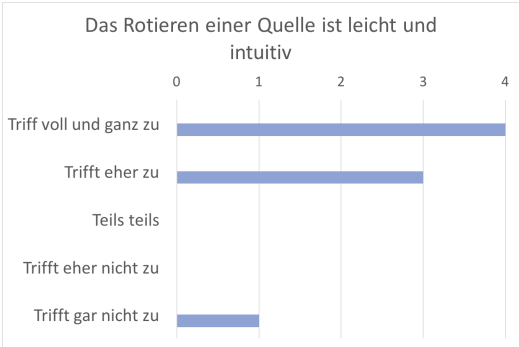
2) Anmerkungen:

3) Angaben zur Person
Alter:
Geschlecht:
Beruf:
Computererfahrung in Jahren:

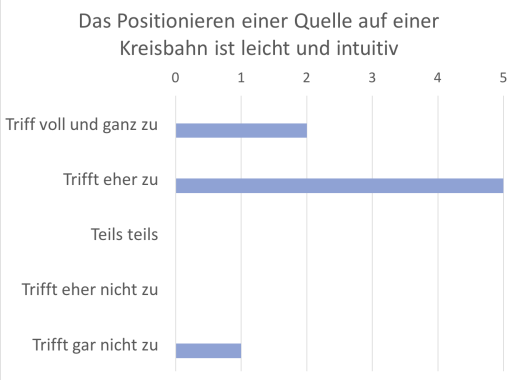
Abbildung A.1.: Der Fragebogen zur Benutzbarkeit



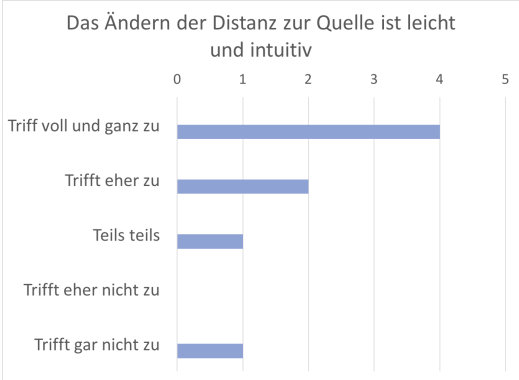
(a) Grafische Darstellung der Auswertungsergebnisse zum Selektieren von WFS-Quellen



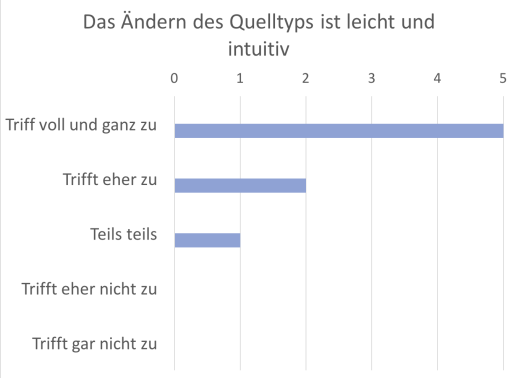
(b) Grafische Darstellung der Auswertungsergebnisse zum Rotieren einer WFS-Quelle



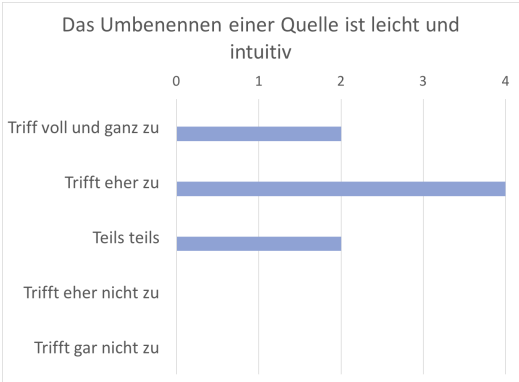
(c) Grafische Darstellung der Auswertungsergebnisse zum Positionieren auf einer Kreisbahn



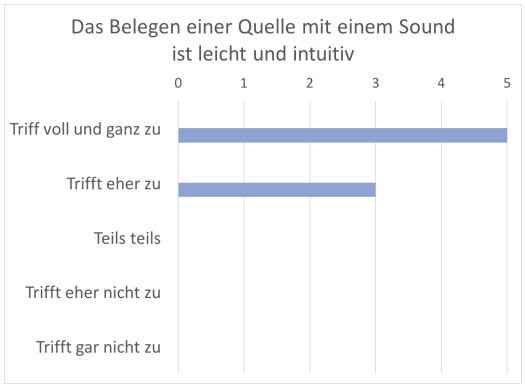
(d) Grafische Darstellung der Auswertungsergebnisse zum Ändern der Distanz zu einer WFS-Quelle



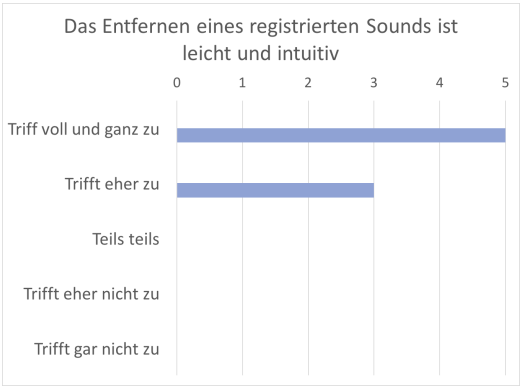
(e) Grafische Darstellung der Auswertungsergebnisse zum Ändern des Quelltyps



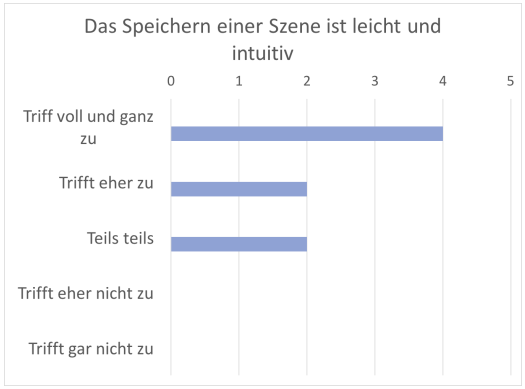
(f) Grafische Darstellung der Auswertungsergebnisse zum Umbennenen einer WFS-Quelle



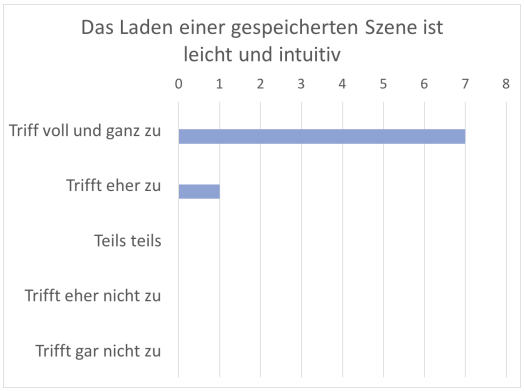
(g) Grafische Darstellung der Auswertungsergebnisse zum Registrieren von Sounds



(h) Grafische Darstellung der Auswertungsergebnisse zum Entfernen registrierter Sounds



(i) Grafische Darstellung der Auswertungsergebnisse zum Speichern einer Szene



(j) Grafische Darstellung der Auswertungsergebnisse zum Laden einer gespeicherten Szene

Abbildung A.2.: Grafische Darstellung aller Auswertungsergebnisse

Glossar

HMD Ein Head Mounted Display ist ein Monitor, welcher am Kopf befestigt ist, sodass optischer Inhalt direkt vor den Augen des Nutzers wiedergegeben werden kann. 63

VR-Brille Eine Brille, an der ein HMD angebracht ist, welches zur Wiedergabe von VR-Inhalten genutzt werden kann. 61

Ableton Live 9 Eine Digital Audio Workstation von der Firma Ableton AG. Weitere Informationen sind auf der Website von Ableton [Abl] zu finden. 39

Cubase 9 Eine Digital Audio Workstation von der Firma Steinberg. Weitere Informationen sind auf der Website von Steinberg [Ste] zu finden. 39

Digital Audio Workstation Ein Programm, welches einem das Aufnehmen, Bearbeiten und Produzieren von Audio-Inhalten ermöglicht. 61

Enum Ein Enum ist das in Programmiersprachen gebräuchliche Konzept eines Aufzählungstyps. 37, 38

HTC Vive Eine von der Firma HTC entwickelte und vermarktete VR-Brille, welche zusätzlich zu der Brille, inklusive zwei Controllern und zwei Lighthouses kommt. 1, 7, 16, 61

Lighthouses Die von der HTC Vive und der Controller benutzten Infrarot-Sendeeinheiten. Diese senden Infrarot Blitze und Fächer aus, die von der HTC Vive und den Controllern benutzt werden, um ihre Position und Rotation zu bestimmen. 7, 61

SuperCollider Eine Programmiersprache, um Echtzeit Audiosynthese zu betreiben. Weitere Informationen sind auf der Website von SuperCollider [Supb] zu finden. ix, 8, 39, 40, 45, 53

VR Virtual Reality (zu deutsch: „virtuelle Realität“) bezeichnet die Nutzung von Computer Modellen und Simulationen, um einem Nutzer die Interaktion mit einer virtuellen dreidimensionalen Welt zu ermöglichen (vgl. [Low15]). 63

Abkürzungen

ART-Trackingsystem Advanced Realtime Trackingsystem. viii, 6, 7

GUI Graphical User Interface. 15

HMD Head Mounted Display. Siehe: HMD. 11, 12, 16, 38, 61, 63

OSC Open Sound Control. v, ix, 8, 18, 23, 28, 32–34, 38–45, 54

UI User Interface. 15

UML Unified Modeling Language. viii, 29

VR Virtual Reality. Siehe: VR. v, viii, 1, 2, 10, 13–16, 18–21, 23–47, 53–55, 61

WFS Wellenfeldsynthese. iii–v, vii, viii, 1, 2, 4–7, 9–11, 15, 18–32, 35, 38, 39, 47–50, 53–55, 59

Literaturverzeichnis

[Abl] Ableton: *Ableton Live 9 Website*. https://www.ableton.com/de/live/, Abruf: 07.06.2017

[Bad13] *Kapitel* W. Fohl: The Wave Field Synthesis Lab at the HAW Hamburg. In: Bader, Rolf: *Sound - Perception - Performance*. Springer International Publishing, 2013

[BP04] Baalman, Marije A. ; Plewe, Daniel: WONDER-a software interface for the application of Wave Field Synthesis in electronic music and interactive sound installations. In: *ICMC*, 2004

[DBGJ13] Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul ; Jung, Bernhard: *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Springer-Verlag, 2013

[EB16] Encyclopædia Britannica, The E.: *Huygens' principle*. Encyclopædia Britannica. https://www.britannica.com/science/Huygens-principle. Version: November 2016, Abruf: 24.06.2017

[FN13] Fohl, Wolfgang ; Nogalski, Malte: A Gesture Control Interface for a Wave Field Synthesis System. In: *NIME*, 2013, S. 341–346

[FW15] Fohl, Wolfgang ; Wilk, Eva: Enhancements to a Wave Field Synthesis System to Create an Interactive Immersive Audio Environment. In: *3rd Int. Conf. on Spatial Audio VDT*, 2015

[Gam95] Gamma, Erich: *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995

[Hei] Heidtmann, Stefan: *Persönliche Mitteilung*

[Her17] Herivel, John: *Christiaan Huygens*. Encyclopædia Britannica. https://www.britannica.com/biography/Christiaan-Huygens. Version: April 2017, Abruf: 24.06.2017

[liga] *Analysis of Valve's 'Lighthouses' Tracking System Reveals Accuracy.* – http://www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/

[ligb] *Lighthouses von VALVE erklärt.* https://vrjump.de/lighthouse-erklaert, Abruf: 29.06.2017

[Low15] Lowood, Henry E.: *virtual reality (VR).* https://www.britannica.com/technology/virtual-reality. Version: Mai 2015, Abruf: 28.06.2017

[MDB06] Moreau, Sébastien ; Daniel, Jérôme ; Bertet, Stéphanie: 3D sound field recording with higher order ambisonics–Objective measurements and validation of a 4th order spherical microphone. In: *120th Convention of the AES*, 2006, S. 20–23

[Nog12] Nogalski, Malte: *Gestengesteuerte Positionierung von Klangquellen einer Wellenfeldsynthese-Anlage mit Hilfe eines kamerabasierten 3D-Tracking-Systems*, HAW Hamburg, Bachelorarbeit, 2012

[Oel] Oellers, Helmut: *Wave Field Synthesis Physical principle.* Website. http://www.syntheticwave.de/Wavefieldsynthesis.htm, Abruf: 28.06.2017

[Opea] OpenSopundControl: *Introduction to OSC.* Website. http://opensoundcontrol.org/introduction-osc, Abruf: 28.06.2017

[Opeb] OpenSoundControl: *The Open Sound Control 1.0 Specification.* Website. http://opensoundcontrol.org/spec-1_0, Abruf: 28.06.2017

[PD15] Preim, Bernhard ; Dachselt, Raimund: *Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces.* Springer-Verlag, 2015

[Ste] Steinberg: *Cubase 9 Website.* https://www.steinberg.net/de/products/cubase/start.html, Abruf: 07.06.2017

[Supa] SuperCollider: *SuperCollider Github.* Website. http://supercollider.github.io/, Abruf: 28.06.2017

[Supb] SuperCollider: *SuperCollider Website.* http://www.audiosynth.com/, Abruf: 07.06.2017

[SW97] Slater, Mel ; Wilbur, Sylvia: A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments. In: *Presence: Teleoperators and virtual Environments* 6 (1997), Dezember, S. 603–616.

[Ver97] Verheijen, Edwin: *Sound Reproduction by Wave Field Synthesis*, Technische Universiteit Delft, Diss., 1997

[viva] http://www.htc.com/managed-assets/shared/desktop/vive/Vive_PRE_User_Guide.pdf

[vivb] https://www.vive.com/de/product/

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 24. August 2017 Felix Baumgartner