Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorarbeit

**Malte Nogalski**

## Gestengesteuerte Positionierung von Klangquellen einer Wellenfeldsynthese-Anlage mit Hilfe eines kamerabasierten 3D-Tracking-Systems

*Fakultät Technik und Informatik*
*Studiendepartment Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Malte Nogalski

# Gestengesteuerte Positionierung von Klangquellen einer Wellenfeldsynthese-Anlage mit Hilfe eines kamerabasierten 3D-Tracking-Systems

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. rer. nat. Wolfgang Fohl
Zweitgutachter: Prof. Dr. rer. nat. Kai von Luck

Eingereicht am: 30. August 2012

**Malte Nogalski**

**Thema der Arbeit**

Gestengesteuerte Positionierung von Klangquellen einer Wellenfeldsynthese-Anlage mit Hilfe eines kamerabasierten 3D-Tracking-Systems

**Stichworte**

Gestensteuerung, Mensch-Maschine-Interaktion, MMI, Bewegungsverfolgung, Wellenfeldsynthese, Echtzeitbedienung

**Kurzzusammenfassung**

In dieser Arbeit wird ein Softwaresystem zur Interpretation von Gesten, mit dem Ziel Soundquellen einer Wellenfeldsynthese-Anlage zu positionieren, entwickelt, implementiert und ausgewertet. Es wird ein Überblick über verschiedene Trackingverfahren und -systeme und über die Wellenfeldsynthese-Anlage gegeben sowie die Anforderungen erläutert, die ein solches System an ein Trackingsystem, zum Erfassen der Bewegungen hat. Vor allem wird auf die Herausforderungen systematischer und implementationstechnischer Natur eingegangen, welche die Entwicklung gestellt hat.

**Malte Nogalski**

**Title of the paper**

Gesture-controlled positioning of sound sources in a wave field synthesis system using of a camera-based 3D-tracking-system

**Keywords**

Gesture Control, Human-Machine-Interaktion, HMI, Motion Tracking, Wave Field Synthesis, Real-Time Control

**Abstract**

This paper dokuments the development, implementation and evaluation of a softwaresystem to interpret gestures, with the aim to position sound sources of a wave field synthesis system. An overview over different tracking methods and systems and over the wave field synthesis system is given, as well as it explaines the requirements, that such a system has to a tracking system. It mainly deals with the challenges of systematic and implementative natures which were given by the development.

# Inhaltsverzeichnis

# 1. Einleitung

## 1.1. Themenbereich eingrenzen

## 1.2. Begriffserklärung

### 1.2.1. Geste/Gestensteuerung

Gesten bieten einem die Möglichkeit, sich nonverbal und dennoch präzise auszudrücken. Dabei liegt der Schwerpunkt der Aussagekraft meist in den Händen und den Armen, jedoch kann der ganze Körper zum Gestikulieren eingesetzt werden. Eine Gestensteuerung ermöglicht es somit ein Computersystem durch Gesten zu bedienen.

Geste:

> Gebärde, sprechende Bewegung.

[WR06]

Geste:

> Gebärde, die Rede begleitende Ausdrucksbewegung des Körpers, besonders der Arme und Hände.

[Dud90]

### 1.2.2. Wellenfeldsynthese-Anlage

Eine WFS-Anlage synthetisiert Schallwellen jeweils aus einer Summe einzelner Schallwellen. Die entstehende synthetische Schallwelle ähnelt einer natürlichen Schallwelle, welche ihren Ursprung an einem Ort hätte, an dem sich in Wirklichkeit kein schallemittierendes Objekt befindet. Die WFS-Anlage ermöglicht es, die Ursprünge dieser synthetischen Schallwellen frei zu platzieren.

Synthese:

> Zusammenfügung, Verknüpfung [einzelner Teile zu einem höheren Ganzen].

[Dud90]

### 1.2.3. 3D-Tracking-System

Tracking (zu deutsch: »Verfolgung«) steht für das Erfassen der Position eines Objektes oder Subjektes über einen Zeitraum und somit der Bewegung dieses Objektes bzw. Subjektes. Ein Jäger »trackt« ein Tier (das Ziel) in der Wildnis, um es aufzuspüren. Dafür liest der Jäger die Spur, welche das Tier hinterlässt und erkennt an der Ausrichtung der Spur auch die Bewegungsrichtung. Ein Trackingsystem oder auch Tracker[1] setzt den Vorgang des »Trackens« um. Es erkennt die Position des Ziels und kann es somit verfolgen, wenn es sich bewegt. Ein 3D-Tracking-System erfasst dabei die Position im dreidimensionaler Raum ($\mathbb{R}^3$) und wird benötigt, um die genaue Position zu erfassen, wenn das Ziel in seiner Bewegung nicht lediglich an die Dimensionen einer Ebene gebunden ist, wie das Ziel des Jägers. Einige Tracker sind ebenfalls in der Lage, zusätzlich zur Position ebenfalls die Ausrichtung des Ziels zu erkennen.

## 1.3. Motivation

Gesten, zum Steuern von Computersystemen, gewinnen in der Mensch-Maschine Kommunikation zunehmend an Bedeutung. Mit ihnen wird eine intuitivere und damit leichtere Bedienung einer Vielzahl von verschiedensten Systemen angestrebt, als sie mit herkömmlichen Eingabegeräten möglich sind [DD11, PSS97, GNHF12]. Im Auto könnten Gesten dabei helfen, bei der Bedienung der Bordinstrumente nicht so sehr den Blick von der Straße abwenden zu müssen [ACBH00]. In der Unterhaltungselektronik wird versucht, einige Funktionen der üblichen Fernbedienung auf Gesten zu übertragen, um die Fernbedienung nicht mehr jederzeit zur Hand haben zu müssen [Sam12].

In anderen Situationen geht der Einsatz von Gesten über einen Komfortgewinn hinaus. In Operationssälen, wo besondere hygienische Bedingungen gelten, ermöglichen Gesten eine sterile Eingabemöglichkeit [CL09]. Alten oder kranken Menschen, welchen die nötige Mobilität fehlt, können durch Gesten neue Möglichkeiten geschaffen werden, um mit Computersystemen zu interagieren und somit selbstständiger zu bleiben [Joh10].

Gesten kommen also in immer mehr Systemen in den verschiedensten Anwendungsgebieten zum Einsatz. In die Reihe dieser Systeme soll sich, mit dieser Arbeit, nun auch die WFS-Anlage der Hochschule für angewandte Wissenschaften Hamburg (HAW Hamburg) einreihen, wie sie in Abschnitt 1.2.2 schon kurz angesprochen wurden und in den Kapiteln 2 und 4 präziser beschrieben wird.

[1] Trackingsystem und Tracker wird im Folgenden synonym verwendet.

### 1.3.1. Bedienmöglichkeiten einer Wellenfeldsynthese-Anlage

Im Auslieferungszustand enthält die WFS-Anlage die Softwarekomponente xWonder, welche eine graphische Oberfläche bietet, um die WFS-Quellen zu modifizieren (verschieben, drehen, usw.). Hierdurch lässt sich die WFS-Anlage zwar in Echtzeit bedienen, jedoch nur mit einer WFS-Quelle zur Zeit und vor allem ist es sehr schwierig, flüssige Bewegungen umzusetzen. Abgesehen von xWonder, lassen sich durch Skripte oder kleine Programme in verschiedenen Programmiersprachen (zum Beispiel, SuperCollider [WCC11]) alle erdenklichen Konstellationen und Bewegungen erzeugen, jedoch müssen diese im Vorwege erstellt werden. Beide Möglichkeiten haben einen sehr abstrakten Ansatz, der sich nicht direkt mit den Klängen der WFS-Anlage, sondern lediglich mit Repräsentanten auseinandersetzt.

### 1.3.2. Gesten als Eingabemethode

Gesten können es ermöglichen, bei der Bedienung der WFS-Anlage direkt mit den synthetisierten Schallwellen bzw. den virtuellen Ursprüngen zu interagieren. Wenn diese Ursprünge als reale Objekte angesehen werden, kann durch Gesten eine sehr intuitive Bedienung geschaffen werden, bei der sich die Benutzer nicht durch Visualisierungen zweidimensionaler Abstraktionen und der Interpretationsleistung ablenken lassen müssen.

### 1.3.3. Vorarbeit für weitere Arbeiten

Neben bzw. vor der Entwicklung und Implementierung der Gestensteuerung gilt es, den Tracker und die WFS-Anlage in einer effektiven Art zur Zusammenarbeit zu bringen. Dies kann der Grundbaustein für viele weitere Arbeiten sein, welche sich mit der Wellenfeldsynthese (WFS) und Positionen oder Orientierungen im Raum befassen. Dies könnten zum Beispiel weitere Bedienmöglichkeiten der WFS-Anlage sowie Programme zum Auswerten und Vergleichen der wahrgenommenen Klänge mit realen Positionen sein.

### 1.3.4. Realtimebedienung mehrerer Personen gleichzeitig

Eine Gestensteuerung, welche in der Lage ist, zwischen verschiedenen Akteuren zu unterscheiden, würde es ermöglichen, dass die WFS-Anlage durch mehrere Personen gleichzeitig in Echtzeit bedient werden kann.

## 1.4. Ähnliche Lösungsansätze

Mir sind zur Zeit keine weiteren Möglichkeiten oder Projekte bekannt, welche sich mit der Echtzeitbedienung einer WFS-Anlage mit Hilfe von Gesten beschäftigen. Im dem Artikel »Mixage mobile« beschreiben Delerue und Warusfel jedoch anhand zweier Beispiele, wie sich die Arbeit von Toningenieuren durch die Anwendung von Wellenfeldsynthese verändert. [DW06]

Es gibt etliche Ansätze, welche eine präzise Positionierung von WFS-Quellen ermöglichen, indem der Ablauf gänzlich im Vorfeld festgelegt wird (zum Beispiel Csound [Cle12], SuperCollider [M+12] oder VST-Plugins [Baa08, S. 52f.]). Andere Ansätze, welche meist ein Graphical User Interface (GUI) bieten, ermöglichen eine Echtzeitbedienung, welche leicht und komfortabel, dabei aber auch unpräziser oder unflexibler, ist, wie zum Beispiel xWonder oder auch Csound, indem hiermit eine GUI erstellt wird. Keiner dieser Ansätze bietet jedoch eine intuitive Echtzeitbedienung, wie sie mit dieser Arbeit angestrebt wird.

## 1.5. Abgrenzungen

In dieser Arbeit wird sich bei der Auswahl der Aktionen, für die Gesten erstellt werden sollen, lediglich auf die direkte Bedienung der WFS-Anlage beschränkt. Es werden dabei keine Möglichkeiten in Betracht gezogen, die Änderungen der WFS-Anlage oder des Trackers notwendig machen würden und es wird kein Einfluss auf die digitalen Quellen der Sounds genommen werden können, welche über die WFS-Anlage angespielt werden.

## 1.6. Aufbau der Arbeit

Im ersten Teil der Arbeit werden die benutzten Systeme vorgestellt. Für die WFS-Anlage soll lediglich ein Überblick über die Funktionsweise und vor allem die Bedienmöglichkeiten gegeben werden. Er soll der Nachvollziehbarkeit späterer Argumentationen in den Kapiteln der Entwicklung und Umsetzung dienen. Auf die Mathematik wird dabei gänzlich verzichtet, da die Berechnung der WFS für diese Arbeit irrelevant ist. Es werden die Anforderungen erläutert, welche die Arbeit an einen Tracker stellt, der für die Erfassung der Bewegungen der Gesten notwendig ist, einige verschiedene Trackingsysteme und schließlich der in dieser Arbeit benutzte ART-Tracker etwas präziser vorgestellt. Zuletzt wird kurz das Labor vorgestellt, in welchem sich die Systeme befinden und in dem die Arbeit entwickelt wurde.

Im zweiten Teil der Arbeit wird der Motion Tracker-Wave Field Synthesis-Connector (MoWeC) vorgestellt, welcher entwickelt und in Java implementiert wurde. Hier wurde die

Vorarbeit umgesetzt, welche nötig ist, um den Tracker und die WFS-Anlage so zu verbinden, dass die eigentliche Gestensteuerung darauf aufbauen kann.

Im dritten Teil der Arbeit wird kurz auf Gesten generell eingegangen. Danach wird die Entwicklung der Gesten behandelt, in der es darum geht, für welche speziellen Aktionen Gesten erstellt werden und wie diese aussehen sollen. Darauf folgt die technische Umsetzung der Gestensteuerung in Form der GestureComponent. Die Prinzipien stehen hier im Vordergrund und die konkrete Implementation wird daher vernachlässigt. Besonders beschäftigt sich das Kapitel mit den wesentlichsten Problemen und deren Bewältigung. Implementiert wurde dieser Teil ebenfalls in Java.

Zum Schluss wird noch ein kurzer Überblick, über die wichtigsten Erkenntnisse und ein Ausblick darüber gegeben, was die nächsten Schritte wären und welche umfangreicheren Vorhaben aus der Arbeit noch entstanden sind.

# 2. Wellenfeldsynthese

Dieses Kapitel bietet einen kurzen Einblick in WFS allgemein und in die WFS-Anlage von Fouraudio[1], welche für diese Arbeit benutzt wurde. Dabei wird bei den Berechnungen nicht ins Detail gegangen, da sich diese Arbeit nicht mit der WFS an sich beschäftigt, sondern mit der Positionierung der Soundquellen.

## 2.1. Motivation

Die Motivation für WFS besteht darin, eine möglichst realitätsnahe akustische Umgebung zu schaffen, welche vor allem nicht nur von einem spezifischen Punkt aus wahrgenommen werden kann. Dabei soll von den einzelnen Lautsprechern und deren Positionen abstrahiert werden, um die Soundquellen frei positionieren und in Echtzeit bewegen zu können. Anders als bei einfacheren Soundsystemen, bei denen ein Geräusch auf den einzelnen Kanälen explizit in der Lautstärke reguliert wird, um den Eindruck zu erwecken, dass es von einem Ort zwischen den Lautsprechern kommt, (vgl. »stereo panning«) [Tza10, And02, FT], wobei hier die Art des Algorithmus auch stark von der Anzahl der Kanäle und des Setups der Lautsprecher abhängt, wird das Augenmerk bei der WFS auf die konkrete Position gelenkt, von der das Geräusch ausgehen soll. Von jedem Ort innerhalb des Bereichs der WFS-Anlage soll das Geräusch dann an der Position lokalisiert werden können, anstatt nur von einem so genannten sweet spot[2] [Baa08, S. 11]

Einsatzmöglichkeiten wären zum Beispiel Testumgebungen, um konkrete akustische Gegebenheiten nachzustellen oder Highend-Entertainmentsysteme wie in Kinos.

## 2.2. Das Prinzip

Das Prinzip der WFS wird hier nur grob angeschnitten. Für ein tieferes Verständnis ist zum Beispiel die Dissertation von Marije Baalman [Baa08] zu empfehlen.

[1] `http://www.fouraudio.com/`

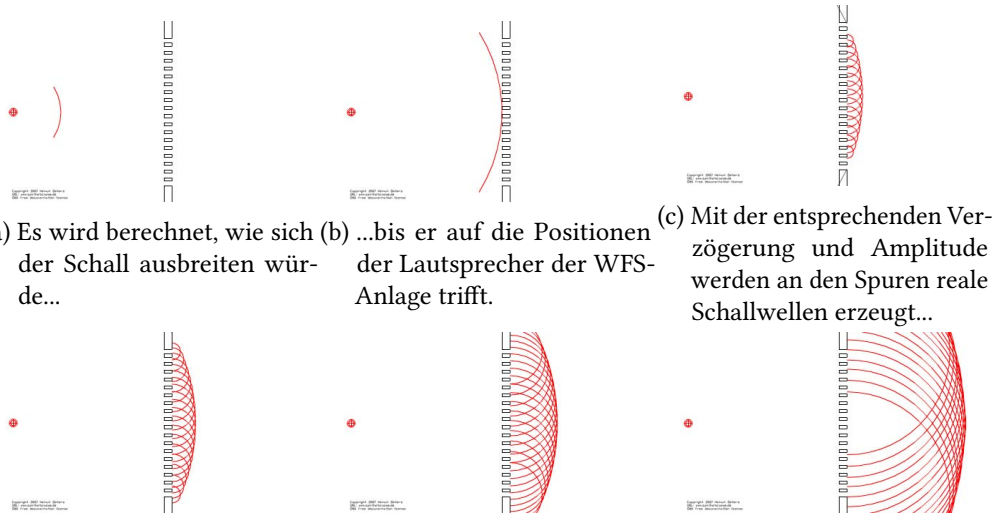
[2] Stereo- oder Surroundsysteme sind immer auf einen Punkt ausgerichtet, an dem die Akustik am besten ist. Je mehr die Position des Empfängers von dieser Position abweicht, desto mehr verfälscht sich die Wahrnehmung.

Die WFS baut auf dem von Christiaan Huygens entdecktem Huygens Prinzip auf, welches für die Ausbreitung jeglicher Wellen (Schall- sowie Wasser- oder Licht-Wellen etc.) gilt und besagt, dass jeder Punkt einer Wellenfront als Ausgangspunkt einer neuen Welle, der so genannten Elementarwelle, betrachtet werden kann. Die neue Lage der Wellenfront ergibt sich durch die Überlagerung sämtlicher Elementarwellen (Abb. 2.1). Zur Veranschaulichung und um der Realisierung der WFS-Anlage schon etwas näher zu kommen, stelle man sich eine massive, von vertikalen Schlitzen in gleichmäßigen Abständen, durchzogene Wand vor. Es befinde sich auf einer Seite dieser Wand eine reale Soundquelle, von der sich naturgemäß die Schallwellen ringförmig in alle Richtungen gleichmäßig und ungestört ausbreiten. Durch die Wand werden die Schallwellen jedoch durchbrochen. Lediglich durch die Schlitze dringen Stücke der originalen Schallwelle, welche sich auf der anderen Seite der Wand wieder in gleichem Maße weiter ausbreiten. Dies sind nun die Elementarwellen. Die vordersten Punkte jeder Elementarwelle (in Ausbreitungsrichtung betrachtet) stehen weiterhin in gleicher Relation zueinander, wie sie es vor der Wand taten. Die Elementarwellen fangen sehr bald nach ihrer Entstehung (abhängig von der Entfernung ihres Ursprunges) an, sich zu überschneiden und bilden somit in der Ausbreitungsrichtung der ursprünglichen Schallwelle wieder eine geschlossene Front. Der Ursprung der originären Soundquelle lässt sich nun anhand der synthetischen Schallwelle von der anderen Seite der Wand immer noch lokalisieren.

Die WFS-Anlage ersetzt nun die zuvor beschriebenen Schlitze in der Wand lediglich durch Lautsprecher. Jeder Schlitz wird durch eine Spur der WFS-Anlage vertreten. Die ursprünglich reale Soundquelle wird durch eine virtuelle WFS-Quelle an gleicher Position ersetzt und die Ausbreitung des Schalls ausgehend von der WFS-Quelle berechnet. Das Ergebnis der Berechnungen ergibt sich aus den Entfernungen der einzelnen Spuren zu der WFS-Quelle und der Verzögerung, mit der ein realer, von der WFS-Quelle ausgehender Schall bis zu der Spur benötigt hätte und mit welcher Amplitude er diese erreicht hätte. Anhand dieser Daten wird zum richtigen Zeitpunkt mit der richtigen Amplitude auf jeder Spur eine reale Elementarwelle erzeugt. Diese Elementarwellen verhalten sich exakt so, wie in dem Szenario mit der durch Schlitze durchzogenen Wand. Somit wurde durch die Elementarwellen eine neue reale Schallwelle synthetisiert (siehe Abb. 2.1).

Folgende Eigenschaften einer WFS-Quelle können modifiziert werden:

- Position
- Ausrichtung
- Typ
- Farbe
- Gruppen-Identifikation (ID)



(a) Es wird berechnet, wie sich der Schall ausbreiten würde...

(b) ...bis er auf die Positionen der Lautsprecher der WFS-Anlage trifft.

(c) Mit der entsprechenden Verzögerung und Amplitude werden an den Spuren reale Schallwellen erzeugt...

(d) ...welche sich kurz darauf anfangen zu überschneiden...

(e) ...und sich zu der synthetischen Schallwelle zu vereinigen.

(f) Die Summe der Elementarwellen bildet die reale Wellenfront.

Abbildung 2.1.: Ablauf der Synthese einer Schallwelle [syn].

- Rotationsrichtung
- Skalierungsrichtung
- Dopplereffekt

Einige Eigenschaften können zudem auf verschiedenen Wegen geändert werden. Zum Beispiel kann angegeben werden, dass die WFS-Quelle über einen bestimmten Zeitraum in gleichmäßiger Geschwindigkeit zu der neuen Position »wandern« soll. Eine komplette Aufstellung der Befehle, durch welche WFS-Quellen manipuliert werden können, befindet sich im Anhang (Kapitel C). Lediglich eine Eigenschaft soll an dieser Stelle präziser erläutert werden, der Typ. Dabei wird auch etwas auf die Ausrichtung eingegangen. Es gibt zwei verschiedene Typen von WFS-Quellen. Die WFS-Punktquelle und die WFS-Linearquelle. Der Typ bestimmt, wie sich der virtuelle (und somit im Endeffekt auch der reale) Schall der WFS-Quelle ausbreitet. Der virtuelle Schall einer WFS-Punktquelle breitet sich ringförmig in alle Richtungen gleichmäßig aus, wie es realer physikalischer Schall tun würde und wie es in Abbildung 2.1 dargestellt ist. Die Ausrichtung einer WFS-Punktquelle, deren Schall sich ohnehin gleichmäßig in alle Richtungen ausbreitet, hat somit keinerlei Effekt. Der Schall einer WFS-Linearquelle hingegen, breitet sich in einer Form aus, wie sie unter Schallwellen in der Natur eigentlich nicht vorkommt. Dabei bilden die Fronten der Elementarwellen nicht wie bei einer WFS-Punktquelle einen Bogen sondern eine Gerade, wie eine Welle, die einen

Ozean durchquert. Dieser Effekt ähnelt jedoch, lokal betrachtet, einer sehr weit entfernten Punktquelle bzw. WFS-Punktquelle. Hierdurch lässt sich der Ursprung des Schalls nicht auf einen konkreten Punkt bestimmen sondern lediglich auf eine Richtung einschränken. Die Richtung, aus der die Schallwelle wahrgenommen wird, ist jedoch unabhängig von der Position des Empfängers der Schallwelle. Da die Schallwelle sich nun nur in eine Richtung ausbreitet, legt die Ausrichtung der WFS-Quelle fest, aus welcher Richtung die Schallwelle kommt.

## 2.3. Technik

Die WFS-Anlage besteht aus dem WFS-Mac (Control Computer), dem WFS-Server, mehrere WFS-Nodes und einer Reihe von Audiomodulen [GMMW07]. Jedes dieser Module hat wiederum acht Spuren. Die Anzahl der WFS-Nodes und Audiomodule hängt von der Größe der WFS-Anlage ab. In dem Labor der HAW Hamburg enthält die WFS-Anlage zwei WFS-Nodes mit jeweils 13 Audiomodule und somit 208 Spuren.

Softwareseitig wird das open-source Softwarepaket sWONDER[3] [BP05, Baa04] verwendet. Es besteht aus mehreren Modulen, welche mittels Open-Sound-Control-Nachrichten (OSC-Nachrichten) kommunizieren und somit auf verschiedenen, sich in einem Netzwerk befindlichen, Rechnern laufen können [BHSK07]. In Abbildung 2.2 sind die Softwaremodule mit den jeweiligen Rechnern abgebildet, auf denen sie laufen. [Baa08, S. 50ff]

Von dem Control Computer wird die WFS-Anlage bedient. Hier befinden sich AppleScript-Programme zum Starten und Stoppen der WFS-Anlage, welche wiederum Skripte auf dem WFS-Server ausführen. Auf dem Control Computer laufen alle Programme, welche den Sound abspielen, der über die WFS-Anlage ausgegeben werden soll. Dies können beliebige Programme sein, die lediglich MP3s abspielen können oder in der Lage sind eine Vielzahl von einzeln manipulierbaren Spuren zu führen. Der Control Computer ist auch der einzige Computer, welcher an externe Netzwerke angeschlossen werden kann. Auf dem Control Computer ist das xWonder-Modul ausführbar, welches die grafische Benutzerschnittstelle der WFS-Anlage darstellt (Abb. 2.3).

cwonder ist das zentrale Modul, welches der Kommunikation zwischen den unterschiedlichen Modulen dient und auf dem WFS-Server läuft. Die Kommunikation findet mittels OSC-Nachrichten statt. Alle Module müssen sich bei cwonder angemeldet haben, um untereinander kommunizieren zu können. Jede OSC-Nachricht, welche bei cwonder eingeht, wird an alle Module weitergeschickt, welche sich angemeldet haben. [Baa08, S. 53]

[3] http://sourceforge.net/projects/swonder

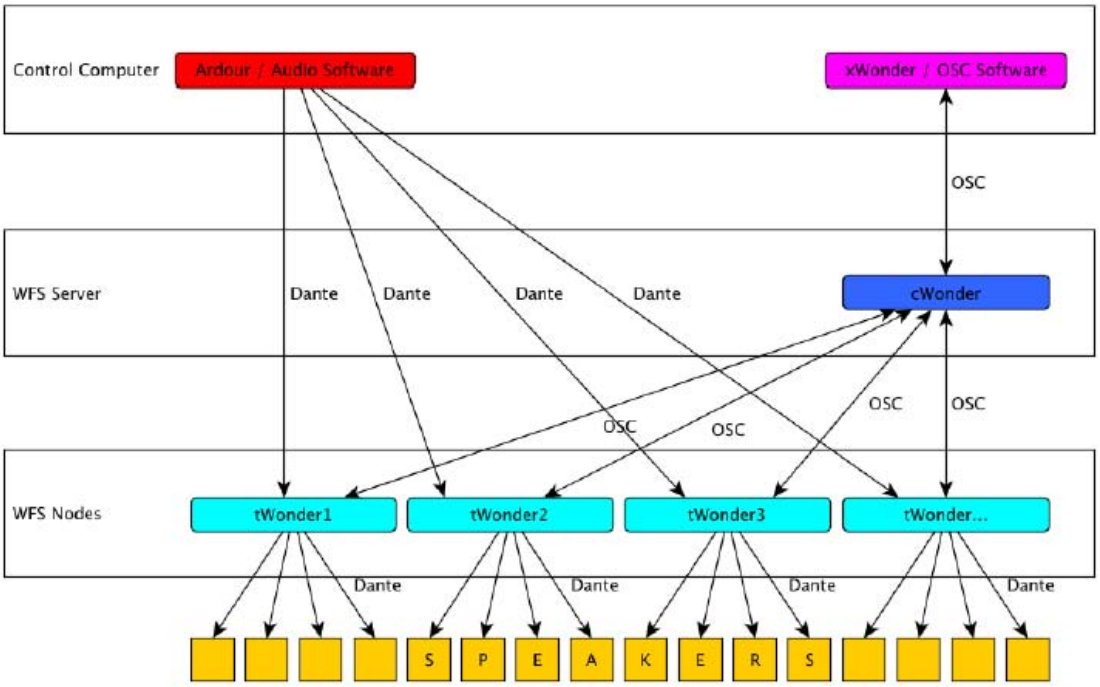



Abbildung 2.2.: Layout der WFS-Anlage [Fou].

twonder ist das Modul der Signalverarbeitung. Es ist ein Jack-Client (http://jackaudio.org/) und hat für jede WFS-Quelle einen Eingans- und für jede Spur einen Ausgangskanal. Anhand der von cwonder übermittelten Positionen der WFS-Quellen berechnen die twonder-Instanzen die Verzögerung und Amplitude pro WFS-Quelle für alle untergeordneten Spuren. Mehrere twonder-Instanzen verteilen die Rechenlast. [Baa08, S. 55]

## 2.4. Bedienmöglichkeiten einer Wellenfeldsynthese-Anlage

Der einzige Rechner der WFS-Anlage, der während des laufenden Betriebes direkt vom Benutzer bedient wird ist der Control Computer. Jegliche Eingaben müssen entweder auf oder über den Control Computer getätigt werden.

### 2.4.1. Open-Sound-Control

Die Kommunikation zwischen allen Wonder-Instanzen findet mittels OSC-Nachrichten statt. Open-Sound-Control (OSC) ist netzwerkfähig und ermöglicht es damit, die Instanzen auf verschiedenen Rechnern laufen zu lassen. Alle Bedienmöglichkeiten benutzen ebenfalls OSC-Nachrichten für die Kommunikation und kommunizieren mit cwonder. Da somit jedes Pro-

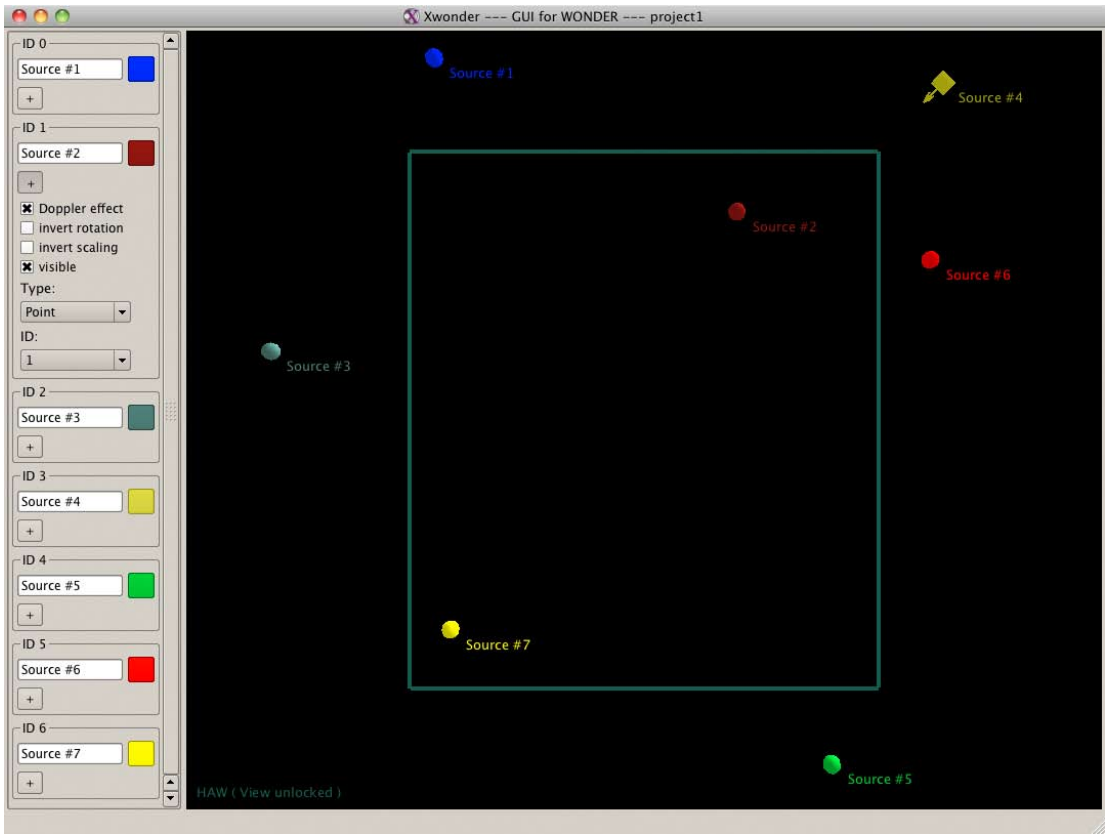



Abbildung 2.3.: Die grafische Benutzerschnittstelle xWonder

gramm, welches in der Lage ist, OSC-Nachrichten zu erstellen und zu verschicken, theoretisch für die Bedienung der WFS-Anlage in Frage kommt, werden hier zusätzlich nur noch ganz knapp die zwei an der HAW Hamburg bisher am meisten gebrauchten beschrieben. Eine Aufstellung aller OSC-Nachrichten, die cwonder versteht, befindet sich im Anhang (C) [WFM03]

### 2.4.2. xWonder

xWonder ist die grafische Benutzerschnittstelle des sWONDER-Softwarepakets. Diese ermöglicht das Erstellen und Entfernen, das Positionieren und Ausrichten sowie das Ändern des Typs und der Farbe von WFS-Quellen. All diese Eingaben werden mittels OSC-Nachrichten an den WFS-Server übermittelt.

Zudem erhält xWonder bei jeder Änderung, die auf dem WFS-Server getätigt wird eine Benachrichtigung hierüber und zeigt somit immer den aktuellen Status aller WFS-Quellen an. Abbildung 2.3 zeigt ein exemplarisches WFS-Setup via xWonder.

### 2.4.3. SuperCollider

SuperCollider ist eine Programmiersprache und -umgebung für Echtzeit Audiosynthese und algorithmische Komposition. SuperCollider ermöglicht es zum Beispiel mit einer Zeile Code eine OSC-Nachricht mit allen nötigen Parametern zu erstellen und an den WFS-Server zu verschicken. Die folgende Zeile SuperCollider-Code erstellt und verschickt eine OSC-Nachricht an den WFS-Server (IP: 192.168.3.1, Port: 58100), um die WFS-Quelle mit der ID 1 an die Position mit den Koordinaten X = -2.5 und Y = 5.8 zu platzieren.

```
1         NetAddr("192.168.3.1", 58100)
2         .sendMsg("/WONDER/source/position", 1, -2.5, 5.8);
```

# 3. Trackingverfahren und -systeme

Dieses Kapitel legt keinen Wert auf Vollständigkeit. Es soll lediglich einen Überblick über einige verschiedene Trackingsysteme bieten, ein paar grundverschiedene Ansätze mit ihren Eigenschaften erläutern, und schließlich genauer auf den in dieser Arbeit benutzten ART-Tracker eingehen.

## 3.1. Was Trackingsysteme leisten

Tracker erfassen die Position und ggf. die Ausrichtung von Objekten in einem begrenzten definierten realen Raum anhand von Bildern, welche teils mit herkömmlichen, teils aber auch mit sehr speziellen Kameras aufgenommen werden.

## 3.2. Unterschiede von Trackingsystemen und -verfahren

Es gibt eine große Anzahl von verschiedenen Methoden des Trackings und der Systeme, welche solche Methoden umsetzten, darum sollen hier nur die wesentlichsten Unterschiede erwähnt werden.

Es gibt markerbasierte Trackingverfahren und markerlose Trackingverfahren. Bei markerbasierten Trackingverfahren wird noch zwischen aktiven und passiven Markern unterschieden. Aktive Marker senden selber irgendetwas aus, was der Erfassung dient und brauchen hierfür in der Regel Elektrizität (Licht, Funksignale, etc.). Passive Marker sind unelektronisch und bieten lediglich eine markante Form oder Farbe oder dergleichen, um gut erkannt zu werden. Hierbei werden an dem zu trackenden Objekt solche speziellen Marker angebracht, welche von dem Trackingsystem erkannt werden. Ein Objekt, welches keine solcher Marker trägt, wäre für das System unsichtbar. Markerlose Verfahren haben den Vorteil, das Objekte ohne weitere Vorbereitung getrackt werden könnne, jedoch zu dem Preis, dass sie oft weniger präzise sind oder durch erhöhten Rechenaufwand eine niedrigere Bildwiederholungsrate erreichen. Es gibt eine Vielzahl verschiedener Ansätze des markerlosen Trackens. Hierbei werden zum Beispiel in den Bildern, welche die Kamera(s) aufnehmen, bestimmte Formen erkannt. Es wird Infrarotlicht ausgestrahlt und durch die Messung der zeitlichen Differenz, bis die Reflektion

wieder bei der Kamera eintrifft, die Entfernung der einzelnen Bildpunkte berechnet oder es werden durch die Verzerrung eines infraroten Gitternetzes, welches auf das Objekt projiziert wird, Rückschlüsse auf das Objekt gezogen. Einige Verfahren erfassen bzw. verarbeiten dreidimensionale andere lediglich zweidimensionale Informationen. In allen Verfahren wird auf verschiedene Arten versucht, möglichst viele Informationen über das Objekt bzw. die Szenerie und die darin enthaltenen Objekte zu bekommen, um die Objekte anhand dieser Informationen in folgenden Bildern an einer möglicherweise anderen Position innerhalb des Bildes wieder zu erkennen und dadurch zu tracken.

## 3.3. Verschiedene Systeme

Es folgt eine exemplarische Aufstellung einiger verschiedener Trackingsysteme mit einem Anriss der Funktionsweise:

- Global Positioning System (GPS) - Mehrere Satelliten in der Erdumlaufbahn senden stetig ihre aktuelle Position und die genaue Uhrzeit aus. Soweit ein Empfänger von mindestens vier unterschiedlichen Satelliten diese Daten erhält, kann er sich anhand der Verzögerung zwischen dem Sende- und dem Empfangszeitpunkt die Distanz des Empfängers zu jedem Satelliten errechnet werden. In Verbindung mit der Position der Satelliten kann die Position des Empfängers errechnet werden. [Rot02]
- Microsoft Kinect (PrimeSense[1] Sensor) - Ein Infrarotlaser projiziert ein statisches Pseudo-Zufallsmuster auf die Szenerie. Gleichzeitig nimmt eine Kamera, deren Position nicht exakt der des Lasers entspricht, das Muster, das auf der Szenerie entsteht auf und vergleicht die Aufnahme mit dem Bild, welches ursprünglich projiziert wurde. Die Projektion wird zum Beispiel auf schrägen Flächen verzerrt, was nur aus einer anderen Perspektive als der, der Kamera wahrgenommen werden kann. Aus dem anderen Blickwinkel verschiebt sich die Projektion, wenn sie in verschiedenen Distanzen aufliegt. Durch den Vergleich der Aufnahme und dem Bild, welches projiziert wurde, lassen sich Rückschlüsse auf die Geometrie schließen. Hierdurch wird für jeden Bildpunkt die Entfernung zum Trackingsystem ermittelt. Indem diese Bilder mit den komplexen Informationen der Szenerie mit den Bildern der folgenden bzw. vorherigen Bildern verglichen werden, können Bewegungen »getrackt« werden. [mir12]

[1] `http://www.primesense.com/`

## 3.4. Anforderungen für die Gestensteuerung an das Trackingsystem

Die Anforderungen der Problemstellung an den Tracker sind relativ hoch.

Es wird eine sehr hohe Bildwiederholungsrate benötigt um Bewegungen schnell erfassen und auf diese reagieren zu können. Wäre die Bildwiederholungsrate zu niedrig, würde dieses ein schleppendes Reagieren des Systems auf die Bewegungen des Akteurs zur Folge haben.

Es muss sowohl die Position als auch die Ausrichtung von Objekten im dreidimensionalen Raum erfasst werden, da Gesten richtungsorientiert ausgeführt werden sollen und die Erfassung muss sehr präzise sein, damit auch geringe Änderungen der Position und vor allem der Ausrichtung erkannt werden können.

Zudem müssen Objekte immer wieder, auch nachdem der Blickkontakt einmal verloren wurde, wieder eindeutig erkannt und von allen anderen Objekten unterschieden werden können.

## 3.5. Trackingsystem der Firma Advanced Realtime Tracking GmbH

Der ART-Tracker benutzt ein markerbasiertes Verfahren, welches sowohl mit aktiven als auch mit passiven Markern arbeitet. Die aktiven Marker senden Infrarotlicht aus, um somit leichter und bei schlechteren Bedingungen[2] erkannt zu werden. Die passiven Marker bestehen lediglich aus kleinen verschieden großen Kugeln, Platten oder Ringen, welche mit einer besonders stark reflektierenden Oberfläche überzogen sind.

Der Bereich des ART-Trackers wird durch die Platzierung mehrerer Infrarotkameras bestimmt, welche allesamt auf das Innere des Bereiches ausgerichtet sind. Diese Kameras senden Infrarotlicht aus und registrieren aus welchen Richtungen besonders starke Reflektionen kommen. Da alle Kameras zueinander kalibriert sind, lassen sich die Bilder der verschiedenen Kameras vergleichen und Überschneidungen finden. Diese Überschneidungen werden als Marker interpretiert und es lässt sich die Position der Überschneidungen im $\mathbb{R}^3$ errechnen (siehe Abb. 3.1).

Um ebenfalls die Ausrichtung von Objekten erfassen zu können gibt es so genannte Targets (siehe Abb. 3.1). Diese bestehen aus mindestens 3 Markern, welche sich in einer festen und einzigartigen Konstellation befinden. Durch die Konstellation kann (im Gegensatz zu einer run-

[2] Starke Infrarotstrahlung, welche nicht von dem System ausgeht oder andere reflektierende Objekte im der Umgebung erschweren das Tracken.

den Kugel) erkannt werden, um wieviel Grad ein Target zu einer definierten Ausgangsposition verdreht ist und durch die Einzigartigkeit der Konstellation wird ein bestimmtes Target immer wieder als dieses erkannt und kann nicht mit anderen Targets verwechselt werden. Diese Targets werden u. U. mit Vorrichtungen geliefert, um sie an Körperteilen des menschlichen Körpers anbringen zu können. Werden so alle Gliedmaßen eines Körpers mit Targets versehen, können alle Bewegungen des Körpers getrackt werden. Zudem liefert der ART-Tracker eine Bildwiederholungsrate von bis zu 60 Hz.

Die Software zu dem A.R.T.-System heißt dtrack. Über diese wird das System unter anderem kalibriert, gestartet und gestoppt. Es können diverse Einstellungen vorgenommen werden und die aktuell getrackten Marker oder wahlweise Targets können textuell oder graphisch verfolgt werden.

Für jede Aufnahme der Kameras wird ein UDP-Datagramm im ASCII-Format erstellt und mittels eines Multicasts über das LAN verschickt. Die Daten, die in den einzelnen Paketen enthalten sind, variieren etwas je nach Einstellung des dtrack. Ein Paket mit der für diese Arbeit benutzten Einstellung enthält eine Framenummer($fr$), einen Zeitstempel($ts$), den Typ($type$) und die Anzahl der erkannten Targets und schließlich für jedes erkannte Target die jeweilige ID($id$), die Qualität, mit welcher das Target erkannt wurde($qu$[3]), die Positionsverschiebung des Targets zum Koordinatensystem des Raumes($s_x$, $s_y$, $s_z$), die Winkel der Ausrichtung($\alpha$, $\beta$, $\gamma$) sowie eine Rotationsmatrix zur Darstellung der Ausrichtung($b_0$, $b_1$, $b_2$, $b_3$, $b_4$, $b_5$, $b_6$, $b_7$, $b_8$), welche wie folgt zu bilden ist:

$$R = \begin{pmatrix} b_0 & b_3 & b_6 \\ b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \end{pmatrix}$$

Der Aufbau eines solchen Paketes sieht folgendermaßen aus:

Fortlaufende Nummer des Paketes: $fr\ integer$

Zeitstempel des Paketes: $ts\ double$

Typ und Anzahl der erkannten Targets: $type\ integer$

ID, Qualität, Position und Ausrichtung für jedes erkannte Target: $[id\ qu][s_x\ s_y\ s_z\ \alpha\ \beta\ \gamma]$

Rotationamatrix des jeweiligen Targets: $[b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8]$

[3] Die Qualität von Targets ist in dieser Version von dtrack noch ungenutzt

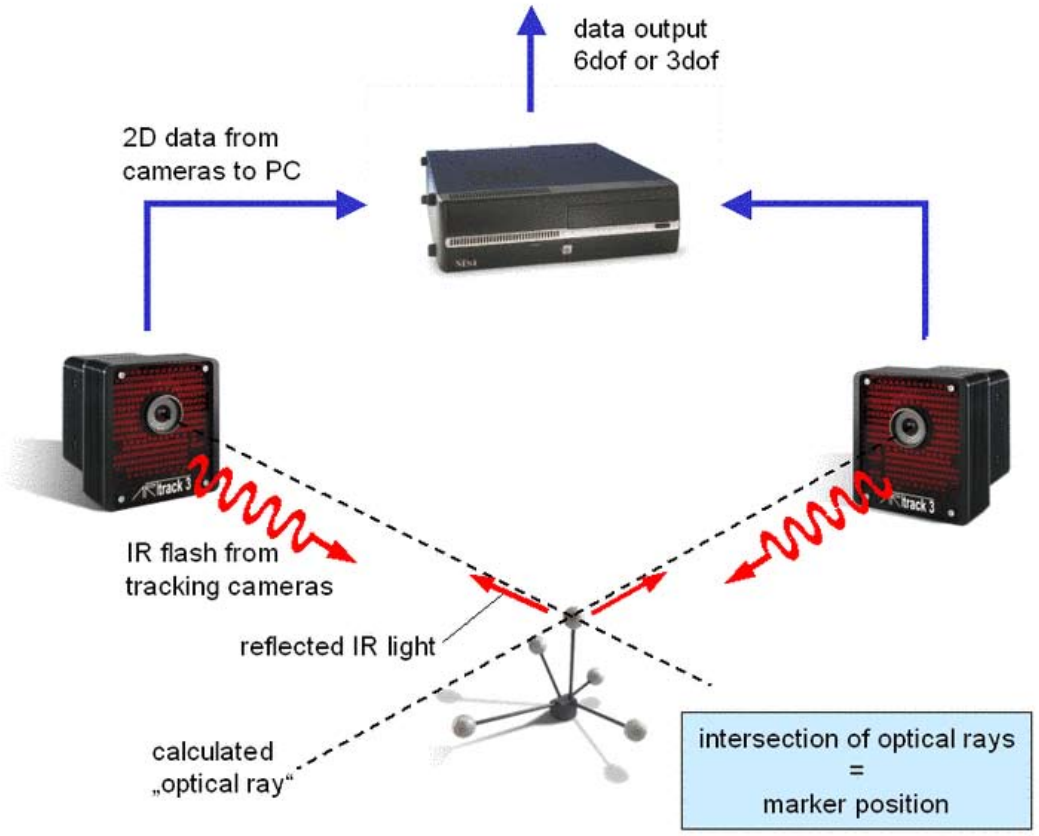



Abbildung 3.1.: Funktionsprinzip des ART-Trackers. Quelle: Advanced Realtime Tracking GmbH (2012)

Exemplarisch könnte ein solches Paket, welches genau ein Target vom Typ $6d$ mit der $id$ 13 enthält, folgendermaßen aussehen:

$fr$ 1578

$ts$ 51442.220161

$6d$ 1

[13 1.000][-472.493 279.846 -289.534 100.1367 20.4116 -4.1646]

[0.934737 0.355192 -0.010270 0.068063 -0.150600 0.986249 0.348761 -0.922582 -0.164946]

[Adv12a]

# 4. Das Labor

Der Bereich der WFS-Anlage an der HAW Hamburg umfasst etwa 5x6 Meter. Die Höhe der unteren Kante der Audiomodule beträgt etwa zwei Meter. Insgesamt gehören 26 Audiomodule mit zusammen 208 Spuren zu der Anlage. Das Terminal mit den Rechnern besteht aus einem rollbaren Container, welcher in der »vorderen«[1] rechten Ecke steht und von dort aus einen Bewegungsspielraum von etwa 1,5 Metern hat.

Der Bereich des Trackers umfasst etwa 4x4 Meter, befindet sich in vollem Umfang innerhalb des Bereiches der WFS-Anlage und eine der Seiten liegt direkt mittig über der kurzen Seite der WFS-Anlage, welche der Vorderseite des Labors entspricht. An der Vorderseite sind insgesamt vier Kameras angebracht. Eine in jeder Ecke und jeweils eine weitere befindet sich in den oberen hinteren Ecken.

Die Powerwall an der HAW Hamburg besteht momentan aus sechs HD-Displays mit jeweils einer Auflösung von 1920x1080, welche mittels der ATI Eyefinity Technologie[2] als ein einziger großer Monitor mit einer Auflösung von 4800x2400 auf einer Fläche von 270x112 Zentimetern benutzt werden kann und in Verbindung mit dem Tracker schon in vielen Projekten wie zum Beispiel den Bachelorarbeiten von Joachim Boetzer[3] und Olaf Potratz[4] zum Einsatz kam. In dieser Arbeit wird die Powerwall nicht benutzt.

Abbildung 4.1 zeigt eine Skizze des Labors mit den Audiomodulen, Infrarotkameras und der Powerwall.

[1] eine Seite des Labors, an welcher sich eine der kurzen Seiten der WFS-Anlage befindet, wird als »Vorseite« bezeichnet, da sich an dieser Seite ebenfalls eine Powerwall befindet und daher oft die zugewandte Seite der Akteure darstellt.

[2] `http://www.amd.com/de/products/technologies/eyefinity/pages/eyefinity.aspx`

[3] Bewegungs- und gestenbasierte Applikationssteuerung auf Basis eines Motion Trackers [Boe08]

[4] Ein System zur physikbasierten Interpretation von Gesten im 3D-Raum [Pot11]

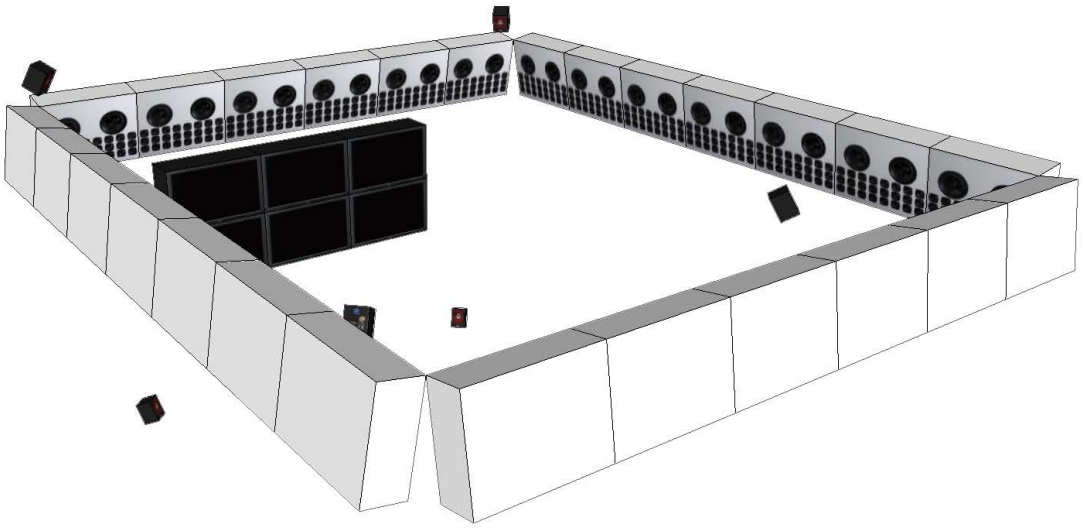


Abbildung 4.1.: Layout des Labors mit Audiomodulen, Infrarotkameras und Powerwall.

# 5. Der MoWeC

## 5.1. Überblick

Bevor mit der Gestensteuerung begonnen werden kann, muss zuerst eine generelle »Schnittstelle« zwischen den beiden vorhandenen Systemen (der Tracker, der später die Bewegungen registrieren soll und die WFS-Anlage, welche die Soundquelle positionieren und selbstverständlich die Klänge erzeugen soll) geschaffen werden. Diese »Schnittstelle« wird im weiteren als der Motion Tracker-Wave Field Synthesis-Connector (MoWeC) bezeichnet.

Im Vordergrund steht, bei der Entwicklung, vor allem ein robustes und flexibles System zu erschaffen, welches dazu taugt, durch weitere Komponenten erweitert zu werden und diesen wichtige und nützliche Funkionen und Werkzeuge zur Verfügung zu stellen.

### 5.1.1. Eingliederung in die Umgebung

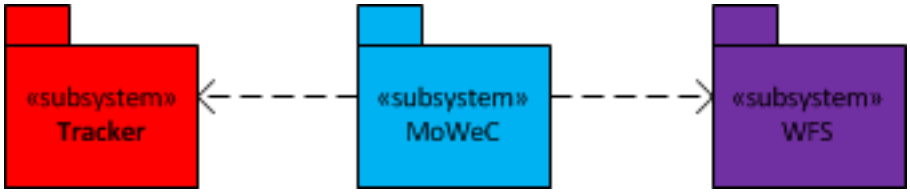



Abbildung 5.1.: Eingliederung des Systems in die Umgebung

Der MoWeC stellt die Verbindung zwischen dem Tracker und der WFS-Anlage her. Diese beiden Systeme sind geschlossen. Daher werden lediglich die angebotenen Schnittstellen benutzt.

### 5.1.2. Schnittstellen der Umgebung

#### 5.1.2.1. Tracker

Der Tracker versendet für jeden Zustand, den das System erfasst, ein UDP-Paket mit sämtlichen Informationen über einen Multicast. Diese Pakete kann somit jedes System empfangen, das sich im gleichen Netzwerk befindet.

Siehe auch Kapitel 3.5.

#### 5.1.2.2. WFS-Anlage

Der WFS-Server der WFS-Anlage kann OSC-Nachrichten empfangen und setzt die Semantik jeder Nachricht unverzüglich um. Da sich der WFS-Server jedoch mit lediglich den anderen Computern der WFS-Anlage in einem geschlossenen Netzwerk befindet (siehe Abb. 2.2), kann diese Schnittstelle auch nur innerhalb dieses kleinen Netzwerkes angeboten werden.

Da der WFS-Mac die Schnittstelle der WFS-Anlage zum Benutzer (sowohl Eingabe als auch Ausgabe) bildet, müssen Programme, die mit dem WFS-Server kommunizieren wollen entweder auf dem WFS-Mac ausgeführt werden oder auf dem WFS-Mac muss ein weiteres Programm laufen, welches die Schnittstelle in ein externes Netzwerk weiterleitet.

## 5.2. Anforderungen

Wie in Abb. 5.1 zu sehen ist, steht der MoWeC direkt zwischen dem Tracker und der WFS-Anlage. Er muss die Pakete, welche der Tracker verschickt, annehmen, verarbeiten und OSC-Nachrichten generieren und an die WFS-Anlage verschicken können.

### 5.2.1. Daten empfangen

Wie schon beschrieben versendet der Tracker die Daten über einen Multicast. Um die Daten zu empfangen muss sich der Empfänger lediglich im selben Netzwerk befinden und sich in der Multicast-Gruppe anmelden.

### 5.2.2. Daten verarbeiten

Der Hauptteil des Arbeitsaufwandes des MoWeC, gilt der Verarbeitung der Daten. Der MoWeC muss in der Lage sein, die Semantik der empfangenen Daten richtig zu interpretieren und für die weitere Verarbeitung vorzubereiten. Dann müssen die empfangenen Daten der Targets (welche innerhalb des A.R.T.-Systems existiert) auf WFS-Quellen (innerhalb der WFS-Anlage) abgebildet werden. Dafür ist eine Transformation notwendig, welche die Unterschiede der Koordinatensysteme präzise berücksichtigt. Für diese Transformationen, aber auch schon für zukünftige Erweiterungen (wie die GestureComponent), soll eine Sammlung an Werkzeugen zum Berechnen und Vergleichen von Positionen und Winkeln geschaffen werden.

Zusätzlich soll es schon ermöglicht werden, Targets direkt auf WFS-Quellen abzubilden, um unabhängig von Gesten diese durch Targets platzieren zu können um zum Beispiel später die Position und Ausrichtung der Hand leicht verfolgen zu können.

### 5.2.3. Open-Sound-Control-Nachrichten verschicken

Abschließend müssen aus den Daten - schnittstellenkonform - für den WFS-Server verständliche OSC-Nachrichten erstellt und vom WFS-Mac an den WFS-Server geschickt werden.

## 5.3. Subsysteme und Komponenten

Um eine möglichst lose Kopplung zum Tracker zu erhalten, wird für den Teil des Systems, welcher für den Empfang der Daten vom Tracker zuständig ist, ein separates Programm eingesetzt, welches die empfangenen Daten an das eigentliche System weiterschickt. Dieses wird Trackerlistener genannt.

Der MoWeC selber wird in einige weitere Komponenten aufgeteilt, welche ebenfalls im Folgenden beschrieben werden.

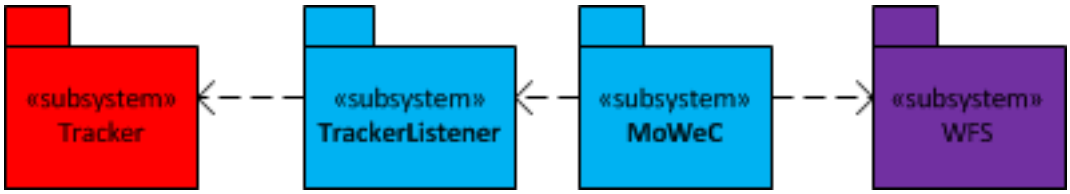



Abbildung 5.2.: Eingliederung des MoWeCs und des Trackerlisteners in die Umgebung

### 5.3.1. Trackerlistener

Der Trackerlistener wurde als separates System entwickelt, um den Tracker zum einen leichter austauschbar zu machen, denn soll ein anderer Tracker benutzt werden, muss lediglich der Trackerlistener angepasst oder ausgetauscht werden, der MoWeC muss jedoch nicht berührt werden und zum anderen, um die sonst bestehende örtliche Bindung zwischen dem System und dem Tracker aufzulösen (siehe Kapitel 5.1.2.1), denn der Trackerlistener kann die Pakete an ein beliebiges Ziel verschicken.

Der Trackerlistener meldet sich, wie vom A.R.T.-System vorgegeben an der Multicastgruppe an und schickt die Daten jedes Pakets unverzüglich an eine beliebige Anzahl beliebiger Empfänger weiter. Sofern der Trackerlistener auf einem Computer ausgeführt wird, der sich

noch in einem zweiten Netzwerk befindet, stellt er damit die Schnittstelle zwischen dem Tracker und diesem zweiten Netzwerk und somit potentiell auch zur Außenwelt dar.

### 5.3.2. Der MoWeC

Der MoWeC nimmt die Daten des Trackerlisteners an und bereitet diese auf, indem sie in eine innerhalb des MoWeCs einheitliche Form (den Markern und schließlich den Sources) gebracht werden. In dieser Form können sämtliche Module mit den Daten arbeiten und zuletzt sorgt der MoWeC dafür, dass die Daten unter Berücksichtigung aller vorgenommenen Änderungen für die WFS-Anlage aufbereitet und dieser zugespielt werden.

In oberster Instanz besteht der MoWeC aus einer Routine, welche bei jedem Eingang neuer Daten eines Trackerlisteners durchgeführt wird. Man kann also von Zyklen sprechen. Der Eingang jedes Paketes löst einen weiteren Zyklus aus. Lediglich die Inhalte der Pakete entscheiden über Unterschiede im Ablauf der Zyklen.

Bei dem Entwurf der Architektur wurde besonders Wert darauf gelegt, dass das System später leicht erweitert werden kann. Daher werden alle Programmteile, welche die Funktionalität des MoWeCs erweitern modular und mit einer minimalen Kopplung angedacht. Als Eingabe der Module soll lediglich eine Menge an Sources dienen und als Ausgabe im Idealfall ebenfalls eine (modifizierte) Menge an Sources. Alle Module werden dabei von der Routine des Coordinators ausgeführt.

Im weiteren werden die Komponenten des MoWeCs einzeln beschrieben.

#### 5.3.2.1. Coordinator

Die Coordinator-Komponente initialisiert alle weiteren Komponenten, nimmt Einstellungen an ihnen vor und bestimmt den Ablauf des Systems. Sie erstellt einen Listener, welcher die Daten eingehender Pakete des Trackerlisteners liefert und enthält eine Routine, welche dann jedes Mal durchlaufen wird. Alle Komponenten laufen hier zusammen und deren Ausführung wird hier bestimmt. Eine GUI[1] würde es Benutzern ermöglichen, Einstellungen an dieser Routine vorzunehmen, um Module zu (de-)aktivieren oder sämtliche Anpassungen vorzunehmen.

#### 5.3.2.2. Listener

Der Listener ist die Schnittstelle für einen Trackerlistener, über die neue Informationen über Targets in das System gelangen. Der Coordinator erstellt einen Listener und definiert dabei eine Methode, welche jedes Mal ausgeführt werden soll, wenn der Listener ein Paket von

[1] Wird aus Zeitgründen und mangels wissenschaftlichen Anspruches vorerst nicht erstellt.

einem Trackerlistener erhält. Standardmäßig wird direkt die Hauptroutine des MoWeCs mit den neuen Daten ausgeführt.

#### 5.3.2.3. Marker

Marker sind die direkten Repräsentanten der Targets innerhalb des Systems. Für jedes Target, was vom Tracker erkannt und dessen Informationen über den Trackerlistener an das System gelangen, wird ein Marker erstellt. Ein Marker enthält alle Informationen, die es über das jeweilige Target gibt, in der gleichen Form, wie sie vom Tracker bzw. des Trackerlisteners übermittelt werden. Dies sind die ID, die X-, Y- und Z-Werte der Position, die X-, Y- und Z-Werte der Drehung um die Achsen, sowie die Rotationsmatrix. Die Klasse der Marker enthält ein Pattern, mit dem die atomaren Informationen über die Targets aus den Paketen des Trackerlisteners ausgelesen werden können, um daraus die jeweiligen Marker zu erstellen. Diese Klasse ist also hochgradig abhängig von dem benutzten Trackerlistener.

#### 5.3.2.4. Source

Sources sind die Repräsentanten der WFS-Quellen innerhalb des MoWeCs. Für alle WFS-Quellen, die in irgendeiner Art durch den MoWeC beeinflusst werden sollen, muss so eine Source innerhalb des MoWeCs existieren. Die Zuweisung erfolgt implizit durch Zuweisung der selben ID. Sämtliche Modifikationen, welche an WFS-Quellen gemacht werden sollen, werden innerhalb des MoWeCs an der repräsentativen Source vorgenommen. Die Sources sind damit der zentrale Datentyp des MoWeCs, mit dem alle Module arbeiten.

#### 5.3.2.5. SourceHistory

Die SourceHistory ermöglicht es, verschiedene Versionen der gleichen Sources zusammengefasst zu speichern. Neue Versionen einer Source entstehen zum Beispiel jedes Mal, wenn eine Source durch ein Target aktualisiert wird, durch die Anwendung von Convertern, einen Positioner, der GestureComponent oder etwaiger anderer Komponenten, welche die Sources ändern.

Wird immer zu einem bestimmten Punkt des Ablaufes der Routine des Coordinators für alle Sources die SourceHistory aktualisiert, können mit Hilfe der SourceHistory Rückschlüsse auf die Entwicklung einer Source bzw. bestimmter Daten einer Source geschlossen werden.

#### 5.3.2.6. Matcher

Der Matcher verbindet Marker mit Sources. Ein Match besteht aus der ID eines Targets und der ID einer Source. Wenn die Eigenschaften eines Markers aktualisiert wurden und ein Match zwischen dem Marker und einer Source besteht, so wird durch einen Matcher die Source ebenfalls aktualisiert. Dabei werden die Werte, um spätere Berechnungen zu vereinfachen, in eine allgemeingültigere Form gebracht, als sie innerhalb des Markers vorliegen. Dadurch wird erreicht, dass eine WFS-Quelle durch ein Target platziert werden kann. Durch den richtigen Einsatz von Convertern kann zum Beispiel erreicht werden, dass sich eine WFS-Quelle immer an genau der Position eines Targets befindet.

#### 5.3.2.7. Positioner

Der Positioner erweitert die Möglichkeit Sources durch Targets zu positionieren und besteht aus zwei Teilen. Einer RelativePosition und einem SourceSourceMatcher.

**Die RelativePosition** bestimmt eine Relation zwischen zwei Positionen. Sie legt entweder einen Versatz auf den Achsen des kartesischen Koordinatensystems oder einen Winkel und eine Entfernung für Polarkoordinaten fest. Mit der RelativePosition kann somit, zu der Position jeder beliebigen Source, eine weitere Position, in der definierten Relation, erstellt werden.

**Der SourceSourceMatcher** legt ein Verbindung von einer Source $source1$ zu einer zweiten Source $source2$ fest und ordnet dieser Verbindung eine RelativePosition zu. Jedes Mal, wenn der Positioner durchläuft, wird die Position von $source2$ in die Relation zu $source1$ gesetzt, welche durch die RelativePosition definiert wurde.

Somit lassen sich zum Beispiel zwei WFS-Quellen, auf welche die linke und rechte Spur eines Musikstückes gelegt wurden, immer genau links und rechts des Kopfes eines Akteurs und in der gleichen Entfernung zu diesem platzieren, sofern der Kopf des Akteurs getrackt wird, um für die Person einen »Kopfhörereffekt« zu erzeugen.

#### 5.3.2.8. Converter

Das System bringt zwei Koordinatensysteme zusammen, das des Trackers und das der WFS-Anlage. Diese sind voneinander völlig unabhängig und können sich in der Dimension, der Ausrichtung der Achsen, des Drehsinns, der Position des Nullpunktes sowie der Skalierung unterscheiden. Dies kann schnell zu ungewollten Ergebnissen führen. So kann zum Beispiel

eine Drehung leicht einen Effekt genau gegensätzlich der Intention haben, wenn der Drehsinn sich unterscheidet.

Doch selbst, wenn die Koordinatensysteme übereinstimmen, kann der Bedarf bestehen eine Ungleichheit zu erzeugen. In der Skalierung zum Beispiel. Da der Bereich der WFS, in dem Soundquellen positioniert werden können, theoretisch unendlich ist, der Bereich, in dem die Targets gesehen werden können, jedoch durch die Kameras beschränkt ist.

Dieses Kapitel beschreibt, wie mit der Problematik der verschiedenen Koordinatensysteme umgegangen wird bzw. wie es ermöglicht wird, die Relation zwischen den Koordinatensystemen zu bestimmen.

**Die Problematik** lässt sich grundlegend in zwei unterschiedliche Anforderungen unterteilen.

- Das Angleichen zwei unterschiedlicher Koordinatensysteme.
- Zwei Koordinatensysteme in eine bestimmte Relation zu bringen.

Da dieses System lediglich die beiden Systeme Tracker und WFS-Anlage verbindet und keinerlei Einfluss auf deren innere Struktur, wie die Koordinatensysteme, nehmen kann, werden die Transformationen nicht an den Koordinatensystemen selber vorgenommen, sondern an allen Daten, welche von einem Koordinatensystem in das andere übertragen werden.

**Verschiedene Koordinatensysteme** können sich in folgenden fünf Aspekten unterscheiden:

- Dimension
- Ausrichtung der Achsen
- Drehsinn
- Position des Nullpunktes
- Skalierung

Da der Tracker die Positionen im $\mathbb{R}^3$ erfasst [Adv12b] , die WFS-Anlage jedoch lediglich ein horizontales zweidimensionales Koordinatensystem hat [Baa08], ist zumindest dieser Unterschied immer gegeben. Im einfachsten Fall wird die Höhe der Position in der Translation von dem Tracker zur WFS-Anlage einfach ignoriert und somit lediglich die Position auf der Ebene berücksichtigt.

**Koordinatensysteme in eine bestimmte Relation zu bringen,** welche nicht der Gleichheit entspricht, kann zum Beispiel nötig sein, wenn WFS-Quellen, welche in statischer Relation zu Targets stehen, zu allen Seiten außerhalb des physikalischen Raumes der WFS-Anlage und des Trackers platziert werden sollen. In diesem Fall müsste das Koordinatensystem des Trackers zu dem der WFS-Anlage herauf-skaliert werden (Vergleiche Abb. 5.3f zu Abb. 5.3e).

Die Mechanismen sind jedoch die gleichen, wie bei der Angleichung der beiden Koordinatensysteme.

**Die Lösung** bietet eine Reihe von Convertern. Der Scaler, Shifter, Mirror, Rotator und der RotationMirror. All diese Converter transformieren die Eigenschaften einer Source oder mehreren Sources und sind so konfigurierbar, dass sie die gewünschte Teilaufgabe erfüllen.

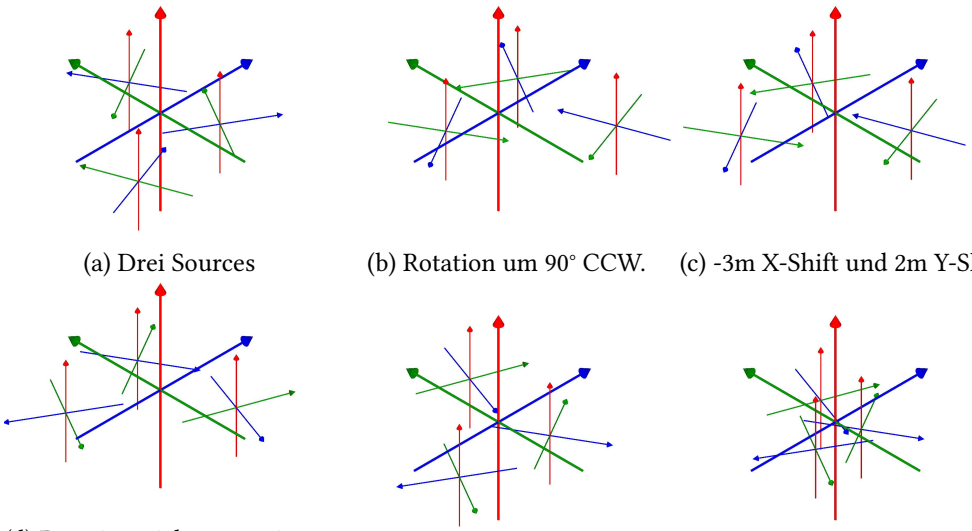
**Der Scaler** kann das Größenverhältnis zwischen den beiden Koordinatensystemen beeinflussen. Ihm kann für alle drei Achsen (X, Y, und Z) ein unterschiedlicher Wert zugewiesen werden. In der Regel wird jedoch für alle Achsen der gleiche Wert gesetzt, um eine gleichmäßige Skalierung zu erhalten. Der Scaler multipliziert nun die X-, Y- und Z-Attribute der Position der Sources, die ihn durchlaufen mit dem jeweiligen gesetzten Wert (siehe Abb. 5.3e auf Abb. 5.3f).

**Der Shifter** verschiebt Sources entlang der Achsen. Ihm kann ebenfalls für alle drei Achsen ein Wert zugewiesen werden. Im Gegensatz zum Scaler unterscheiden sich die Werte doch in der Regel voneinander. Die Arbeitsweise ist dieselbe, wie die des Scalers, nur wird nicht multipliziert, sondern zu den Werten der Position werden die eingestellten Werte addiert und somit die Position der Sources verschoben (siehe Abb. 5.3b auf Abb. 5.3c).

**Der Mirror** spiegelt die Positionen der Sources um die Achsen des Koordinatensystem des Raumes. Hier wird lediglich eingestellt, um welche Achsen gespiegelt werden soll (siehe Abb. 5.3d auf 5.3e).

**Der Rotator** dreht die Sources um alle drei Achsen, wobei die jeweilige Ursprungsgerade immer parallel zu der jeweiligen Achse des Raumkoordinatensystems und durch die Position der Source verläuft. Dem Rotator kann für die Rotation um jede Achse ein separater Wert zugewiesen werden (siehe Abb. 5.3a auf Abb. 5.3b).

**Der RotationMirror** ändert den Drehsinn der Achsen. Die Drehung, die eine Source zuvor im Uhrzeigersinn hatte, hat die Source nach der Transformation im gleichen Maß gegen den



(a) Drei Sources (b) Rotation um 90° CCW. (c) -3m X-Shift und 2m Y-Shift

(d) Rotationsrichtungsspiegelung der Z-Achse (RotationMirror). (e) Spiegelung an den X-Achse (f) Skalierung um 0.5

Abbildung 5.3.: Durchlaufen von drei Sources einer ConverterBox mit einem Rotator, Shifter, RotationMirror, Mirror und einem Scaler. Die Kreuze stehen für Koordinatensysteme. Das große steht für das Koordinatensystem des Raumes, die kleinen für die der Sources. Die blaue Achse entspricht der X-, die grüne der Y- und die rote der Z-Achse. Dieses Konzept wird im mehreren Abbildungen in den folgenden Kapiteln wieder aufgegriffen, wobei die kleinen Koordinatensysteme in den jeweiligen Abbildungen auch für WFS-Quellen, Targets oder Händen stehen können.

Uhrzeigersinn und umgekehrt ebenso. Angegeben wird lediglich, welche Achsen von der Transformation betroffen sein sollen (siehe Abb. 5.3c auf Abb. 5.3d).

**Für die komplette Transformation** von einem Koordinatensystem in das andere, müssen möglicherweise mehrere verschiedene Converter angewandt werden. Dazu dient die ConverterBox, in der eine beliebige Anzahl von Convertern unter Einbehaltung der Reihenfolge zusammengefasst werden kann. Eine ConverterBox transformiert die Eigenschaften einer Source oder mehreren Sources entsprechend der beinhaltenden Convertern (siehe Abb. 5.3).

#### 5.3.2.9. ModificationFactory

Die ModificationFactory identifiziert anhand der SourceHistory alle Unterschiede der Sources zu ihren vorherigen Versionen, welches sich in der SourceHistory befinden. Dazu kann eine

Toleranz eingestellt werden und ob ein Jitter-Filter angewandt werden soll. Anhand der Toleranz werden nur Modifications für Änderungen erstellt, welche die Toleranz übersteigen. Durch diese Funktion kann die Anzahl an Modifications um ein vielfaches heruntergesetzt werden, indem, im geringsten Fall, Änderungen, die lediglich für das System aber nicht für die Menschen im Bereich der WFS-Anlage erkennbar sind vernachlässigt werden. Hierdurch verringert sich im gleichen Maße die Anzahl der OSC-Nachrichten, welche erstellt, an die WFS-Anlage gesandt und vor allem von ihr verarbeitet werden müssen. Zu viele aufeinanderfolgende OSC-Nachrichten an den WFS-Server haben akustische Artefakte zur Folge. Je höher die Toleranz eingestellt wird, um so stärker ist natürlich die Entlastung, jedoch bringt auch eine Toleranz der Position im Millimeterbereich schon einen erheblichen Vorteil. Um dennoch schleichende Änderungen zu bemerken und berücksichtigen zu können, vermerkt die ModificationFactory intern für jede Source die Daten, zu denen zuletzt eine Modification erstellt wurde und vergleicht die Toleranz immer mit dem Unterschied dieser Daten zu den aktuellen.

#### Modifications

Für jede Änderung, die an einer Source vorgenommen werden kann und die per OSC-Nachricht an die WFS-Anlage geschickt werden soll, gibt es einen Typ von Modification. Sie enthält die Art der Änderung (zum Beispiel Positions- oder Ausrichtungsänderung), die ID der Source, aus der die Modification extrahiert wurde sowie alle nötigen Parameter wie die Koordinaten oder einen Winkel. Die Erstellung jeder Modification soll schlussendlich eine Modifikation des WFS-Setups zur Folge haben.

#### 5.3.2.10. OSC-Adapter

Der OSC-Adapter ist in der Lage, aus Modifications, die von der ModificationFactory identifiziert, extrahiert und erstellt wurden, die entsprechenden OSC-Nachricht der JavaOSC-Bibliothek zu erstellen.

#### 5.3.2.11. OSC-Sender

Der OSC-Sender bedient sich ebenfalls der JavaOSC-Bibliothek und ermöglicht es, dem System OSC-Nachrichten an beliebige Empfänger zu verschicken.

#### 5.3.2.12. Values und Utilities

Die Pakete Values und Utilities stellen einige einfachere und komplexere Datentypen mitsamt Rechen- und Vergleichsoperatoren zur Verfügung. Enthalten sind:

- Degree180 - Winkeldatentyp, welcher Werte zwischen -180 und +180 wiedergibt.
- Degree360 - Winkeldatentyp, welcher Werte zwischen 0 und +360 wiedergibt.
- Length - Datentyp, der für eine Distanz zwischen zwei Punkten steht.
- Matrix - Datentyp, der eine zweidimensionale Matrix darstellt, mit der alle einfachen Rechenoperatoren durchgeführt werden können.
- Radiant - Windeldatentyp, welcher Werte zwischen 0 und $2\pi$ wiedergibt.
- Rotationsmatrix - Datentyp, der eine Rotationsmatrix verkörpert, auf den nicht nur Matrixrechenoperationen ausgeführt werden können, sondern dessen Werte sich durch Angabe von Winkeln und Achsen entsprechend der jeweiligen Rotation modifizieren, lassen sowie die Möglichkeit bietet, sich die eulerschen Winkel ausgeben zu lassen.
- Straight - Datentyp, der eine Gerade innerhalb des Koordinatensystem darstellt.

Die Degree-Datentypen dienen lediglich der Darstellung und sollen es Programmierern ermöglichen, je nach Aufgabe den passendsten Datentyp benutzen zu können. Auf unterster Ebene werden Winkel immer im Bogenmaß mit dem Datentyp Radiant angegeben und alle Winkeldatentypen sind untereinander vergleichbar und können beliebig in Rechenoperatoren zusammen benutzt werden.

#### 5.3.2.13. JavaOSC

JavaOSC ist eine Java-Bibliothek, welche von Illposed Software entwickelt und bereitgestellt wird. Sie enthält Methoden zum Erstellen, Verschicken und Empfangen von OSC-Nachrichten. [IS12]

#### 5.3.2.14. Ablauf und Zusammenspiel der Komponenten

Abbildung 5.4 zeigt sämtliche Pakete des MoWeCs mit ihren jeweiligen Konnektoren. Abbildung 5.5 zeigt ein Signalflussdiagramm, welches den Fluss der Daten vom Tracker bis zur WFS-Anlage darstellt.

Nachdem der Listener ein neues Paket vom Trackerlistener erhalten hat, wird mit dem Inhalt dieses Paketes eine Methode des Coordinators aufgerufen. Der Coordinator bestimmt, wie die Daten das System durchlaufen. Zuerst werden mittels des Matchers die Daten des Paketes ausgelesen, interpretiert und für die Daten aller Targets des Trackers, welche in dem Paket

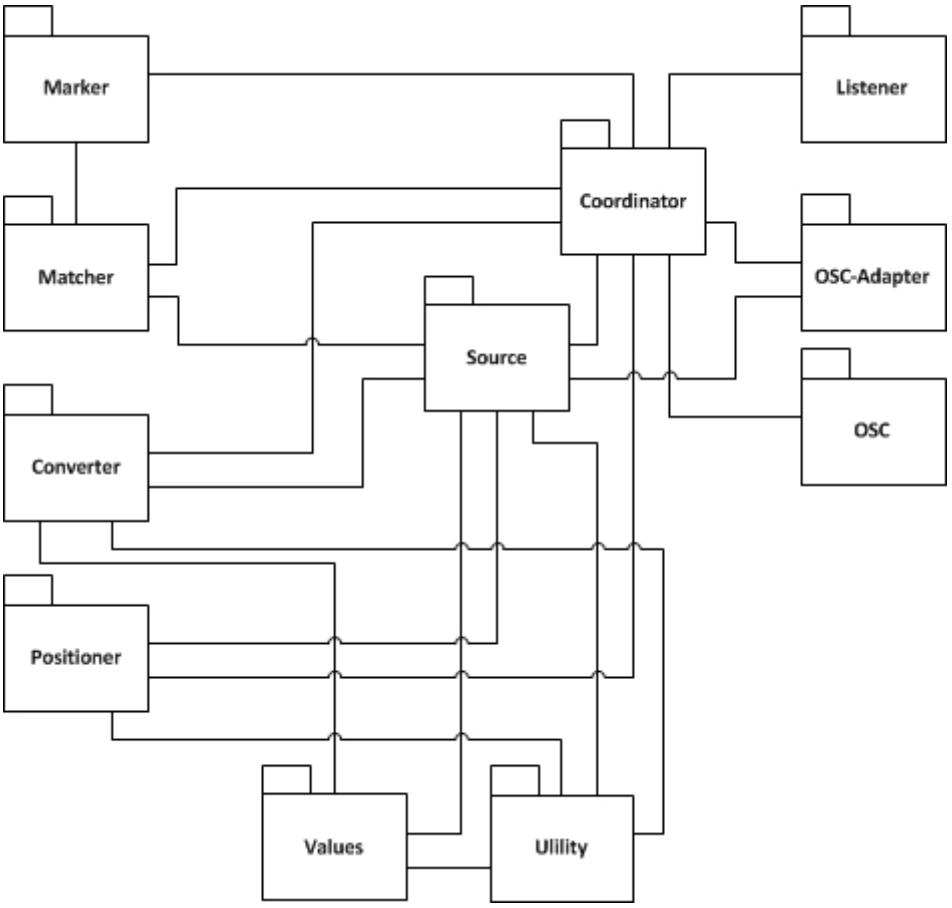



Abbildung 5.4.: Paketdiagramm des MoWeCs

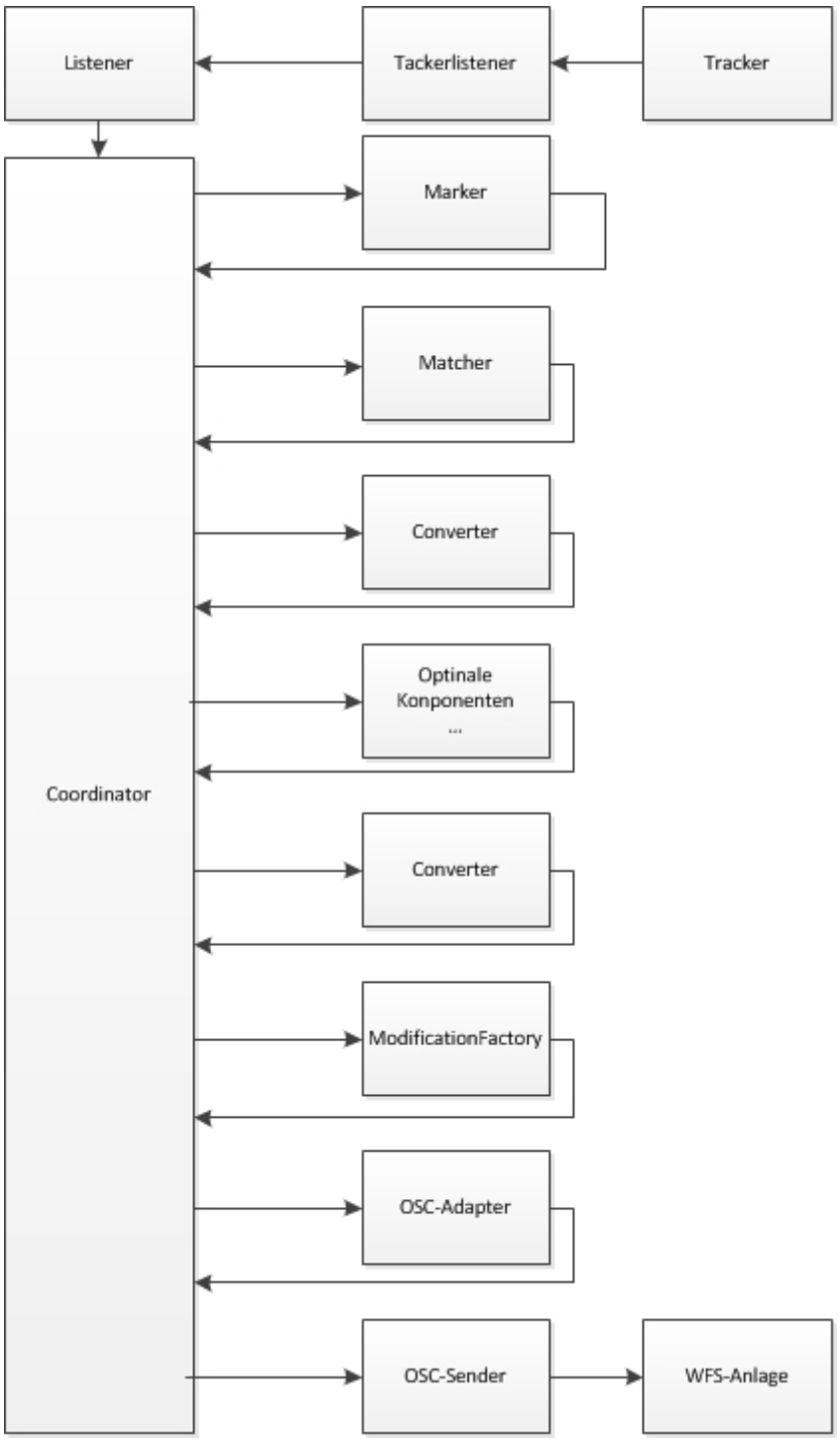



Abbildung 5.5.: Signalflussdiagramm des MoWeCs

enthalten sind, wird ein dementsprechendes Marker-Objekt erstellt oder, falls diese schon aus vorherigen Zyklen bestehen, aktualisiert. Die erstellten Marker werden an den Coordinator zurückgegeben. Im nächsten Schritt werden mittels des Matchers alle Sources aktualisiert, für die dem Matcher ein Match mit einem der in diesem Zyklus aktualisiertem Marker vorliegt. Nachdem alle Sources aktualisiert wurden, durchlaufen diese eine ConverterBox, welche die Daten der Sources von dem Koordinatensystem des Trackers in das der WFS-Anlage transformieren. Welche Converter in der ConverterBox enthalten sind und wie diese konfiguriert sind, hängt davon ab, wie sich die Koordinatensysteme unterscheiden. Es können auch mehrere ConverterBoxen oder einzelne Converter durchlaufen werden, jedoch können alle nötigen Aufgaben in einer ConverterBox zusammengefasst werden.

Die nun vorliegenden Sources können eine Reihe optionaler Komponenten durchlaufen. Eine optionale Komponente, welche dem MoWeC schon hinzugefügt wurde, ist der SourceSourceMatcher. Mit diesem können weitere Sources, welche nicht in einem direkten Match zu einem Marker und damit Target stehen, anhand der eingegangenen Daten modifiziert werden. Eine weitere optionale Komponente ist die GestureComponent. An dieser Stelle können auch noch weitere Komponenten eingefügt und aktiviert bzw. deaktiviert werden.

Nach dem optionalen Teil der Routine werden wieder durch eine ConverterBox Transformationen an den mittlerweile möglicherweise modifizierten Sources vorgenommen, um diese in das Koordinatensystem der WFS-Anlage zu übertragen. Nun werden für alle Sources mittels einer SourceHistory die jeweiligen Einträge angelegt bzw. aktualisiert. Im Anschluss werden anhand der SourceHistory durch die ModificationFactory die Änderungen, welche an den Sources vorgenommen, wurden extrahiert. Dabei werden die in der SourceHistory angegebenen Toleranzen berücksichtigt, um zu geringe Änderungen zu vernachlässigen und ggf. ein vorhandener Jitter geglättet. Das Ergebnis sind eine Menge an Modifications, aus denen mittels des OSC-Adapters und JavaOSC für jede Modification eine OSC-Nachricht erstellt wird. Der OSC-Sender verschickt mit Hilfe des JavaOSC die OSC-Nachrichten dann entweder direkt an den WFS-Server oder an eine weitere dazwischen liegende Instanz einer dritten Partei.

# 6. Gesten

## 6.1. Was sind Gesten

Gesten bieten einem die Möglichkeit, sich nonverbal und dennoch präzise auszudrücken. Dabei liegt der Schwerpunkt der Aussagekraft meist in den Händen und der Arme, jedoch kann der Ganze Körper zum Gestikulieren eingesetzt werden. Oft werden Gesten auch Rede begleitend eingesetzt, um den verbal kommunizierten Inhalt zu unterstreichen oder zu ergänzen [Fri12].

Übliche Gesten, welche die meisten Menschen im Alltag gebrauchen, sind zum Beispiel das Winken mit der Hand, der Händedruck oder das Händeschütteln sowie eine Umarmung als Begrüßung oder Verabschiedung, das Nicken des Kopfes oder das Kopfschütteln um die Wörter »ja« und »nein« zu ersetzten oder lediglich das Reichen der Hand als Zeichen der Zuneigung, des Vertrauens oder der Versöhnung. Das Deuten mit dem ganzen Arm, der Hand oder eines Fingers in eine Richtung, auf einen Gegenstand oder eine Person bietet oft, auch redegewandten Personen, eine Möglichkeit sich auf einfache Art sehr präzise auszudrücken [Fri12].

Besonders für Personen mit Hörbehinderungen wurden komplette Sprachen entwickelt, welche sich ausschließlich der Gesten der Arme und Hände sowie der Mimik bedienen. Die so genannten Gebärdensprachen. Diese ermöglichen es Personen sich komplett nonverbal und dennoch äußerst umfangreich auszudrücken [EHH12].

## 6.2. Gesten für die Mensch-Maschine-Kommunikation

Gesten tauchen in immer mehr Anwendungen und Anwendungsgebieten auf. Sie helfen dabei, die nötige Hardware, welche der Eingabe von Daten in Computersysteme dient, zu vermindern oder soweit vom Anwender zu entfernen, dass der Eindruck entsteht, dass sie gar nicht mehr vorhanden wäre, indem die Systeme den Anwender nur noch aus der Ferne »beobachten«. Zudem schaffen es einige Verfahren oder Anwendungen, die Gesten so intuitiv zu gestalten, dass die Bedienung durch sie noch einfacher wird, indem zum Beispiel mit einem Finger direkt auf ein (reales) Objekt gezeigt werden kann, anstatt mit einer Eingabehilfe wie der üblichen »Maus« auf eine digitale abstrakte Repräsentation [LK09].

## 6.3. Andere Gestenverfahren

Hier sollen lediglich einige übliche Systeme aus dem Alltag genannt werden, welche mit Gesten arbeiten.

- Touchscreens - Kommen vor allem in Smartphones und Tablet-PCs zum Einsatz. Mit den Fingern oder einem speziellen Stift werden zweidimensionale Gesten auf den Bildschirm gezeichnet, um Aktionen auszuführen.
- Fernseher - Einige Fernseher unterstützen mittlerweile Gesten, um zum Beispiel Programme zu wechseln oder die Lautstärke zu regulieren. [Sam12]
- Digitales Schaufenster - Das Frauenhofer Heinrich-Hertz-Institut entwickelte ein System, welches einen Bildschirm und Kameras in einem Schaufenster platziert, um Passanten die Möglichkeit zu geben durch Gesten mit den Händen durch Waren zu blättern, welche dann auf dem Bildschirm angezeigt werden. [FI12]
- Spielekonsolen - Einige Konsolen wie die Wii oder die Xbox[1] bieten Spiele an, die durch Gesten bedient werden.

## 6.4. Was sollen die Gesten leisten?

Die Gesten sollen die verschiedenen Möglichkeiten der Eingabe von Kommandos an die WFS-Anlage erweitern und es ermöglichen, die WFS-Anlage ohne visuelles Feedback und somit rein nach dem Gehör zu bedienen. Anstatt Modifikationen der Positionen der WFS-Quellen anhand zum Beispiel einer zweidimensionalen Abstraktion vorzunehmen, wie sie xWonder bietet (siehe Abb. 2.3), soll sich der Stärke der WFS-Anlage bedient werden. Diese besteht darin, es Personen zu ermöglichen, die Ursprungspositionen der synthetisierten Schallwellen akustisch bestimmen zu können. Somit können die WFS-Quellen auch anhand ihrer Akustik identifiziert und lokalisiert werden, was die Basis bildet, um auch direkt mit ihnen interagieren zu können. Des weiteren soll die Bedienung möglichst intuitiv sein, so dass Benutzer wenig bis gar keine Einführung benötigen, indem die WFS-Quellen bzw. die Ursprünge der synthetisierten und somit hörbaren Schallquellen als reale Objekte betrachtet werden und direkt mit diesen interagiert wird.

Die erste Frage richtet sich daher danach, **was** für Kommandos die WFS-Anlage versteht (eine umfangreichere Aufstellung wird in Kapitel 2) vorgestellt). Dicht gefolgt von der Frage, **wie häufig** diese Kommandos in der Regel ausgeführt werden, denn Gesten sollen vor allem für die häufig benötigten Kommandos geschaffen werden.

[1] Mit zusätzlichem Kauf einer Microsoft Kinect

Da die WFS-Anlage sich im Kern damit beschäftigt, Wellenfelder zu berechnen und zu erschaffen, welche eine nicht vorhandene Soundquelle suggerieren sollen, steht in dieser Arbeit die Positionierung der WFS-Quellen an erster Stelle.

Die Ausrichtung beeinflusst den Effekt von WFS-Punktquellen nicht, da die virtuellen Schallwellen sich gleichmäßig in alle Richtungen ausbreiten und ebene WFS-Linearquellen können rein durch das Gehör gar keine konkrete Position zugewiesen werden (Siehe Kapitel 2.2). Darum wird die Ausrichtung vorerst gänzlich vernachlässigt.

Alle OSC-Nachrichten, welche einen Parameter für die Zeit haben, zu der oder über welchen Zeitraum ein Befehlt ausgeführt werden soll, werden ebenfalls außer acht gelassen, da dies der Idee, direkt mit den WFS-Quellen wie mit Objekten interagieren zu können, widerspräche, da die Effekte nicht direkt eintreten würden.

Das Aktivieren einer WFS-Quelle durch eine Geste macht insofern keinen Sinn, da wenn eine unbestimmte WFS-Quelle aktiviert werden soll, zunächst noch das Routing gesetzt werden muss, damit überhaupt irgendeine Tonspur über die WFS-Quelle ausgegeben wird. Sollte eine bestimmte WFS-Quelle aktiviert werden, für die zuvor bereits ein Routing gesetzt wurde, so müsste die Geste die konkrete ID der WFS-Quelle enthalten, was bei einer Anzahl von momentan 64 möglichen WFS-Quellen ein zu komplexes Vorhaben darstellen würde. Das Deaktivieren einer WFS-Quelle wäre theoretisch unkompliziert, doch da dieser Effekt durch eine Geste nicht rückgängig gemacht werden könnte, wird auf diese Möglichkeit ebenfalls verzichtet.

Die weiteren Aktionen, welche direkt auf WFS-Quellen ausgeführt werden können (Typ, Farbe, Gruppen-ID, Rotationsrichtung, Skalierungsrichtung und Dopplereffekt) stellen allesamt eher Einstellungen dar, welche vorgenommen werden, bevor die Echtzeitsteuerung durch die Gesten angetreten wird.

**Die Positionierung der WFS-Quellen** auf der Ebene der WFS-Anlage soll also das Ziel der Gesten sein. Dabei soll jede beliebige Position erreicht werden können.

## 6.5. Entwicklung der Gesten

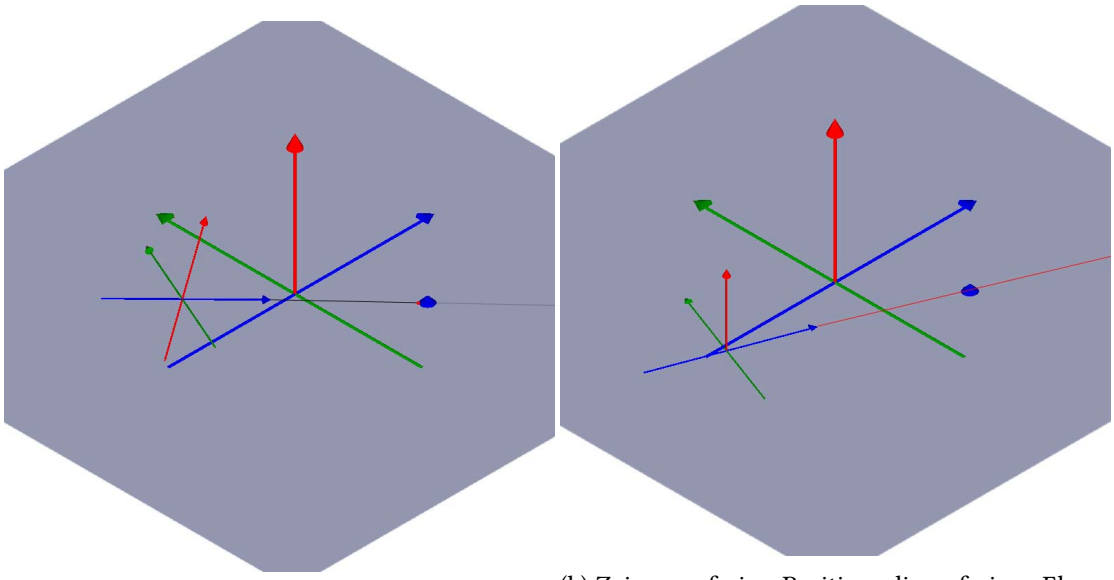
In diesem Kapitel wird beschrieben, welche Gesten notwendig sind, um die in Kapitel 6.4 gesteckten Ziele zu erreichen, wie diese genau aussehen sollen und wie sie erarbeitet wurden.

### 6.5.1. Ziel

Grob ist das Ziel schon bekannt. Beliebige WFS-Quellen sollen durch möglichst intuitive Gesten frei auf der Ebene der WFS-Anlage positioniert werden können.

### 6.5.2. Zusammenstellung eines Gestenrepertoire

Obwohl die WFS-Quellen eigentlich »nur« bewegt werden sollen, sind dafür mehrere Schritte notwendig. Zuerst muss dem System mitgeteilt werden, um welche WFS-Quelle es sich handelt (Auswählen). Danach kann die Positionierung durchgeführt werden (Bewegen) und es muss möglich sein, die Wahl der WFS-Quelle zu ändern, wozu die Auswahl der zuvorigen WFS-Quelle zunächst aufgehoben werden muss (Abwählen), um dann erneut eine WFS-Quelle auswählen zu können. Absehbar war schon vor den Tests, dass es zum Bewegen der WFS-Quellen wahrscheinlich mehrere Gesten geben wird, denn würde man sich außerhalb des Systems befinden, wie bei der Bedienung über xWonder (Kapitel 2.4.2), hätte man eine Draufsicht auf die Ebene der WFS-Anlage und der Ansatz für die Positionierung wäre dann trivial. Ein Punkt der Ebene ließe sich leicht eindeutig bestimmen, da der Strahl, welcher durch die Blickrichtung oder ein Zeigewerkzeug gebildet wird, die Ebene genau ein mal schneidet. Dieser Schnittpunkt markiert die Position auf der Ebene, wie in Abbildung 6.1a zu sehen. Da das System jedoch rein durch akustisches Feedback bedienbar sein soll, reicht eine abstrakte Darstellung der Umgebung (wie sie zum Beispiel xWonder bietet) nicht aus. Der Anwender muss sich selbst auf die Ebene der WFS-Anlage begeben, um die Positionen der WFS-Quellen orten zu können. Damit lässt sich aber eine eindeutige Position auf der Ebene nicht mehr so leicht bestimmen, denn ein Strahl, welcher nun durch ein Zeigewerkzeug gebildet wird, liegt ebenfalls komplett auf der Ebene. Somit liegt jeder beliebige Punkt des Strahls ebenfalls auf der Ebene und könnte somit auch das intendierte Ziel darstellen (siehe Abb. 6.1b). Zur Bestimmung eines konkreten Punktes der Ebene bedarf es also weiterer Informationen. Es muss zwischen zwei Arten der Bewegung unterschieden werden. Eine dieser Bewegungen positioniert eine WFS-Quelle in eine, vom Akteur aus gesehen, andere Richtung, wobei die Entfernung zum Akteur jedoch nicht verändert wird. Dies ergibt eine Bewegung der WFS-Quelle auf einer Kreisbahn, mit dem Akteur im Mittelpunkt des Kreises. Die zweite Bewegung verändert die Distanz der WFS-Quelle zum Akteur, dabei bleibt der Winkel, in dem die WFS-Quelle zum Akteur steht jedoch unverändert. Präziser wird dies im Kapitel 6.6.4 unter Technische Umsetzung erläutert und behandelt.



(a) Zeigen auf eine Position, die auf einer Ebene liegt ohne sich selber auf der Ebene zu befinden. Der Strahl des Zeigers schneidet die Ebene auf einem ganz konkreten Punkt. Das Ziel wird eindeutig bestimmt.

(b) Zeigen auf eine Position, die auf einer Ebene liegt, während man sich selber ebenfalls auf der Ebene befindet. Der Strahl befinde sich komplett in der Ebene. Somit könnte jeder Punkt, der auf diesem Strahl liegt, das Ziel sein. Das Ziel wird nicht eindeutig bestimmt.

Abbildung 6.1.: Zeigen auf einen Punkt im $\mathbb{R}^3$ und $\mathbb{R}^2$

### 6.5.3. Tests

Um die Gesten so intuitiv wie möglich zu kreieren, wurde zuerst ein Versuch entworfen, um zu ermitteln, wie verschiedene Personen so eine Steuerung erwarten würden.

10 Probanden mit sehr verschiedenem technischen Allgemeinwissen und ohne Vorwissen über diese Arbeit wurden einzeln in einen Raum geführt. Ihnen wurde erklärt, dass sie sich vorstellen sollen, dass um sie herum, in unerreichbarer Entfernung, so dass sie nicht einfach hingehen können, Soundquellen in der Luft schweben und dass sie nun aus der Distanz einzelne Quellen unter Zuhilfenahme lediglich einer Hand repositionieren sollen.

Die verschiedenen Ergebnisse sollten dann analysiert werden, in de Annahme, dass sich in ihnen genügend Gemeinsamkeiten befinden, um daraus Gesten zu entwickeln, die es der Großzahl dieser Probanden und damit hoffnungsvoll auch der Großzahl aller weiterer Personen, ermöglichen das System später ohne weitere Einführung genau so benutzen zu können, wie sie es jetzt versucht haben.

Um die Gesten im Anschluss so simpel wie möglich zu gestalten und um außerdem das bei Gesten übliche Start-Ende Problem[2] zu vereinfachen, wird das Augenmerk bei den Tests neben den Bewegungen vor allem auf die von einer Bewegung unabhängigen Stellungen der Hand bei bestimmten Intentionen gerichtet, da so eine Handhaltung anhand eines Bildes zu erkennen ist, ohne eine Bewegung interpretieren zu müssen und daher kein Start und kein Ende hat.

Im folgenden werden die Versuche und die ausschlaggebenden Ergebnisse beschrieben. Eine ausführlichere Aufstellung aller Probanden befindet sich im Anhang (Kapitel B).

**Für den ersten Versuch** wurden die Probanden in die Mitte des Bereiches der WFS-Anlage geführt. Von dort aus wurde in eine Richtung gedeutet, in der sich eine angeblich fiktive zu repositionierende Soundquelle außerhalb des realen Raumes befindet (Die Versuche wurden in geräuschloser Umgebung durchgeführt). Danach wurde in eine andere Richtung gedeutet, in welche die Soundquelle bewegt werden soll. Vorweggenommen wurden erstens, dass nur eine Hand benutzt werden darf und zweitens, dass es bei dem ersten Versuch vor allem um die Richtung geht, was, wie schon beschrieben (Kapitel 6.5.2), die Endposition lediglich auf eine unendliche Anzahl von Möglichkeiten auf dem Strahl beschränkt (siehe Abb. 6.1b), was aber nicht explizit ausgesprochen wurde.

Was die Gesten aller Probanden beim Selektieren der Soundquelle gemein hatten, war, dass sie den Arm der Soundquelle entgegen ganz ausgestreckt halten. Dabei wurde die Hand meist mit dem Handrücken nach oben und leicht zum Körper geneigt gehalten. Die Handfläche öffnete sich damit in Richtung Soundquelle. Die größten Unterschiede lagen darin, dass einige Probanden die Hand in der offenen Haltung beließen, andere eine greifende Bewegung ausführten und die Hand im geschlossenen oder halb geschlossenen Zustand hielten.

Nach der Auswahl einer Soundquelle wurde der Arm meist in gleicher Haltung durch eine Drehung des Schultergelenks oder des Körpers auf die gewünschte Richtung ausgerichtet und im Anschluss ggf. wieder geöffnet und unachtsam in eine natürlichere Haltung gesenkt. Einige Probanden hielten die Hand, nachdem die Soundquelle ausgewählt war, etwas zum eigenen Körper herangezogen, dann die Drehung ausgeführt und den Arm in der Wunschrichtung zum »platzieren« der Soundquelle wieder gestreckt. Damit wurden (teils unbewusst) drei Gesten ausgeführt: Auswählen, Bewegen und Abwählen einer Soundquelle.

Im Anschluss wurden die Probanden befragt, wie genau sie sich den Verlauf der Position auf der Ebene der WFS-Anlage während der Bewegung vorstellen bzw. vorgestellt haben.

[2] Schwierig ist bei Gesten oft, zu erkennen, wann eine Geste beginnt und wann sie endet. Welcher Teil einer Bewegung also interpretiert werden soll.

Alle Probanden sind zu dem Entschluss gekommen, dass die Soundquelle sich auf einer kreisförmigen Bahn bewegen muss, bei welcher der Mittelpunkt des Kreises die Position des Probanden ist und sich somit immer im gleichen Abstand zum Probanden befand.

**Für den zweiten Versuch** wurde den Probanden aufgetragen die besagte Soundquelle nun näher an die eigene Position heranzuholen, da sie sich bei dem ersten Versuch immer in gleicher Entfernung zur eigenen Position befunden hatte. Die Auswahl der Soundquelle fand (bis auf eine Ausnahme) immer genau so statt, wie bei dem ersten Versuch. Danach wurde der Arm gebeugt und die Hand somit näher an den Körper herangebracht und abschließend wurde der Arm wieder gesenkt.

Bei der Frage nach dem genauen Effekt dieser Geste gingen die Erwartungen etwas auseinander.

Der erste Impuls war meist, dass die Entfernung der Soundquelle zur eigenen Position sich linear um die gleiche Distanz verringert, um die sich die Entfernung der Hand zum eigenen Körper verringert. Auf die Frage, wie man dann die Distanz einer Soundquelle um mehrere Meter oder Dekameter verändert, waren die meisten Probanden eher ratlos. Irgend eine Art kontinuierlicher Veränderung wäre für die meisten denkbar.

**Die letzte Aufgabe** bestand darin, die Distanz der besagten Soundquelle zur eigenen Position nun zu erhöhen. Die Auswahl der Soundquelle unterschied sich dabei in der Regel nicht von den vorherigen Aufgaben. Die Intention war nun meistens, die Hand dann weiter von der eigenen (ursprünglichen) Position der Soundquelle entgegen zu bewegen, wobei dafür meist ein Schritt nach vorne von Nöten war, da die Soundquelle, wie zuvor, schon mit ausgestrecktem Arm ausgewählt wurde und die Hand somit nur noch durch eine Bewegung des ganzen oder des Oberkörpers weiter der Soundquelle entgegen bewegt werden konnte. Bei der Wiederholung des gleichen Tests, wurde die Quelle dann in der Regel mit leicht angewinkeltem Arm ausgewählt, um durch die anschließende Streckung des Arms die gewünschte Repositionierung der Hand zu erreichen.

### 6.5.4. Auswertung der Tests

Die Tests haben die Vermutung bekräftigt, dass für das Bewegen der WFS-Quellen mehrere verschiedene Gesten notwendig sein werden. Zumindest eine für die Bewegung auf der Kreisbahn um die Position des Akteurs und eine oder zwei weitere, um die Entfernung zu beeinflussen (verringern, erhöhen).

Abbildung 6.2.: Verschiedene Möglichkeiten der Handhaltung zum Auswählen einer WFS-Quelle

#### 6.5.4.1. Auswählen

Eine sehr wichtige Erkenntnis der Versuche war, dass die Geste zum Auswählen einer WFS-Quelle bei allen Probanden sehr ähnlich war. Die Stellung und Bewegungen der Finger können direkt vernachlässigt werden, da das Target auf dem Handrücken angebracht wird und somit lediglich die Position und Ausrichtung des Handrückens bzw. der Handfläche misst. Unter Einbezug dieser Einschränkung werden die Gesten noch ähnlicher. Es stellt sich heraus, dass der Handrücken immer aus der Waagerechten um grob 45 Grad nach oben zum Körper hin geneigt wird (siehe Abb. 6.2). Als Erkennungsmerkmal der LockGesture soll also diese Haltung der Hand bzw. des Handrückens dienen.

#### 6.5.4.2. Bewegen (kreisförmig)

Der Ansatz zum Repositionieren einer WFS-Quelle auf einer kreisförmigen Bahn ist naheliegend: Nachdem eine WFS-Quelle ausgewählt wurde, wird diese stetig in der Richtung, in die gedeutet wird, in der gleichen Entfernung zur Hand platziert, wie zuvor.

#### 6.5.4.3. Bewegen (Distanz)

Die Tests haben die Vermutung bestätigt, dass die Geste,die die Soundquelle dem Akteurs näher zu bringen soll, üblicherweise einer ziehenden Bewegung ähnelt. Die Hand wird also, analog zu dem Effekt, der auf die Soundquelle ausgeübt werden soll, näher geholt und die Soundquelle damit »herangezogen«. Analog dazu wird die Soundquelle, um die Distanz zu erhöhen, »weggeschoben«. Die Hand wird also mit ausgewählter Quelle weiter vom Körper des Akteurs, in die Richtung der Soundquelle, entfernt.

Den exakten Effekt, den diese Gesten auf die ausgewählte Soundquelle haben sollen, konnte leider nicht ermittelt werden, da die meisten Probanden sich darüber selber unschlüssig waren. Herausgearbeitet wurde jedoch, dass es eine Kontinuität in der Bewegung der Soundquelle

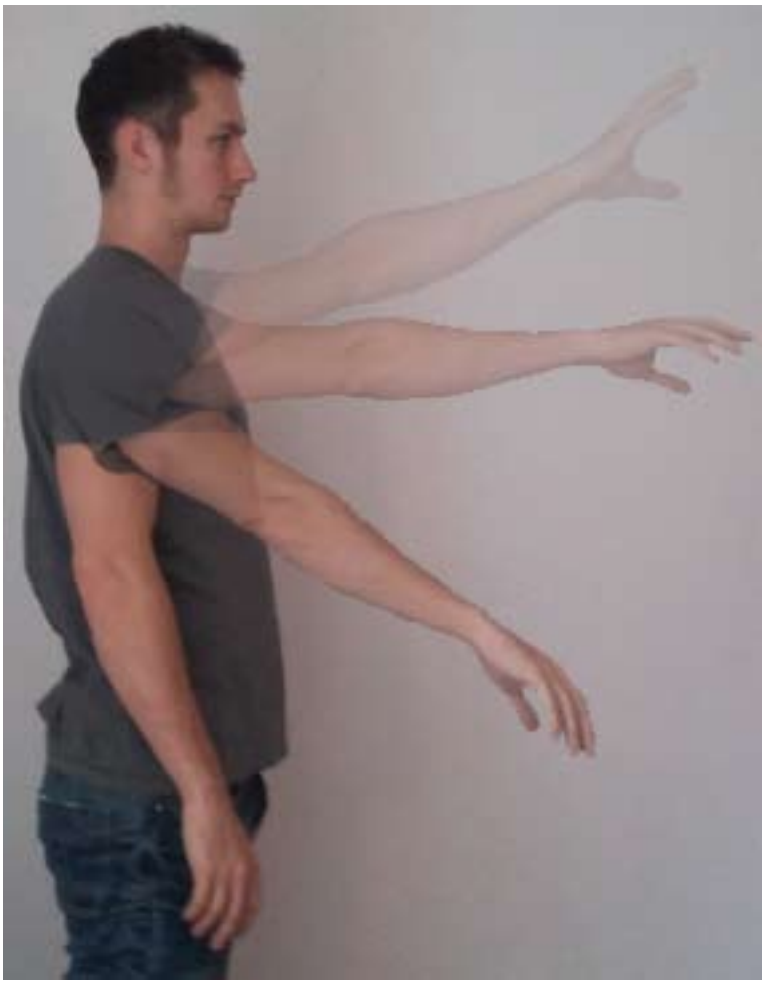


Abbildung 6.3.: Bewegung des Arms und der Hand nach Abschluss einer Geste

geben muss, die nicht der Bewegung der Hand entspricht, damit große Distanzen erreichbar werden.

#### 6.5.4.4. Abwählen

Eine weitere Gemeinsamkeit aller Versuche war, dass die Probanden nach Abschluss einer Aufgabe ihren Arm und damit auch die Hand wieder in eine natürlichere Haltung gesenkt haben. Wenn man dabei das Target auf dem Handrücken betrachtet bedeutet das, dass sich die Höhe der Position verringert und die Hand sehr stark nach unten geneigt wird (siehe Abb. 6.3). Die Position der Hand zu benutzen hätte jedoch einen entscheidenden Nachteil. Wenn mehrere Gesten nacheinander auf verschiedene WFS-Quellen ausgeführt werden sollen, müsste der

Abbildung 6.4.: Abwinkeln der Hand zum Abwählen einer WFS-Quelle

ganze Arm jedes Mal zwischen den Gesten gesenkt und wieder angehoben werden, um eine neue WFS-Quelle auszuwählen. Den Neigungswinkel der Hand zu benutzen hat dagegen den Vorteil, dass man in so einem Fall lediglich den Neigungswinkel der Hand verringern und den Arm in gehobener Position verweilen lassen könnte (siehe Abb. 6.4). Das spart Kraft und Zeit und die Geste würde dennoch erkannt, wenn der Arm einfach wieder gesenkt würde.

## 6.6. Technische Umsetzung

Die GestureComponent ist ein Modul, welches dem MoWeC hinzugefügt wird und stellt für das gesamte System eine optionale Komponente dar. Sie wird vom MoWeC ausgeführt und bedient sich dann der vom MoWeC zur Verfügung gestellten Funktionen (siehe Abb. 6.5).

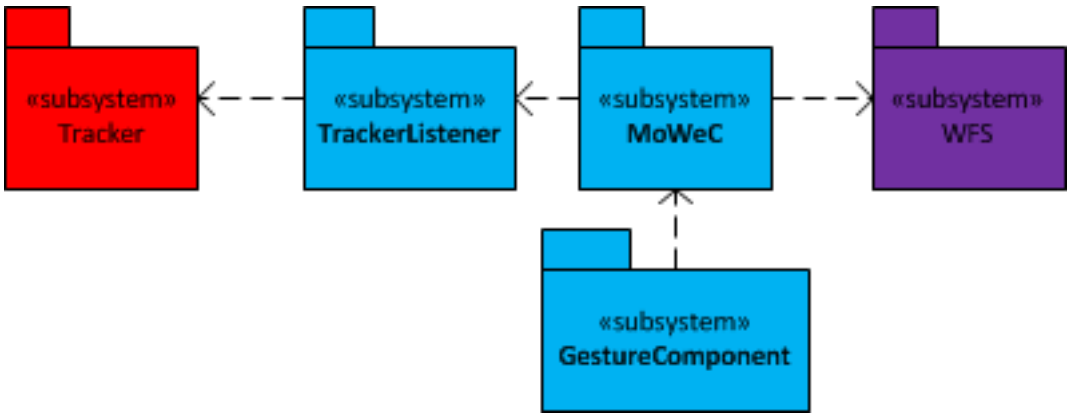



Abbildung 6.5.: Eingliederung der GestureComponent in die Umgebung

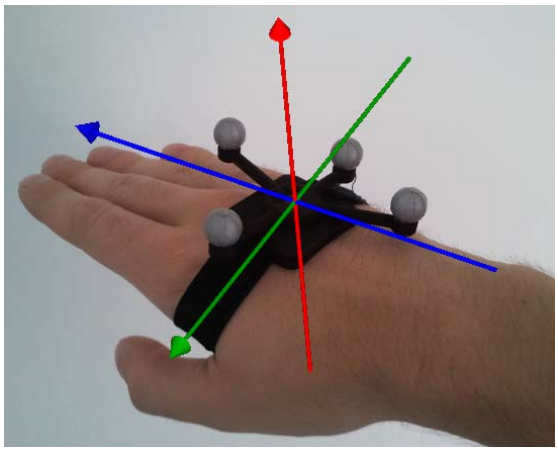


Abbildung 6.6.: Hand mit aufgebrachtem Target und eingeblendetem Koordinatensystem des Targets

Um die Position und Ausrichtung der Hand erfassen zu können, wird ein Target des A.R.T.-Systems auf dem Handrücken angebracht (siehe Abb. 6.6).

Das Target wird so auf der Hand platziert, dass der Strahl der X-Achse des Targets ($X_T$) der Zeigerichtung der Hand bzw. einer Verlängerung des Arms entspricht. Die Y-Achse des Targets ($Y_T$) verläuft auf der Ebene, auf welcher der Handrücken liegt orthogonal zu $X_T$ und die positive Richtung verläuft (bei der rechten Hand) von der Handaußenkante in Richtung des Daumens. Die Z-Achse des Targets ($Z_T$) verläuft wiederum orthogonal zu beiden anderen Achsen und die positive Richtung verläuft von der Handfläche in Richtung des Handrückens. Gemessen wird die Position und die Rotation des Targets relativ zum Koordinatensystem des Raumes (X-Achse des Raumes ($X_R$), Y-Achse des Raumes ($Y_R$) und Z-Achse des Raumes ($Z_R$)). Die X-, Y- und Z-Werte der Position geben an, um welche Distanz das Target innerhalb des Koordinatensystem des Raumes auf den jeweiligen Achsen verschoben wurde. Die Winkel $\alpha$, $\beta$ und $\gamma$ geben an, um wie viel Grad das Target zum Koordinatensystem des Raumes um die Achsen $X_R$, $Y_R$ und $Z_R$ rotiert wurde. Die Rotationsmatrix gibt die Richtungskosinusse der Achsen des Koordinatensystem des Targets zu den Achsen des Koordinatensystem des Raumes an [BSMM05, S. 217]. Für eine ausführlichere Erklärung von Rotationsmatrizen siehe Rotationsmatrizen (Kapitel A).

Da der MoWeC bereits die Funktionalität bereitstellt, Targets Sources zuzuweisen, soll die Repräsentation der Hand ganz einfach durch eine Source geschehen. Der GestureComponent müssen lediglich alle Sources übergeben werden, die berücksichtigt werden sollen. Dazu gehört ebenfalls die Source, welche die Hand repräsentiert. Durch die GestureComponent können alle Sources modifiziert werden, welche nicht Händen zugewiesen sind. Das Ergebnis und die

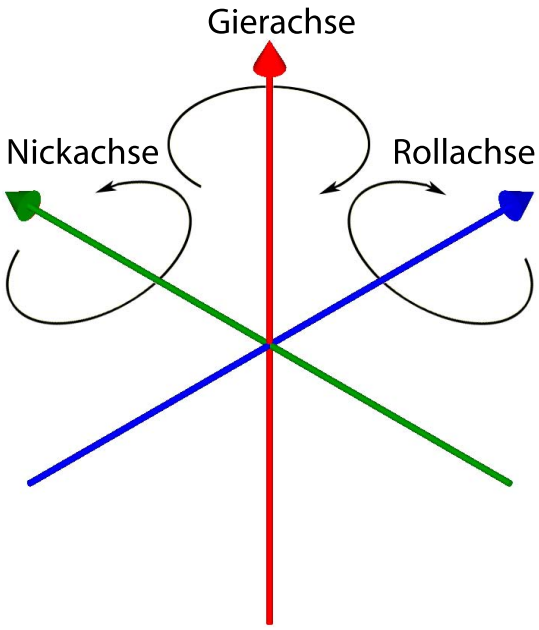



Abbildung 6.7.: Roll-Achse, Nick-Achse und Gier-Achse mit Drehsinn

Rückgabe der GestureComponent an der Coordinator, ist die gleiche Menge an Sources, die zu Beginn übergeben wurde, nur wurden einige Sources möglicherweise durch die Ausführung von Gesten verändert.

Für die Rotation eines Objektes im $\mathbb{R}^3$ um die eigenen drei Achsen ($X_T$, $Y_T$ und $Z_T$) wurden (zuerst für Luftfahrzeuge) eigene Begriffe eingeführt. Rollen, Nicken und Gieren, welche Rotationen um die Roll-Winkel, Nick-Winkel und Gier-Winkel um die Roll-Achse, Nick-Achse und Gier-Achse beschreiben (siehe Abb. 6.7) [Wik12a].

- **Rollen**: Rotation um die X-Achse, welche in »Fahrtrichtung« verläuft. In diesem Fall der Zeigerichtung. Also ein Kippen zu den Seiten.
- **Nicken**: Rotation um die Y-Achse, welche von links nach rechts verläuft. Bewirkt das Heben oder Senken der Schnauze (in diesem Fall der Fingerspitzen) und hat den entgegengesetzten Effekt auf das Heck.
- **Gieren**: Rotation um die Z-Achse, welche von unten nach oben verläuft. Bewirkt (in Ausgangsstellung) eine Rotation auf der horizontalen Ebene.

In der Regel ist, wenn von Rotationen gesprochen wird, die relative Ausrichtung des Koordinatensystem eines Objektes zu einem Referenz-Koordinatensystem (oft das Koordinatensystem des Raumes, in dem sich das Objekt befindet) gemeint. Dieses Koordinatensystem dreht sich

nicht mit und die Auswirkung, welche eine Rotation um diese Achsen auf das Objekt haben, sind stark abhängig von der momentanen Ausrichtung des Objektes. Wenn von Rollen, Nicken und Gieren die Rede ist, verstehen sich die Angaben immer relativ zu der aktuellen Ausrichtung des Objektes. Folgendes Beispiel sollte dies verdeutlichen: Ein Pilot, welcher an seinem Steuerknüppel zieht, zieht sein Flugzeug damit hoch und gewinnt an Höhe. Nicht jedoch, wenn das Flugzeug kopfüber fliegt. In diesem Fall hätte ein »hochziehen« des Piloten ein Verlust an Höhe zur Folge. Dies ist darauf zurückzuführen, dass die Z-Achse des Flugzeuges, welche sich bei der Rotation des Flugzeugs auf den Kopf ebenfalls mitgedreht hat und nun in die entgegengesetzte Richtung zeigt. Es ist also substanziell, dass eindeutig ist, auf welches Koordinatensystem sich bei Richtungsangaben bezogen wird. Durch diese eindeutigen Begriffe, ist das sichergestellt.

### 6.6.1. Zustände der Hand

Die Hand kann durch Gesten in verschiedene Zustände (States) gebracht werden. Abbildung 6.8 zeigt ein Zustandsdiagramm der Hand. Die States werden also durch Gesten erstellt. Diese States enthalten Informationen, welche von der Art des States und von den Gegebenheiten bestimmt werden, unter welchen der gegebene State erstellt wurde. Diese Gegebenheiten sind der Zeitpunkt, die Position und Ausrichtung der Hand sowie die Source, mit all ihren Informationen, auf welche die Geste abzielte, durch welche der State erstellt wurde.

Von der Art des States hängt in erster Linie ab, welche Gesten ausgeführt werden können bzw. welche erkannt werden können. Durch die verschiedenen States einer Hand wird eine Logik unter den Gesten erzeugt, denn nicht jede Geste macht zu jedem Zeitpunkt Sinn. Dies beugt Konflikten zwischen verschiedenen Gesten vor, da die Anzahl der Gesten, welche zu einem gegebenen Zeitpunkt ausgeführt werden kann, eingeschränkt wird und wirkt sich zugleich positiv auf die Rechenlast aus, indem nicht ausführbare Gesten auch keinerlei Rechenlast verursachen. Zu Beginn befindet sich die Hand im FreeState. In diesem State wird nur eine Geste ermöglicht. Die LockGesture. So lange diese Geste nicht erkannt wird, verbleibt die Hand im FreeState. Wird die LockGesture erkannt, so wechselt die Hand zuerst in den LockingState, von dort weiter in den LockedState oder, wenn die Geste zuvor abgebrochen wurde, zurück in den FreeState. Von allen übrigen States kann immer durch Ausführen der UnlockGesture in den FreeState zurückgekehrt werden. Von dem LockedState kann zusätzlich durch Ausführen der MoveGesture in den MoveState und durch die PushPullGesture in den PushPullState gewechselt werden. Von diesen beiden States kann jeweils auch wieder in den LockedState zurückgewechselt werden.

#### 6.6.1.1. Informationen der Zustände

In diesem Abschnitt wird kurz erklärt, welche Informationen in welchen States enthalten sind und wann diese geändert werden. Warum dies in den einzelnen Fällen so ist, wird in den Kapiteln 6.6.2 bis 6.6.4 erklärt.

Die Informationen der States dienen dazu, zu einem späteren Zeitpunkt wichtige Daten, die nur zu bestimmten Zeitpunkten vorlagen, für die Ausführung der Gesten jedoch notwendig sind, zu sichern und bereitzustellen.

In jedem State ist festgelegt, welche Gesten erkannt werden sollen. Dies wurde bereits erläutert und ist ebenfalls in Abbildung 6.8 zu erkennen. Der FreeState enthält auch keine weiteren Informationen, da dieser der Ausgangszustand ist. In allen weiteren States sind folgende Informationen enthalten:

- Die Source, auf welche die Gesten abzielen (jedem State, bis auf den FreeState, ist genau eine Source zugewiesen, welche beim Start des Auswahlvorganges festgelegt und dann immer weitergereicht wird).
- Die Source, auf welche die Gesten abzielen zu dem Zeitpunkt und in dem Zustand, als sie ausgewählt wurde. Insbesondere die Position $sourcePositionAtLock$ ist relevant.
- Die Hand, zu dem Zeitpunkt und in dem Zustand, als die Source ausgewählt wurde. Dazu gehören zum Beispiel:
  - Die Position der Hand $handPositionAtLock$ und
  - die Ausrichtung der Hand $handAngleAtLock$, welche beide ebenfalls ihren Zustand behalten.
- Die Position der Hand, als der State erstellt wurde $P_S$.
- Die Ausrichtung der Hand, als der State erstellt wurde ($A_S$).
- Der Zeitpunkt, zu dem der State erstellt wurde ($T_S$).

### 6.6.2. Auswählen

Um eine WFS-Quelle auszuwählen, soll eine bestimmte Handhaltung erkannt werden, welche durch eine Rotation der Hand gegeben sein soll. Um die erforderlichen Winkel der Hand (Kapitel 6.5.4.1) zu ermitteln, muss zunächst ermittelt werden, was das genau für die Rotation des Targets zum Koordinatensystem des Raumes bedeutet.

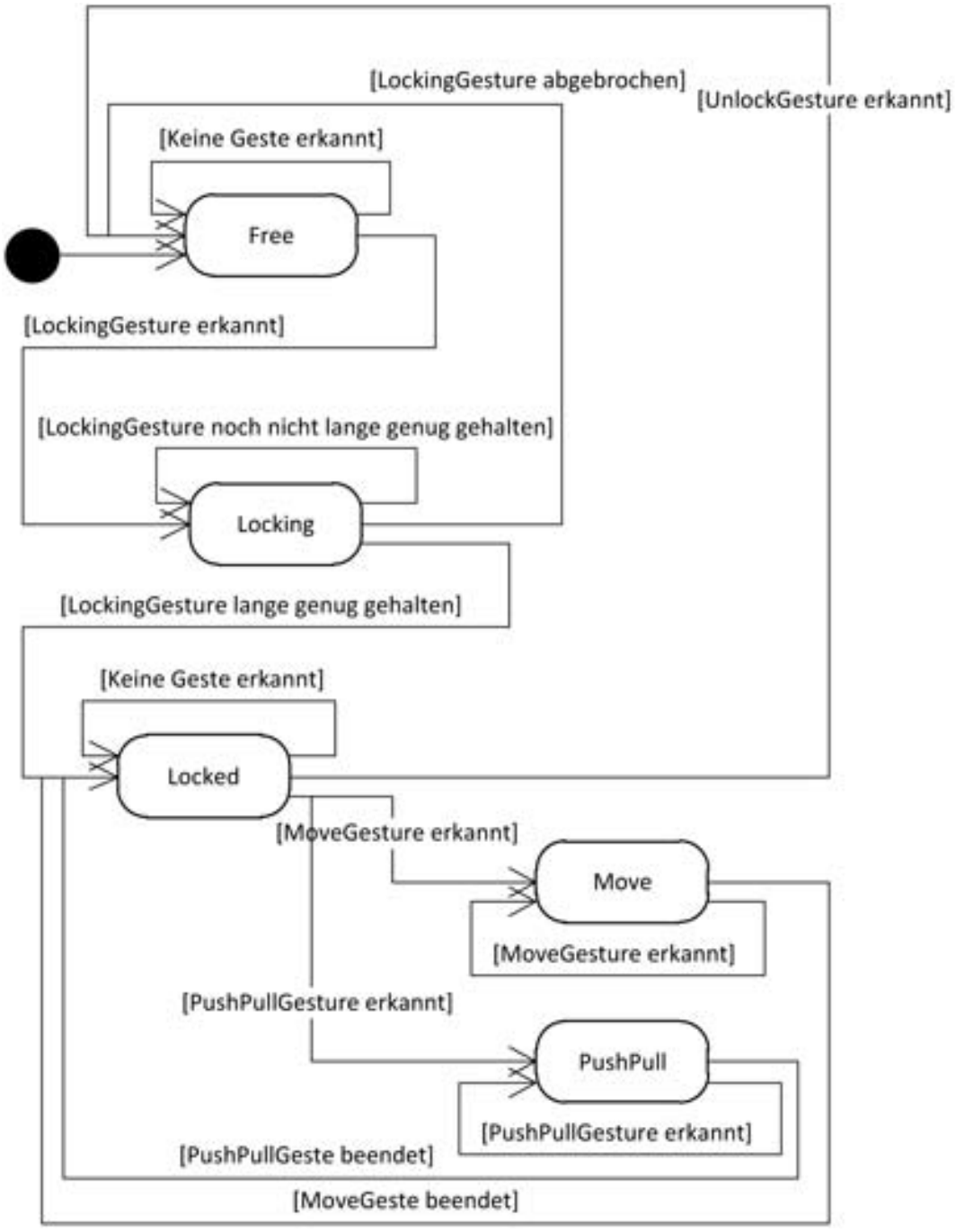
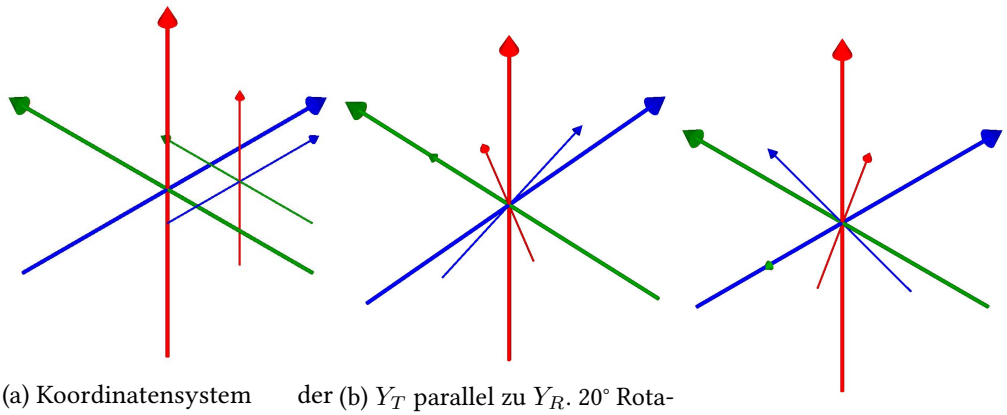



Abbildung 6.8.: Zustandsautomat der Hand



(a) Koordinatensystem der Hand auf allen Achsen parallel zu dem Koordinatensystem des Raumes.

(b) $Y_T$ parallel zu $Y_R$. 20° Rotation um $Y_R$( und in diesem Fall somit auch um $Y_T$).

(c) Zusätzlich 90° Rotation um $Z_R$.

Abbildung 6.9.: Ausrichtungen

Betrachte man das Koordinatensystem der Hand bzw. des Targets der Hand[3], so liegt bei der Ausführung der Geste ein Nicken vor (siehe Abb. 6.10). So lange $X_R$ und $X_T$ parallel zueinander stehen (siehe Abb. 6.9a), entspricht diese Rotation auch einer Rotation um $Y_R$ (siehe Abb. 6.9b). Da die Hand sich jedoch im Raum frei bewegt, ist dieser Zusammenhang in der Regel nicht gegeben. In Abbildung 6.9c wurde die Hand exemplarisch zuvor um 90° um $Z_R$ rotiert. In diesem Fall entspricht die Rotation für die Geste einer Rotation um $X_R$. Es müsste also ebenfalls die Rotation um $Z_R$ berücksichtigt werden. Die Rotation um $Z_R$ bestimmt, wie stark sich die Rotation der Geste anteilig auf $X_R$ und $Y_R$ auswirkt. Befindet sich $Y_T$ parallel zu entweder $X_R$ oder $Y_R$, wie in Abb. 6.9b und 6.9c, so findet die Rotation zu hundert Prozent auf dieser Achse statt. In diesem Fall beträgt der Winkel zwischen $Y_T$ und der jeweils anderen Achse des Raumes 90° und auf der Achse findet keine Rotation statt, welche zu der Geste beiträgt. Lediglich ein Rollen der Hand würde eine Rotation auf dieser Achse zur Folge haben (siehe Abb. 6.7). Beträgt die Rotation um $Z_R$ 180°, so wird wieder um $Y_R$ rotiert, jedoch in die entgegengesetzte Richtung. Die X-Achsen wären antiparallel und ein positives Nicken der Hand hätte, in diesem Fall, die gleiche Rotation zur Folge, wie ein negatives Nicken bei parallelen X-Achsen. Beschränkt sich die Rotation um $Z_R$ nicht auf ein Vielfaches von +/-90° oder 0°, so hat das Nicken der Hand eine Rotation um $X_R$ **und** $Y_R$ zur Folge. Jeweils zu einem Anteil von $0\% <= x <= 100\%$. Genauer soll darauf in dieser Arbeit nicht eingegangen

[3]Das Koordinatensystem der Hand und das Koordinatensystem des Targets werden im Folgenden synonym betrachtet

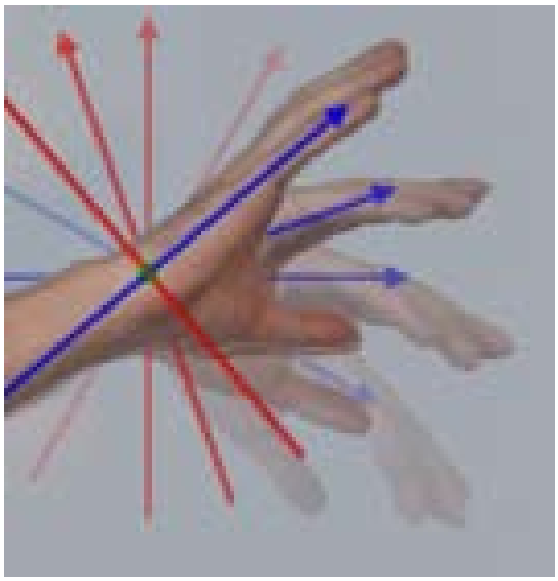


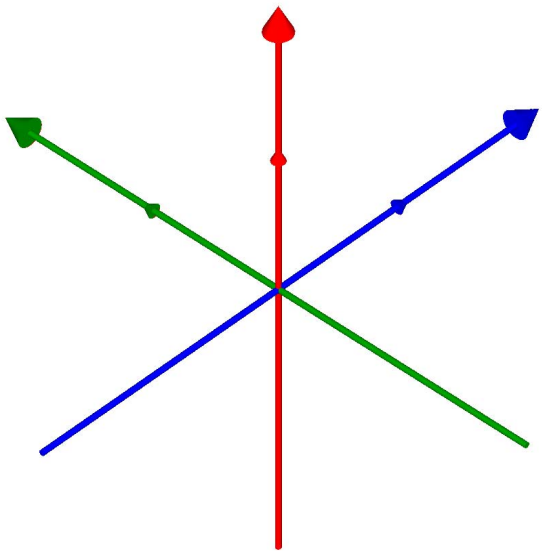
Abbildung 6.10.: Handstellungen mit eingeblendetem Koordinatensystem

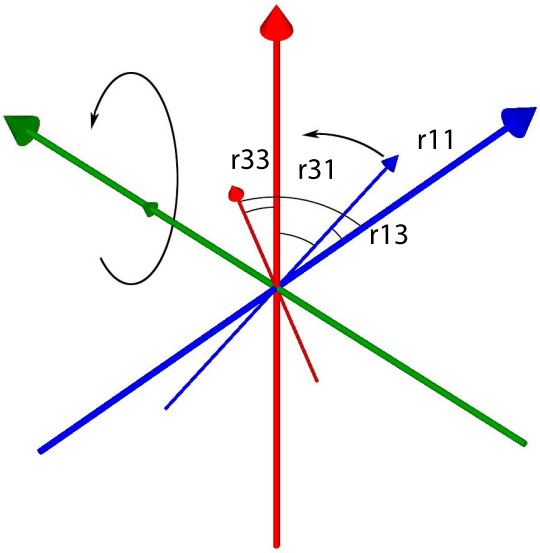
werden. Die Umsetzung ist auf diesem Wege möglich, jedoch recht umständlich und nicht nötig, wenn man sich die Rotationsmatrix des Targets zur Hilfe nimmt.

Die neun Felder der Rotationsmatrix geben an, in welchem Winkel jede einzelne Achse der Hand zu jeder Achse des Raumes steht. Genauer gesagt die Kosinusse der Winkel. Die einzelnen Felder der Rotationsmatrix werden daher durch unterschiedlichste Rotationen beeinflusst und jede beliebige Rotation beeinflusst mindestens vier bis zu alle neun Felder. Betrachtet man unter diesem Gesichtspunkt die beiden Koordinatensysteme beim Nicken der Hand, so »bewegen« sich $X_T$ und $Z_T$. In der Rotationsmatrix drückt sich das durch Änderungen in den Feldern $r11$ ($X_T$ entfernt sich von $X_R$ - der Winkel vergrößert sich), $r13$ ($Z_T$ nähert sich $X_R$ - der Winkel verringert sich), $r31$ ($X_T$ nähert sich $Z_R$ - der Winkel verringert sich) und $r33$ ($Z_T$ entfernt sich von $Z_R$ - der Winkel vergrößert sich) aus (siehe Tabelle 6.1). Das Ziel besteht nun darin, einen Zusammenhang zu finden, der von dem Gier-Winkel unabhängig ist. Der Winkel zwischen $X_R$ und $X_T$ ($r11$) ändert sich auch bei einer Rotation um $Z_R$, da sich $X_T$ hierbei von $X_R$ entfernt und $Y_R$ nähert (siehe Abb. 6.11). Damit verbleiben $r13$, $r31$ und $r33$, welche Informationen über den Nick-Winkel tragen und vom Gieren jedoch unbeeinflusst bleiben.

Am präzisesten trifft der Kosinus des Winkels zwischen $Z_R$ und $X_T$ auf ein Nicken zu, welcher in $r31$ hinterlegt ist, da dieser den Winkel, den die Roll-Achse zur senkrechten $Z_R$ bildet, beschreibt, denn jedes Mal, wenn die Fingerspitzen im Vergleich zum Schwerpunkt der Hand angehoben werden, nähert sich die mit den Fingern verlaufende $X_T$ der Ausrichtung von

Tabelle 6.1.: Koordinatensystem der Hand in Ausgangsposition (links) und mit einer Rotation von 20° um die Nick-Achse (rechts) jeweils mit Rotationsmatrix






$$D_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D = \begin{pmatrix} 0.9396926207859084 & 0 & 0.3420201433256687 \\ 0 & 1 & 0 \\ -0.3420201433256687 & 0 & 0.9396926207859084 \end{pmatrix}$$

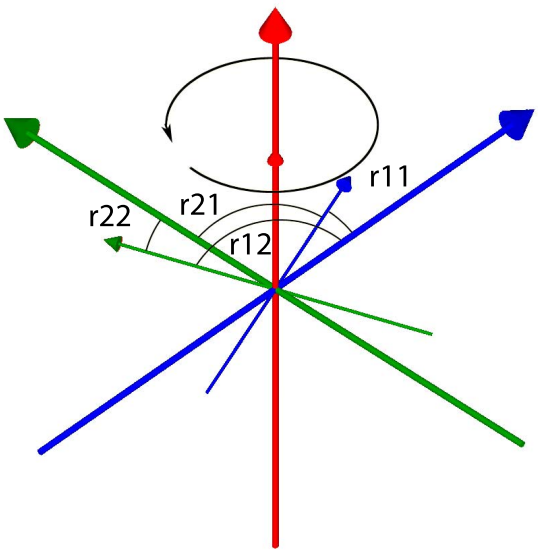
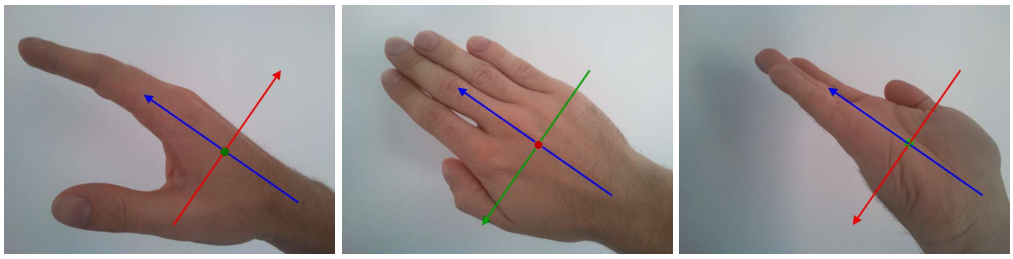



Abbildung 6.11.: Rotation um die Gier-Achse mit beeinflussten Winkeln und Bezeichnung der Felder der Rotationsmatrix, welche die Winkel angeben



(a) Richtige Handhaltung (b) Falsche Handhaltung (c) Falsche Handhaltung

Abbildung 6.12.: Haltungen der Hand, welche die Bedingungen des Nicken erfüllen würden, würde lediglich $r31$ der Rotationsmatrix als Indikator genommen

$Z_R$. Allein reicht dies aber noch nicht aus, da dieser Winkel auch noch durch Handhaltungen gegeben sein kann, welche nicht der entworfenen Geste entsprechen, den ein Rollen der Hand, welches diesen Winkel definitionsgemäß nicht ändert, ließe noch ein weites Spektrum anderer Handhaltungen zu (siehe Abb. 6.12). Um das Rollen zu erkennen und damit ausschließen zu können, wird der Winkel zwischen der Nick-Achse (da diese sich ja durch das Nicken nicht verändert) und $Z_R$ hinzugezogen, welcher in $r32$ der Rotationsmatrix hinterlegt ist. Dieser Winkel ändert sich ausschließlich durch Rollen der Hand. Damit steht fest, welche Winkel für die Erkennung der Geste relevant sind. Der nächste Schritt ist es, zu definieren, welche Werte diese Winkel haben müssen bzw. dürfen. Durch die Tests wurde ermittelt, dass ein Auswahl-Nick-Winkel des Handrückens von etwa 45° umgesetzt werden soll, was einem Kosinus von 0,707 entspricht (dieser Wert wird nun exemplarisch vorausgesetzt, kann jedoch in der Implementierung leicht geändert werden). Diesem Wert ist jedoch eine Toleranz hinzuzufügen, da es unmöglich ist diesen Winkel in der Präzision des A.R.T.-Systems zu erreichen. Diese Toleranz ist ebenfalls ein Winkel, welcher zum einen zu dem Auswahl-Nick-Winkel addiert wird, um den $maxNickWinkel$ zu erhalten und welcher vom Auswahl-Nick-Winkel subtrahiert wird, um den $minNickWinkel$ zu erhalten. Diese beiden Winkel bilden die Grenzen, in denen sich der Nick-Winkel befinden muss. Je höher der Wert der Toleranz gewählt wird, desto leichter wird die Geste erkannt aber je leichter ist es auch möglich, dass eine Handhaltung ungewollt als Geste erkannt wird. Diese Werte können nach Bedarf auf Anforderungen oder Personen angepasst werden. Mit dem Auswahl-Roll-Winkel wird ebenso verfahren, um den $maxRollWinkel$ und den $minRollWinkel$ zu ermitteln. Nun müssen lediglich die entsprechenden Felder der Rotationsmatrix ($r31$ und $r32$) mit diesen Grenzwerten verglichen werden. Damit ist das System in der Lage, verschiedene aber ähnliche Handhaltungen als Intention für die LockGesture zu interpretieren.

Nachdem die Intention des Auswählens einer WFS-Quelle erkannt wurde, muss die intendierte WFS-Quelle ermittelt werden. Diese muss sich in Richtung der Roll-Achse befinden. Bei der Richtung wird wie beim Auswahl-Nick-Winkel und Auswahl-Roll-Winkel mit einer Toleranz gearbeitet. Hierzu werden zwei Geraden gebildet, indem der Winkel der Toleranz für die eine Gerade auf die Ausrichtung der Roll-Achse addiert und für die andere von dieser subtrahiert wird und die sich auf der Position der Hand kreuzen. Somit spannen sie die Auswahlebene in Zeigerichtung auf. Auch hier gilt: Je großzügiger die Toleranz gewählt wird, desto leichter ist es, eine WFS-Quelle auszuwählen, auch wenn die Position nicht so exakt bestimmt wird. Wird die Toleranz jedoch zu hoch gewählt, so befinden sich schneller mehrere WFS-Quellen innerhalb der Auswahlebene und die intendierte WFS-Quelle ist nicht mehr so leicht zu bestimmen.

Für alle Sources, die der GestureComponent zur Verfügung stehen, wird nun der Winkel von der Hand zur der jeweiligen Source berechnet und unter all jenen, zu denen der Winkel sich zwischen den beiden Winkeln befindet, welche die Auswahlebene aufspannen, wird diejenige ausgewählt, bei welcher der Winkel am ähnlichsten ist. Das heißt, sie liegt am nächsten an dem Strahl, welcher die Zeigerichtung angibt.

Dies würde im Prinzip reichen, um per Geste eine WFS-Quelle auszuwählen. Jedoch könnte es immer noch leicht passieren, dass eine unachtsame Bewegung der Hand diese für einen kurzen Moment in die für die LockGesture nötige und hier beschriebene Haltung bringt und somit eine WFS-Quelle versehentlich ausgewählt wird. Daher wechselt die Hand zuerst in den LockingState und muss für einen kurzen Zeitraum (aktuell 250 ms) in einer Haltung verweilen, welche weiterhin all diesen Kriterien entspricht. Dazu wird im LockingState die Zeit gespeichert, zu der die LockGesture zuerst erkannt wurde und in jedem Zyklus des Systems die aktuelle Zeit mit der gespeicherten verglichen. Ist die benötigte Zeit verstrichen, so ist die Geste vollendet. Die WFS-Quelle ist somit ausgewählt und die Hand wechselt in den LockedState. Sowie während der Dauer des Auswählens die Hand in eine Ausrichtung gebracht wird, welche nicht mehr den Kriterien der LockGesture entspricht, kehrt die Hand unverzüglich in den FreeState zurück.

### 6.6.3. Abwählen

Abgewählt werden kann eine Source jederzeit sobald eine ausgewählt wurde. Das heißt sowie die Hand sich nicht im FreeState oder LockingState befindet. Die Kriterien zum Abwählen einer Source orientieren sich stark an denen zum Auswählen, nur werden sie gegensätzlich betrachtet. Es wird ebenfalls der Nick-Winkel herangezogen, wie er in Kapitel 6.6.2 (Auswählen) beschrieben wurde und verglichen wird er mit dem $minNickWinkel$ und $maxNickWinkel$

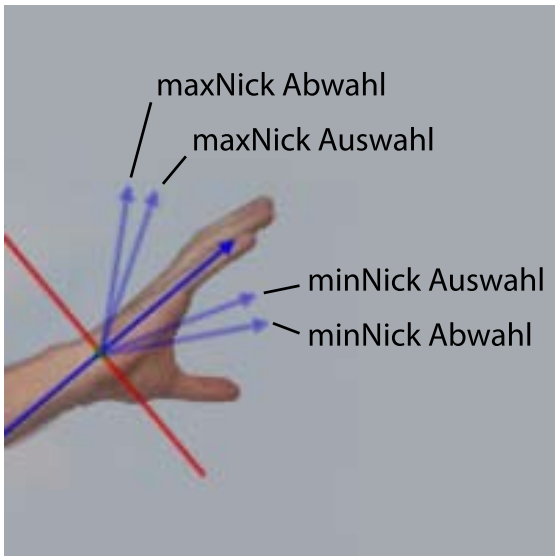



Abbildung 6.13.: Winkelbereich zum Auswählen und Abwählen einer Source

der LockGesture. Um jedoch zu vermeiden, dass nach dem Auswählen einer Source diese direkt wieder abgewählt wird, weil die Handhaltung den Nick-Winkel gerade an der Grenze zum $minNickWinkel$ oder $maxNickWinkel$ platziert und somit eine minimale Bewegung zum Abwählen reicht, wird für die UnlockGesture eine weitere Toleranz berücksichtigt und somit ein Hystereseeffekt erzeugt. Dieser erweitert den Nick-Winkel, den die Hand beim Auswählen einnehmen musste um einen in der UnlockGesture festgelegten Wert (momentan 10°) auf einen weiteren Winkel, der somit für die Erhaltung der Auswahl nötig ist (siehe Abb. 6.13). Mit dem Roll-Winkel wird ebenso verfahren. Befinden sich die entsprechenden Kosinusse der Rotationsmatrix außerhalb der durch diese Winkel begrenzten Bereiche, wird die UnlockGesture als erkannt betrachtet und die Hand kehrt in den FreeState zurück.

### 6.6.4. Bewegen

Zuvor wurde schon kurz das Problem angesprochen, welches bei dem Versuch entsteht, von einem Punkt auf einer Ebene auf einen anderen Punkt der gleichen Ebene zu deuten. Dies ist nicht möglich, wenn man sich lediglich eines Winkels bedient, da der Strahl, welcher sich in die angedeutete Richtung erstreckt, sich komplett auf der Ebene befindet und der intendierte Zielpunkt somit nicht eindeutig definiert werden kann (siehe Abb. 6.1b). Würde man sich im $\mathbb{R}^3$ an einem Punkt befinden, der sich nicht auf der Ebene befindet, auf welcher sich das Ziel befindet, so würde der Strahl die Ebene an genau einem definierten Punkt schneiden, der dadurch schon eindeutig bestimmt werden kann (siehe Abb. 6.1a).

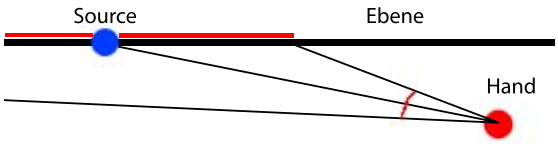



Abbildung 6.14.: Winkelbereich zum Auswählen und Abwählen einer Source. Geringe Änderungen des Winkels hätten einen großen Entfernungsunterschied auf der Ebene der WFS-Anlage zur Folge.

Das A.R.T.-System liefert uns eine Position im $\mathbb{R}^3$ und die Lautsprecher der WFS-Anlage befinden sich in einer Höhe, welche in der Regel über der Position der Hand liegen sollte (siehe Abb. 4.1). Akustisch wird die Ebene der WFS-Anlage also ungleich der Ebene empfunden, auf der sich der Akteur befindet. Damit sind im Prinzip die Voraussetzungen für eine eindeutige Positionsbestimmung gegeben. Dadurch, dass sich die Ebene der WFS-Anlage und die Ebene, welche durch die X- und Y-Achsen des Trackers jedoch in der Höhe nur leicht unterscheiden würden und zudem die WFS-Quellen sich in der Regel außerhalb des Raumes (unter Umständen weit außerhalb) befinden, würde eine minimale Änderung des Winkels der Hand schon eine große Veränderung der Position bewirken. Je weiter die WFS-Quelle entfernt ist, desto stärker wäre der Effekt und würde ein präzises Platzieren wieder sehr erschweren, wenn nicht unmöglich machen (siehe Abb. 6.14). Daher wird darauf verzichtet, von dieser Möglichkeit Gebrauch zu machen und das Bewegen von WFS-Quellen wird auf zwei unterschiedliche Gesten aufgeteilt. Die MoveGesture ermöglicht es, die WFS-Quelle in einer kreisförmigen Bahn um die aktuelle Position der Hand des Akteurs herum zu platzieren. Die Distanz zur Hand bleibt dabei immer gleich. So ist mit der Richtung und der konstanten Distanz immer ein konkreter Punkt eindeutig bestimmbar. Die PushPullGesture ermöglicht es dem Akteur, die Entfernung der WFS-Quelle zur eigenen Position zu verändern. Mit der Kombination beider Gesten ist jede Position auf der Ebene der WFS-Anlage bestimmbar.

#### 6.6.4.1. Kreisförmig

Die MoveGesture ermöglicht es, den Winkel einer WFS-Quelle zur Position der Hand zu verändern und sie somit aus der Sicht des Akteurs in eine andere Richtung zu bewegen. Hierbei bleibt die Position der WFS-Quelle im Koordinatensystem der Hand jedoch stets konstant, wenn diese sich dreht oder bewegt. Durch eine Rotation der Hand im Koordinatensystem des Raumes wird die WFS-Quelle somit jedoch vom Koordinatensystem des Raumes gesehen, relativ zur Hand in eine andere Position gebracht (siehe Abb. 6.15).

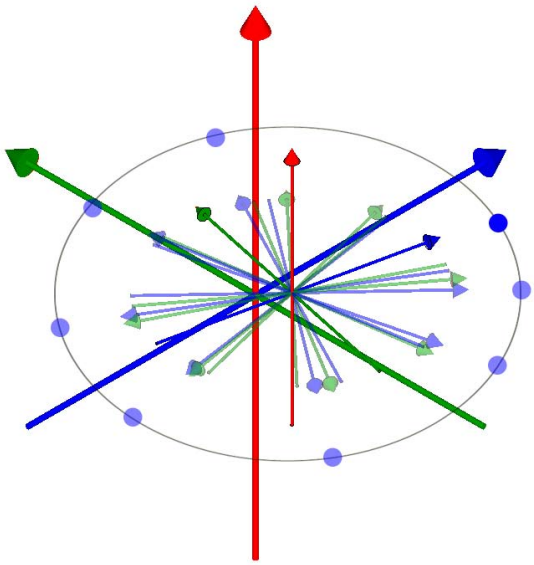


Abbildung 6.15.: Bewegen einer WFS-Quelle auf einer Kreisbahn um den Akteur.

Für die Berechnung wird sich der Positionen $handPositionAtLock$, $sourcePositionAtLock$ und der aktuellen Position und Ausrichtung der Hand bedient. Als Grundlage für die Entfernung gilt immer die Entfernung zwischen $handPositionAtLock$ und $sourcePositionAtLock$, welche sich während der Ausführung der Geste nicht ändert. Bei jedem Zyklus des Systems, in der die Hand sich im LockedState befindet, wird geprüft, ob der aktuelle Winkel der Hand um mehr als die Toleranz vom $handAngleAtLock$ abweicht, wenn dies der Fall ist, wechselt die Hand in den MoveState. Bei jedem Zyklus des Systems, in der sich die Hand im MoveState befindet, wird die Distanz zwischen $handPositionAtLock$ und $sourcePositionAtLock$ berechnet. Danach wird sich des Positioners des MoWeCs bedient, um anhand der Distanz und der Ausrichtung der Hand neue X- und Y-Positionsdaten relativ zur Hand wie folgend zu errechnen:

Seien $X_r$ und $Y_r$ Positionsdaten relativ zur Hand, $\gamma$ der Rotationswinkel um die Gier-Achse der Hand und $dist$ die Entfernung zwischen $handPositionAtLock$ und $sourcePositionAtLock$, dann gilt:

$$\begin{aligned} X_r &= dist * \cos\ \gamma \\ Y_r &= dist * \sin\ \gamma \end{aligned} \tag{6.1}$$

Seien $X_H$ und $Y_H$ die Positionsdaten der Hand und $X$ und $Y$ die neuen Ergebnispositionsdaten der WFS-Quelle der Geste, so gilt:

$$\begin{aligned} X &= X_H + X_r \\ Y &= Y_H + Y_r \end{aligned} \tag{6.2}$$

Somit bewegt sich die WFS-Quelle nicht wirklich auf einer Kreisbahn, denn dadurch, dass die Distanz zur Hand gewahrt bleibt, wirken sich Positionsänderungen der Hand zusätzlich in gleichem Maße auf die Position der WFS-Quelle aus. Somit ist es auch möglich die WFS-Quelle zum Beispiel seitlich zu verschieben. Jedoch lediglich um eine Distanz, welche auch durch die Hand zurückgelegt werden kann.

Nachdem die Source der WFS-Quelle mit den neuen Positionsdaten versehen wurde, ist die Geste beendet.

#### 6.6.4.2. Distanz

Die PushPullGesture ermöglicht es, die Distanz einer WFS-Quelle zur eigenen Position zu beeinflussen. Dabei soll eine Bewegung der Hand näher zum eigenen Körper die WFS-Quelle kontinuierlich näher bringen (siehe Abb. 6.16), bis die Hand wieder in die Ausgangsposition gebracht wird (siehe Abb. 6.18a) oder die WFS-Quelle abgewählt wird. Eine Bewegung der Hand vom Körper weg soll die WFS-Quelle eben so kontinuierlich immer weiter vom eigenen Körper entfernen.

Da der eigene Körper jedoch kein Target trägt und auch nicht extra eins bekommen soll, seine Position dem System dadurch nicht bekannt ist und daher nicht als Referenzpunkt genommen werden kann, wird die Distanzveränderung der Hand zum Körper anhand der Distanzveränderung der Hand zur Source gemessen. Da sich die Distanz der WFS-Quelle jedoch durch die Ausführung der PushPullGesture verändert, würde in dem nächsten Zyklus des Systems diese durch die Geste hervorgerufene Distanzänderung wiederum bei der Berechnung der neuen Distanz berücksichtigt und damit dem Effekt der Geste entgegen wirken. Aus diesem Grund wird beim Auswählen einer WFS-Quelle durch die LockGesture im Locked-State direkt der zu dem Zeitpunkt aktuelle Zustand der Source hinterlegt. Anstatt bei dieser Berechnung nun immer wieder die aktuelle Position der Source zu benutzen, wird, so lange die PushPullGesture aktiv ist, immer diese ursprüngliche Position ($sourcePositionAtLock$) zur Berechnung genommen (siehe Abb. 6.16c).

Seien $x_h, y_h$ Positionsdaten der Hand und $x_s, y_s$ Positionsdaten der $sourcePositionAtLock$. Dann gilt:

$$DistanzNeu = \sqrt{(x_s - x_h)^2 + (y_s - y_h)^2} \tag{6.3}$$

Diese Distanz muss mit der Distanz verglichen werden, welche beim Auswählen der WFS-Quelle zwischen der Source und der Hand bestand.

Seien $x_h$, $y_h$ Positionsdaten der $handPositionAtLock$ und $x_s$, $y_s$ Positionsdaten der $sourcePositionAtLock$. Dann gilt:

$$DistanzAlt = \sqrt{(x_s - x_h)^2 + (y_s - y_h)^2} \tag{6.4}$$

Die Differenz der beiden Längen errechnet sich ganz einfach durch

$$DistanzDiff = DistanzAlt - DistanzNeu \tag{6.5}$$

Enthalten in $DistanzDiff$ ist auch schon die Information, ob die WFS-Quelle gedrückt oder gezogen wird, denn ist die Differenz negativ, so bedeutet dies, dass die Hand weiter von der WFS-Quelle entfernt und somit dem Körper näher gebracht wurde. Das soll ebenfalls ein Näherbringen der WFS-Quelle zur Folge haben, was durch eine Subtraktion von der Distanz $DistanzNeu$ erreicht wird.

Damit nicht jede minimale und daher möglicherweise ungewollte Bewegung eine Relokalisierung der WFS-Quelle zur Folge hat, wird auch in dieser Geste eine Toleranz berücksichtigt. Die Toleranz ($T$) ist eine Länge (aktuell 5 cm), um welche die Hand bewegt werden kann, ohne durch die Geste einen Effekt hervorzurufen. Hierzu wird $T$ von $DistanzDiff$ subtrahiert, sofern $DistanzDiff$ positiv ist, ansonsten wird $T$ addiert:

$$DistanzAbzglT = \begin{cases} DistanzDiff - T \; für \; DistanzDiff >= 0 \\ DistanzDiff + T \; für \; DistanzDiff < 0 \end{cases} \tag{6.6}$$

Ist das Ergebnis kleiner oder gleich null, so wird kein Effekt hervorgerufen. Ansonsten dient $DistanzAbzglT$ als weitere Berechnungsgrundlage.

Sollte in vorherigen Zyklen $i_x, ..., i_{t-1}$ des MoWeCs, bei der Ausführung der GestureComponent, bereits die Toleranz überschritten und damit in den PushPullState gewechselt und die WFS-Quelle somit bewegt worden sein, so ist dies auch als Ende der Geste zu interpretieren, da dies bedeutet, dass die Hand wieder in ihre Ausgangsposition zurückgekehrt ist und die WFS-Quelle somit wieder zum Stillstand kommt. In diesem Fall wird wieder in den LockedState gewechselt.

Im nächsten Schritt wird die neue Distanz ermittelt, in welcher die WFS-Quelle neu platziert werden soll. Hierzu wird $DistanzAbzglT$ mit einem frei wählbaren Faktor multipliziert, welcher den Effekt verstärkt oder verringert. Je höher der Faktor, desto stärker ist der Effekt. Eine weitere Gegebenheit, welche mit der Anbindung zum Tracker zu tun hat, muss jedoch noch berücksichtigt werden, denn die Datenpakete, welche dieses System vom Tracker bekommt, werden mittels unzuverlässiger UDP-Pakete übermittelt, welche nicht gewährleisten, dass alle Pakete ankommen und somit auch nicht, dass sie in gleichmäßigen Abständen ankommen. Da jedes empfange Paket unverzüglich einen Zyklus des Systems und somit ggf. auch diese Berechnung auslöst, würde der Effekt durch die variierenden Intervalle zwischen den Paketen beeinflusst werden. Je mehr Pakete empfangen würden, desto stärker wäre der Effekt und umgekehrt eben so. Um diesen Einfluss zu eliminieren, wird anhand des Zeitstempels des States (welcher bei dem vorherigen Zyklus gesetzt wurde) und der aktuellen Zeit die zeitliche Differenz $t$ zum letzten Zyklus berechnet. Mit dieser wird das zuvorige Ergebnis erneut multipliziert. Somit wird der Effekt, der durch die unterschiedlichen Intervalle erzeugt wird, ausgeglichen, indem bei zum Beispiel doppelter verstrichener Zeit ebenfalls ein doppelt so starker Effekt hervorgerufen wird.

$$DistanzFaktorisiert = DistanzAbzglT * Faktor * t \tag{6.7}$$

Diese Distanz wird von der Distanz zwischen $handPositionAtLock$ und der Source subtrahiert, um die neue Distanz ($DistanzDerPlatzierung$) zu erhalten, in welcher die WFS-Quelle zur $handPositionAtLock$ platziert werden soll.

Um die Position der WFS-Quelle lediglich in der Entfernung zur Hand zu verändern, jedoch nicht in der Richtung, durch versehentliches und in der Realität schwer zu vermeidender Rotation der Hand um die Gier-Achse, wird der Winkel ermittelt, in dem die Source sich beim Auswählen der WFS-Quelle zur Hand befunden hatte. Seien $x_h$ und $y_h$ Positionsdaten der Hand im aktuellen Zustand und $x_n$ und $y_n$ die Positionsdaten der neuen Position der Source, so gilt:

$$\begin{aligned} x_n &= DistanzDerPlatzierung * \cos\ handAngleAtLock + x_h \\ y_n &= DistanzDerPlatzierung * \sin\ handAngleAtLock + y_h \end{aligned} \tag{6.8}$$

Nun wird die Position der Source mit diesen Koordinaten aktualisiert und die Geste erfüllt ihren Effekt.

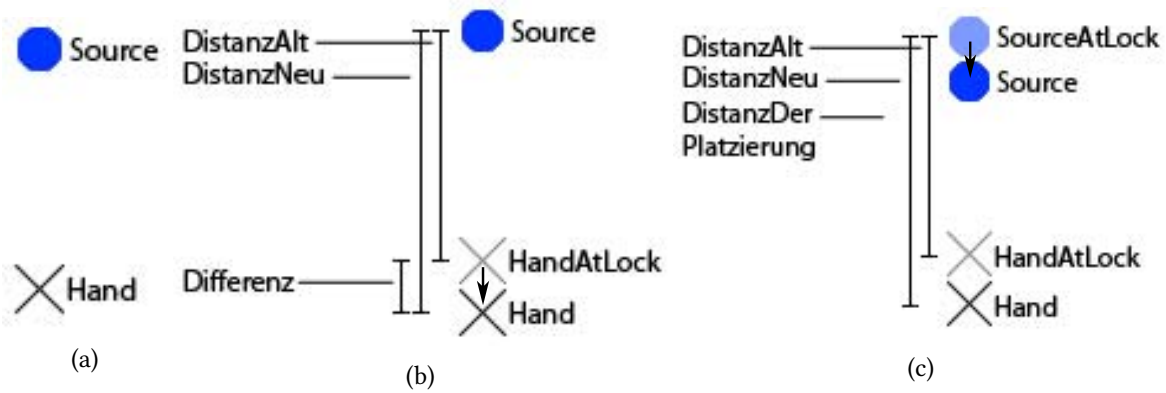



Abbildung 6.16.: Distanz einer WFS-Quelle in drei Schritten ändern.
(a): Die Hand hat eine Source ausgewählt. Die Positionen der Hand und der Source werden im State vermerkt.
(b): Die Hand wird näher zum Körper gezogen und entfernt sich damit von der Source
(c): Durch den Unterschied in den Distanzen der Hand und Source beim Auswählen und aktuell wird eine neue Position für die Source berechnet

#### 6.6.4.3. Konflikte

Eigentlich besteht zwischen den beiden Gesten MoveGesture und PushPullGesture ein Konflikt, welcher das parallele Ausführen beider Gesten verhindert. Nur die Einführung der beiden States MoveState und PushPullState ermöglichen es, beide Gesten ausführen zu können.

Die PushPullGesture reagiert einzig auf die Entfernung der Hand zur Source. Genauer gesagt, auf den Unterschied in der Entfernung zwischen $handPositionAtLock$ und $sourcePositionAtLock$ sowie zwischen der Position der Hand im aktuellen Zustand und $sourcePositionAtLock$. Die Relation zwischen all diesen Positionen ist also relevant. Wird die Hand jedoch auf einer Kreisbahn um den Körper bewegt, um die MoveGesture auszuführen, so bringt dies auch eine starke Veränderung dieser Distanzen mit sich, was ein ungewolltes Auslösen der PushPullGesture zur Folge hat. Dieses Problem könnte am besten gelöst werden, wenn beim Ausführen der MoveGesture ebenfalls neue Positionen für alle für die PushPullGesture relevanten Positionen gefunden würden, so, dass die Relationen erhalten bleiben. Das ungelöste Problem hierbei ist das Finden eines zuverlässigen Referenzpunktes, zu dem die Positionen neu bestimmt werden. Nähme man die Hand im aktuellen Zustand mit samt ihrer Ausrichtung, was naheliegend erscheint, und platziere alle Punkte in der neuen Richtung mit den zuvorigen Entfernungen, so würde dies wiederum die Ausführung der PushPullGesture unmöglich machen, da diese genau auf den Unterschieden dieser Entfernungen basiert, welche so aufgehoben werden würden.

Aus diesem Grund wurden die beiden States für die Gesten eingeführt, welche die Ausführungen der beiden Gesten trennen. Sobald eine der beiden Gesten erkannt wurde, wird in den jeweiligen State gewechselt und die jeweils andere Geste wird nicht mehr beachtet. Dies wirft jedoch noch zwei zu klärende Punkte auf. 1.: Wann soll in den einen oder den anderen State gewechselt werden? und vor allem 2.: Wann soll dieser State wieder verlassen werden? Gewechselt wird in einen der States, wenn die jeweilige Geste zuerst erkannt wird. Dies ist jedoch stark von den jeweiligen Toleranzen abhängig. Werden diese zu gering gewählt, so ist es leicht möglich versehentlich die nicht beabsichtigte Geste zuerst zu aktivieren und somit keinen Effekt mehr von der Geste zu erhalten, welche eigentlich ausgeführt werden sollte und deren Effekt der Erwartung entspricht.

So passiert es zum Beispiel, in der Praxis leicht, dass beim Ausführen eines »pulls«, also bei dem Versuch, die PushPullGesture so auszuführen, dass die WFS-Quelle näher kommt, die Hand aus Versehen durch das Beugen des Armes ebenfalls leicht auf der Gier-Achse gedreht wird und somit fälschlicher Weise die MoveGesture zuerst erkannt wird. Würden die Toleranzen sehr hoch gewählt, so wird dieser Fall immer unwahrscheinlicher, jedoch bestimmt die Höhe der Toleranz der MoveGesture die minimale Erstbewegung, welche auf die WFS-Quelle ausgeübt werden kann, da die Bewegung durch die Toleranz gar nicht berücksichtigt wird. So hat zum Beispiel eine Toleranz von 10° zur Folge, dass sich eine WFS-Quelle zu Beginn der Geste um keinen Winkel, der geringer als diese 10° ist, bewegt werden kann. Sobald die Geste erkannt und in den MoveState gewechselt wurde, lässt sich jedoch jeder beliebige Winkel einstellen, auch einer, der näher an der ursprünglichen Ausrichtung liegt, als die Toleranz.

Die PushPullGesture wird anstatt der MoveGesture nicht so leicht versehentlich ausgeführt. Wenn es jedoch passiert, hat es einen noch überraschenderen und schwerer zu korrigierenden Effekt. Die WFS-Quelle nähert sich unverhofft, oft schnell, da bei der MoveGesture oft größere Positionsänderungen der Hand ausgeführt werden, und in der Distanz kann die WFS-Quelle (über die Akustik) schwieriger wieder an die ursprüngliche Position bewegt werden.

Um die States wieder zu verlassen, muss die Intention des Akteurs hierfür erkannt werden. Dies ist bei der PushPullGesture relativ einfach, denn da die WFS-Quelle sich während der Ausführung der Geste stetig bewegt, ist ein eindeutiges Handeln des Akteurs notwendig, um die WFS-Quelle an dem Zielort stoppen zu lassen. Dies geschieht entweder durch Abwählen der WFS-Quelle oder indem die Hand wieder in die Ausgangsposition gebracht wird, bzw. den Bereich um die Ausgangsposition, welcher durch die Toleranz gebildet wird (siehe Abb. 6.18b). In diesem Fall kann die PushPullGesture sofort als beendet erkannt und die Hand in den LockedState versetzt werden. Dabei wird der LockedState jedoch mit der vorherigen $handPositionAtLock$ erstellt, anstatt die aktuelle Position der Hand zu nehmen, da sich die

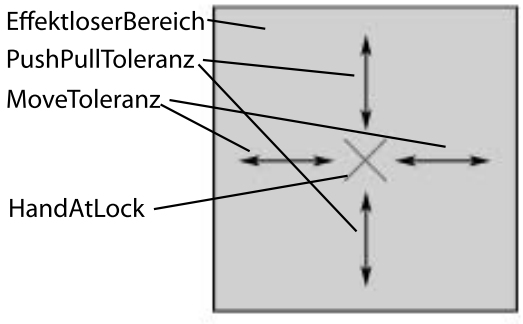
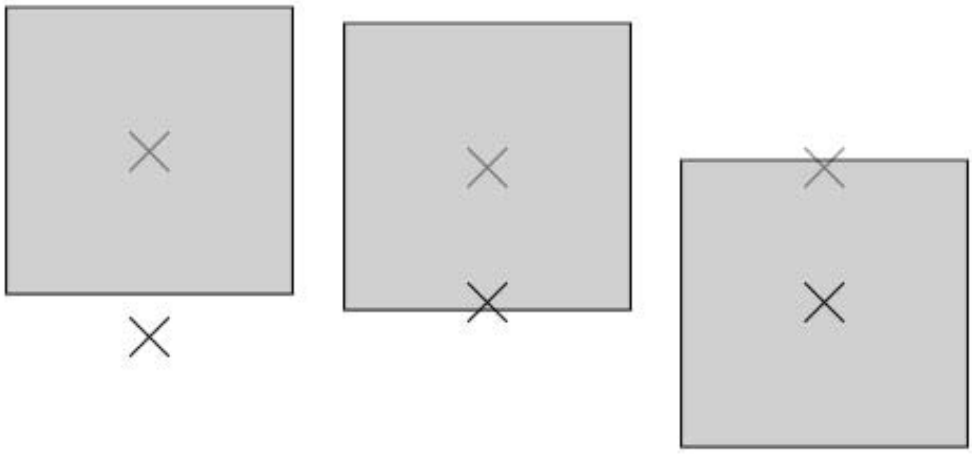



Abbildung 6.17.: Die Position der Hand beim Auswählen mit visualisiertem Toleranzbereich.

Hand sonst direkt wieder im Mittelpunkt der Toleranzen befinden würde, wie in Abbildung 6.18c dargestellt ist. Diese Aktion wird jedoch ausgeführt, sobald sich die Hand wieder in dem Bereich, den die Toleranzen aufspannen, befindet. Somit befindet sich die Hand gerade an einem der Ränder der Toleranzen, wie in Abbildung 6.18b richtig umgesetzt wurde. Würde die neue Position genommen werden, würde der Bereich der Toleranzen in diesem Moment einen »Sprung« machen und die Hand würde sich entgegen der Erwartung nicht mehr am Rand der Toleranzen befinden sondern in der Mitte.

Bei der MoveGesture ist das Erkennen dieser Intention nicht ganz so einfach. Da die WFS-Quelle sich nur bewegt, solange sich auch die Hand in Bewegung befindet, könnte jeder Stillstand der Hand direkt als Vollendung der Geste interpretiert werden. Da es jedoch unmöglich ist, eine Hand freischwebend wirklich exakt stillzuhalten, muss hier ebenfalls mit einer Toleranz gearbeitet werden und da eine Bewegung oder, wie in diesem Fall, ein Nichtvorhandensein einer Bewegung, nur in einem zeitlichen Rahmen gemessen werden kann, muss auch dieser festgelegt werden. Anders ausgedrückt bedeutet das, dass der Stillstand der Hand eigentlich durch lediglich geringe Bewegungen (unterhalb der Toleranz) innerhalb eines bestimmten Zeitrahmens definiert werden muss. Der Zeitrahmen und die Toleranz sind dabei wieder frei wählbar, haben jedoch beide eine starke Auswirkung und ein sorgfältiges Abwägen ist angebracht. Hierfür werden alle Positionen und die dazugehörigen Ausrichtungen der Hand innerhalb des Zeitrahmens, dessen letzter Eintrag der aktuelle ist, gespeichert. Aus diesen Positionen und Ausrichtungen werden die Mittelwerte gebildet und wenn alle Mittelwerte innerhalb der Toleranzen zu den aktuellen Werten liegen, so wird dies als »Stillstand« der Hand interpretiert.

Besonders wichtig ist hier, wie schon erwähnt, den Zeitrahmen und die Toleranz sorgfältig zu wählen. Wird die Toleranz zu niedrig gewählt (etwa < 1 mm), so wird es für den Akteur zu schwierig still genug zu halten, um die Geste zu beenden. Wird hingegen die Toleranz zu



(a) Die PushPullGesture wird ausgeführt, da die Hand sich außerhalb des Toleranzbereiches befindet.

(b) Beenden der PushPullGesture mit richtigem Setzten der $handPositionAtLock$.

(c) Beenden der PushPullGesture mit falschem Setzten der $handPositionAtLock$.

Abbildung 6.18.: Der Toleranzbereich beim Beenden der PushPullGesture.

großzügig gewählt, würden schon geringe Bewegungen, welche eigentlich noch zur Positionierung der WFS-Quelle dienen, schon als Stillstand der Hand interpretiert, die Geste damit als beendet betrachtet und damit feine Positionierungen der WFS-Quelle unmöglich gemacht. Ein zu geringer Zeitrahmen hat ebenfalls zur Folge, dass die Hand zu schnell als stillstehend erkannt wird. Ein zu hoher Zeitrahmen jedoch, dass die Akteure bereits mit der nächsten Geste fortfahren wollen, dazu die Hände wieder bewegen und es für das System gar nicht zum Stillstand der Hände kommt. Bei der Wahl beider Toleranzen muss zusätzlich gleichermaßen noch berücksichtigt werden, dass beim Vollenden der Geste und dem damit verbundenen Zurückkehren der Hand in den LockedState die Referenzpositionen $sourcePositionAtLock$ und $handPositionAtLock$ neu gesetzt werden. Die $handPositionAtLock$ ist Grundlage und Mittelpunkt für die Toleranzen. Sobald diese neu gesetzt werden, wird also ein Bereich um die Position der Hand gebildet, in welchem Bewegungen keinen Effekt haben, da sie durch die Toleranzen unterdrückt werden. Würde während der Ausführung der MoveGesture regelmäßig in den LockedState gewechselt werden, würde dies damit ein Springen der WFS-Quelle zur Folge haben, da immer wieder die Toleranz überwunden werden muss, bis es wieder zu einer Verschiebung der WFS-Quelle kommt.

## 6.7. Bewertung der Umsetzung

In diesem Abschnitt, wird darauf eingegangen, wie gut die Umsetzung im praktischen Gebrauch funktioniert. Es wird auf die Intuitivität eingegangen und darauf, wie gut sich welche Gesten bei bekannter Funktionsweise ausführen lassen. Die Bewertung ist wieder in die einzelnen Gesten unterteilt.

### 6.7.1. Auswählen

Das Auswählen einer WFS-Quelle funktioniert schon bei Probanden, welche die Funktionsweise lediglich ein Mal gesehen haben, ziemlich gut. Ohne jegliche Einführung gelingt die Geste nur etwa jedes zweite Mal. Auch Probanden, denen die Geste zwar schon mal gelang, über die Funktionsweise jedoch nicht genau Bescheid wissen, gelingt die Ausführung nicht sicher. Anwendern, denen bekannt gemacht wurde, worauf es bei der Geste exakt ankommt, haben jedoch eine sehr hohe Erfolgsquote. Die meisten Fehler entstehen dadurch, dass entweder nicht lange genug gewartet wird, bis die Hand in den LockedState gewechselt hat oder, dass zum Auswählen so ausladend gestikuliert wird, dass die WFS-Quelle in der Zwischenzeit schon ausgewählt wurde, die Hand aber immer noch irgendwie bewegt wird und somit direkt eine Folgegesten ausgeführt wird.

### 6.7.2. Bewegen (kreisförmig)

Die Ausführung der MoveGesture, nachdem eine WFS-Quelle ausgewählt wurde, hat eine hundertprozentige Erfolgsquote. Hier gibt es keinerlei Probleme zu berichten. Auch das Wechseln des States der Hand in den MoveState und teilweise auch durch Stillstand der Hand zwischenzeitlich zurück in den LockedState, geschieht ohne Kenntnisnahme und somit auch ohne Irritation der Anwender. Die einzigen Probleme bei der Ausführung dieser Geste ist auf ein zuvor nicht erfolgreiches Auswählen der WFS-Quelle zurückzuführen.

### 6.7.3. Bewegen (Distanz)

Das Ausführen der PushPullGesture birgt einige Probleme. Probanden, welche keinerlei Einführung bekommen haben, gelingt es zwar in vielen Fällen, eine WFS-Quelle zu sich heranzuziehen, jedoch dauert die Realisierung dessen, was die Geste dann genau bewirkt (das kontinuierliche Annähern der WFS-Quelle), meist so lange, dass die WFS-Quelle schon bis direkt an die Hand heran geholt wurde, die Lokalisierung der WFS-Quelle, durch die sich nun

innerhalb des Raumes befindende Position, oft nicht mehr ohne weiteres möglich ist und die Probanden mit der Situation überfordert sind.

Hier bedarf es dann einer ausführlicheren Einführung, bis die Geste sicher und mit dem erwarteten Effekt ausgeführt wird. Insbesondere besteht ein Problem, wenn die WFS-Quelle bis zum Anschlag (also bis zur Position der Hand) an den Körper heran geholt wurde. Denn wird die WFS-Quelle in diesem Zustand abgewählt, was oft auch gerade bei überraschten Probanden der Fall ist, befindet sich die Position der WFS-Quelle direkt vor dem Probanden, da die Hand zuvor direkt vor den Körper gezogen wurde. Wird nun versucht, die WFS-Quelle, mit gestrecktem Arm, erneut auszuwählen, so befindet sich die WFS-Quelle zwischen dem Körper und der Hand. Somit kann die WFS-Quelle nicht ausgewählt werden. Nun ist es das einfachste, selber eine neue Position einzunehmen, um die WFS-Quelle wieder wie gehabt auswählen zu können, jedoch ist die Lokalisierung der WFS-Quelle, nun innerhalb des Raumes, so unzuverlässig, dass meist die visuelle Darstellung von xWonder zur Hilfe genommen werden muss, um die Hand wieder zum Auswählen auf die WFS-Quelle ausgerichtet zu bekommen. Zudem ist es schwierig eine WFS-Quelle anhand der akustischen Lokalisierung in oder in der unmittelbaren Nähe des Raumes präzise zu Positionieren, da die Bewegung der WFS-Quelle oft zu schnell ist, um rechtzeitig wahrgenommen zu werden. Diese Probleme haben auch Anwender, welche mit der Funktionsweise der Geste bekannt sind.

Das Bewegen einer WFS-Quelle in die entgegengesetzte Richtung birgt ähnliche Probleme. Zumeist ist der Beginn schwieriger, da, durch die LockGesture der Arm meist bereits gestreckt ist und nun verschiedene Möglichkeiten ausprobiert werden, um der WFS-Quelle dennoch irgendwie einen Impuls in die entgegengesetzte Richtung zu geben. Oft ist dies mit einem vorherigen Heranziehen der Hand verbunden, was nun allenfalls ein kurzzeitiges Annähern der WFS-Quelle zur Folge hat und durch das anschließende »wieder Ausstrecken« des Armes, lediglich diesen ungewollten Effekt wieder stoppt. Distanzveränderungen in einer bereits großen Entfernung, haben zudem eine nur sehr geringe Veränderung der Akustik zur Folge und sind somit, anhand dieser, schwer präzise bestimmbar. Das höchste Ausmaß dieses Umstandes wird erreicht, wenn die Geste auf eine WFS-Quelle in hoher Entfernung zwar korrekt ausgeführt, der Effekt jedoch nicht unmittelbar wahrgenommen und somit an der korrekten Ausführung der Geste gezweifelt wird. Dies gilt selbstverständlich für die PushPullGesture ungeachtet dessen, ob eine weit entfernte WFS-Quelle nun näher heran oder noch weiter entfernt werden soll.

Zudem kommt es in beiden Fällen, vor allem jedoch beim »pull«, oft noch dazu, dass durch eine versehentliche Rotation der Hand um die Gier-Achse, fälschlicher Weise in den MoveState gewechselt und die PushPullGesture somit vorübergehend nicht ausführbar gemacht wird.

### 6.7.4. Abwählen

Die Ausführung der UnlockGesture funktioniert ebenso, wie die Ausführung der MoveGesture, äußerst zuverlässig. Sie wird immer erkannt und es werden nicht fälschlicherweise andere Gesten ausgeführt. Das einzige Problem, was hin und wieder aufgetreten ist, ist dass gelegentlich, durch Unachtsamkeit, doch aus Versehen, vorzeitig ausgeführt wird und somit Ratlosigkeit über den Verbleib des im Anschluss intendierten Effektes einer MoveGesture oder PushPullGesture entsteht.

# 7. Schluss

## 7.1. Zusammenfassung

Ziel dieser Arbeit war es, ein möglichst einfaches und intuitives Repertoire an Gesten zur Positionierung von Klangquellen einer WFS-Anlage zu erarbeiten und umzusetzen. Hierzu wurden zunächst die vorhandenen Systeme (ART-Tracker und WFS-Anlage) und anschließend ein eigens entwickeltes System beschrieben, das die vorhandenen Systeme koppelt und dabei wichtige Grundvoraussetzungen für die GestureComponent liefert.

Anschließend wurde ein kurzer Überblick über Gesten generell und deren Anwendung in anderen Systemen gegeben. Daraufhin wurde unter anderem durch Tests mit zehn verschiedenen Probanden erarbeitet, für welche konkreten Aktionen bei der Steuerung der WFS-Anlage Gesten geschaffen werden und wie diese Aussehen sollen. Danach wurden diese Ergebnisse auf ihre mathematischen Eigenschaften hin überprüft und somit die technische Umsetzung erarbeitet.

Zuletzt wurden die einzelnen Gesten der entwickelten und in Java umgesetzten Gestensteuerung auf ihre Anwendbarkeit, Intuitivität und Fehlertoleranz hin untersucht und sowohl die positiven als auch die negativen Aspekte erläutert.

## 7.2. Fazit

Diese Arbeit hat gezeigt, dass es möglich ist, beliebige Klangquellen einer WFS-Anlage mit einem Repertoire an Gesten beliebig zu positionieren. Hierbei stellt die dreidimensionale Umgebung besondere Anforderungen an die Auswahl des Repertoires an Gesten und der Festlegung, wie diese konkret aussehen sollen. An vielen Stellen birgt ein Abwägen zwischen Präzision und Fehlertoleranz Konflikte. Neben den Herausforderungen an die Gesten, stellt vor allem das präzise und schnelle Lokalisieren der WFS-Quellen rein durch ihre Akustik eine große Herausforderung dar. Wesentliche Probleme, welche im Laufe dieser Arbeit herausgearbeitet und beschrieben wurden und die auch zum Abschluss dieser Arbeit noch bestehen, sind auf Probleme der Lokalisierung zurückzuführen. Offen gelassen wird die Frage, ob so eine Steuerung in der Praxis gänzlich ohne visuelle Unterstützung praktikabel sein wird.

## 7.3. Ausblick

Ein paar Ideen, welche in der Entwicklung und der Auswertung der GestureComponent entstanden aber nicht mehr umgesetzt wurden, werden im Folgenden kurz angerissen:

- Der Informationsfluss von der WFS-Anlage zum MoWeC wurde bisher ignoriert. Die GestureComponent braucht jedoch Informationen über die WFS-Quellen, um mit ihnen interagieren zu können. Bisher geschieht dies auf Annahmen, die lediglich innerhalb des MoWeCs getroffen werden. Der MoWeC müsste von der WFS-Anlage unmittelbar über die Zustände und Änderungen aller WFS-Quellen informiert werden und diese Informationen berücksichtigen. Dies stellt vor allem Anforderungen an die Synchronisation, da nun von verschiedenen Quellen Informationen über WFS-Quellen eingehen, welche durch den möglicherweise zeitlichen Versatz widersprüchlich zueinander sein können.
- Ein Head-up-Display, welches auf dem Kopf getragen würde und eine transparente Projektionsfläche vor einem der Augen hätte, könnte die WFS-Quellen im $\mathbb{R}^3$ visualisieren. Der Nachteil wäre das zusätzliche, möglicherweise kabelgebundene, Gerät.
- Die PushPullGesture könnte so angepasst werden, dass die WFS-Quelle sich langsamer bewegt, je näher sie der Hand ist, um somit in diesem Bereich eine präzisere Positionierung zu erleichtern.
- Ein markerloses Trackingverfahren würde das Target auf der Hand unnötig machen und somit den Bedienungskomfort steigern.
- Um dem versehentlichen vorzeitigen Auswählens einer WFS-Quelle eines uninformierten Anwenders, welcher eine komplexere Bewegung zum Auswählen ausführen möchte entgegenzuwirken (siehe Kapitel 6.7.1), könnte die Bedingung zum erkennen der LockGesture die Bedingung erweitert werden, die Hand für einen gewissen Zeitraum stillzuhalten sowie es nötig ist, um vom MoveState in den LockedState zurück zu wechseln.
- Ein zusätzliches eindeutiges Feedback bei erfolgreicher Ausführung der Gesten, würde die Bedienung in einigen Fällen erleichtern. Denkbar wäre dies zum Beispiel durch das zuvor angesprochene Head-up-Display, einem akustischen Signal oder einem visuellen Signal an einer separaten Projektionsfläche. All diese Möglichkeiten haben ihre eigenen Vor- und Nachteile.
- Durch die präzise Interpretation des Nick-Winkels und der Höhe der Position der Hand beim Ausführen der LockGesture, ließe sich der Strahl, mit dem die Hand in eine Richtung deutet von der Ebene in die dritte Dimension erweitern und könnte es somit

ggf. ermöglichen, an WFS-Quellen zu kommen, welche auf der Ebene durch andere WFS-Quellen verdeckt wären.

- Eine sehr umfangreiche Erweiterung des Gestenrepertoire, vor allem auch der Funktionsweise des Systems, könnte es ermöglichen, direkten Einfluss auf die Sounds im herkömmlichen Sinne zu bekommen, die unabhängig von der Position sind, welche ihnen von der WFS-Anlage zugewiesen wird.

# Glossar

$A_S$ Die Ausrichtung der Hand zu dem Zeitpunkt, als ein State erstellt wurde. 47

$P_S$ Die Position der Hand zu dem Zeitpunkt, an dem ein State erstellt wurde. 47

$T_S$ Zeitpunkt, zu dem ein State erstellt wurde. 47

$handAngleAtLock$ Die Ausrichtung der Hand zu dem Zeitpunkt, an dem eine WFS-Quelle ausgewählt wurde. 47, 56

$handPositionAtLock$ Die Position der Hand zu dem Zeitpunkt, an dem eine WFS-Quelle ausgewählt wurde. 47, 56, 58–61, 63

$maxNickWinkel$ Der Nick-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle nicht überschritten werden darf. 52–54

$maxRollWinkel$ Der Roll-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle nicht überschritten werden darf. 52

$minNickWinkel$ Der Nick-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle nicht unterschritten werden darf. 52–54

$minRollWinkel$ Der Roll-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle nicht unterschritten werden darf. 52

$sourcePositionAtLock$ Die Position einer WFS-Quelle zu dem Zeitpunkt, an dem sie ausgewählt wurde. 47, 56–58, 60, 63

**OSC-Adapter** Transformiert Modifications des MoWeCs in OSC-Nachricht. 29, 33

**OSC-Nachricht** Open-Sound-Control-Nachricht. 9–12, 21, 22, 29, 30, 33, 36, 70

**OSC-Sender** Versendet OSC-Nachricht an einen definierten Empfänger. 29, 33

**MoWeC** Der Grundstock des in dieser Arbeit entwickelten und beschriebenen Systems, der dazu nötig ist, den Tracker mit der WFS-Anlage zu verbinden. Die GestureComponent ist nicht Teil des MoWeCs. 4, 20–24, 30–33, 43, 44, 56, 58, 68, 70–72, 75, 76

**A.R.T.-System** Tracker der Advanced Realtime Tracking GmbH. 16, 21, 22, 44, 52, 55, 70, 71

**Advanced Realtime Tracking GmbH** Entwickler des A.R.T.-Systems. 17, 70

**Akteur** Urheber einer Handlung. In dieser Arbeit meist eine Person, welche das System bedient. 3, 15, 18, 25, 37, 40, 41, 55, 56, 61–63, 73, 77

**ART-Tracker** Tracker der Firma Advanced Realtime Tracking GmbH. 4, 13, 15–17, 67, 76

**Auswahl-Nick-Winkel** Der Nick-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle notwendig ist. 52, 53

**Auswahl-Roll-Winkel** Der Roll-Winkel der Hand, welcher zum Auswählen einer WFS-Quelle notwendig ist. 52, 53

**Auswahlebene** Eine Ebene, welche durch zwei sich auf der Position der Hand kreuzenden Geraden in Zeigerichtung ($X_T$) gebildet werden, welche durch addieren und subtrahieren einer Toleranz gebildet werden. 53

**Converter** Die Bestandteile der ConverterComponent, welche Sources von einem Koordinatensystem in ein anderes transformieren. Konkrete Converter sind der Scaler, Shifter, Mirror, Rotator und der RotationMirror. 24, 25, 27, 28, 33, 71

**ConverterBox** Ein Bestandteile der ConverterComponent, welchem mehrere Converter hinzugefügt werden können, um eine Reihe verschiedener transformationen auf gegebene Sources anzuwenden. 28, 33, 76

**ConverterComponent** Die Komponente des MoWeCs, welche Sources von einem Koordinatensystem in ein anderes überträgt. 71

**Coordinator** Die Komponente des MoWeCs, welche für die Koordination der einzelnen Komponenten zuständig ist. 23, 24, 30, 33, 45

**cwonder** Kommunikationsmodul von sWONDER. 9–11, 77, 92

**Dekameter** Zehn Meter. 40

**Drehsinn** Der Drehsinn eines Koordinatensystem gibt an, .... 25–27, 45, 76

**dtrack** Die Software zu dem A.R.T.-System. 16

**Ebene** zweidimensionaler Raum. 2, 26, 36–39, 54, 55, 68, 69, 84

**Einheitsvektor** Vektor mit der Länge 1 [BSMM05, S. 186]. 82, 83, 85

**Elementarwelle** Setzt sich eine Schallwelle aus mehreren einzelnen Schallwellen zusammen, so sind die einzelnen Schallwellen die Elementarwellen der Schallwelle. Die WFS synthetisiert Schallwellen aus Elementarwellen. 7, 8, 71

**FreeState** State der Hand, welcher symbolisiert, dass keine WFS-Quelle ausgewählt wurde. 46, 47, 53, 54

**Geste** Bewegung, die der nonverbalen Kommunikation dienen. 1–5, 15, 22, 34–43, 45–47, 49, 52, 53, 55–68, 72–76, 87

**Gestensteuerung** Die Möglichkeit, ein System allein mit den Bewegungen (vor allem der Arme und Hände) bedienen zu können. 1, 3, 5, 20, 67

**GestureComponent** Der Teil des Systems, welcher die Gesten umsetzt. 5, 21, 24, 33, 43–45, 53, 58, 67, 68, 70, 76

**Gier-Achse** Die Achse eines Objektes, welche vertikal verläuft. 45, 51, 56, 59, 61, 65, 72, 76

**Gier-Winkel** Der Winkel um die Gier-Achse. 45, 50, 72

**gieren** Die Änderung des Gier-Winkels. 45, 46, 50

**Head-up-Display** Gerät, welches Informationen im Blickfeld des Anwenders einblendet.. 68

**Identifikation** Merkmal eines Objektes (oft eine Zahl), welche unter allen vergleichbaren Objekten einzigartig ist. 7

**JavaOSC** Java Bibliothek von Illposed Software, welche das OSC-Protokoll implementiert. 29, 30, 33

**Jitter** Bezeichnet ein »zittern« der Daten. Dabei ändern sich aufeinanderfolgende Daten jeweils nur zu einem sehr geringen Maße, so in verschiedene Richtungen, dass sich die einzelnen kleinen Veränderungen gegenseitig in kurzem Zeitraum wieder aufheben und sich nicht zusammen in eine Richtung bewegen. 29, 33

**Koordinatensystem** Ein Koordinatensystem dient zur eindeutigen Bezeichnung der Position von Punkten und Objekten in einem geometrischen Raum. 16, 21, 25–28, 30, 33, 44–47, 49–51, 55, 71, 72, 74, 76, 82–85

**Listener** Komponente des MoWeCs, welche die Pakete des Trackerlisteners annimmt und somit die aktuellen Informationen der Targets in das System bringt. 23, 30

**LockedState** State der Hand, welcher symbolisiert, dass eine WFS-Quelle ausgewählt wurde. 46, 53, 56–58, 61, 63, 64, 68

**LockGesture** Geste, welche dem auswählen oder selektieren einer WFS-Quelle dient. 41, 46, 52–54, 57, 65, 68

**LockingState** State der Hand, welcher symbolisiert, dass gerade eine WFS-Quelle ausgewählt wird, die benötigte Zeit aber noch nicht erreicht wurde. 46, 53

**Marker** Repräsentanten der Targets innerhalb des Systems. 23–25, 33, 72

**Match** Zu deutsch »Paarung«. Stellt eine Verbindung von einem Marker zu einer Source her. 25, 33

**Matcher** Weist Markern zu Sources zu. 25, 30, 33

**Mirror** Teil des Systems, welches Sources über die Achsen ihres Koordinatensystem spiegelt. 27, 28, 71

**Modification** Ausgaben einer ModificationFactory. Jede Modification steht für eine Änderung einer Source, welche an die WFS-Anlage übertragen werden soll. 29, 33, 70, 73

**ModificationFactory** Extrahiert aus Sources die Änderungen, welche an ihnen vorgenommen wurden. 28, 29, 33, 73

**MoveGesture** Geste, welche dem Bewegen einer WFS-Quelle auf einer Kreisbahn um den Akteur dient. 46, 55, 60–64, 66

**MoveState** State der Hand, welcher symbolisiert, dass gerade eine WFS-Quelle bewegt wird. 46, 56, 60, 61, 64, 65, 68

**Multicast** Eine Nachrichtenübertragung von einem Punkt zu einer Gruppe. Empfänger der Nachrichten sind alle und ausschließlich Mitglieder der Gruppe. 16, 20, 21, 73

**Multicastgruppe** Eine Gruppe, deren Mitglieder Computer sind, welche die Empfänger eines Multicasts sind. 22

**Nick-Achse** Die Achse eines Objektes, horizontal quer zum Objekt verläuft. 45, 51, 52, 73, 76

**Nick-Winkel** Der Winkel um die Nick-Achse. 45, 50, 52–54, 68, 70, 71, 73

**nicken** Die Änderung des Nick-Winkels. 45, 46, 49, 50, 52, 76

**Obergriff** Handgelenkstellung, bei der die Handfläche nach unten und die Finger vom Körper weg zeigen. 89

**Open-Sound-Control** Ein Protokoll für die Kommunikation zwischen Computern, Synthesizern und anderen Multimediageräten über ein Netzwerk. 9

**orthogonal** Orthogonalität; Rechtwinklig; Zwei Vektoren sind orthogonal zueinander, wenn ihr Skalarprodukt null ist. [HSZ03, S. 882]. 84

**Positioner** Positioniert Sources in bestimmten Relationen zu anderen Sources. 24, 25, 56

**Powerwall** Beschreibt im wesentlichen eine besonders große und trotzdem sehr hochauflösende Anzeige, welche meist aus mehreren kleineren Monitoren besteht. 18

**PushPullGesture** Geste, welche dem Bewegen einer WFS-Quelle auf einem Strahl dient, der von dem Akteur in Zeigerichtung ausgeht. 46, 55, 57, 60, 61, 63–66, 68, 77

**PushPullState** State der Hand, welcher symbolisiert, dass gerade eine WFS-Quelle herangezogen oder weggeschoben wird. 46, 58, 60

**RelativePosition** Definiert eine ortsbezogene Relativität zu einer Position. 25, 74

**Richtungsvektor** Ein Vektor, welcher lediglich eine Ausrichtung beschreibt. 77, 82–84

**Roll-Achse** Die Achse eines Objektes, welche längs zum Objekt verläuft. 45, 50, 53, 73, 76

**Roll-Winkel** Der Winkel um die Roll-Achse. 45, 52–54, 70, 71, 73

**rollen** Die Änderung des Roll-Winkels. 45, 46, 49, 52

**RotationMirror** Teil des Systems, welches die Drehrichtung einer Sources umkehrt. 27, 28, 71

**Rotationsmatrix** Eine Rotationsmatrix ist in der Mathematik eine Matrix, die eine Drehung im euklidischen Raum beschreibt. 16, 24, 30, 44, 50–52, 54, 75, 76, 82–85

**Rotator** Teil des Systems, welches Sources zu ihrem Koordinatensystem um den Nullpunkt rotiert. 27, 28, 71

**Scaler** Teil des Systems, welches Sources zu ihrem Koordinatensystem neu skaliert. 27, 28, 71

**Shifter** Teil des Systems, welches Sources zu ihrem Koordinatensystem verschiebt. 27, 28, 71

**Soundquelle** Eine Position im realen Raum, von der sich ringförmig Schallwellen ausbreiten. 7, 20, 36

**Source** Zu deutsch: Quelle. Die Repräsentation einer virtuellen Klangquelle innerhalb des Systems. Nicht zu verwechseln mit WFS-Quelle. 23–25, 27–29, 33, 44–47, 53–55, 57–60, 71–74, 76, 77

**SourceHistory** Kann Einträge für alle Sources haben, in denen die Zustände der letzten Zyklen vermerkt sind, anhand derer Rückschlüsse auf die Veränderungen der Sources gezogen werden können. 24, 28, 33

**SourceSourceMatcher** Stellt die Verbindung zwischen zwei Sources her, um die zweite Source in einer Relation zur ersten Source zu platzieren, welche durch eine RelativePosition definiert wird. 25, 33

**State** Zustand der Hand. Ändert sich durch das Ausführen von Gesten und enthält unter anderem Informationen, welche vom jeweiligen Zustand abhängen. 46, 47, 59–61, 64, 70–73

**Strahl** Als Strahl oder Halbgerade wird eine Gerade bezeichnet, welche auf einer Seite durch einen Anfangspunkt begrenzt ist, sich auf der anderen Seite aber ins unendliche erstreckt. Die Orientierung des Strahlt geht von dem Anfangspunkt aus und erstreckt sich entlang des Strahls. [BSMM05, S. 131]. 37–39, 44, 53, 54, 68, 73

**SuperCollider** SuperCollider ist eine Programmiersprache und -umgebung für Echtzeit Audiosynthese und algorithmische Komposition. 3, 12

**sWONDER** Modulares Softwarepaket für die Bedienung und die Berechnungen von WFS-Anlagen. 9, 11, 71, 75

**System** Das in dieser Arbeit entwickelte und beschriebene System mit allen Komponenten. 15, 20, 22–24, 26, 29, 30, 37, 38, 43, 52, 53, 56, 57, 59, 63, 70, 72, 74, 76

**Target** Ein Konstrukt, dessen Position so wie die Ausrichtung von der A.R.T-Software erkannt wird. 15–17, 21–28, 30, 33, 41, 42, 44, 47, 49, 50, 57, 68, 72, 76, 82

**Tracker** Siehe Trackingsystem. 2–5, 13, 15, 18, 20–27, 30, 33, 55, 59, 70, 75

**Trackerlistener** Horcht und nimmt alle Pakete an, die der Tracker verschickt und schickt diese an gewünschte Empfänger weiter. 22–24, 30, 72, 76

**Trackingsystem** Ein System, welches die Position und ggf. Ausrichtung von Objekten im realen dreidimensionalen Raum erfasst. 2, 4, 13, 14, 75

**twonder** Modul für die Signalverarbeitung von sWONDER. 10

**UnlockGesture** Geste, welche dem abwählen einer ausgewählten WFS-Quelle dient. 46, 54, 66

**Utilities** Paket des MoWeCs, welches komplexere Datentypen mitsamt Rechen- und Vergleichsoperatoren zur Verfügung stellt. Hierzu gehören unter anderem Positionen, Ausrichtungen und Rotationsmatrizen. 30

**Values** Paket des MoWeCs, welches einfache Datentypen mitsamt Rechen- und Vergleichsoperatoren zur Verfügung stellt. Hierzu gehören unter anderem Längen und Winkel. 30

**WFS-Anlage** Wellenfeldsynthese-Anlage. 1–11, 18, 20, 21, 23, 25–27, 29, 30, 33, 35–37, 39, 55, 67–70, 73–76

**WFS-Linearquelle** Eine WFS-Quelle, die so eingestellt ist, dass sie eine gerichtete gerade Welle erzeugt. Wie eine Welle, die den Ozean durchkreuzt. 8, 36

**WFS-Mac** Ein Teil der WFS-Anlage. Bildet die Schnittstelle zwischen der WFS-Anlage und Anwendern so wie anderen Programmen. 9, 21, 22

**WFS-Node** Ein Teil der WFS-Anlage. Berechnet die Audiosignale für die untergeordneten Spuren. 9

**WFS-Punktquelle** Eine WFS-Quelle, die so eingestellt ist, dass sich die virtuelle Schallwelle von dieser Quelle aus ringförmig und gleichmaßig in alle Richtungen verbreitet. Wie die Welle, welche entsteht, wenn man einen Stein ins Wasser wirft. 8, 9, 36

**WFS-Quelle** Virtuelle Klangquelle der WFS-Anlage. 3, 4, 7–12, 21, 22, 24, 25, 27, 28, 35–37, 40–43, 47, 53, 55–65, 67–77

**WFS-Server** Ein Teil der WFS-Anlage. 9, 11, 12, 21, 22, 29, 33

**WFS-Setup** Beschreibt die Summe aller Eigenschaften aller WFS-Quellen. 11, 29

**xWonder** Graphische Benutzeroberfläche der WFS-Anlage. 3, 4, 9, 11, 35, 37, 65, 76

# Abbildungsverzeichnis

# Literaturverzeichnis


[ACBH00] Akyol, Suat ; Canzler, Ulrich ; Bengler, Klaus ; Hahn, Wolfgang: *Gestensteuerung für Fahrzeugbordsysteme*. http://www.akyol.de/mediapool/82/829906/data/V008_2000.pdf. Version: 2000

[Adv12a] Advanced Realtime Tracking GmbH: *Advanced Realtime Tracking*. http://www.ar-tracking.com/. Version: 2012

[Adv12b] Advanced Realtime Tracking GmbH: *Arttrack System*. http://www.ar-tracking.com/products/tracking-systems/arttrack-system/. Version: 2012

[And02] Andresen, Gavin: Playing by Ear: Creating Blind-Accessible Games-Tasks. In: *Gamesultra* (2002). http://www.conceptlab.com/uci/us12b/us12b-week9-andresen-playingbyear.pdf

[Baa04] Baalman, Marije: Updates of the WONDER software interface for using wave field synthesis International Computer Music Conference 2004, 2004

[Baa08] Baalman, Marije: *On Wave Field Synthesis and Electro-Acoustic Music*. 2008

[BHSK07] Baalman, Marije ; Hohn, Torben ; Schampijer, Simon ; Koch, Thilo: Renewed architecture of the sWONDER software for a wave field synthesis on large scale systems Linux Audio Conference 2007, 2007

[Boe08] Boetzer, Joachim: *Bewegungs- und gestenbasierte Applikationssteuerung auf Basis eines Motion Trackers*, HAW Hamburg, Bachelorarbeit, 2008

[BP05] Baalman, Marije ; Plewe, Daniel: WONDER - a software interface fpr the application of Wave Field Synthesis in electronic music and interactive sound installations 3rd International Linux Audio Conference, 2005

[BSMM05] Bronstein ; Semendjajew ; Musiol ; Mühling: *Taschenbuch der Mathematik*. Sechste Auflage. 2005

[CL09] Chojecki, Paul ; Leiner, Ulrich: Berührungslose Gestik-Interaktion im Operationssaal. In: *Mensch-Computer-Interaktion im Operationssaal* (2009), 13-18. http://www.oldenbourg-link.com/doi/abs/10.1524/icom.2009.0003?journalCode=icom

[Cle12] Clements, John: *Csound*. http://www.csounds.com/. Version: 2012

[DD11] Dorau, Rainer ; Dorau, Rainer: Prinzipien der Gestensteuerung. Version: 2011. http://dx.doi.org/10.1007/978-3-642-03101-4_3. In: *Emotionales Interaktionsdesign*. Springer Berlin Heidelberg, 2011 (X.media.press). – ISBN 978–3–642–03101–4, 102-133

[Dud90] Dudenredaktion, Wissenschaftlicher R.: *Duden Fremdwörterbuch*. Bd. 5. 5., neu bearbeitete und erweiterte Auflage. 1990. – ISBN 3–411–20915–1

[DW06] Delerue, Olivier ; Warusfel, Olivier: Mixage mobile. In: *IHM '06: Proceedings of the 18th International Conference of the Association Francophone d'Interaction Homme-Machine*. New York, NY, USA : ACM, 2006, S. 75–82

[EHH12] Eichmann, Hanna ; Hansen, Martje ; Heßmann, Jens: *Handbuch Deutsche Gebärdensprache*. Seedorf, 2012. – ISBN 978–3–936675–20–7

[FI12] Frauenhofer-Institut: *Digitales Schaufenster interpretiert Körpergesten*. http://www.hhi.fraunhofer.de/de/press/press-archive/press-releases-2011/the-digital-shop-window-that-can-read-body-signs/. Version: 2012

[Fou] Four Audio GmbH: *Kurze Einführung: Wellenfeldsynthese-Software und -Cluster*

[Fri12] Fricke, Ellen: *Wie Wörter und Gesten zusammenwirken*. 1. Auflage. 2012. – ISBN 978–3–11–021888–6

[FT] Floros, Andreas ; Tatlar, Nicolas-Alexander: *SPATIAL ENHANCEMENT FOR IMMERSIVE STEREO AUDIO APPLICATIONS*. http://avarts.ionio.gr/~floros/pubs/DSP2011%20Paper%20A.pdf

[GMMW07] Goertz, Anselm ; Makarski, Michael ; Moldrzyk, Chrustoph ; Weinzierl, Stefan: Entwicklung eines achtkanaligen Lautsprechermoduls für die Wellenfeldsynthese, 2007

[GNHF12] Glende, S. ; Nedopil, C. ; Hunstock, V. ; Friess, M.: Unberührte Möglichkeiten-Ideen und Anforderungen zur Gestensteuerung aus Nutzersicht. In: *Technik für ein selbstbestimmtes Leben* (2012)

[HSZ03] Hackebusch ; Schwarz ; Zeidler: *Teubner Taschenbuch der Mathematik.* Zweite Auflage. 2003

[IS12] Illposed-Software: *JavaOSC.* `http://www.illposed.com/software/javaosc.html`. Version: 2012. – letzter Zugriff 21.07.2012

[Joh10] John, Thilo; Klose Stefan; Häusler Benny; Frenzel Mirco; Michaelis T. Michael; Ernst E. Michael; Ernst: Reha@home ? Technisches Konzept und Prototyp für ein telemedizinisches Übungsprogramm im häuslichen Umfeld. In: *3. Deutscher Ambient Assisted Living Kongress* 26.-27 (2010), Januar, S. 6

[LK09] Lalanne, Denis ; Kohlas, Jürg: *Human Machine Interaction - Research Results of the MMI Program.* Berlin, 2009. – ISBN 978–3–642–00436–0

[M+12] McCartney, James u. a.: *SuperCollider.* `http://supercollider.sourceforge.net/`. Version: 2012. – letzter Zugriff 21.07.2012

[mir12] mirror2image: *How Kinect depth sensor works? Stereo triangulation?* `http://mirror2image.wordpress.com/2010/11/30/how-kinect-works-stereo-triangulation/`. Version: 2012

[Pot11] Potratz, Olaf: *Ein System zur physikbasierten Interpretation von Gesten im 3D-Raum*, HAW Hamburg, Bachelorarbeit, 2011

[PSS97] Pavlovic, Vladimir I. ; Sharma, Rajeev ; Sharma, Rajeev: Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1997), Nr. 7, 677-695. `http://www.cs.rutgers.edu/~vladimir/pub/pavlovic97pami.pdf`

[Rot02] Roth, Jörg: *Mobile Computing: Grundlagen, Technik, Konzepte.* 2002

[Sam12] Samsung: *SmartTV.* `http://smart-tv.samsung.de/`. Version: 2012

[syn] synthetic wave audio products GmbH: *WFS principle.* `http://www.syntheticwave.de/Wavefieldsynthesis.htm`. – letzter Zugriff 29.08.2012

[Tza10] Tzanetakis, Luis Gustavo; McNally Kirk; Jones R. George; Martins M. George; Martins: Stereo Panning Information for Music Information Retrieval Tasks. In: *J. Audio Eng. Soc* 58 (2010), Nr. 5, 409–417. http://www.aes.org/e-lib/browse.cfm?elib=15454

[WCC11] Wilson, Scott ; Cottle, David ; Collins, Nick: *The SuperCollider Book*. 2011

[WFM03] Wright, Matthew ; Freed, Adrian ; Momeni, Ali: Open Sound Control: State of the Art 2003, 2003

[Wik12a] Wikipedia: *Roll-Nick-Gier-Winkel — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Roll-Nick-Gier-Winkel&oldid=106571028. Version: 2012. – [Online; Stand 10. August 2012]

[Wik12b] Wikipedia: *Rotation matrix — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Rotation_matrix&oldid=506574157. Version: 2012. – [Online; Stand 13. August 2012]

[WR06] WAHRIG-Redaktion: *Die deutsche Rechtschreibung*. 2006

# A. Rotationsmatrizen

Eine Rotationsmatrix beschreibt den Rotationsanteil einer Koordinatentransformation von einem kartesischen Koordinatensystem in ein anderes (Zu einer vollständigen Koordinatentransformation würde noch der Translationsanteil fehlen). [BSMM05, S. 196] In diesem Fall vom Raumkoordinatensystem (Im folgenden als (A) bezeichnet) in das Koordinatensystem des Targets (im folgenden als (B) bezeichnet).

Zur Vereinfachung soll zunächst die Rotationsmatrix für den $\mathbb{R}^2$ beschrieben werden. Diese Matrix hat die Dimension (2x2). Die Spalten der Matrix sind die Einheitsvektoren der Richtungsvektoren der Achsen des Koordinatensystem B im Koordinatensystem A.

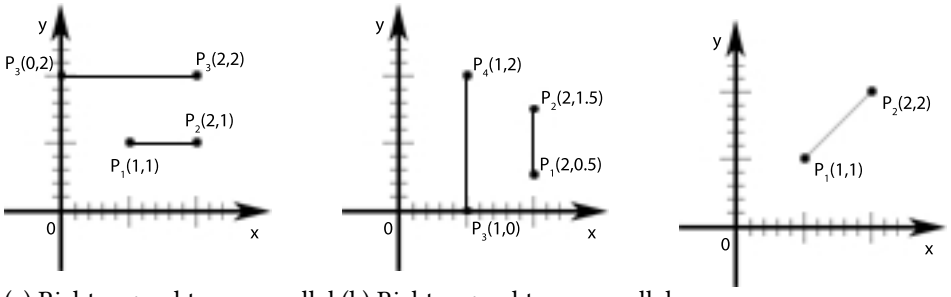
Ein Richtungsvektor beschreibt, wie der Name schon sagt, eine Ausrichtung, indem die Differenz von einem beliebigen Punkt P1 zu einem weiteren, in der Richtung liegenden, Punkt P2 beschrieben wird.

In Abbildung A.1a sind zwei Vektoren abgebildet, welche die gleiche Richtung angeben. Beide erstrecken sich parallel zur X-Achse. $\vec{v}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ wird durch die Punkte P1 und P2 beschrieben, da sie sich beide auf der gleichen Höhe der Y-Achse befinden und eine Einheit auf der X-Achse zwischen ihnen liegt. $\vec{v}_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ wird durch die Punkte P3 und P4 beschrieben, da sie sich beide auf der gleichen Höhe der Y-Achse befinden und zwei Einheiten auf der X-Achse zwischen ihnen liegt.

In Abbildung A.1b sind zwei Vektoren abgebildet ($\vec{v}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ und $\vec{v}_4 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$), welche Analog zu dem Beispiel A.1a Richtungen parallel zur Y-Achse beschreiben.

In Abbildung A.1c wird durch den Vektor $\vec{v}_5 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ ein Winkel von 45 Grad beschrieben.

Wie man sieht, spielt die Länge für einen Richtungsvektor keine Rolle. Daraus folgt, dass die Länge eines Richtungsvektors frei wählbar ist. Wird die Länge eines Vektors auf eins gesetzt, so ist dies ein Einheitsvektor. Die Vektoren $\vec{v}_1$ und $\vec{v}_3$ sind Einheitsvektoren.




(a) Richtungsvektoren parallel zur X-Achse (b) Richtungsvektoren parallel zur Y-Achse (c) Richtungsvektor in 45°

Abbildung A.1.: Richtungsvektoren

Alle Vektoren, welche sich parallel zu einer Achse befinden, erstrecken sich definitionsgemäß ausschließlich und in Ihrer ganzen Länge entlang dieser Achse und zu keinem Teil entlang einer anderen Achse. So ein Vektor der Länge 1 (Einheitsvektor) kann somit im $\mathbb{R}^2$ lediglich zwei verschiedene Ausprägungen haben: Entlang der X-Achse $\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ oder entlang der Y-Achse $\vec{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. In dem Fall, dass die beiden Koordinatensysteme nicht gegeneinander verdreht sind, also einer Drehung um 0 Grad unterliegen, liegen die jeweiligen X- und Y-Achsen parallel zueinander. Das heißt, ein Richtungsvektor, welcher die Ausrichtung der X-Achse von B innerhalb von A angibt, erstreckt sich ebenfalls gänzlich entlang der X-Achse von A. Er wäre daher der oben angegebene Vektor $\vec{x}$. Für Y würde das analog dasselbe bedeuten.

Wie schon erwähnt, geben die Spalten der Rotationsmatrix die Projektionen der Achsen von B auf A an. Werden die beiden Vektoren nun zu einer Matrix zusammengefasst, ergibt sich folgende Rotationsmatrix ($D_0$), welche eine Drehung um 0 Grad beschreibt:

$$D_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Dies entspricht:

$$D_0 = \begin{pmatrix} \cos\ 0 & -\sin\ 0 \\ \sin\ 0 & \cos\ 0 \end{pmatrix}$$

Analog dazu lautet eine Rotationsmatrix ($D_\alpha$) für eine Drehung um einen beliebigen Winkel $\alpha$

$$D_\alpha = \begin{pmatrix} \cos\ \alpha & -\sin\ \alpha \\ \sin\ \alpha & \cos\ \alpha \end{pmatrix}$$

und beschreibt beliebige Drehungen im $\mathbb{R}^2$.

Eine Drehung im $\mathbb{R}^2$, dessen Ebene durch die Achsen X und Y aufgespannt wird, ist eigentlich eine Drehung um die nicht explizit vorhandene Achse der dritten Dimension (der Z-Achse). Daher ändern sich die Einträge der Rotationsmatrix für den $\mathbb{R}^2$ auch nicht, wenn der Raum um eine weitere Dimension erweitert wird, sonder die Rotationsmatrix wird lediglich ergänzt. Zum einen müssen die beiden schon vorhandenen Vektoren $\vec{x}$ und $\vec{y}$ um die Angabe der dritten Dimension erweitert werden. Da $\vec{x}$ sich weiterhin exakt parallel zur X-Achse befindet, beträgt die Projektion auf der Z-Achse (so wie bisher schon auf der Y-Achse) ebenfalls 0. Der Vektor lautet für den $\mathbb{R}^3$ also $\vec{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$. Analog dazu $\vec{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$. Der Richtungsvektor der Z-Achse muss sich um ein orthogonales Koordinatensystem aufzuspannen und eine Drehung um 0 Grad auch um die Dritte Achse beizubehalten, direkt parallel zur Z-Achse des Raumes erstrecken und daher folgendermaßen aussehen $\vec{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Die Rotationsmatrix, welche eine Drehung um 0 Grad im $\mathbb{R}^3$ beschreibt lautet daher:

$$D_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Wie schon angesprochen, ist eine Drehung im $\mathbb{R}^2$ (Aufgespannt durch die Achsen X und Y) eigentlich eine Drehung um die dritte nicht vorhandene Achse Z. Im $\mathbb{R}^3$ lässt sich jede Ausrichtung durch höchstens drei aufeinanderfolgende Drehungen um die jeweiligen Achsen (X, Y und Z) darstellen. Die Rotationsmatrix für die Drehung um die Z-Achse im $\mathbb{R}^3$ lässt sich ganz leicht von der Rotationsmatrix der Drehung im $\mathbb{R}^2$ ableiten und lautet:

$$D_z(\gamma) = \begin{pmatrix} \cos\ \gamma & -\sin\ \gamma & 0 \\ \sin\ \gamma & \cos\ \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Analog dazu lauten die Rotationsmatrizen für die Drehungen um die X- und Y-Achsen:

$$D_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\ \alpha & \sin\ \alpha \\ 0 & -\sin\ \alpha & \cos\ \alpha \end{pmatrix} D_y(\beta) = \begin{pmatrix} \cos\ \beta & 0 & -\sin\ \beta \\ 0 & 1 & 0 \\ \sin\ \beta & 0 & \cos\ \beta \end{pmatrix}$$

Die allgemeine Rotationsmatrix ($D$) für eine Drehung um alle Achsen ergibt sich aus der Multiplikation der atomaren Rotationen wie folgt:

$$\begin{aligned} D &= D_z(\gamma)D_y(\beta)D_x(\alpha) \\ &= \begin{pmatrix} \cos\ \gamma & -\sin\ \gamma & 0 \\ \sin\ \gamma & \cos\ \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\ \beta & 0 & -\sin\ \beta \\ 0 & 1 & 0 \\ \sin\ \beta & 0 & \cos\ \beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\ \alpha & \sin\ \alpha \\ 0 & -\sin\ \alpha & \cos\ \alpha \end{pmatrix} \\ &= \begin{pmatrix} \cos\ \beta \cos\ \gamma & -\cos\ \alpha \sin\ \gamma + \sin\ \alpha \sin\ \beta \cos\ \gamma & \sin\ \alpha \sin\ \gamma + \cos\ \alpha \sin\ \beta \cos\ \gamma \\ \cos\ \beta \sin\ \gamma & \cos\ \gamma \cos\ \alpha + \sin\ \alpha \sin\ \beta\ \sin\ \gamma & -\sin\ \alpha \cos\ \gamma + \sin\ \beta \sin\ \gamma \cos\ \alpha \\ -\sin\ \beta & \sin\ \alpha \cos\ \beta & \cos\ \alpha \cos\ \beta \end{pmatrix} \end{aligned} \tag{A.1}$$

[Wik12b] Exemplarisch könnte eine mit Werten gefüllte Rotationsmatrix folgendermaßen aussehen:

$$D = \begin{pmatrix} 0.934737 & 0.068063 & 0.348761 \\ 0.355192 & -0.150600 & -0.922582 \\ -0.010270 & 0.986249 & -0.164946 \end{pmatrix}$$

Seien $x$, $y$ und $z$ die Einheitsvektoren des Koordinatensystem eines Objektes und $x'$, $y'$ und $z'$ die Einheitsvektoren des Koordinatensystem des Raumes, in dem sich das Objekt befindet, so bilden die Felder der Rotationsmatrix die Projektionen folgendermaßen:

$$\begin{pmatrix} x \to x' & x \to y' & x \to z' \\ y \to x' & y \to y' & y \to z' \\ z \to x' & z \to y' & z \to z' \end{pmatrix}$$

Die Felder einer Rotationsmatrix tragen folgende Bezeichnungen:

$$\begin{pmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{pmatrix} \tag{A.2}$$

# B. Tests zur Gestenentwicklung

- Protagonist 1 (25 J., männlich, Einzelhändler):
  - Auswählen: Der Arm wird horizontal in Richtung der Quelle ausgestreckt und die Hand im offenen Griff gehalten, wobei die Handfläche der Quelle entgegengestreckt ist. Der Handrücken ist dabei etwa 45° nach oben geneigt.
  - Abwählen: Der Arm und die Hand werden gesenkt.
  - Verschieben: Die Hand verbleibt in Auswahlposition. Der Arm bleibt starr und wird in die gewünschte Richtung ausgerichtet.
  - Heranziehen: Die Hand verbleibt in Auswahlposition und wird an den Körper herangezogen. Eine stetige Annäherung der Quelle wird erwartet, solange die Hand in einer dem Körper näheren Position verweilt. Durch Abwählen der Quelle oder das Zurückbewegen der Hand in die Position beim Auswählen der Quelle, wird die Bewegung gestoppt.
  - Wegschieben: Die Hand verbleibt in Auswahlposition und wird weiter vom Körper entfernt. Hierzu musste beim ersten Versuch ein Schritt nach vorne gemacht werden. Beim nächsten Mal wurde der Arm beim Auswählen schon gebeugt gehalten, um die Hand weiter von Körper entfernen zu können ohne einen Schritt tun zu müssen. Analog zum Heranziehen wird erwartet, dass die Quelle sich stetig entfernt, bis durch das Abwählen der Quelle oder das Bewegen der Hand zurück in die Position beim Auswählen der Quelle wird die Bewegung gestoppt wird.
- Protagonist 2 (28 J., weiblich, Psychologiestudentin):
  - Auswählen: Wie Protagonist 1.
  - Abwählen: Wie Protagonist 1.
  - Verschieben: Wie Protagonist 1.
  - Heranziehen: Die Hand wird um 180° so gedreht, dass die Handfläche nach oben zeigt, dann wird die Hand zum Körper geneigt und herangezogen. Erwartet wurde, dass die Quelle dabei die gleiche Distanz zurücklegt, wie die Hand. Nach dem Hinweis auf das Problem, dass die Quelle so nur schwer über große Distanzen bewegt werden könne, war die zweite Idee: Eine lineare Funktion, bei der Bewe-

gungen einen stärkeren Effekt hervorrufen, wenn sich die Hand näher am Körper befindet, sich die Quelle aber in beiden Fällen nicht weiter bewegt, wenn die Hand stillgehalten wird.
  - Wegschieben: Analog zum Heranziehen.

- Protagonist 3 (52 J., weiblich, Buchhalterin):
  - Auswählen: Die offene Hand wird mit ausgestrecktem Arm auf die Quelle ausgerichtet und hochkant gestellt, so dass die Handfläche der Richtung zugewandt ist, in welche die Quelle verschoben werden soll.
  - Abwählen: Wie Protagonist 1.
  - Heranziehen: Wie Protagonist 2.
  - Wegschieben: Wie Protagonist 1, nur mit gestreckten Fingern.
- Protagonist 4 (25 J., männlich, Lehramtsstudent(Biologie, Philosophie))
  - Auswählen: Die Hand wird der Quelle entgegen gestreckt, gegriffen und etwas an den Körper herangezogen.
  - Abwählen: Der Arm wird wieder ausgestreckt und die Hand geöffnet.
  - Verschieben: Der Arm verbleibt in Auswahlposition und wird mit oder ohne Körper in die gewünschte Richtung gedreht. Die Entfernung der Quelle zum Protagonisten wird dabei als gleichbleibend vorausgesetzt.
  - Heranziehen: Die Quelle wird anstatt mit der normalen Auswahlgeste mit einer Handhaltung im Untergriff, die sonst der normalen Geste ähnelt, ausgewählt. Die Quelle wird dann ruckartig herangezogen. Der Effekt wird ähnlich dem Scrollen erwartet, wie es auf den meisten Touchscreens umgesetzt wird. Der Bildlauf wird dabei nach Beendigung der Geste fortgeführt und nimmt stetig ab, bis er zum endgültigen Stillstand.
  - Wegschieben: Die Quelle wird normal ausgewählt und ebenfalls ruckartig weggeschoben mit einem Effekt, wie beim Heranziehen.
- Protagonist 5 (20 J., weiblich, Sozialökonomiestudentin):
  - Auswählen: Der Arm wird der Quelle entgegen ausgestreckt. Die Handfläche nach vorn gerichtet.
  - Abwählen: Der Arm wird zurückgezogen.
  - Verschieben: Arm und Hand bleiben in der Position der des Auswählens und der Arm wird durch Drehung des Schultergelenks oder/und des Körpers in die gewünschte Richtung ausgerichtet.
  - Heranziehen: Nach der Auswahlgeste wird die Hand zum Körper herangezogen.

  - Wegschieben: Nach der Auswahlgeste wird die Hand vom Körper weggeschoben.

- Protagonist 6 (25 J., männlich, Mediendesignstudent);
  - Auswählen: Der Arm wird ausgestreckt. Die Handhaltung hängt von der Folgegeste ab.
    * Soll die Quelle verschoben werden, ist die Hand hochkant. Finger zeigen nach vorne. Die Handfläche zeigt in die Richtung, in welche die Quelle verschoben werden soll.
    * Soll die Quelle weggeschoben werden, zeigt die Handfläche nach vorne und die Fingerspitzen nach oben.
    * Soll die Quelle herangezogen werden, bildet die Hand einen Untergriff.
  - Abwählen: Der Arm wird gesenkt.
  - Verschieben: Die Handfläche zeigt in die Richtung, in welche die Quelle verschoben werden soll und der Arm wird durch Drehung des Schultergelenkes oder/und des Körpers in die gewünschte Richtung ausgerichtet.
  - Heranziehen: Im Untergriff wird die Hand an den Körper herangezogen.
  - Wegschieben: Während die Handfläche zur Quelle gerichtet ist und die Fingerspitzen nach oben zeigen wird die Hand vom Körper weggeschoben.

- Protagonist 7 (männlich, Informatikprofessor der HAW Hamburg):
  - Auswählen: Mit dem Zeigefinger der rechten Hand wird, während die anderen Finger angewinkelt bleiben, auf die Quelle gezeigt. Der Handrücken ist dabei nur leicht nach oben und etwa 30° nach rechts geneigt
  - Abwählen: Der Arm wird gesenkt.
  - Heranziehen: Die Hand wird nach dem Auswählen in einen Untergriff gebracht und zum Körper herangezogen. Über die genaue Erwartung besteht Unklarheit. Ein kontinuierliches Annähern der Quelle erscheint am wahrscheinlichsten/logischsten.
  - Wegschieben: Die Hand wird nach dem Auswählen in einen offenen Obergriff gebracht, etwas an den Körper herangezogen um dann durch das wieder Abstoßen der Hand den Effekt zu erzielen. Ein kontinuierliches Entfernen der Quelle wird erwartet, bis die Hand wieder etwas zum Körper herangezogen wird.

- Protagonist 8 (30 J., männlich, VWLer):
  - Auswählen: Der Arm wird der Quelle entgegen ausgestreckt und die Hand im Obergriff geballt.
  - Abwählen: Die Hand wird geöffnet und der Arm ganz leicht gebeugt.

  - Verschieben: Die Hand und der Arm verbleiben in der Position des Auswählens und werden durch Drehung des Schultergelenkes oder/und des Körpers in die gewünschte Richtung ausgerichtet.
  - Heranziehen: Die Quelle wird normal ausgewählt und die Hand anschließend herangezogen. Die Quelle kommt exponentiell schneller, wenn schneller gezogen wird. Wenn die Hand nicht bewegt wird, bewegt sich die Quelle auch nicht.
  - Wegschieben: Analog zum Heranziehen mit einem Schritt nach vorne, um die Hand nach dem Auswählen mit gestrecktem Arm der Quelle noch näher bringen zu können.

- Protagonist 9 (25 J., männlich, Informatik-Master-Student):
  - Auswählen: Der Arm wird der Quelle entgegen ausgestreckt und die Hand im Obergriff geballt.
  - Abwählen: Die Hand wird geöffnet und der Arm gesenkt.
  - Verschieben: Der Arm wird mit »gegriffener« Quelle in die gewünschte Richtung ausgerichtet und die Hand geöffnet.
  - Heranziehen: Die Hand wird mit »gegriffener« Quelle zum Körper herangezogen und während die Hand am Körper verbleibt, nähert sich die Quelle, bis die Hand geöffnet wird.
  - Wegschieben: Analog zum Heranziehen.
- Protagonist 10 (24 J., männlich, Student der Angewandten Informatik):
  - Auswählen: Der Arm wird der Quelle entgegen ausgestreckt und die Hand im offenen Obergriff gehalten, wobei die Handfläche der der Quelle entgegen geöffnet ist.
  - Abwählen: Der Arm wird gesenkt und die Hand entspannt.
  - Verschieben: Der Arm und die Hand verbleiben in der Auswahlstellung und werden durch Drehung des Schultergelenkes und/oder des Körpers in die gewünschte Richtung ausgerichtet, wobei die Distanz zur Quelle gleichbleibend erwartet wird.
  - Heranziehen: Die Hand wird nach dem Auswählen in gleichbleibender Stellung an den Körper herangezogen. Ein kontinuierliches Annähern der Quelle wird erwartet, bis die Hand wieder in die Ausgangsposition gebracht wird.
  - Wegschieben: Analog zum Heranziehen. Nach dem Erkennen der Problematik, dass beim Auswählen der Quelle mit ausgestrecktem Arm nun jedoch ein Schritt nach vorne getan werden muss, wird die Quelle zukünftig mit angewinkeltem Arm

ausgewählt, um die Hand durch Strecken des Armes der Quelle entgegenbringen zu können.

# C. OSC-Commands

OSC Complete

# Wonder OSC-Commands

## Complete OSC reception implementation chart

Parametertypes: i = int, f = float, s = string, NULL = arbitrary number

**Note:** all commands have to be prefixed with /WONDER

| Command | Parameters | content | > | ... | ... | ... | ... | ... |
|---|---|---|---|---|---|---|---|---|
| **cwonder** | | | | | | | | |
| /source/activate | i | id | | | | | | |
| /source/deactivate | i | id | | | | | | |
| /source/type | i i | id | | type [ 0=plane, 1=point ] | | | | |
| /source/type | i f | id | | type [ 0=plane, 1=point ] | timestamp [ seconds ] | | | |
| /source/angle | i f | id | | angle [ degrees ] | | | | |
| /source/angle | i f f | id | | angle [ degrees ] | duration [ seconds ] | | | |
| /source/angle | i f f f | id | | angle [ degrees ] | duration [ seconds ] | timestamp [ seconds ] | | |
| /source/position | i f f | id | | x coordinate [ meters ] | y coordinate [ meters ] | | | |
| /source/position | i f f f | id | | x coordinate [ meters ] | y coordinate [ meters ] | duration [ seconds ] | | |
| /source/position | i f f f f | id | | x coordinate [ meters ] | y coordinate [ meters ] | duration [ seconds ] | timestamp [ seconds ] | |
| /source/position | i f f f f f | id | | x coordinate [ meters ] | y coordinate [ meters ] | not used | timestamp [ seconds ] | duration [ seconds ] |
| /source/color | i i i | id | | red [ 0;255 ] | green [ 0;255 ] | blue [ 0; 255 ] | | |
| /source/groupId | i i | id | | groupId | | | | |
| /source/rotationDirection | i i | id | | inverted [ 0=false, 1=true] | | | | |
| /source/scalingDirection | i i | id | | inverted [ 0=false, 1=true] | | | | |
| /source/dopplerEffect | i i | id | | on [ 0 = false, 1 = true ] | | | | |
| /group/activate | i | groupId | | | | | | |
| /group/deactivate | i | groupId | | | | | | |
| /group/position | i f f | groupId | | x coordinate [ meters ] | y coordinate [ meters ] | | | |
| /group/color | i i i | groupId | | red [ 0;255 ] | green [ 0;255 ] | blue [ 0; 255 ] | | |
| /project/createWithScore | s | projectname | | | | | | |
| /project/createWithoutScore | s | projectname | | | | | | |
| /project/addScore | none | | | | | | | |
| /project/load | s | projectname | | | | | | |
| /project/save | | | | | | | | |



Abbildung C.1.: Liste aller OSC-Commands, welche cwonder versteht

OSC Complete

| | | | |
|---|---|---|---|
| /project/save | s | projectname | |
| /project/snapshot/take | i | snapshotId | |
| /project/snapshot/take | i s | snapshotId | name |
| /project/snapshot/recall | i f | snapshotId | duration [ seconds ] |
| /project/snapshot/delete | i | snapshotId | |
| /project/snapshot/rename | i s | snapshotId | name |
| /project/snapshot/copy | i i | snapshotId (from) | snapshotId (to) |
| /score/play | none | | |
| /score/stop | none | | |
| /score/setStartScenario | none | | |
| /score/enableRecord | i | 0 = off, 1 = on | |
| /score/enableRead | i | 0 = off, 1 = on | |
| /score/reset | | | |
| /score/newtime | f | time [ seconds ] | |
| /score/enableMMC | i | 0 = off, 1 = on | |
| /score/source/enableRecord | i i | sourceId | 0 = off, 1 = on |
| /score/source/enableRead | i i | sourceId | 0 = off, 1 = on |
| /jfwonder/frametime | i | currenttime [ jackframe ] | |
| /jfwonder/connect | none | | |
| /jfwonder/error | s | errormessage | |
| /stream/render/connect | s | name | |
| /stream/render/connect | s s | host | port |
| /stream/render/connect | none | | |
| /stream/render/disconnect | none | | |
| /stream/render/pong | i | count | |
| /stream/render/send | NULL | | |
| /stream/score/connect | s | name | |
| /stream/score/connect | s s | host | port |
| /stream/score/connect | none | | |
| /stream/score/disconnect | none | | |
| /stream/score/pong | i | count | |
| /stream/score/send | NULL | | |
| /stream/visual/connect | s | name | |
| /stream/visual/connect | s s | host | port |

OSC Complete
/stream/visual/connect
none
/stream/visual/disconnect
none
/stream/visual/pong
i
count
/stream/visual/send
NULL
/stream/timer/connect
s s
host
port
/stream/timer/connect
none
/stream/timer/pong
i
count
/reply
s i s
reply to message
state
message

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

Hamburg, 30. August 2012 Malte Nogalski