Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Sebastian Jandt

Visualisierung und Steuerung von Wellenfeldsynthese-Quellen mit Android-Geräten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Jandt

Visualisierung und Steuerung von Wellenfeldsynthese-Quellen mit Android-Geräten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Betreuender Prüfer: Prof. Dr. Wolfgang Fohl
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 12. Juni 2014

Sebastian Jandt

Thema der Arbeit
Visualisierung und Steuerung von Wellenfeldsynthese-Quellen mit Android-Geräten

Stichworte
Wellenfeldsynthese, Android, Augmented Reality, Mensch-Maschine-Interaktion, Echtzeitbedienung

Kurzzusammenfassung
Diese Arbeit beschreibt Ansätze zur Visualisierung und interaktiven Steuerung von Soundquellen einer Wellenfeldsynthese-Anlage. Dafür soll ein Mobilgerät, welches mit dem Android-Betriebssystem ausgestattet ist, genutzt werden. Es werden Grundlagen über Augmented Reality Lösungen, die Wellenfeldsynthese und das eingesetzte System erläutert. Abschließend werden die Konzepte dieser Ansätze beschrieben und implementiert.
Sebastian Jandt

Title of the paper
Visualisation and control of wave field synthesis sources using Android devices

Keywords
Wave field synthesis, Android, Augmented Reality, Human-Machine-Interaction, Real-Time Control

Abstract
This thesis describes approaches for the visualisation and interactive control of sound sources of a wave field synthesis system. For this we'll be using a mobile device running the Android operating system. Fundamentals about augmented reality solutions, the wave field synthesis and the used systems will be shown. Finally, the concepts for these approaches will be explained and implemented.

Inhaltsverzeichnis

1 Einleitung

1.1 Mensch-Maschine-Schnittstelle

Täglich umgeben uns eine große Anzahl an Maschinen welche wir, ohne dass es uns bewusst ist, mit Leichtigkeit Steuern können. Die Bedienung von Maschinen wie Telefon, Fernseher, Auto, Computer, etc. gehört zu unserem alltäglichen Leben dazu und wird gar nicht mehr richtig wahrgenommen. Diese Maschinen sind für uns zu unverzichtbaren Werkzeugen in jeglichen Lebenslagen geworden.

Für die Bedienung muss der Mensch in einen Dialog mit diesen Maschinen treten. Dieser Prozess, auch *Mensch-Maschine-Interaktion (MMI)* genannt, findet über sogenannte *Mensch-Maschine-Schnittstellen (MMS)* statt. Diese Schnittstellen liefern Informationen, die der Mensch verstehen und wiederum zur Bedienung der Maschinen nutzen kann.

Im Zusammenhang mit dieser Arbeit sei eine spezielle Form der MMI betrachtet, der sogenannten *Mensch-Computer-Schnittstelle (MCS)*, welche aus einer Kombination von zusammenpassender Soft- und Hardware besteht. Als Beispiel sei einmal das Display eines Mobiltelefons, welches diverse Informationen wie ein Kamerabild und zusätzlich darüber generierte Objekte anzeigen kann, welche durch Berührungen wiederum Steuerungsdaten an die aktuelle Anwendung zurückliefern. Das Verständnis über die Arbeitsweise einer Maschine sollte für den bedienenden Menschen dabei schnell und einfach zu verstehen sein, um ihn in seiner Arbeit zu unterstützen und Fehler zu vermeiden. Dafür ist es von großen Wert auf sofortige und nachvollziehbare Rückmeldung zu achten, sodass der Benutzer die Auswirkung auf die Anwendung in direkten Zusammenhang mit der von ihm ausgeführten Aktion bringen kann.

1.2 Gegenstand der Arbeit

Gegenstand dieser Arbeit ist eine Wellenfeldsynthese-Anlage (siehe 2) an der HAW-Hamburg welche um eine weitere Eingabemethode ergänzt werden soll. Die aktuell zur Verfügung

stehenden Mittel zur interaktiven Steuerung der WFS-Quelle sind auf einen stationären Arbeitsplatz, oder eine Gestensteuerung, welche rein auf der akustischen Ortung des Nutzers basiert, limitiert. Die Möglichkeit WFS-Quelle auch mobil, also an jeder Stelle im Raum der WFS-Anlage, interaktiv zu steuern und dabei ein direktes visuelles Feedback zu erhalten ist bislang nicht möglich.

Malte Nogalski beschreibt diese ebenfalls im Fazit seiner Bachelorarbeit:

> 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. [Nog12]

1.3 Zielsetzung

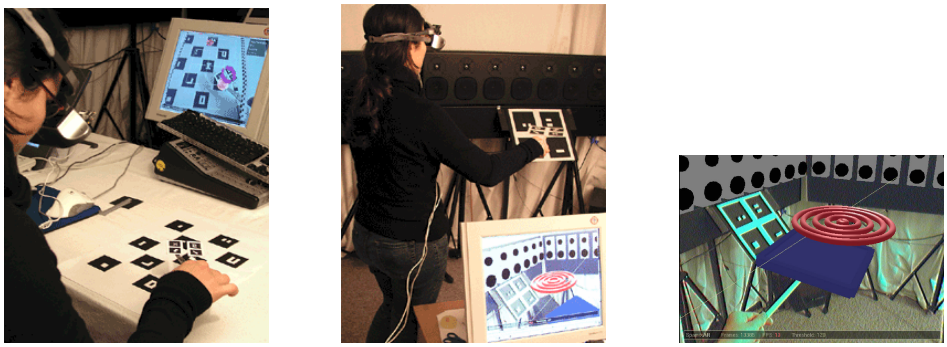
Das angestrebte Ziel besteht darin, die Quellen der WFS-Anlage mit Hilfe eines Mobilgerätes, welches das Android Betriebssystem verwendet, zu visualisieren und steuern zu können.

Dabei werden verschiedenen Szenarien betrachtet. Zunächst soll die Möglichkeit betrachtet werden, bestehende Bedienkonzepte der WFS-Anlage auf einem Mobilgerät verfügbar zu machen, um Änderungen auch fern eines stationären Arbeitsplatzes zu sehen und durchzuführen. Danach soll die Möglichkeit betrachtet werden, in der das Android Gerät sich an einem stationären Punkt befindet und die WFS-Quelle durch tragbare Marker von der Kamera des Android Gerätes erkannt werden. Zuletzt soll betrachtet werden, dass das Android Gerät sich selber, gegebenenfalls anhand von stationären Markern, im Raum orientiert. Dabei soll dem Nutzer ermöglicht werden, die Quellen der WFS-Anlage durch Objekte, welche über das Kamerabild gelegt werden, auf dem Bildschirm zu sehen und zu steuern.

1.4 Ähnliche Ansätze

Einen vergleichbaren Ansatz beschreiben Melchior, Laubach und de Vries in ihrem Paper [MLV05]. Sie verwenden dabei allerdings kein Mobilgerät sondern ein sogenanntes Head Mounted Display (HMD), ein auf dem Kopf getragenes visuelles Ausgabegerät. Dieses HMD besitzt ebenfalls eine Kamera mit welcher ein Bild in Blickrichtung des Benutzers an einen

Computer zur Verarbeitung weitergegeben werden kann, welcher wiederum ein Bild an das HMD ausgibt. Sie beschreiben dabei zwei Ansichten. Zum einen die exozentrische Ansicht auf das WFS-System im Miniaturformat, wo der Benutzer durch Marker die Position der WFS-Quelle verändern kann. Zum anderen eine egozentrische Ansicht, wo die Position des HMDs im Raum der WFS erkannt wird um dann anschließend durch Gesten mit einem größeren Marker, Quellen auszuwählen und in ihrer Position zu verändern.



(a) Exozentrische Ansicht (b) Egozentrische Ansicht (c) Bild des HMDs

Abbildung 1.1: Visualisierung und Steuerung über ein Head Mounted Display [MLV05]

1.5 Aufbau

Zunächst sollen einmal die Grundlagen der Wellenfeldsynthese in Kapitel 2 beschrieben werden. Es wird darauf eingegangen, welche Vorteile das System gegenüber herkömmlichen Audiosystemen bietet. Die physikalischen Grundlagen werden kurz angerissen, allerdings wird hier nicht allzu sehr ins Detail über benötigten Algorithmen eingegangen, da diese nicht von Relevanz für diese Arbeit sind. Das Labor an der HAW-Hamburg, in der die Wellenfeldsynthese-Anlage steht, an welcher im Rahmen dieser Arbeit getest wurde wird, ebenfalls beschrieben.
Die Wahl des Android-Betriebssystems, welche für die Interaktion mit der Wellenfeldsynthese-Anlage im Rahmen dieser Arbeit als Zielsystem dienen soll, wird in Kapitel 3 erläutert.
In Kapitel 4 wird der Begriff der *Augmented Reality* erläutert. Dabei werden ein paar Ansätze und die Verwendung von Augmented Reality Umgebungen beschrieben. Die Visualisierung und Steuerung der Quellen soll mit Hilfe einer solchen Umgebung umgesetzt werden.
Kapitel 5 beschäftigt sich mit dem Augmented Reality SDK Vuforia der Firma Qualcomm, welches für die Erstellung der Android Anwendung im Rahmen dieser Arbeit genutzt wird.
Die Verfahren zur Visualisierung und Steuerung mit einem Android Gerät werden dann in Kapitel 6 erarbeitet und umgesetzt.

Abschließend wird in Kapitel 7 zusammengefasst was Entwickelt wurde, ein kurzes Fazit zur Nützlichkeit wird abgegeben und ein paar Ideen aufgelistet, welche auf die in dieser Arbeit entstandenen Verfahren aufbauen könnten.

2 Wellenfeldsynthese

In diesem Kapitel wird nur kurz auf die Grundlagen der *Wellenfeldsynthese* (kurz WFS) und den grundlegenden Aufbau der Anlage im Labor der HAW-Hamburg eingegangen. Für nähere Informationen über den Aufbau verweise ich auf die Bachelor-Arbeit von Malte Nogalski [Kapitel 2] [Nog12] und für tiefere Einblicke über die Theorie und Umsetzung der WFS siehe [Baa06] und [Baa07].

2.1 Hintergrund

Das Ziel von herkömmlichen zwei- und mehrkanaligen Audiowiedergabe-Systemen besteht darin, Soundquellen für das menschliche Ohr im Raum ortbar zu machen. Das gelingt allerdings meist nur, wenn sich der Zuhörer an einem ganz speziellen Ort, dem so genannten Sweet Spot befindet. Der Grund dieses Phänomens liegt darin, dass der Ort und die Anzahl der Lautsprecher im voraus bekannt sind und von dem Tonproduzenten, welche die Audiospur erstellt hat, berücksichtigt wurde. Quellen die für das menschliche Ohr nicht genau an der Position eines Lautsprechers wahrzunehmen sein sollen, werden dafür auf mehrere Lautsprecher gelegt und in ihrer Lautstärke auf den einzelnen Spuren angepasst.
Bekannt ist diese Methode auch unter dem Namen des Panning. Hierbei wird zum Beispiel bei einer Surround-Ausgabe eine Monospur zwischen den einzelnen angedachten Lautsprechern verschoben. Exemplarisch sind hier Bilder des Surround-Pan Fensters aus der Software-Audio- und Midisequenzer-Software *Logic Pro 9* der Firma *Apple* in Abbildung 2.1a und das Äquivalent der open-source Audio Workstation Ardour [1] mit dem Namen VBAP in Abbildung 2.1b zu sehen. Für Zuhörer außerhalb des Sweet Spot kann der Klangeindruck schnell von den Vorstellung des Tonproduzenten abweichen.
Im Kontrast dazu ist das Ziel der WFS, einen Raum zu erstellen, in dem die Ortung von Soundquellen an jeder Position für den Zuhörer möglich ist. Dabei soll dann nicht mehr die Position der einzelnen Lautsprecher, sondern die virtuelle Position der einzelnen Soundquellen für die räumliche Wahrnehmung wichtig sein.

[1] http://ardour.org/

(a) Surround-Pan Fenster in Logic Pro 9 (b) Vector-base Amplitude Panning in Ardour

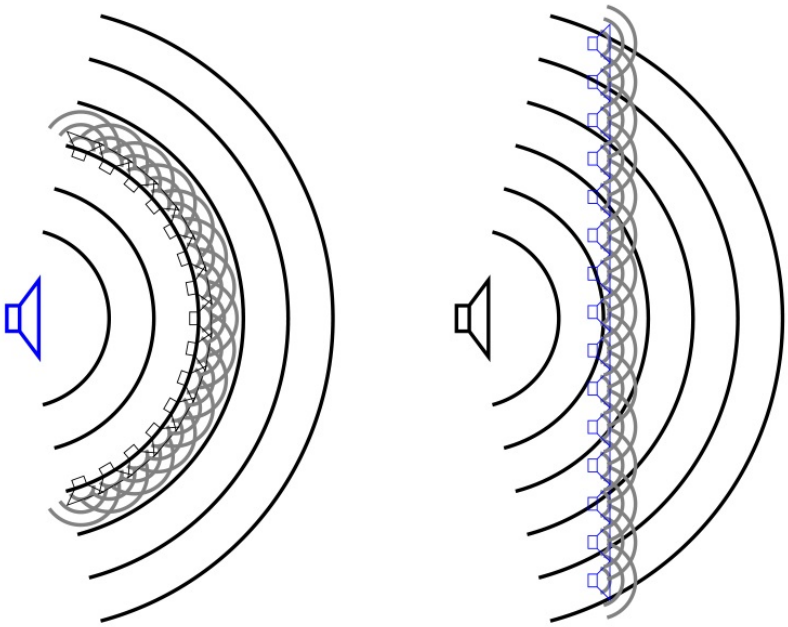
Abbildung 2.1: Surround-Panning Fenster

Der wohl gravierendste Unterschied zu herkömmlichen Zwei- und Mehrkanal Systemen (z.B. Stereo, 5.1, …) besteht darin, dass deutlich mehr Lautsprecher und Audiokanäle mit deutlich geringerem Abstand zueinander benötigt werden. Weitere Informationen über die physikalischen Hintergrunde sind dazu in Kapitel 2.2 zu finden. Die Anzahl der benötigten Kanäle, welche individuell angesteuert und berechnet werden müssen, steigt dabei leicht auf mehrere hundert an. Eine Hochrechnung am Beispiel des Labors an der HAW ist unter Kapitel 2.4.1 zu finden. Diese hohe Anzahl der Kanäle stellt einen nicht zu unterschätzenden Rechenaufwand an die WFS-Anlage. Dank des technologischen Fortschritts der letzten Jahre, insbesondere der Leistungsfähigkeit für parallele Aufgaben der kommerziell verfügbaren Rechnersysteme und der Verfügbarkeit der entsprechenden Software, ist die WFS technisch umsetzbar geworden.

2.2 Physikalische Grundlagen

Die WFS beschreibt ein Verfahren zur räumlichen Klangerzeugung welches auf dem *huygens-sches Prinzip*, auch *huygens-fresnelsches Prinzip* (nach Christiaan Huygens und Augustin Jean Fresnel) basiert. Dieses Prinzip besagt, dass jeder Punkt einer Wellenfront (hier Schallwellen) als Ausgangspunkt einer neuen Welle dienen kann. Diese neuen Wellen, auch Elementarwellen genannt, bilden durch Überlagerung wiederum eine Wellenfront welche der Ausgangswelle ähnlich ist (siehe Abbildung 2.2a).

Bei der WFS werden nun Lautsprecher in diskreten Abständen als Ausgangspunkte von Elementarwellen verwendet und einzeln von jeweils einer Audiospur der WFS-Anlage angesteuert (siehe Abbildung 2.2b). Die Positionen der Ausgangswellen werden durch virtuelle WFS-Quellen beschrieben. Die WFS-Anlage berechnet nun die Ausbreitung der Ausgangswellen, also deren Verzögerung und Amplitude für jede Elementarwelle, aus Abstand zwischen WFS-Quellen und den Lautsprechern und der Dauer welcher eine reale Wellenfront benötigen würde.



(a) Huygenssches Prinzip (b) Wellenfeldsynthese

Abbildung 2.2: Vom *Huygenssches Prinzip* zur *Wellenfeldsynthese* [Mon07]

2.3 Das Labor

Für Projekte steht eine 5m x 6m große Wellenfeldsynthese-Anlage der Firma FourAudio [Fou] mit insgesamt 26 Audiomodulen und einem 19“ Rackwagen, welcher wiederum drei Linux und einen Apple Computer beherbergt, zur Verfügung.
Von den drei Linux Computern dient einer als WFS-Server. Der WFS-Server führt die Komponente cwonder aus (siehe Kapitel 2.4.1) und muss vor den restlichen Rechnern hochgefahren werden. Die beiden restlichen Linux Computer dienen als WFS-Nodes. Diese WFS-Node übernehmen das Rendering der einzelnen Ausgabekanäle. Die Anzahl der WFS-Node hängt zusammen mit der Anzahl der benötigten Audiomodule von der Größe des Raumes, in dem die

WFS-Anlage aufgebaut werden soll, ab. Die verwendeten Lautsprechermodule (siehe Abbildung 2.4a) bieten 8 Kanäle mit einem Abstand von jeweils 10cm. Bei einer Größe von 5m x 6m, wie im Labor an der HAW, ergibt sich dabei eine Anzahl von 6 Modulen für die kurzen Seiten (resultiert in 480cm) und 7 Modulen für die langen Seiten (560cm). Die Anzahl der WFS-Node berechnet sich dadurch, das die verwendete Dante Soundkarte eine Grenze von 128 Kanälen besitzt und maximal 16 der oben genannten Module mit Daten versorgen kann. Der Apple Computer, auch als WFS-Mac bezeichnet, stellt die Hauptschnittstelle zwischen der WFS-Anlage und äußeren Systemen dar. Auf ihm befinden sich unter anderem Scripte zum starten und herunterfahren des WFS-Servers und der WFS-Nodes, der GUI Komponenten von swonder und Programme zum Abspielen der Audiostreams für die WFS-Quellen. Zum Abspielen können dabei einfache Anwendungen wie Medienplayer oder komplexe mehrkanal-fähige Audio Workstations dienen. Die Anordnung der Komponenten ist in Abbildung 2.3 zu sehen.

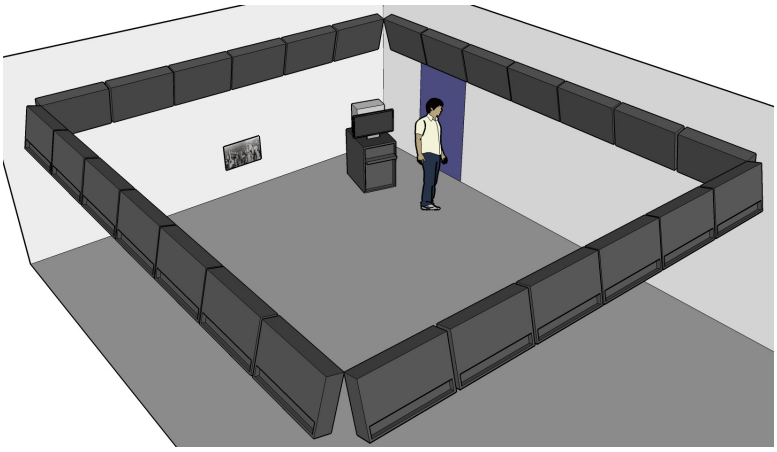


Abbildung 2.3: Das Labor an der HAW mit den WFS Komponenten und dem für in Kapitel 6.4 wichtigen Image-Target

Die verwendeten Lautsprechermodule basieren auf den von Goertz et al. im Jahr 2007 entwickelten achtkanaligen WFS-Lautsprechermodulen (Siehe [GMMW07]), welche für eine WFS-Anlage im großen Hörsaal H104 der TU Berlin entstanden (siehe [MGM+07]).

Jedes Modul bietet dabei acht unabhängige Audiokanäle, welche von jeweils drei Breitband Lautsprecherchassis in einer Zeile und je vier Kanälen auf zwei Tieftonchassis wiedergegeben werden. Das Layout des Moduls ist in Abbildung 2.4a zu erkennen.

Im Vergleich zu den Modulen an der TU Berlin wurde mehr Wert auf universelle Einsatzmöglichkeiten gelegt. Die Gehäuse der Module wurden zum Beispiel wie in Abbildung 2.4b

zu erkennen, für die Montage an Traversensystemen optimiert. Wo bei den Modulen der TU Berlin noch aufwendige und teure MADI und ADAT Netzwerke zur Signalübertragung zum Einsatz kamen, setzen die Module im Labor auf ein System namens Dante der Firma Audinate [Aud]. *Dante* setzt dabei auf ein Layer 3 IP Protokoll auf und kann daher über ein gewöhnliches und kostengünstiges Computernetzwerk übertragen werden. Die WFS-Module bieten hierfür zwei Netzwerkschnittstellen, welche wahlweise für eine Redundanz oder zum durchschleifen der Netzwerkverbindung zur Minderung des Verkabelungsaufwandes dienen [MGTK12].

(a) Layout eines WFS-Moduls (b) Traversenmontage

Abbildung 2.4: WFS Lautsprechermodul von FourAudio

2.4 swonder

Im Labor der Hochschule für angewandte Wissenschaften Hamburg (HAW-Hamburg) kommt das open-source Paket swonder (Wave field synthesis Of New Dimensions of Electronic music in Realtime) zum Einsatz. Es ist ein modulares und hardwareunabhängiges Softwaresystem welches seit 2002 unter Leitung von Marije Baalman entwickelt wird (siehe [Baa05] [Baa07], [BHSK07]). Die Aufteilung in einzelne Module erlaubt es, den aufkommenden Rechenaufwand auf mehrere Rechner zu verteilen.

2.4.1 Aufbau

Die aktuell zum Betrieb und der Bedienung wichtigen Modulen der Anlage hier einmal kurz vorgestellt:

cwonder Die zentrale Anlaufstelle von Steuerbefehlen. Module können sich hier anmelden um dann eingehende Nachrichten selber zu empfangen. Wird auf dem WFS-Server ausgeführt.

twonder Die eigentlichen, zur WFS benötigten, Berechnungen für die einzelnen Audiokanäle werden hier ausgeführt. Mehrere Instanzen von twonder werden, der Lastenverteilung zugute, auf den WFS-Node verteilt.

xwonder Grafische Oberfläche mit Übersicht über alle aktiven WFS-Quellen und deren Eigenschaften (Position, Art, Name, ...). Sie entstand im Laufe der Magisterarbeit von Hans-Joachim Mond, da die vorher verfügbaren Benutzeroberflächen eher für Nutzer mit fundiertem technischen Wissen über die WFS-Anlage waren und eine übersichtliche und ergonomisch einfach zugängliche Benutzeroberfläche für wonder benötigt wurde (siehe Abbildung 2.5, [Mon07]). Verfügbar auf dem WFS-Mac.

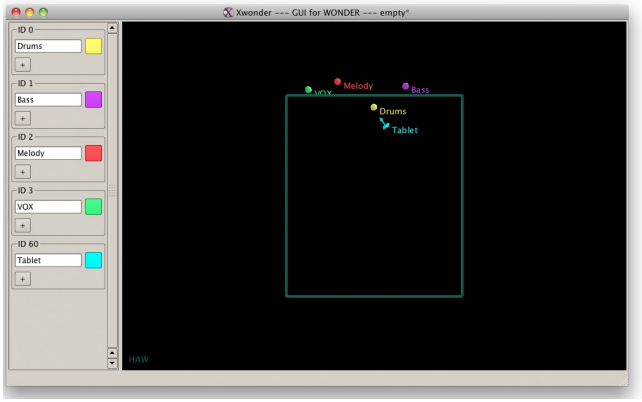


Abbildung 2.5: xwonder

2.4.2 Steuerung

Die einzelnen Module des swonder Pakets verwenden zum Informationsaustausch OSC (Open Sound Control). OSC ist ein auf Nachrichten basiertes Kommunikationsprotokoll zur Übermittlung von Steuersignalen unter Computern, Synthesizern und anderen multimedialen Geräten. Im Gegensatz zum klassischen MIDI (Musical Instrument Digital Interface) wurde OSC für moderne Computernetzwerke entwickelt und kann über eine große Anzahl an verschiedenen Kommunikationswegen verteilt werden. Es eignet sich zur Verbreitung von Steuersignalen über weite und lokale Netzwerke, Interprozesskommunikation oder sogar innerhalb einer

einzelnen Anwendung [WFM03].

Die OSC Messages bestehen im Wesentlichen aus einer Adresse des zu ändernden Parameters (OSC Address Pattern) gefolgt von einem ’,’ (Komma) und der Anzahl und Art der übermittelten Parameter beschrieben als Zeichenkette (OSC Type Tag String), abschließend dann den Argumenten (OSC Arguments) selbst.

z.B.: “/wonder/source/position“ ’,’ “iff“ 1 0.4 -1.3

Die Standard Type Tag Strings sind in Tabelle 2.1 zu finden. Es gibt darüber hinaus bereits weitere, von verschiedenen Anwendungen genutzte, Type Tags um z.B. Arrays oder 64 bit und boolesche Werte zu übermitteln.
Alle Argumente werden in Big-Endian Byte-Reihenfolge erwartet und müssen in der Länge

Tabelle 2.1: OSC Standard Type Tag Strings

Type Tag	Beschreibung
i	32 bit langer, vorzeichenbehafteten Integer
f	32 bit lange Fließkommazahl (Nach IEEE 754)
s	Nullterminierte ASCII Zeichenfolge
b	Ein 32 bit Integer gefolgt von dieser Anzahl an beliebigen binären Inhalt

ein Vielfaches von 4 sein. Um dies zu gewährleisten müssen durch Padding 0 bis 3 zusätzliche Nullen hinter Zeichenketten oder binären Inhalt angehängt werden.
Um mehrere Messages zusammen zu versenden wird den Daten das Schlüsselwort „#bundle“ und ein OSC Time Tag vorangestellt, gefolgt von der jeweiligen Datenlänge der Message (als 32 bit Integer) und der Message selbst. Genannt werden diese Pakete OSC Bundles.
Wo Messages darauf aus sind, beim Empfänger sofort bzw. schnellst möglich verarbeitet zu werden, sind Bundles dazu gedacht, erst zu dem im Time Tag beschriebenen Zeitpunkt verarbeitet zu werden. Das Time Tag besteht dabei aus einer 64 bit langen Festkommazahl, die ersten 32 bit beschreiben die vergangenen Sekunden seit der Mitternacht am 1. Januar 1900, die zweiten 32 bit beschreiben den Sekundenbruchteil. Es kommt damit dem Zeitstempel des Network Time Protocol (NTP) gleich.
Einen Auszug der Verfügbaren Adressen und der dazugehörigen Parameter sind in Anhang 7.3 zu finden. Eine ausführlichere Liste mit den OSC-Nachrichten, welche cwonder versteht, findet sich in Anhang C der Bachelorarbeit von Malte Nogalski [Nog12].

3 Android

Die geplante Umsetzung setzt in Rahmen dieser Arbeit auf das freie und quelloffene Android-Betriebssystem der Open Handset Alliance. Android ist aktuell eines der meistverbreiteten Mobilen Betriebssysteme [Anda]. Nach Angaben des *IDC Worldwide Mobile Phone Tracker* besaßen Android-Geräte Ende 2013 einen Marktanteil von über 78% im Vergleich mit Apple iOS, Microsoft Windows Phone und BlackBerry. Im April 2013 verkündete Google, dass pro Tag über eineinhalb Millionen neue Geräte aktiviert werden, im September 2013 waren dann weltweit über einer Milliarde Geräte aktiviert. Die hohe Verbreitung und der freie Zugang zu Entwicklungsumgebungen macht es daher interessant, Anwendungen für Mobilgeräte auf Android zu entwickeln.

3.1 Entwicklungsumgebung

Android bietet durch das Android SDK eine komplett freie und vorkonfigurierte Entwicklungsumgebung an. Dazu gehört das ADT Plugin (Android Developer Tools) für die freie IDE Eclipse [Ecl], den Android Platform-tools und einem Emulator zur Ausführung und Debuggen von Anwendungen auf einem Computer. Die ADT bieten eine Reihe an Werkzeugen, welche in die Eclipse IDE integriert sind. Dazu gehört das erstellen, kompilieren, installieren und debuggen von Android Projekten und den Java und XML Editoren zur Erstellung und Bearbeitung von Programmcode und Konfigurationsdateien. Die Platform-tools bieten die Möglichkeit, verschiedene SDK-Versionen und Systemabbilder für den Emulator aus dem Internet herunterzuladen und aktuell zu halten.
Das SDK ist für Windows, Mac OS X und Linux Betriebssysteme erhältlich [Andb].

3.2 Geräte

Die während der Entwicklung verfügbaren Geräte mit Android-Betriebssystem waren ein Samsung Galaxy Note und ein Google Nexus 10. Beide Geräte liefen mit der aktuellen Android Version 4.4.3 „KitKat“. Beim Samsung Galaxy Note handelt es sich um ein im Oktober 2011 erschienenes 5,3 Zoll großes Smartphone welches von einer Samsung Exynos 4210 CPU mit

2x1.4 GHz angetrieben wird. Mit seiner Größe zählt es zu den sogenannten Smartlets (bzw. Phablet im englischen). Sie bezeichnen eine Mischform aus Smartphone und Tablet-Computern. Das Google Nexus 10 gehört zu der Klasse der Tablet-Computer und bietet mit seinem 10 Zoll großen Bildschirm und einer Samsung Exynos 5250 CPU mehr Bildschirmfläche und Prozessorleistung.
Beide Geräte bieten für die im Rahmen dieser Arbeit wichtigen Merkmale wie Multitouch-Eingaben und hochauflösende Kameras auf Rück- und Vorderseite.

(a) Google Nexus 10 (b) Samsung Galaxy Note

Abbildung 3.1: Während der Entwicklung genutzten Android Geräte

4 Augmented Reality

4.1 Idee

Die Idee hinter einer Augmented Reality (*kurz AR, zu Deutsch „erweiterte Wirklichkeit“*) besteht darin, die Wahrnehmung der realen Welt um virtuelle Elemente zu ergänzen. Sie stellt damit einen Teilgebiet der der Mensch-Maschine-Interaktion dar. Die Augmented Reality bezieht sich dabei auf die Erweiterung mehrerer Sinne. Darunter können visuelle, akustische und haptische Informationen verstanden werden. In der heutigen Forschungslandschaft wird der Begriff der Augmented Reality allerdings eher für die visuellen Informationen verwendet, so auch in dieser Arbeit.

Dabei gibt es keine genaue Definition in welchem Verhältnis die virtuellen Informationen zu den realen stehen sollen. Milgram et al. beschreiben dies über ein *Realtity-Virtuality (RV) Continuum* [MTUK95]. Dabei steht am linken Ende des Kontinuums eine Umgebung die ausschließlich aus realen Informationen besteht (zum Beispiel ein Kamerabild). Am rechten Ende steht dem eine rein virtuelle Umgebung gegenüber, welche realen Umgebungen nachempfunden sein kann aber nicht muss. Alle Punkte zwischen diesen beiden Extrema stellen eine *Mixed Reality* dar. Dabei überwiegen im linken Bereich die realen Informationen, daher wird dieser Bereich der Augmented Reality zugeteilt. Im rechten Bereich überwiegen die virtuellen Informationen, daher der Begriff Augmented Virtuality. Abbildung 4.1 skizziert diese Zusammenhänge.

Abbildung 4.1: Vereinfachte Repräsentation des RV-Kontinuums nach [MTUK95]

Gerade für mobile Endgeräte bieten sich weite Einsatzmöglichkeiten von AR-Anwendungen auf. Wo zu Beginn der Entwicklung in dem Bereich der AR noch angepasste Hard- und Software benötigt wurde, ist in den letzten Jahren eine große Anzahl von geeignetem mobilen Endgeräten verfügbar geworden. Diese Mobilgeräte verfügen über die benötigte Sensorik wie Kameras, Kompass, GPS, Gyroskop und einer ausreichenden Leistung durch schnelle Mehrkern-Prozessoren und grafischer Beschleunigung (vgl. [ABK11]). Dadurch können AR-Anwendungen kostengünstig für eine breite Masse zugänglich gemacht werden.

Kommerzielle Nutzung

Werbung in Printmedien kann durch die Überlagerung von 3D-Objekten, Videomaterial oder weiterführenden Informationen für den Kunden interessanter gestaltet werden. Das Magazin *AutoBild* bietet zum Beispiel eine AR-Anwendung für Android un iPhone um Videos, Soundbeispiele und 3D-Modelle von in ihrem Magazin vorgestellten Fahrzeugen (siehe Abbildung 4.2a).

Unterhaltung

Es gibt bereits eine Anzahl an Spielen, welche sich zur Steuerung und Generierung des Spielgeschehens eine AR zu nutze machen. Wer zum Beispiel einmal mit einem Toyota 86 über seinen Schreibtisch fahren möchte, sollte einmal einen Blick auf auf *Toyota86AR*[1] werfen (siehe Abbildung 4.2b).

Konstruktion und Wartung

Zur Erleichterung können Konstruktions- und Wartungsanleitungen bzw. die einzelnen Schritte aus diesen, für einen Techniker über eine Maschine gelegt werden. Dies kann nicht nur zu einer Zeitersparnis, sondern auch zur Vermeidung von Fehlern durch vergessen eines Vorganges führen.

Navigation

Auch in der Navigation kann eine AR hilfreich sein. Mögliche Informationen könnten Fahrlinien, Schilder und Kennzeichnungen oder Abstände zu anderen Verkehrsteilnehmern sein. Eine Umsetzung hiervon bietet *Wikitude Drive*, zu sehen in Abbildung 4.2c.

[1] http://www.toyota86ar.com/

(a) AutoBild App

(b) Toyota86AR

(c) Wikitude Drive

Abbildung 4.2: Einsatzmöglichkeiten von Augmented Reality

4.2 Aufgaben von AR

Zu den Aufgaben um eine AR zu ermöglichen gehören drei grundlegende Teilbereiche.

Identifizierung Was soll erkannt werden und welche Informationen sollen verfügbar gemacht werden

Tracking Wo befinden sich die zu suchenden Objekte (in der Regel Positionsdaten)

Visualisierung Wie werden die Objekte und dazugehörigen Informationen dem Benutzer präsentiert.

Darüber hinaus muss zwischen verschiedenen Herangehensweisen des Trackings unterschieden werden. Dabei sind die Zielsetzungen der einzelnen Methoden und deren erreichbare Qualität des Trackings vielseitig. Im Folgenden einmal einige Ansätze kurz erläutert.

Sensor basiertes Tracking

Hier wird anhand von GPS-, Kompass-, Gyroskop- und Beschleunigungssensordaten die Position und Ausrichtung des Gerätes erkannt. Große Toleranzen und Beeinflussbarkeit durch die Umwelt machen diese Art des Trackings allerdings für die in dieser Arbeit betrachteten Verfahren uninteressant.

Marker basiertes Tracking

Hier werden über die Kamera spezielle Marker erkannt. Diese Marker haben oftmals die Form eines Quadrates welches ein eindeutiges Muster zur Identifizierung und Ausrichtung besitzt. Die Funktionsweise dieses Tracking wird kurz am Beispiel der Technik, welche dem ARToolKit

[ART] zugrunde liegt, erläutert.
Das Kamerabild wird zunächst nach Bereichen abgesucht, deren Konturen denen eines Viereck ähneln. Anschließend wird, sobald ein solcher Bereich gefunden wurde, das Bild auf ein Bit Farbtiefe reduziert. Anhand der Konturen des erkannten Vierecks wird nun eine entsprechende perspektivische Transformation vorgenommen, um die vier Seiten des Vierecks parallel mit den Bildseiten zu bekommen. Das so angepasste Bild wird dann auf die Ausmaße von Referenzfelder skaliert und mit ihnen verglichen. Wurde eine Übereinstimmung gefunden, so kann anhand der zuvor erkannten Konturen des Ausgangsbildes eine Ebene aufgespannt werden, welche zusammen mit angegebenen „realen“ Abmessungen der Marker dann auf die Position und Ausrichtung relativ zur Kamera schließen lässt.

Natural Feature Tracking

Die Kamera wird hier dazu verwendet, Merkmale von im voraus analysierten Bilder zu erkennen. Hierfür können Alltagsgegenstände wie ein Bild oder eine Postkarte verwendet werden. Als Beispiel wie diese Features aussehen können, ist in Abbildung 4.3 das Ergebniss der Bildanalyse des Vuforia SDKs für des in Kapitel 6.4 wichtige ImageTarget zu sehen.

(a) Ausgangs Bild (b) Erkannte Features

Abbildung 4.3: Bildanalyse durch das Vuforia SDK [Vuf]

Echtzeit Umgebungserkennung

Ein Tracking der Position der Kamera wird hier dadurch ermöglicht, das der Tracker eigenständig eine dreidimensionale Karte der ihm unbekannten Umgebung erstellt und sich gleichzeitig anhand der bereits gefundenen Informationen lokalisiert. Bekannte Methodiken auf diesem Gebiet sind zum Beispiel das *Simultaneous localization and mapping* (kurz SLAM) [DNC+01] und das darauf aufbauende Parallel Tracking and Mapping (kurz PTAM) [KM07] [KM09].

Weitere

Weitere Verfahren gibt zur es zum Beispiel zur Erkennung von Gesichtern und Körperhaltung, diese sind aber für diese Arbeit nicht von Relevanz.

5 Qualcom Vuforia SDK

Es gibt eine große Anzahl an fertigen Bibliotheken, welche die in Kapitel 4 aufgezählten Verfahren umsetzen. Dabei wird der aufwendige Teil der Verarbeitung des Kamerabildes dem Anwendungsentwickler abgenommen.

Durch die einfache Integration, der aktiven Entwicklung und freien Verfügbarkeit fiel die Wahl über das zu verwendende SDK auf das *Vuforia Augmented Reality SDK* von Qualcomm. Es bietet eine Software Plattform für mobile Endgeräte, welche die Erstellung von *Augmented Reality* Anwendungen vereinfachen soll. Es ist zum Zeitpunkt der Erstellung dieser Arbeit für Mobilgeräte mit Android und Apple iOS verfügbar und bietet mit Schnittstellen in C++ Entwicklungsmöglichkeiten auf nativer Ebene, über Java für Android und Objective-C für iOS Zugriff durch Plattform typische Sprachen. Darüber hinaus ist eine Erweiterung für die populäre Spiele Laufzeit- und Entwicklungsumgebung Unity[1] verfügbar.

Das *Vuforia Augmented Reality SDK* unterstützt das Marker- und Natural Feature Tracking. Die Marker werden als FrameMarker bezeichnet. Sie bieten eine einfach und universell einsetzbare Möglichkeit, eine große Anzahl von bis zu 512 an eindeutig identifizierbaren Objekten zu benutzen. Die Bilddateien für alle FrameMarker liegen dem Vuforia SDK bereits bei. Für das Natural Feature Tracking wird das sogenannte ImageTarget verwendet, welches sich unauffällig und neutral in die Umgebung integrieren lässt. Hierfür können beliebige Bilddaten bequem über das Onlineportal des Vuforia SDKs hochgeladen, analysiert und als Datensatz für die Integration in die Anwendung heruntergeladen werden. Dabei sollte darauf geachtet werden, dass das Bild ausreichend kontrastreiche Details für ein zuverlässiges Tracking aufweist. Die erreichbare Qualität wird einem nach der Analyse im Onlineportal in Form einer 0 bis 5 Sterne Bewertung angezeigt.

[1] http://unity3d.com/

Das Vuforia SDK bietet auch noch weitere, für diese Arbeit aber uninteressante Targets wie zylindrische Objekte oder die Erkennung von Text an. Beispiele sind in Abbildung 5.1 zu sehen.

Mit der Version 2.8 des Vuforia SDKs kam ein weiteres, für diese Arbeit interessantes Feature hinzu. Das als *Extended Tracking* bezeichnete Verfahren erlaubt es ein stabiles Tracking von Image Targets, auch wenn sich diese aus dem Sichtfeld der Kamera bewegen. Da es sich bei Vuforia trotz der freien Verfügbarkeit um ein kommerzielles Produkt handelt, welches nicht quelloffen ist, sind genaue Informationen über die verwendeten Algorithmen nicht verfügbar. Es liegt allerdings nahe das hier Algorithmen zum Erkennen und Festhalten von Features der Umgebung zum Einsatz kommen[2].

(a) Image Target (b) Frame Marker (c) Cylinder Targets

Abbildung 5.1: Beispiele von Qualcomm Vuforia *Targets*

Pose Matrix

Die Daten der Orientierung und Position eines jeden Targets, wird in einer 4x4 Werte großen Matrix bereit gestellt. Die Richtungsvektoren der einzelnen Achsen und der Positionsvektor können dabei direkt aus dieser Matrix ausgelesen werden:

$$PoseMatrix = \begin{pmatrix} X_{Right} & X_{Up} & X_{Forward} & X_{Pos} \\ Z_{Right} & Y_{Up} & Y_{Forward} & Y_{Pos} \\ Y_{Right} & Z_{Up} & Z_{Forward} & Z_{Pos} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[2]Es wurde im Entwicklerforum von Vuforia auf SLAM/PTAM [KM07] verwiesen

6 Visualisierung und Steuerung

Für das Ziel, die Quellen der WFS-Anlage mit einem Android Mobilgerät zu visualisieren und zu steuern, machen wir uns die offene Architektur von swonder zu nutze. Die Möglichkeit Steuerbefehle und Benachrichtigungen über Änderungen, in Form von OSC-Nachrichten mit externen Anwendungen auszutauschen, erlaubt vielseitige Interaktionsmöglichkeiten (siehe Kapitel 2.4.2). Die Grundlage der hier vorgestellten Methoden bildet also die Kommunikation zwischen dem Mobilgerät und der WFS-Anlage.

6.1 Grundbausteine

6.1.1 Hauptbildschirm

Beim Öffnen der zu entwickelnden Android Anwendung muss zunächst die Zieladresse für die OSC-Nachrichten angegeben werden. Durch eine Bestätigung der Verbinden-Schaltfläche wird versucht sich mit der WFS-Anlage zu verbinden. Aus- und eingehende Nachrichten werden dabei in einer Liste angezeigt. Am unteren Bildschirmrand befinden sich die Schaltflächen zum starten der einzelnen Anwendungsteile, welche im Folgenden beschrieben werden. Das Layout ist in Abbildung 6.1 zu sehen.

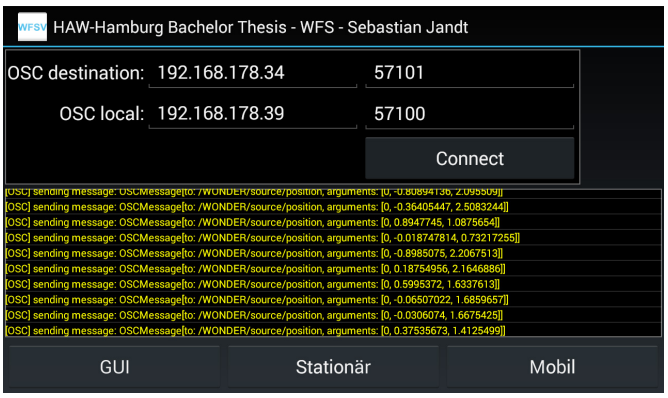


Abbildung 6.1: Hauptbildschirm der Android Anwendung

6.1.2 OSC Interface

Für die Kommunikation über Open Sound Control wird die Java-Bibliothek JavaOSC von *Illposed Software* verwendet [IS]. Für die Nutzung mit Android mussten allerdings kleine Änderungen vorgenommen werden. Android erlaubt es zum Beispiel nicht, das Netzwerk-Kommunikation im Haupt-Thread der Anwendung ausgeführt wird. Da für die Empfangsrichtung bereits ein eigener Thread genutzt wird, musste nur die Senderichtung angepasst werden. Beim Aufruf zum Senden einer OSC-Nachricht wird nun ein AsyncTask erstellt, welcher dann asynchron zum Haupt-Thread die OSC-Nachricht durch die ursprüngliche Methode versendet.

SourceStorage

Um die Eigenschaften aller WFS-Quellen von einem zentralen Punkt zugänglich zu machen, gibt es diese eine Liste mit allen Quellen. Diese Klasse kann sich bei der WFS-Anlage am *render* und *visual* Stream anmelden (siehe Anhang 7.3), um über Änderungen der Eigenschaften der Quellen informiert zu werden. Hierfür werden sogenannte OSCListener, welche von JavaOSC bereitgestellt werden, für entsprechende OSC Adressen beim Empfänger registriert. Die SourceStorage folgt dabei dem Entwurfsmuster eines Singleton, es gibt also immer nur eine Instanz. Diese kann über eine Zugriffsmethode erreicht werden kann.

Source

Jede Instanz dieser Klasse repräsentiert eine WFS-Quelle und ist für die Speicherung ihrer Eigenschaften zuständig. Sie werden von der SourceStorage erstellt und sind nur über diese zugänglich. Neben der Speicherung der Eigenschaften läuft auch das Setzen und Senden von Änderungen über die Source. So gibt es für jede Eigenschaft eine dazugehörige Methode wie activate(), angle(), position(), ….

SourceChangedListener

Um über Änderungen der Eigenschaften von Quellen benachrichtigt zu werden, bietet die Source-Klasse die Möglichkeit, einen SourceChangedListener zu registrieren. Dieser bietet die Methode *sourceChanged* welche bei Änderungen aufgerufen wird und zusätzlich die betroffene Source mitliefert.

SpamPreventer

Um bei schnellen Änderungen ein zu hohes Netzwerkaufkommen zu verhindern, gibt es die Klasse SpamPreventer. Sie stellt sicher, dass eine gewisse Anzahl an Nachrichten pro Sekunde

nicht überschritten wird. Verwendet wird diese Klasse von der Source bei Änderungen in Winkel und Position.

6.1.3 Router

Da das interne Netzwerk der WFS-Anlage, wie in Kapitel 2.4.1 beschrieben, nicht von außen zugänglich ist, musste eine kleine Anwendung geschrieben werden, welche die Nachrichten von außen in das Netzwerk der WFS weiterleitet. Hierzu wurde das Utility *Router* erstellt. Es listet in einem kleinen Benutzerinterface die verfügbaren Netzwerkschnittstellen auf, welche dann vom Nutzer entsprechend ausgewählt werden. Nach Bestätigung durch die Start-Schaltfläche werden dann, ebenfalls unter Nutzung von *JavaOSC* eintreffenden OSC-Nachrichten zwischen den beiden ausgewählten Netzwerken ausgetauscht.

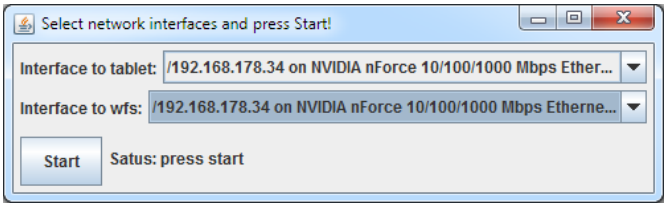


Abbildung 6.2: Router

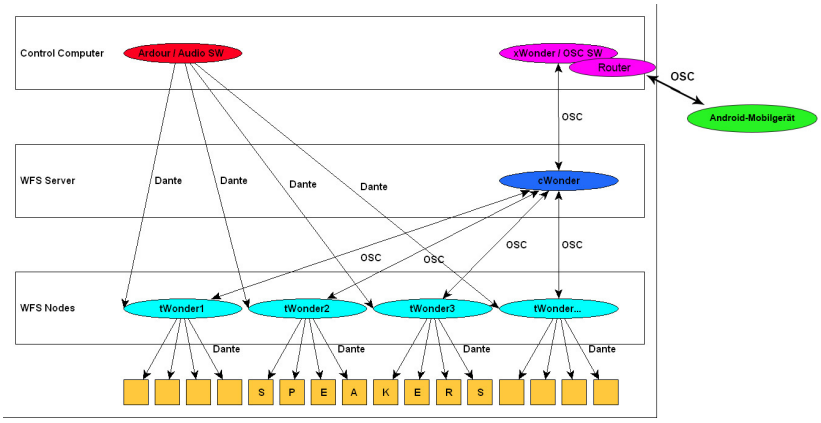


Abbildung 6.3: Zugang zum WFS internen Netzwerk über den Control Computer (WFS-Mac). Die Router Anwendung wird dabei als OSC Software eingesetzt.

6.1.4 Logging

Für eine Übersicht über die aktuelle Kommunikation mit der WFS-Anlage und weiteren hilfreichen Informationen gibt es ein Nachrichten-Log, welcher im Starter angezeigt wird.

6.1.5 Vektoren- und Matrizen- Arithmetik

Für alle Vektor und Matrizen basierten Berechnungen wird eine Portierung der *OpenGL Mathematics* Bibliothek von C++ zu Java verwendet [JGL]. Alle Klassen sind hier Immutable aufgebaut. Das heißt das die Werte der Instanzen selber nicht mehr verändert werden können und Ergebnisse von Berechnungen durch Erstellung von neuen Instanzen zurückgeliefert werden.

6.2 GUI basiert

Als Benutzerschnittstelle stellt swonder standardmäßig ein GUI namens xwonder bereit. xwonder wurde von Hans-Joachim Mond mit dem Ziel entwickelt, die Positionierung und Konfiguration der einzelnen WFS-Quellen übersichtlich und ergonomisch leicht zugänglich zu machen. Die bis dahin vorhandene GUI von Marije Baalman bot zwar umfangreiche Möglichkeiten an, welche allerdings eher für Nutzer mit detaillierten Kenntnissen auf dem Gebiet der Wellenfeldsynthese nützlich waren [Mon07].

In Anlehnung an die Oberfläche von xwonder (siehe Abbildung 2.5) wurde ein vergleichbares Layout für eine Oberfläche für die touchbasierte Eingabe entwickelt.
Es besteht dabei aus zwei getrennten Bereichen, wie in Abbildung 6.4 zu sehen. Zum einen gibt es eine Liste mit den verfügbaren Quellen, der SourceList, und deren Eigenschaften welche in SourceControl Blöcken zusammengefasst sind. Zum anderen eine grafische Darstellung der Positionen der Quellen in Relation zu den realen Ausmaßen der WFS-Anlage, der Overview. Für die Darstellung wurde eine zwei dimensionale Draufsicht von oben auf die Anlage gewählt.

Über die SourceListe sollen Eigenschaften zugänglich sein, welche rein informativer Natur sind. Dazu gehören die ID der Quelle, ihres Namens und ihrer Farbe für die Darstellung, ob die Quelle eine Punkt- oder Linearschallquelle darstellen soll und gegebenenfalls den dazugehörigen Winkel. Dabei werden bei deaktivierten Quellen alle Eigenschaften außer der ID ausgeblendet. Dies nimmt dem Benutzer die Möglichkeit, Eigenschaften welche aktuell keinerlei Auswirkungen auf den Zustand der WFS-Anlage haben zu verändern. Nebenbei verschafft

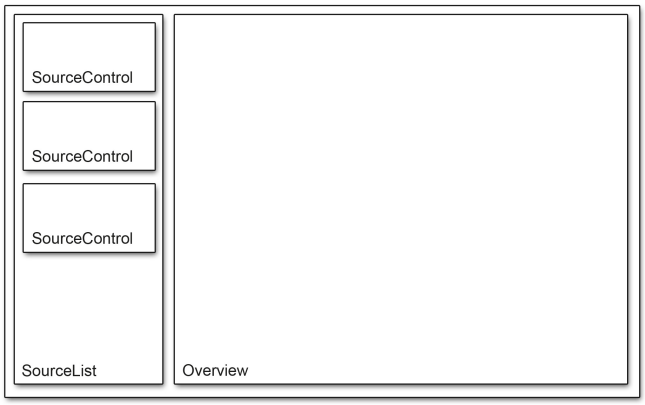


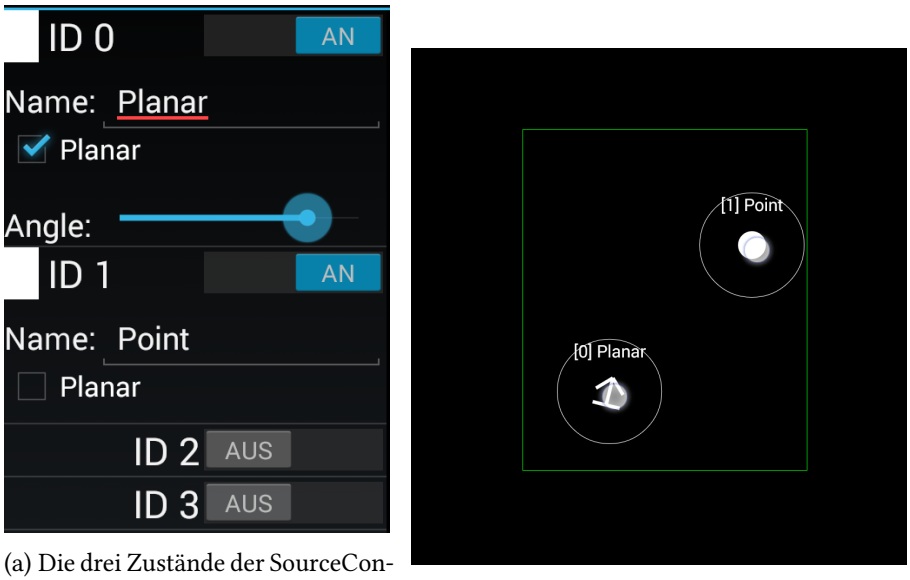
Abbildung 6.4: GUI - Übersicht

dies einen besseren Fokus auf die Eigenschaften der aktiven Quellen. Wie die drei verschiedenen Zustände (aktiviert als Linearschallquelle, aktiviert als Punktschallquelle, deaktiviert) der SourceControl Blöcke aussehen, ist in Abbildung 6.5a zu sehen.

Innerhalb der Overview sollen, ähnlich wie bei xwonder, die physikalischen Lautsprecher-Arrays als Linien, Punktschallquellen als Kreis und Linearschallquellen als Pfeile dargestellt werden. Die Quellen selbst sollen dabei farblich hervorgehoben und mit Zusatzinformationen wie der ID und des vergebenen Namens identifizierbar sein. Dem Benutzer soll die Positionsänderung der Quellen über touchbasierte Gesten ermöglicht werden. Dabei wird bei Berührung überprüft, ob im Bereich der Berührung eine aktive Quelle zu finden ist. Wird eine Quelle gefunden, so wird ein Kreis um die Quelle angezeigt um die Quelle während der Berührung, unter dem Finger des Benutzers, weiterhin sichtbar zu machen. Mehrere Quellen sollen so durch Berührung mit mehreren Fingern parallel verschoben werden können. Diese Darstellungsform bietet gegenüber einer rein formular- oder textbasierten Eingabe den Vorteil, dass der Benutzer intuitiv und ohne Umwege über eine Bildschirmtastatur, einen Bezug zwischen den angezeigten Positionen auf dem Mobilgerät und der Positionen der Quellen im Raum der WFS herstellen kann. Ein Beispiel wie die Overview während der Berührung beider aktivierter Quellen aus Abbildung 6.5a aussehen kann, ist in Abbildung 6.5b zu sehen.

6.2.1 Umsetzung

Die Umsetzung auf Android nutzt zur Aufteilung in SourceList und Overview die sogenannten Fragments. Fragments wurden dafür Entwickelt, größere Teile einer Anwendung, im besonderen Teile der Benutzeroberfläche, in separate Bausteine aufteilen zu können. Ein Vorteil



(a) Die drei Zustände der SourceControl

(b) Dazugehöriges Aussehen der Overview

Abbildung 6.5: GUI - Aussehen

hiervon ist, dass bei der recht großen Anzahl an verfügbaren Geräten und Bildschirmgrößen, große Teile einer Anwendung nur einmal geschrieben werden müssen, um diese dann passend für die Bildschirmgröße alleine oder mit anderen Fragmenten zusammen kombiniert genutzt werden können.

SourceList und SourceControl

Die SourceList erweitert hier ein ListFragment. Die Klasse ListFragment bietet, wie der Name bereits vermuten lässt, die Funktionalität eine Liste zu verwalten. Um die SourceList mit Inhalt zu füllen gibt es den SourcesAdapter. Der SourcesAdapter gibt der SourceList Auskunft über Anzahl der anzuzeigenden Elemente und gibt darüber hinaus die Klassen, welche in der SourceList angezeigt werden sollen, an diese zurück. Hier sind es die SourceControl Klassen, welche wiederum Standardklassen des Android Betriebssystems für die Darstellung und Eingabe von Eigenschaften der dazugehörige WFS-Quelle nutzen (siehe Abbildung 6.5a). Jede SourceControl Instanz registriert bei ihrer Erzeugung dabei eine Reihe an Listenern, welche auf die Änderungen durch den Nutzer reagieren, um die Parameter sofort an die Quellen der WFS weiter zu geben. Die SourceControl implementiert darüber hinaus einen SourceChangedListener, der bei der dazugehörigen Source registriert wird, um über eintreffende Parameteränderungen durch das WFS-System informiert zu werden (siehe Kapitel 6.1.2).

Overview

Die Overview selbst stellt nur einen Container für die OverviewView dar. Die OverviewView ist der Hauptbestandteil der GUI. Sie bietet die von xwonder her bekannte Draufsicht auf das WFS-System mit den Positionen der WFS-Quellen. Sie kümmert sich um die Weitergabe von Touch-Eingaben an den MultitouchSourceTracker und um das eigentliche Anzeigen der Quellen. Für das benötigte Umrechnen zwischen den Koordinatensystemen des Bildschirms und der WFS dient die WorldScreenTransformation Klasse.

MultitouchSourceTracker

Der MultitouchSourceTracker erhält alle Touch-Eingaben des Benutzers auf der Overview. Er unterscheidet dabei zwischen dem Anfang, der Bewegung und dem Endes einer Berührung und gibt basierend darauf Positionsänderungen an die WFS-Anlage weiter. Bei Multitouch-Eingaben wird zwischen den einzelnen Berührungen unterschieden. Bei Anfang einer Berührung wird zunächst über die WorldScreenTransformation im SourceStorage überprüft, ob sich eine WFS-Quelle nahe der Berührung befindet. Wurde eine Quelle gefunden so wird sich eine Referenz der Quelle zusammen mit der ID der Berührung gemerkt. Bei einer Bewegung wird zunächst geprüft, ob sich zu der ID der Berührung eine Quelle gemerkt wurde. Ist eine Quelle bekannt, so wird die neue Position (umgerechnet durch die WorldScreenTransformation) an die WFS-Anlage gesendet. Beim Ende der Berührung wird die Referenz zur Quelle, welche zusammen mit der ID der Berührung gemerkt wurde, wieder gelöscht.

WorldScreenTransformation

Die WorldScreenTransformation Klasse ist für das Umrechnen von Bildschirm zu WFS Koordinaten und umgekehrt zuständig. Sie generiert basierend auf der Größe der Overview, der Ausmaße der WFS-Anlage und einem einzustellenden Abstand zwischen Bildschirmrand und Rand der WFS-Anlage die Skalierung und Transformation der Koordinaten. Die Größen sind in Abbildung 6.6 skizziert.

Für die Skalierung wird zunächst geprüft, in welcher Richtung das kleinere Verhältnis benötigt wird.

$$Bounds = WFS + 2 * Border$$

$$Scale = min\left(\frac{Breite_{Overview}}{Breite_{Bounds}}, \frac{Höhe_{Overview}}{Höhe_{Bounds}}\right)$$

Zur Bestimmung der Verschiebung wird anschließend der Mittelpunkt von Overview und Bounds berechnet. Vom Bounds-Mittelpunkt wird dann der Ausgangspunkt der WFS (Origin)

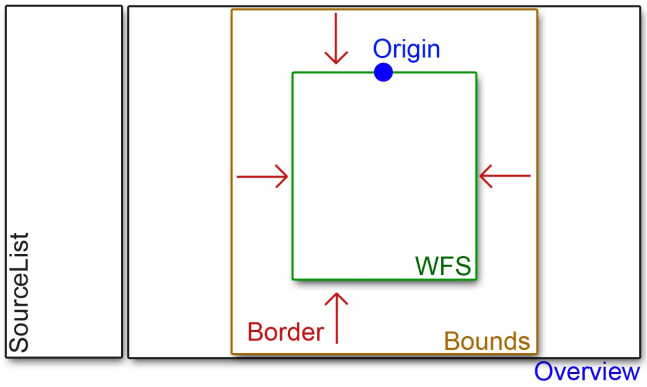


Abbildung 6.6: Berechnung der Transformation zwischen WFS- und Bildschirmkoordinaten

und der gewünschte Rand abgezogen und mit der Skalierung multipliziert. Dieses Ergebnis wird wiederum vom Mittelpunkt der Overview subtrahiert.

$$Center_{Overview} = \frac{Overview}{2}$$

$$Center_{Bounds} = \frac{Bounds}{2}$$

$$Offset = (Center_{Bounds} - Origin - Border) * Scale$$

$$Translation = Center_{Overview} - Offset$$

Nun kann zwischen beiden Koordinatensystemen konvertiert werden:

$$Pos_{Screen} = (Pos_{World} * Scale) + Translation$$

$$Pos_{World} = (Pos_{Screen} - Translation) / Scale$$

6.3 Marker als Quelle

Um einen Einstieg in das Kamera-basierte Tracking von Markern im Zusammenspiel mit der WFS-Anlage zu erlangen, wurde zunächst der Ansatz verfolgt, die Kamera des Android Gerätes zur Erkennung von Markern im Raum der WFS zu verwenden und die Positionsdaten anschließend zur Platzierung von Quellen an die WFS-Anlage weiter zu leiten. Das Mobilgerät mit der Kamera wurde hierzu an den Ausgangspunkt des Koordinatensystems der WFS-Anlage platziert.

6.3.1 Aufbau

Als Tracker kommt hier der „MarkerTracker“ des Vuforia SDKs zum Einsatz, welcher die sogenannten FrameMarker erkennen und deren Position und Orientierung erfassen kann. Um eine möglichst präzise Positionsangabe zu erzeugen, müssen die Maße der Marker eingemessen und während der Initialisierung der Marker an den Tracker weitergegeben werden. Für die Tests im Labor der HAW wurden zum Beispiel die Marker auf DIN A4 ausgedruckt und hatten eine reelle Größe von 14cm in der Breite sowie in der Höhe.

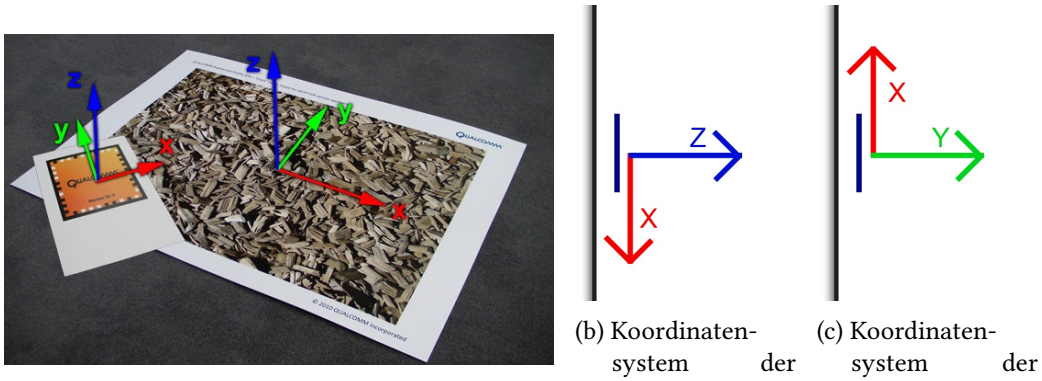
6.3.2 Umrechnung der Koordinaten

Ähnlich wie bei der GUI gibt es auch hier wieder zwei verschiedene Koordinatensysteme. Die Koordinaten welche vom Tracking zurückgeliefert werden, befinden sich in einem drei Dimensionalen Raum. Da die WFS-Anlage nur mit Positionsdaten auf der Ebene, also nur in zwei Dimensionen arbeitet, müssen wir diese umrechnen.

Gewählt wurde ein vergleichsweise einfacher Weg. Die Kamera des Android Gerätes, welche für das Tracking verwendet wird, wird an der selben Stelle platziert, wo für die WFS-Anlage der Ausgangspunkt des Koordinatensystems liegt. Anschließend muss sie in der Richtung der Y-Achse der WFS-Anlage ausgerichtet werden, sodass sie in den Raum hinein sieht. Bei dieser Ausrichtung können die X- und Z-Koordinaten der PoseMatrix (siehe 5) des Markers für die Position der WFS-Quellen verwendet werden. Die Achsen der Marker sind zur Veranschaulichung in Abbildung 6.7a zu sehen. Dabei muss darauf geachtet werden, mit welcher Einheit die Größe der Marker initialisiert wurde. Bei einer Initialisierung mit Werten in Zentimetern müssen die Koordinaten entsprechend in Meter, welche die WFS-Anlage nutzt, umgerechnet werden. Bei Betrachtung der Achsen der Marker fällt auf, dass wenn ein Marker in Richtung der Kamera gehalten wird, die X-Koordinate in die verkehrte Richtung zeigt. Diese muss also für die WFS-Anlage invertiert werden (siehe Abbildungen 6.7b und 6.7c).

6.3.3 Probleme

Ein Nachteil dieser Methode ist, dass das Tracking der Marker nur in einem eingeschränkten Bereich vor der Kamera verwendbare Ergebnisse liefert. Dies machte sich beim Testen des Systems bemerkbar. Der Bereich vor der Kamera konnte dabei in drei Kategorien unterteilt werden. Die Marker, welche innerhalb des Sichtwinkels der Kamera und maximal in etwa 4 Meter Entfernung gehalten wurden, konnten direkt Erkannt und bei Bewegung verfolgt werden. Beim erhöhen der Entfernung der Marker zur Kamera blieben diese weiterhin im



(a) Achsen der Marker [Vuf]

(b) Koordinatensystem der Marker

(c) Koordinatensystem der WFS

Abbildung 6.7: Marker Koordinaten

Tracking. Bei schnelleren Bewegungen ging das Tracking dann allerdings verloren. Neue Marker konnten in diesem Bereich dann nicht mehr erkannt werden. Im Bereich außerhalb des Sichtwinkels der Kamera können logischerweise keine Marker erkannt werden. Die Bereiche sind in Abbildung 6.8 skizziert.

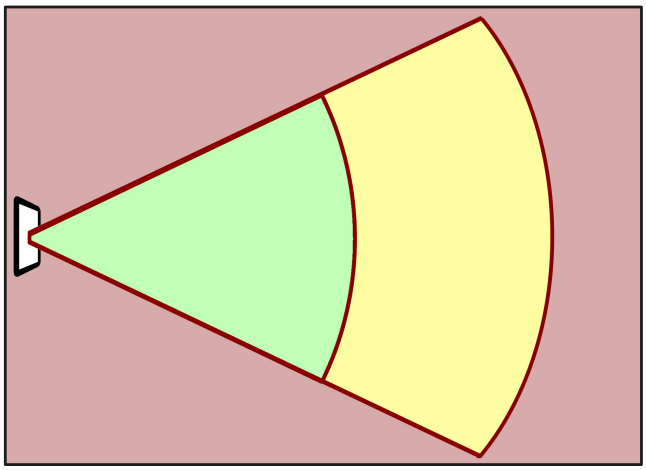


Abbildung 6.8: Stationäre Tracking-Bereiche: Erkennung und Tracking (grün), nur Tracking (gelb), unmöglich (rot)

6.3.4 Andere Systeme

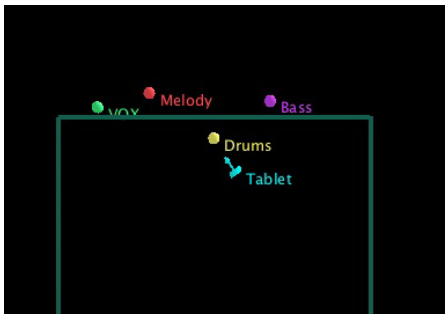
Die in diesem Kapitel vorgestellte Methode zur Positionsänderung von WFS-Quellen ist ebenfalls mit anderen Systemen möglich. Das Bild kann auch von anderen stationären Kameras, die nicht in ein Mobilgeräte eingebaut sind, stammen. Auch die Art des Tracking Systems

und der Marker kann variiert werden. Ein gutes Beispiel hierzu ist das ebenfalls im Labor der HAW verfügbare Motion-Tracking System der Firma Advanced Realtime Tracking GmbH, den ART-Tracker. Er verfügt über sechs Infrarotempfindliche Kameras die zueinander Kalibriert sind, um aktive und passive Marker, welche selber Infrarotlicht senden oder reflektieren, im dreidimensionalen Raum zu lokalisieren. Es dient zum Beispiel für die Gestensteuerung von WFS-Quellen welche von Malte Nogalski für seine Bachelorarbeit erstellt wurde (siehe [Nog12]).

6.4 Marker als Referenz

Das interessantere Konzept ist die Steuerung und Visualisierung der WFS-Quelle über eine Augmented Reality Umgebung. Dabei soll ein Mobilgerät in den Händen des Benutzers zu halten sein, auf dessen Bildschirm die Quellen, durch 3D-Objekte repräsentiert, mit der realen Umgebung verschmelzen. Die Position, im Fall von Linearschallquellen zusätzlich die Rotation, der Quellen soll dabei durch touchbasierte Gesten auf dem Bildschirm zu steuern sein. Das Mobilgerät soll frei im Raum bewegt werden können und dabei seine eigene Position und Orientierung erkennen. Zusammengefasst soll also dem Benutzer das Gerät als ein Fenster zum betrachten der Quellen der WFS-Anlage dienen. Abbildung 6.9a zeigt was auf dem Bildschirm des Mobilgerätes zu sehen sein soll. Dazu passend ist ein Ausschnitt von xwonder mit den Quellen und einer Linearschallquelle, welche als Visualisierung der Position des Mobilgerätes dient, in Abbildung 6.9b zu sehen.

(a) Augmented Reality Ansicht auf dem Mobilgerät



(b) xwonder Ansicht

Abbildung 6.9: Beispiel der Augmentierung

6.4.1 Positionsermittlung des Gerätes

Für die Ermittlung der eigenen Position und Orientierung des Mobilgerätes wird ein Image-Target des Vuforia SDKs verwendet. Dafür wurde ein Bild der Skyline von New York City analysiert und dem „ImageTracker“ als Marker übergeben. Das Bild wurde dann am Ausgangspunkt des Koordinatensystems der WFS-Anlage auf Haltehöhe des Mobilgerätes aufgehängt (siehe Abbildung 2.3).

Sobald dieses ImageTarget vom Tracker erkannt wird, muss die zurückgelieferte PoseMatrix invertiert werden, da in diesem Fall nicht die Position des Targets relativ zur Kamera, sondern die Position der Kamera relativ zum Target gebraucht wird. Aus dieser invertierten PoseMatrix können dann die Position und die Ausrichtung des Mobilgerätes ausgelesen werden (siehe 5). Zur Verwendung mit der WFS-Anlage muss der Positionsvektor noch mit der realen Breite des ImageTargets multipliziert werden. Bei dem ImageTarget, welches für die Entwicklung im Labor der HAW genutzt wurde, beträgt dies 57,4 Zentimeter. Da das Koordinatensystem der WFS-Anlage in Meter angegeben ist muss auch hier wieder umgerechnet werden.

$$P = Pos * Breite_{ImageTarget}/100$$

Um Die Position des Mobilgerätes für swonder bekannt zu machen, um es zum Beispiel auf xwonder zu betrachten, können ähnlich wie bei der Umrechnung in Kapitel 6.3 die X- und Z-Koordinaten für die Position verwendet werden.

$$P_{WFS} = \begin{pmatrix} P_X \\ P_Z \end{pmatrix}$$

Um auch die Ausrichtung des Mobilgerätes bekannt zu machen, wird aus den X- und Z-Werten des Forward-Vektors der PoseMatrix der Arkustangens gebildet.

$$\theta = \operatorname{atan2}\left(P_Z, P_X\right)$$

Extended Tracking

Um Quellen zu visualisieren und zu steuern auch wenn das ImageTarget nicht im Kamera-Sichtfeld des Mobilgeräts liegt, wird für dieses das „Extended Tracking“ aktiviert. Dabei versucht der ImageTracker des Vuforia SDKs sich anhand der Umgebung zu orientieren. Das ImageTarget dient dabei als Ausgangspunk.

Allerdings ist die Zuverlässigkeit dieses Trackings stark von den Lichtverhältnissen und den verfügbaren, kontrastreichen Merkmalen der Umgebung abhängig. Außerdem dürfen bereits gefundene Merkmale sich nicht mehr bewegen da sonst der Bezug zum Ausgangspunkt nicht mehr funktioniert. Hier muss dann das Tracking erneut gestartet werden. Tests im Labor mit der WFS an der HAW-Hamburg zeigten, das der Raum ausreichend Merkmale für ein zuverlässiges Tracking bietet.

6.4.2 Gesten

Zur Steuerung der WFS-Quelle werden im Folgenden ein paar touchbasierte Gesten vorgestellt. Dabei wurde auf Gesten, wo mehr als ein Finger benötigt wird verzichtet. Dies soll dadurch begründet werden, dass das sichere Halten des Mobilgerätes, insbesondere bei größeren Geräten wie einem Tablet, mit nur einer Hand gewisse Schwierigkeiten bereiten kann.

Auswählen und halten

Um eine Quelle in ihrer Position oder bei Linearschallquellen in ihrer Richtung zu steuern, muss diese zunächst ausgewählt werden. Die Suche nach einer Quelle wird durch die Berührung an beliebiger Stelle auf dem Bildschirm gestartet. Dabei wird, solange noch keine Quelle erfolgreich ausgewählt wurde, der Unterschied zwischen dem Winkel der Ausrichtung des Mobilgerätes und dem Winkel aller aktiven Quellen zum Mobilgerät verglichen. Sobald eine oder mehrere Quellen einen gewissen Winkelunterschied unterschreiten, wird die Quelle mit dem geringsten Abstand zum Mobilgerät ausgewählt. Die drei möglichen Szenarien sind in Abbildung 6.10 skizziert.

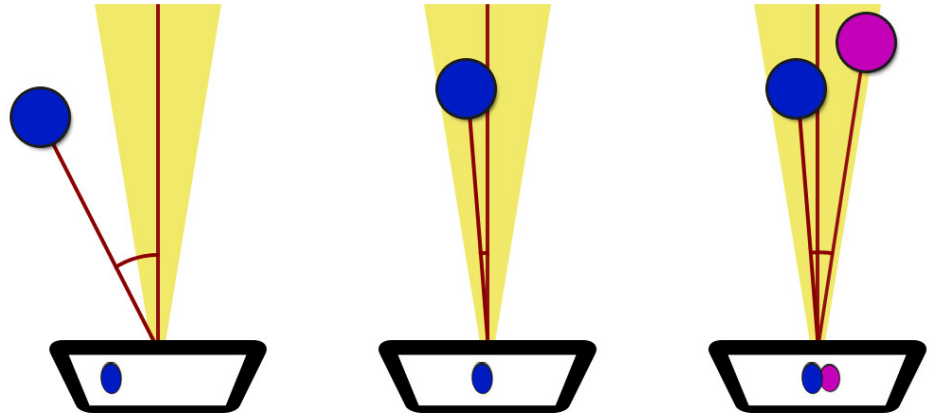
Die nun ausgewählte Quelle wird bis zum Ende der Berührung des Bildschirmes festgehalten und folgt der Bewegung des Mobilgerätes. Der Abstand der Quelle bei der Auswahl wird zwischengespeichert und zusammen mit der Position und der Ausrichtung des Gerätes zur Berechnung der neuen Position verwendet.

$$Dir_{Gerät} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

$$Pos_{Quelle} = Pos_{Gerät} + Dir_{Gerät} * Abstand$$

Entfernungsänderung

Zur Veränderung der Entfernung zwischen festgehaltener Quelle und des Mobilgerätes wurde eine Geste gewählt, bei der Punkt der Berührung auf dem Bildschirm nach oben und unten



(a) Keine Quelle innerhalb des auswählbaren Bereiches

(b) Quelle innerhalb des auswählbaren Bereiches, Blau ausgewählt

(c) Zwei Quellen innerhalb des auswählbaren Bereiches, Blau ausgewählt

Abbildung 6.10: 3D Modelle der Quellen

verschoben wird. Dabei wird der minimale und maximale Abstand auf einen Bereich zwischen 25 cm und 10 m limitiert um die Quellen in Sichtweite de Nutzers zu behalten.

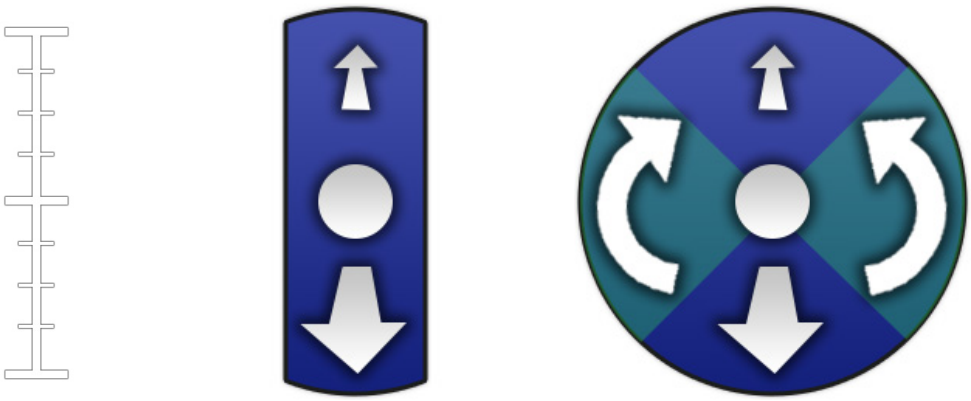
Rotation

Zur Rotation von Linearschallquellen gibt es eine Geste, wo der Punkt der Berührung nach links und rechts verschoben wird. Je weiter der Abstand zur ursprünglichen Berührung ist, umso schneller wird die Rotation verändert.

6.4.3 User Interface

Um dem Nutzer zu zeigen, in welchem Bereich Quellen ausgewählt werden können, wird in der Mitte des Bildschirmes eine Auswahlhilfe, welche einem Fadenkreuz ähnelt, angezeigt (siehe Abbildung 6.11a).

Zusätzlich wird in der Zeit, in der eine Quelle festgehalten wird, an der Position der Berührung ein kleines Steuerfeld eingeblendet. Dieses Steuerfeld soll dem Nutzer dabei helfen, die aktuell verfügbaren Gesten intuitiv zu Erlernen und Umzusetzen. Welches Steuerfelder angezeigt wird, hängt dabei von der Art der festgehaltenen Quelle ab. Da Punktschallquellen keine Rotation anbieten, wird hier nur das Ändern der Entfernung angezeigt (siehe Abbildung 6.11b). Für Linearschallquellen sind beide Gesten möglich und werden auch so angezeigt (siehe Abbildung 6.11c).

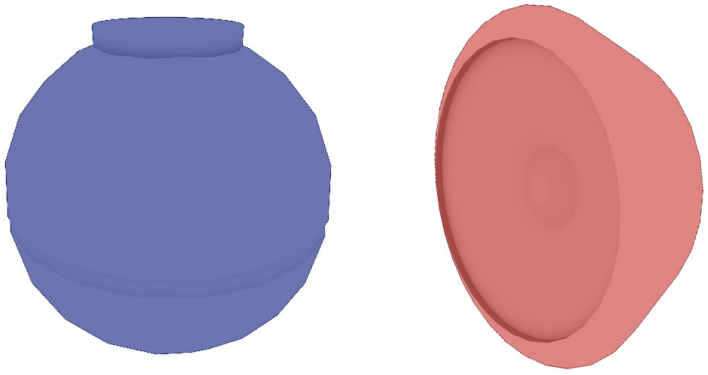


(a) Auswahlhilfe (b) Steuerfeld bei Punktschallquellen (c) Steuerfeld bei Linearschallquellen

Abbildung 6.11: User Interface für die Augmented Reality

6.4.4 3D Modelle

Die für die Visualisierung verwendeten 3D Modelle wurden mit Hilfe der freien 3D-Grafiksoftware Blender [Ble] erstellt. Dabei sollten Punkt- und Linearschallquellen eindeutig voneinander zu unterscheiden sein. Die Punktschallquelle wird als Kugel und die Linearschallquelle als gerichtetes Lautsprecher-Chassis dargestellt. Auf dem Bildschirm werden diese Objekte dann passend zu den Farbinformation der WFS-Quelle eingefärbt. Die aktuell ausgewählte Quelle wird zur Hervorhebung schwarz umrandet.



(a) Repräsentation von Punktschallquellen (b) Repräsentation von Linearschallquellen

Abbildung 6.12: 3D Modelle der Quellen

Zur Verwendung in der Anwendung wurden die Modelle im Format *Wavefront .OBJ* exportiert und in ein eigenes Modellformat mit Endung .bbmf konvertiert. Diese Konvertierung

wurde durch ein eigens dafür erstelltes Utility mit dem Namen *BBMF-Converter* durchgeführt. Das Modellformat entstand durch frühere Grafikprojekte und ist nur für nicht animierte Modelle geeignet.
Der Grund der Nutzung eines eigenen Modellformates besteht darin, dass die Datensätze direkt in den Speicher des Grafikchips geladen werden können, ohne auf dem Mobilgerät noch aufwendig konvertiert werden zu müssen.

7 Schluss

7.1 Zusammenfassung

Das Ziel dieser Arbeit, die virtuellen Quellen der WFS-Anlage mit einem Android-Gerät zu visualisieren und zu steuern wurde erfolgreich umgesetzt. Es wurden hierzu verschiedene Verfahren beschrieben und Implementiert. Bekannte Bedienkonzepte einer Standardkomponente des Wellenfeldsynthese-Systems wurde auf Android umgesetzt und für Multitouch-Eingaben optimiert. Daneben wurde untersucht, in welcher Form eine Augmented Reality Umgebung zur Visualisierung und Steuerung eingesetzt werden kann. Dabei wurden zwei Konzepte untersucht. Eines davon bietet die Möglichkeit, die Quellen der WFS-Anlage wie durch eine Art Fenster zu betrachten, in denen sie als virtuelle 3D-Objekte dem Kamerabild überlagert angezeigt werden. Hierfür wurden dann Gesten zur Bildschirmeingabe gezeigt, welche es dem Benutzer auf eine leichte Weise erlauben die Quellen in ihrer Position und Ausrichtung zu verändern.

7.2 Fazit

Gerade die Umsetzung der xwonder Anwendung auf Android und der dadurch entstandenen Möglichkeit, über Multitouch-Eingaben mehrere Quellen gleichzeitig zu verschieben, ist intuitiv und macht bei der Benutzung eine menge Spaß. Auch die Sicht auf die virtuellen Quellen durch die Augmented Reality Umgebung bietet eine interessante neue Sicht und erlaubt eine neue visuelle Möglichkeit der Steuerung.

7.3 Ausblick

Die Möglichkeiten zur Visualisierung der WFS-Quellen ist nicht nur auf abstrakte Objekte zu reduzieren. Eine weiterführende Idee wäre es, in dem Augmented Reality Bild anstelle der hier vorgestellten 3D-Objekte andere, zum Audiomaterial passende Objekte anzuzeigen. So könnte hier zum Beispiel eine 3D-Animation mit passender Tonuntermalung durch die WFS-Anlage dargestellt werden. Oder bei Tonaufnahmen mehrerer Musikinstrumente könnten diese dem

Nutzer als ein Orchester zusammen auf dem Bild gezeigt werden.

Die Positionserkennung des Mobilgerätes könnte auch für das in der Bachelorarbeit von Carola Christoffel vorgestellte Verfahren zur Verbesserung der Lokalisierung von WFS-Quellen, welche sich vor den physikalischen Lautsprechern der WFS-Anlage befinden, verwendet werden (siehe [Chr14]).

Glossar

ADT Android Developer Tools, ein Plugin für die Eclipse IDE. 12

Augmented Reality Die erweiterung der realität um virtuelle Informationen. 14, 31

Dante Das Dante Audionetzwerk der Firma Audinate [Aud] ist ein IP-basiertes Netzwerk, das die Übertragung der Audiodaten über den Layer 3 des ISO/OSI Modells umsetzt. Für die Übertragung können daher Standardnetzwerkkomponenten verwendet werden. 8, 9

FrameMarker Marker mit speziellem Muster zur Erkennung durch das Vuforia SDK. 19, 29

Head Mounted Display Ein Bildschirm, welcher an dem Kopf des Benutzers befestigt werden kann. Kommt oft mit einer Kamera daher.. 2

IDE Eine integrierte Entwicklungsumgebung (Abkürzung IDE, von englisch integrated development environment) welche der Softwareentwicklung dient.. 12

ImageTarget Marker mit frei wählbaren Bildinformationen. Teil des Natural Feature Tracking des Vuforia SDK. 17, 19, 32

Marker Objekte zur Erkennung durch eine Kamera. 2, 3, 32

MCS Mensch-Computer-Schnittstelle. 1

MIDI Musical Instrument Digital Interface. Ein Industriestandard für den Austausch digitaler Steuersignale zwischen elektronischen Instrumenten.. 10

MMI Mensch-Maschine-Interaktion. 1

MMS Mensch-Maschine-Schnittstellen. 1

Mobilgerät Ein tragbarer Computer welcher mobil eingesetzt werden kann. Vereint Stromversorgung, Hardware und Eingabegeräte in einem Gehäuse.. 2, 21, 25, 28, 31, 33, 36

Multitouch Eine Eingabemethode für Bildschirme welche mehrere Berührungspunkte gleichzeitig erlaubt. 13

Open Handset Alliance Ein Konsortium aus über 80 Firmen unter Leitung von Google, welches unter anderem das Andrid Betriebssystem zur verfügung stellt.. 12

OSC Open Sound Control. Ein Protokoll zur Übertragung von Steuerbefehlen zwischen Computern, Synthesizern und weiteren Audiogeräten über ein herkömmliches IP-Netzwerk.. 10

Panning Der Vorgang der Lautstärkeverteilung zwischen mehreren Audiokanälen.. 5

PoseMatrix 4x4 Matrix mit der Position und Ausrichtung eines Markers. 29, 32

Smartphone Ein Mobiltelefon welches mehr Computer-Funktionalität zur Verfügung stellt.. 12

Sweet Spot Der Sweet Spot beschreibt den Ort in einem Audiosystem, an dem ein optimales Hörerlebnis gewährleistet ist.. 5

swonder Das verwendete Softwarepaket für die Wellenfeldsynthese [Baa07]. 32

Tablet Ein tragbarer, flacher Computer ohne Tastatur. 13, 33

Tracker Ein Softwarebaustein zum Erkennen von Markern. 32

WFS-Anlage Wellenfeldsynthese-Anlage. 2, 6–8, 10, 21, 22, 24, 27–29, 31, 32

WFS-Mac Teil der WFS-Anlage welcher die Schnittstelle zu äußeren Systemen bildet.. 8, 10, 23, 40

WFS-Node Teil der WFS-Anlage welcher der Verarbeitung der Audiosignale dient.. 7, 8, 10

WFS-Quelle Wellenfeldsynthese-Quellen. 2, 3, 31, 33

WFS-Server Teil der WFS-Anlage welcher zur Verwaltung und Kommunikation zwischen den einzelnen Modulen des swonder Paketes dient.. 7, 10

xwonder Die GUI zur intuitiven Steuerung der WFS-Quellen, bestandteil von swonder.. 31, 32

Abbildungsverzeichnis

Literaturverzeichnis

[ABK11] Azuma, Ronald ; Billinghurst, Mark ; Klinker, Gudrun: Special Section on Mobile Augmented Reality. In: *Computers and Graphics* 35 (2011)

[Anda] *Android, the world's most popular mobile platform.* http://developer.android.com/about/, Abruf: 2014-06-10

[Andb] *Android SDK.* http://developer.android.com/sdk/, Abruf: 2014-06-10

[ART] *ARToolKit.* http://www.hitl.washington.edu/artoolkit/, Abruf: 2014-06-11

[Aud] *Audinate.* http://www.audinate.com/, Abruf: 2014-06-10

[Baa05] Baalman, Marije A. J.: Updates of the WONDER software interface for using Wave Field Synthesis. (2005)

[Baa06] Baalman, Marije A.: Reproduction of arbitrarily shaped sound sources with wave field synthesis - theory and implementation. In: *Tonmeistertagung*, 2006

[Baa07] Baalman, Marije A.: *On wave field synthesis and electro-acoustic music - state of the art.* 2007

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

[Ble] *Blender 3D.* http://www.blender.org/, Abruf: 2014-06-11

[Chr14] Christoffel, Carola: *Modifikation der Software einer Wellenfeldsyntheseanlage zur Wiedergabe fokussierter Quellen in Abhängigkeit der Zuhörerposition.* 2014

[DNC+01] Dissanayake, Gamini ; Newman, Paul M. ; Clark, Steven ; Durrant-Whyte, Hugh F. ; Csorba, M.: A solution to the simultaneous localization and map

building (SLAM) problem. In: *IEEE Transactions on Robotics and Automation* 17 (2001), S. 229–241

[Ecl] *Eclipse Project.* `http://eclipse.org/`, Abruf: 2014-06-10

[Fou] *FourAudio.* `http://www.fouraudio.com/`, Abruf: 2014-06-10

[GMMW07] Goertz, Anselm ; Makarski, Michael ; Moldrzyk, Christoph ; Weinzierl, Stefan: Entwicklung eines achtkanaligen Lautsprechermoduls fur die Wellenfeldsynthese. In: *FORTSCHRITTE DER AKUSTIK* 33 (2007), Nr. 2, S. 681

[IS] Illposed-Software: *JavaOSC.* `http://www.illposed.com/software/javaosc.html`, Abruf: 2014-05-22

[JGL] *JGLM - Java OpenGL Mathematics Library.* `http://github.com/jroyalty/jglm/`, Abruf: 2014-06-11

[KM07] Klein, Georg ; Murray, David: Parallel Tracking and Mapping for Small AR Workspaces. In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07).* Nara, Japan, November 2007

[KM09] Klein, Georg ; Murray, David: Parallel Tracking and Mapping on a Camera Phone. In: *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09).* Orlando, October 2009

[MGM+07] Moldrzyk, Christoph ; Goertz, Anselm ; Makarski, Michael ; Feistel, Stefan ; Ahnert, Wolfgang ; Weinzierl, Stefan: *Wellenfeldsynthese für einen großen Hörsaal.* 2007

[MGTK12] Makarski, Michael ; Goertz, Anselm ; Thaden, Rainer ; Kleber, Jochen: Entwicklung einer WFS-Anlage mit FIR-Entzerrung und Dante Audionetzwerk. In: *27. Tonmeistertagung.* Köln, 2012

[MLV05] Melchior, Frank ; Laubach, Tobias ; Vries, Diemer de: Authoring and User Interaction for the Production of Wave Field Synthesis Content in an Augmented Reality System. In: *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality.* Washington, DC, USA : IEEE Computer Society, 2005 (ISMAR '05). – ISBN 0–7695–2459–1, S. 48–51

[Mon07] Mond, Hans-Joachim: *Implementierung einer graphischen Benutzeroberfläche zur Steuerung eines Wellenfeldsynthesesystems.* 2007

[MTUK95] Milgram, Paul ; Takemura, Haruo ; Utsumi, Akira ; Kishino, Fumio: Augmented reality: a class of displays on the reality-virtuality continuum, 1995, S. 282–292

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

[Vuf] Vuforia, Qualcomm: *Vuforia Developer Portal.* `http://developer.vuforia.com/resources/api/main`, Abruf: 2014-06-02

[WFM03] Wright, Mathhew ; Freed, Adrian ; Momeni, Ali: OpenSound Control: State of the Art 2003. In: *New Interfaces for Musical Expression*, 2003, S. 153–159

Anhang A. OSC Nachrichten

Hier eine kurze Auflistung der in dieser Arbeit verwendeten OSC Nachrichten mit ihren zugehörigen Adressen und Parametern. Alle Adressen müssen mit dem Vorsatz */WONDER* übergeben werden!

Tabelle 1: swonder OSC Nachrichten

Adresse	Type Tag	Inhalt der Parameter
/source/activate	i	Id
/source/deactivate	i	Id
/source/type	ii	Id, Quellen Art (0: Planar, 1: Punkt)
/source/angle	if	Id, Winkel (Grad)
/source/position	iff	Id, X-Koordinate (Meter), Y-Koordinate (Meter)
/source/color	iiii	Id, Rot (0-255), Grün (0-255), Blau (0-255)
/source/name	is	Id, Name
/stream/render/connect	ss	Hostname/Ip, Port
/stream/visual/connect	ss	Hostname/Ip, Port

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

Hamburg, 12. Juni 2014 Sebastian Jandt