Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Martin Kucharczyk

Szenenvisualisierung und -steuerung für ein Wellenfeldsynthese-System mit Hilfe eines Android-Applets

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Martin Kucharczyk

Szenenvisualisierung und -steuerung für ein Wellenfeldsynthese-System mit Hilfe eines Android-Applets

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Technische 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.-Ing. Andreas Meisel

Abgegeben am 1. August 2011

Martin Kucharczyk

Thema der Bachelorarbeit

Szenenvisualisierung und -steuerung für ein Wellenfeldsynthese-System mit Hilfe eines Android-Applets

Stichworte

Wellenfeldsynthese, Android, Open Sound Control, WONDER, DTrack, perspektivische Projektion, Augmented-Reality, Kamera-Tracking, Kameratransformation

Kurzzusammenfassung

Diese Arbeit beschreibt die Entwicklung einer Android-Applet, die Szenen eines Wellenfeldsynthese-Systems mit der WONDER-Softwaresuite visualisiert und steuert. Dafür werden kartesische Koordinaten der Quellpositionen mit dem Open Sound Control-Protokoll übertragen, kartesische Koordinaten der Eigenposition über Multicast Datagramme übertragen und anschließend durch perspektivische Projektion vom Dreidimensionalem auf den Bildschirm ins Zweidimensionale transformiert. Die Quellen werden dabei auf dem Kamerabild der Frontkamera abgebildet, wodurch man von einer Augmented-Reality-Applikation spricht. Die Eigenposition erhält das Android-Gerät von der DTrack-Software, die das Kamera-Tracking-System betreibt.

Title of the paper

Scene visualization and control for a wave field synthesis system using an Android-applet

Keywords

Wavefield synthesis, Android, Open Sound Control, WONDER, DTrack, perspective projection, Augmented Reality, Camera Tracking, camera transformation

Abstract

This paper covers the development of an Android applet, which scenes of a wave field synthesis system, operated by the WONDER software suite, visualizes and controls. For this, cartesian coordinates of the source positions are transfered with the Open Sound Control protocol, cartesian coordinates of the users position are transfered via multicast datagrams and then transformed by perspective projection from three dimensions to the two-dimensional screen. The sources are overlayed on the camera image of the front camera, whereby this applet can be named an Augmented Reality applet. The Android device receives the users position from the DTrack software that operates the camera tracking system.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Vorwort

Wellenfeldsynthese, ein räumliches Audiowiedergabeverfahren, findet eine immer weitere Verbreitung. Schon heute werden große Wellenfeldsynthese-Systeme für kommerzielle Zwecke genutzt, wie z.B. bei den Linden Lichtspielen in Ilmenau (IDM), den Bregenzer Festspielen (AG) und den Seefestspielen Mörbisch (Gmb). Wellenfeldsynthese gibt ein bisher unvergleichliches Klangerlebnis, da Wellenfronten physikalisch korrekt nachgebildet werden und die Lokalisation von Soundquellen der Realität sehr nahe kommen. Bisher werden diese System linear genutzt, das Publikum kann in die Szene nicht eingreifen. Eine logische Erweiterung ist die Interaktion mit dem Wellenfeldsynthese-System. Der Zuhörer könnte einzelne Quellen umpositionieren, verstummen lassen oder weitere Quellen hinzufügen. Weiterhin ist eine Leinwand oder eine Bühne nicht immer die beste Lösung zur Visualisierung der Szenerie. Mit heute weit verbreiteten Smartphones und Tablet-Computern könnten Szenen direkt durch die Kamera auf den Bildschirm um einen herum angezeigt werden. Die Gesten zur Steuerung sind zumeist sehr vertraut und sonst auch leicht erlernbar, da nicht mehr, als die eigenen Finger zur Steuerung auf dem Bildschirm genutzt werden. Die Anwendungsfelder sind sicherlich zahlreich, so kann eine solche Anwendung für Unterhaltungszwecke genutzt werden, aber auch zur Produktion von Sounds, dank intuitiver Steuerung. Aus dieser Sicht eröffnen sich viele neue Mögichkeiten, aber auch zahlreiche Probleme, die neue Überlegungen benötigen.

1.2 Ziele

Diese Bachelor-Arbeit hat das Ziel ein Android-Applet für die Visualisierung und Steuerung von virtuellen Soundquellen einer Wellenfeldsyntheseanlage zu entwickeln. Die Wellenfeldsynthese-Anlage wird vorraussichtlich durch eine neuere Version der WON-

DER Software in Betrieb genommen und kann somit mit dem Open Sound Control Protokoll angesprochen werden. Um eine genaue Positionierung der steuernden Person im Raum zu erhalten, kommt das A.R.T Kamera-Tracking System mit der DTrack Software zum Einsatz. Der Benutzer soll durch die Frontkamera seines Androidgeräts ein reales Bild seiner Umgebung auf dem Bildschirm erhalten, das durch das Einzeichnen von virtuellen Schallquellen, wenn im Sichtfeld der Kamera, ergänzt wird. Diese Soundquellen sollen durch übliche Touch- und Multitouchgesten gesteuert werden, wie z.B. das Umpositionieren der Quellen im Raum durch Drag'n'Drop mit einem Finger oder der Anpassung der Lautstärke mit zwei zusammen- oder auseinanderziehenden Fingern.

1.3 Übersicht über die Arbeit

In dieser Arbeit wird wie folgt vorgegangen. In Kapitel 2 werden die einzelnen Teilaspekte, die es für die Erstellung der Android-Applikation analysiert und erläutert. In 2.2 wird das Wellenfeldsynthese-System vorgestellt, erläutert wie Wellenfeldsynthese funktioniert und WONDER, eine Software-Suite, für den Betrieb einer Wellenfeldsynthese vorgestellt. Das übliche Kommunikationprotokoll in Soundumgebungen ist das Open Sound Control Protokoll, das näher im Abschnitt 2.3 erläutert wird. Es wird genau auf die Spezifikation eingegangen und bereits existierende Implementierungen vorgestellt (2.3.4). Um die Soundquellen des Wellenfeldsynthese-Systems visualisieren zu können, muss die eigene Position bekannt sein. In dieser Arbeit kommt ein Kamera-Tracking-System mit der DTrack-Software zum Einsatz, dessen Funktionsweise in Abschnitt 2.4 beschrieben ist. Zum Abschluss des Kapitels soll der Begriff Augmented-Reality in Abschnitt 2.5 definiert werden und anschließend im Abschnitt 2.5.2 evaluiert werden, auf welchem Framework eine solche Android-Applikation entwickelt werden kann.

Bevor eine mögliche Implementierung in Kapitel 4 beschrieben wird, zeigt Kapitel 3 das dazugehörige Design der Software. Die Bestandteile Augmented-Reality Engine in 3.1, den Erhalt der Quellpositionen in 3.2, den Erhalt der Eigenposition in 3.3, sowie das Gesamtkonzept in Abschnitt 3.4, sollen den Aufbau der Software aufzeigen.

Kapitel 2

Analyse

In diesem Kapitel wird zunächst in Abschnitt 2.1 ein Gesamtüberblick über das System gegeben. Eine Einführung in die Wellenfeldsynthese gibt Abschnitt 2.2.1, gefolgt von der Beschreibung der eingesetzten Software zum Betrieb des Wellenfeldsynthese-Systems in Abschnitt 2.2.2. Abschnitt 2.3 gibt einen detailierten Überblick über das Open Sound Control Protokoll, das zur Kommunikation mit dem Wellenfeldsynthese-System benötigt wird. Das Kamera-Tracking-System wird in Abschnitt 2.4 beschrieben. Abschließend wird in Abschnitt 2.5 evaluiert welche Augmented-Reality Engine für diese Arbeit sinnvoll ist.

2.1 Das Gesamtsystem

Die Android Applet soll für ein Wellenfeldsynthese-System mit Hilfe eines Indoor-Tracking Systems nüztliche Informationen, sowie die Steuerung des Wellenfeldsynthese-Systems ermöglichen. Für das Wellenfeldsynthese-System wird eine neuere Version der WONDER (s. 2.2.2) Software zum Einsatz kommen. Für das Tracking-System wird die DTrack (s. 2.4) Software genutzt. Die WONDER Software nutzt das Opensoundcontrol zur Kommunikation und zur Visualisierung und Steuerung im dreidimensionalem Raum bietet sich eine Augmented-Reality Applikation an.

2.2 Das Wellenfeldsynthese-System

2.2.1 Wellenfeldsynthese allgemein

2.2.1.1 Einführung

Um zu verstehen, wofür die Applikation genutzt wird, soll im Folgenden erklärt werden, wie Wellenfeldsynthese funktioniert. Die Wellenfeldsynthese ist ein räumliches Audiowiedergabeverfahren zur Rekonstruktion des Orginalschallfeldes. Die akustische Lokalisation ist nicht

von psychoakustischen Effekten wie der Phantomschallquellenbildung abhängig, im Gegensatz zu konventionellen, kanalorientieren Audiowiedergabeverfahren. Eine größere Lautsprecheranordnung emuliert eine Vielzahl von Elementarwellen um so das Orginalschallfeld physikalisch korrekt zu rekonstruieren. Dazu werden rings um den Zuhörer aufgestellte, einzeln steuerbare, Lautsprechermembranen exakt in dem Moment ausgelenkt, wenn die Wellenfront einer virtuellen Schallquelle ihren Raumpunkt durchlaufen würde. Die entstehende Wellenfront ist durch eine ausreichend große Anzahl solcher Elementarwellen physikalisch nicht mehr von der realen Wellenfront zu unterscheiden. (Hel)

2.2.1.2 Mathematische Grundlagen

Mathematische Basis ist das Kirchhoff-Helmholtz-Integral, das beschreibt, dass der Schalldruck in jedem Punkt innerhalb eines quellfreien Volumens bestimmt ist, wenn Schalldruck und Schallschnelle in allen Punkten seiner Oberfläche determiniert sind. Der Vollständigkeit halb sei es nachfolgend genannt, aber nicht näher erläutert:

$$P(w,z) = \int \int_{dA} (G(w,z|z') \frac{\delta}{\delta_n} P(w,z') - P(w,z') \frac{\delta}{\delta_n} G(,z|z'))) dz'$$

Nach Rayleigh II ist der Schalldruck innerhalb eines Halbraumes auch schon bestimmt, wenn nur die Druckverteilung auf einer Ebene bekannt ist, da der Vektor der Schallschnelle dann durch die Superposition der Elementarwellen rekonstruiert werden. (Hel)

2.2.1.3 Physikalische Grundlagen

Virtuelle Schallquellen Für eine räumlich korrekte Reproduktion der Aufnahme darf die wahrgenommene Position der Schallquelle nicht von der Zuhörerposition abhängen.

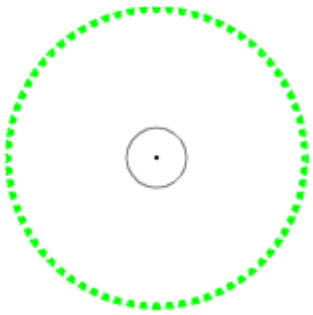


Abbildung 2.1: schwarz: Schallquelle blau: reale Wellenfront grün: Lautsprecher

Bei einer kugelförmigen Lautsprecheranordnung (s. 2.1) und einer Quelle im Zentrum dieser Anordnung, ist leicht einzusehen, dass der Ausgangspunkt der Wellenfront immer im

Zentrum der Anordnung wahrgenommen wird, völlig unabhängig von der Zuhörerposition (s. 2.2).

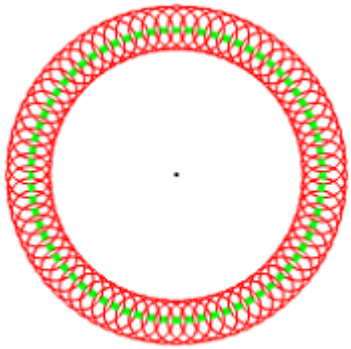


Abbildung 2.2: schwarz: Schallquelle grün: Lautsprecher rot: reproduzierte Wellenfront

Durch die Wellenfeldsynthese können solche virtuellen Schallquellen auch in ebenen Lautsprecheranordnungen erzeugt werden. Christiaan Huygen entdeckte, das jeder Punkt einer Wellenfront Ausgangspunkt einer Elementarwelle ist. Nach diesem Prinzip hatte Prof. Berkhout an der TU Delft 1987 ein Verfahren entwickelt, das jede einzelne um den Zuhörer angeordnete Lautsprechermembran genau in dem Moment bewegt, wenn die eigentliche Wellenfront die virtuelle Punktquelle erreichen würde. (de.b)

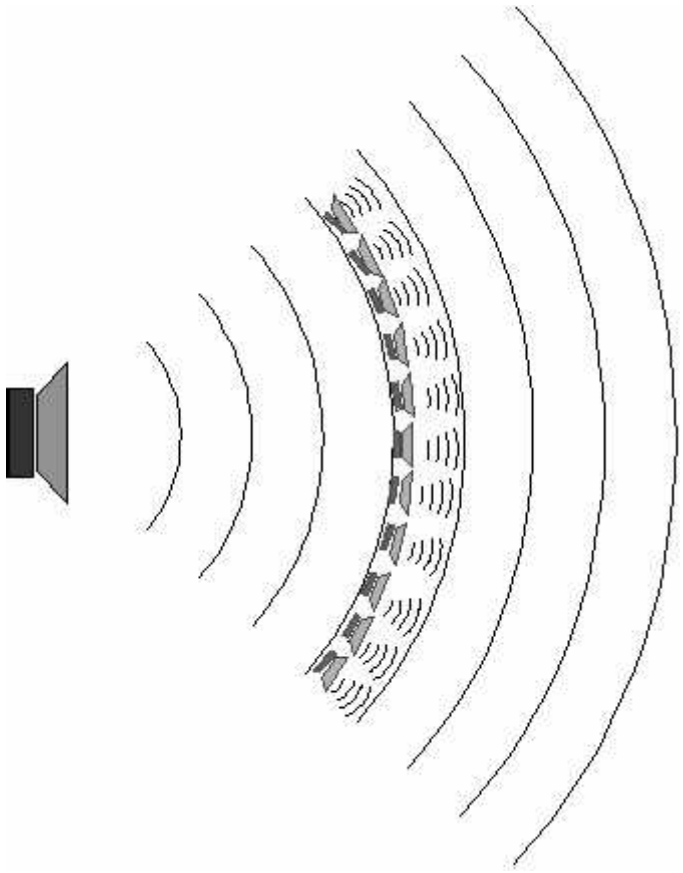


Abbildung 2.3: Nach Huygens' Prinzip ist jeder Punkt einer Wellenfront Ausgangspunkt einer Elementarwelle

Wie zuvor beschrieben, beruht Wellenfeldsynthese auf Huygens' Prinzip. Abbildung 2.3 soll verdeutlichen, wie man sich eine Wellenfront aus einzelnen Elementarwellen zusammengesetzt vorstellen kann.

Hat man nun einen geraden Lautsprecherverlauf, so müssen die einzelnen Lautsprecher zu unterschiedlichen Zeiten angesteuert werden je nachdem, wann die Wellenfront den Lautsprecher passieren würde. Ein einfaches Beispiel dazu ist auf Abbildung 2.4 zu sehen.(en.c)

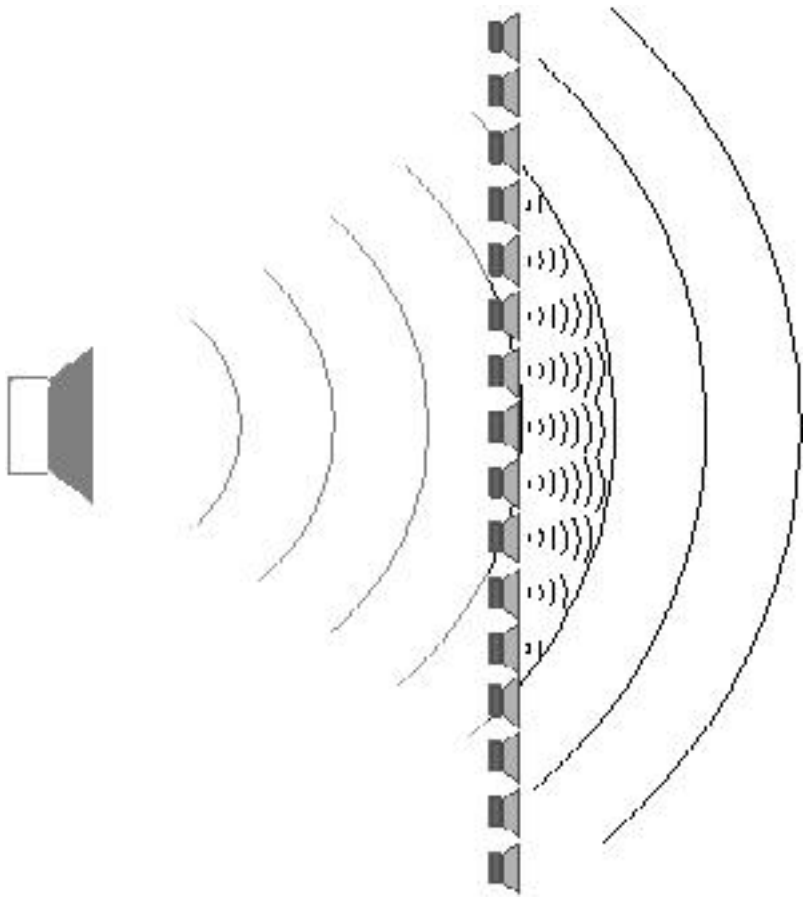


Abbildung 2.4: Das Wellenfeldsynthese Prinzip - jeder einzelne Lautsprecher wird angesteuert, sobald die eigentliche Wellenfront den Lautsprecher passieren würde

2.2.2 WONDER

WONDER (**W**ave field synthesis **o**f **n**ew **d**imensions of **e**lectronic music in **r**ealtime) ist eine Software-Suite zur Nutzung von Wellenfeld- und Binauralsynthesesystemen. WONDERs primäre Plattform ist Linux, kann aber auch unter Mac OSX genutzt werden.

Das Programm wurde ursprünglich von Marje Baalman und Daniel Plewe unter der GNU General public license veröffentlicht. (BBH$^+$)

2.2.2.1 Systemanforderungen und Installation

Die Installation von WONDER wird für eine Linux (X11) Umgebung empfohlen und benötigt folgende Bibliotheken zur Kompilierung und Installation. (BBH$^+$)

- scon
- pkg-config
- libxml++2.6
- libfftw3
- libsndfile
- libasound
- liblo
- QT4.4x
- gcc/g++ compiler

Um WONDER beispielsweise auf einer frischen Ubuntu 10.04 Installation einzurichten sind nur folgende Schritte nötig:

```
sudo apt-get install subversion scons libxml++2.6-dev libjack-dev
sudo apt-get install libfftw3-dev libsndfile-dev libasound-dev
sudo apt-get install liblo-dev libqt4-dev jackd
svn co https://swonder.svn.sourceforge.net/svnroot/swonder swonder
cd swonder/wonder3.1.9
scons wfs=1
sudo scons install wfs=1
```

Zu beachten ist, dass die Version der WONDER Software abweichen kann und der „cd Befehl“ entsprechend angepasst werden muss.

2.2.2.2 Softwarekomponenten von WONDER

Für ein Wellenfeldsynthesesystem werden folgende Kompenenten der WONDER-Softwaresuite benötigt: lib, jackpp, cwonder, twonder, xwonder, scoreplayer und jwonder.

Die zwei erst genannten Komponenten sind Bibliotheken, bei allen folgenden Komponenten handelt es sich um ausführbare Programme.

- cwonder ist die zentrale Kontrolleinheit
- twonder dient als zeitbasierte Rendereinheit
- xwonder ist ein GUI zur Übersicht und Steuerung des Systems
- scoreplayer kann zur Aufnahme und Wiedergabe von midi-synchronisierbaren Partituren genutzt werden
- jfwonder ist ein Hilfsprogramm für sehr genaue Timings zwischen cwonder und den verschiedenen twonder Instanzen

In einem größerem Wellenfeldsynthese Setup würde man mehrere dedizierte Rechner mit twonder für das Rendering nutzen, einen Rechner für die Koordination der Renderer, sowie einen weiteren Rechner für xwonder und eventuell scoreplayer um als Schnittstelle zum Menschen zu fungieren. Weiterhin wird genau eine Instanz von jfwonder benötigt, die nach Empfehlung auf einem der Renderer laufen sollte. Alle Instanzen können auch nur auf einem Rechner gestartet werden, was zu Testzwecken sinnvoll sein kann.(BBH$^+$)

2.2.3 Schnittstellen zu WONDER

WONDER nutzt sowohl für interne als auch externe Kommunikation das Open Sound Control Protokoll. Für weitere Details s. 2.3

Es folgt eine kurze Beschreibung die für diese Arbeit wichtigsten Befehle[1]:

/source/activate i	aktiviert Quelle mit ID i
/source/deactivate i	deaktiviert Quelle mit ID i
/source/position i x y	positioniert Quelle mit ID i bei x,y
/stream/visual/connect s s	als Zuhörer für visuelle Änderungen anmelden

Tabelle 2.1: WONDER OSC Befehle

Bereits mit diesen wenigen Befehlen kann man sich bei cwonder anmelden, um beispielsweise die Positionierung von Quellen zu erhalten, sowie Änderungen an Quellen vorzunehmen. Für weitere Befehle s. (BBH$^+$).

[1] allen Befehlen ist /WONDER voranzustellen

2.3 Open Sound Control

2.3.1 Einführung

Für den Erhalt von Quellpositionen und das verändern von Daten am Wellenfeldsynthesesystem wird das Open Sound Control Protokoll benötigt und soll deshalb im folgendem vorgestellt und erklärt werden.

Open Sound Control (OSC) ist ein Protokoll zur Kommunikation zwischen Computern, Sound-Synthesizer und anderen Multimediageräten. Das Protokoll ist für den Einsatz in modernen Hochgeschwindigkeitsnetzwerken optimiert, sodass einige der Vorteile des Protokolls hohe Kompatibilität, Genauigkeit und Flexibilität sind. (MBa)

2.3.2 Spezifikation

Im folgenden ist die Syntax und Semantik von OSC Nachrichten beschrieben. (MBb)

2.3.2.1 Datentypen

Die Größe von OSC Nachrichten entspricht immer einem Vielfachen von 32 bits.

Ist der erste Block einer Nachricht an einer 32-bit Grenze ausgerichtet, so soll dadurch garantiert sein, dass Folgedaten ebenso 32-Bit ausgerichtet sind.

Datentyp	**Beschreibung**
int32	32bit big-endian Zweierkomplement Ganzzahl
OSC-timetag	64bit big-endian Fixpunkt Zeitformat
float32	32-bit big-endian IEEE 754 Fließpunktzahl
9 OSC-String	Eine Sequenz von nicht null ASCII-Zeichen, die mit null beendet wird und deren bits auf ein Vielfaches von 32 durch null-Zeichen ergänzt wird
OSC-blob	Eine Struktur die aus einer Anzahl Bytes besteht, die durch einen int32 angegeben wird. Anschließend folgen der Anzahl entsprechend viele Bytes, sowie 0-3 zusätzlich Nullbytes um die Größe auf ein Vielfaches von 32 zu bekommen

Tabelle 2.2: OSC Datentypen

2.3.2.2 OSC Pakete

OSC nutzt zur Übertragung sogenannte OSC-Pakete. Diese Pakete bestehen aus einem zusammenhängendem Block von Binärdaten und einem Byte, das die Größe des Blocks und somit des eigentlichen Inhalts angibt.

Die Netzwerkschicht ist für die Übertragung von Größe und Inhalt selbst verantwortlich. So kann ein OSC Paket als UDP datagram versendet werden oder beispielsweise auch als Stream mit TCP.

Die ersten zwei Bytes des Pakets bestimmen ob es sich um eine „OSC Message“ oder ein „OSC Bundle“ handelt. Dazu im folgendem mehr.

2.3.2.3 OSC Message

Eine OSC Nachricht beste aus einem „OSC Address Pattern“, der Adresse, gefolgt von beliebig vielen „OSC Type Tag Strings“, den Datentypen.

2.3.2.4 OSC Address Pattern

Diese Adressen erinnern an URL Datenpfade, sie beginnen mit einem ’/’ gefolgt von einem OSC-String.

2.3.2.5 OSC Type Tag String

„OSC Type Tag Strings“ sind OSC-Strings, die mit einem ’,’ beginnen gefolgt von der genauen Anzahl Zeichen, enstsprechend der Anzahl der Argumente der Nachricht. Die einzelnen Zeichen nach dem Komma werden als „OSC Type Tag“ bezeichnet.

OSC Type Tag	Dazugehöriger Datentyp
i	int32
f	float32
s	OSC-String
b	OSC-Blob

Tabelle 2.3: Standard OSC Type Tags

Weiterhin können nicht standardisierte OSC Type Tags genutzt werden, die aber nicht von allen OSC Applikationen unterstützt werden müssen und dementsprechend von solchen Applikationen verworfen werden sollten. Folgende Tabelle beschreibt auf welche Typen dabei zurückgegriffen werden kann.

OSC Type Tag	Dazugehöriger Datentyp
h	64 bit big-endian Zweierkomplement Ganzzahl
t	OSC-timetag
d	64 bit IEEE 754 Fließpunktzahl
S	OSC-String der z.B. für Systeme genutzt werden, die zwischen Symbolen und Strings unterscheiden
c	32 bit ASCII Zeichen
r	32 bit RGBA Farbe
m	4 byte MIDI Nachricht, wobei von MSB zu LSB: port id, status byte, data1 und data2
T	Wahr. Es werden keine bytes allokiert.
F	Falsch. Es werden keine bytes allokiert.
I	Unendlich. Es werden keine bytes allokiert.
N	Nil. Es werden keine bytes allokiert.
[	Anfang eines Arrays
]	Ende eines Arrays

Tabelle 2.4: Nicht-Standardisierte OSC Type Tags

2.3.2.6 OSC Bundles

Ein OSC-Bundle besteht aus dem OSC-string „bundle“ gefolgt von einem OSC timetag und einer beliebigen Anzahl von OSC Bundle Elementen. Das OSC timetag ist eine 64 bit Fixpunktzahl.

Ein OSC Bundle Element besteht aus einer Größe und dem eigentlichem Inhalt. Die Größe wird als int32 angegeben und der Inhalt ist, wie bei allen Datentypen bisher 32 bit ausgerichtet. Es sei erwähnt, das Bundles weitere Bundles enthalten können.

Nachfolgende Tabelle zeigt, wie ein solches OSC Bundle aussehen könnte:

Daten	Größe	Nutzen
OSC-string „#bundle“	8 bytes	um zu erkennen, dass dies ein bundle ist
OSC-timtag	8 bytes	Zeitstempel, der für das ganze bundle gilt
Größe erstes bundle Element	4bytes	erstes Bundle Element
Inhalt erstes bundle Element		
Größe zweites bundle Element	4bytes	zweites Bundle Element
Inhalt zweites bundle Element		

Tabelle 2.5: OSC Bundle Beispiel

2.3.3 OSC Semantik

2.3.3.1 OSC Adressen und Adressraum

Jeder OSC Server hat eine Menge von OSC Methoden. Das verschicken von OSC Nachrichten, kommt einem Methodenaufruf gleich. Die adresse gibt an welche Methode aufgerufen werden sollen und die Type Tags sagen aus, welche Argumente übergeben werden sollen. Anschließend führt der Server aufgerufene Methode aus.

OSC Methoden eines Servers werden in einem Baum angeordnet. Man spricht hier vom OSC Address Space. Die Blätter stellen hierbei die Methoden dar und die Knoten werden als OSC Container bezeichnet. Der Adressraum eines Servers ist dynamisch, d.h. er kann sich zur Laufzeit ändern.

Nicht alle ASCII Zeichen können für die Benennung der Methoden genutzt werden:

Zeichen	**ASCII code (dezimal)**
' '	32
#	35
*	42
,	44
/	47
?	63
[	91
]	93
{	123
}	125

Tabelle 2.6: für OSC Methoden und Knoten nicht erlaubte ASCII Zeichen

Die Syntax von OSC Adressen sind an URL Pfade angelehnt und beginnen mit einem '/' (forward slash). Nachfolgend einige Beispiele:

```
/dies/ist/ein/pfad
/dies/ist/ein/pfad2
/list/1
/list/2
/list/3
```

2.3.3.2 OSC Message Dispatching und Pattern Matching

Nach Erhalt einer OSC Nachricht muss der Server die auf dem Adress Pattern der OSC Nachricht basierenden, korrekten OSC Methoden innerhalb des OSC Adressraums aufru-

fen. Dieser Vorgang wird als dispatching bezeichnet. Alle dem Pattern übereinstimmenden Methoden werden mit den selben Argumenten aufgerufen.

Ein Substring zwischen zwei '/' wird als „part“ bezeichnet. Das Adress Pattern einer OSC Nachricht stimmt mit einer OSC Adresse unter folgenden zwei Bedingungen überein:

1. OSC Adresse und OSC Address Pattern haben die selbe Anzahl an „parts“
2. alle Substrings (parts) der OSC Adresse stimmen mit ihren in der Hierarchie entsprechenden Substrings der OSC Nachricht überein

Um mit einer Nachricht mehrere Methoden anzusprechen können folgende Kontrollzeichen zusätzlich verwendet werden:

1. '?' wildcard für genau ein Zeichen
2. '*' wildcard für eine beliebige Zeichenfolge
3. '[]' eine Übereinstimmung ist gegeben, wenn mindestens einer der Zeichen innerhalb der Klammern übereinstimmt. Innerhalb von [] kann außerdem genutzt werden:
   - zwei durch ein '-' getrennte zeichen geben einen Bereich ein (z.B. zwischen [a-z])
   - ein '!' negiert angegebene Zeichen und besagt, dass jene nicht vorkommen sollen
4. bei Komma separierten Strings in geschweiften Klammern, muss nur einer der Strings übereinstimmen (z.B. {foo,bar})

2.3.3.3 Zeitliche Semantik und OSC timtags

OSC benötigen Zugriff auf die korrekte, absolute Zeit, da OSC keine Zeit-Synchronisations-Mechanismen unterstützt. Einzelne OSC-Nachrichten, also ohne timetag, sollten umgehend verarbeitet werden. Handelt es sich um ein OSC Bundle muss das OSC timetag beachtet werden. Liegt angegeben Zeit in der Zukunft, so wird bis zu diesem Zeitpunkt mit der Ausführung gewartet. Zeitstempel in der Vergangenheit bzw. Gegenwart werden unverzüglich ausgeführt, außer der OSC Server ist so eingerichtet, dass er zu alte Nachrichten verwirft.

Wie bereits erwähnt werden timetags durch 64 bit Fixpunktzahlen repräsentiert. Die ersten 32 bit bestimmen die Sekunden seit dem 1. Januar 1900 um 00:00 Uhr. Die letzten 32 bit spezifizieren die Dezimalen einer Sekunden in einer Auflösung von 200 ps. Diese Darstellung ist an die NTP Zeitstempel angelehnt. Ein Zeitstempel mit 63 0-bits und einer 1 im LSB ist ein Spezialfall und heißt „unverzüglich“.

OSC Nachrichten innerhalb eines Bundles sollen atomar verarbeitet werden. Sie erhalten alle den Zeitstempel des Bundles. Werden mehrere Methoden mit einer Nachricht angesprochen ist die Reihenfolge unspezifiziert. Weiterhin gilt, dass Zeitstempel von verschachtelten OSC Bundles größer sein müssen. als ihre umkapselnden Bundles.

2.3.4 Bibliotheken

2.3.4.1 liblo in C

liblo ist eine Open Sound Control implementierung für POSIX Systeme, wie Windows, Mac OS X und Linux. liblo ist in C geschrieben und unterstützt alle Standard OSC Typen, sowie „threaded Server“, dispatching und Zeitbasierte bundles. liblo wurde unter der GNU General Public Licence veröffentlicht. (MBc)

Transportmöglichkeiten

- TCP
- UDP

Features

- Paket Parsing
- Paket Erstellung
- Bundle Unterstützung
- timetag Unterstützung
- Paketraten von > 100Hz möglich

Plattformen

- Windows
- Linux
- Mac OS X

Bundle Unterstützung

- Erstellen
- Lesend

Unterstützte Type Tags

- i: int32
- b: blob
- s: string
- f: float32
- h: int64
- t: timetag
- d: float64
- S: symbol
- T: true
- F: false
- N: null
- I: infinitum

2.3.4.2 Java OSC

JavaOSC ist eine einfache, aber eingeschänkte Open Sound Control Implementierung in Java. (MBd)

Transportmöglichkeiten

- UDP

Plattformen

- Plattformunabhängig

Features

- Paket Parsing
- Paket Erstellung

Unterstützte Type Tags

- i: int32
- b: blob
- s: string
- f: float32

2.4 Das Kamera-Tracking-Systems

2.4.1 Beschreibung DTrack

Das Dtrack System ist ein IR optisches Tracking-System, das die Position von bis zu 20 Objekten liefern kann. Das System kann etwa alle 20ms bis 40ms neue Daten versenden, weswegen man von einem Realtime Tracking System sprechen kann. Für das Tracking werden spezielle Marker benötigt, sogenannte „Rigid Bodys“, diese bestehen aus 4 bis 20 fest arrangierten, kugelförmigen, bewusst refklektierenden Markern.

Zu jedem Objekt werden kartesische Koordinaten (x,y,z), sowie drei Winkel, die die Orientierung des Objekts angeben, gesendet. (A.R07)

2.4.2 Schnittstellen

DTrack nutzt UDP/IP Datagramme um Messdaten an andere Applikationen zu verschicken. Die Pakete können binär oder in einem ASCII Format verschickt werden, je nachdem wie das System konfiguriert wird. Vom Hersteller empfohlen wird das ASCII Format, da es deutlich mehr Informationen enthält. Im Normalfall wird nach jeder Messung ein Datagramm verschickt und jedes Datagramm enthält die Informationen über genau eine Messung. Die Frequenz mit der gesendet wird, kann im System konfiguriert werden. (A.R07)

2.4.3 ASCII Datagramme

Ein UDP Datagramm im ASCII Format enthält mehrere Zeilen, die durch CR/LF (hex 0D 0A)getrennt sind. Jede Zeile beginnt mit einer ID und enthält dementsprechend Daten in einem bestimmten Datentyp. In der Konfiguration des Systems kann angegeben werden, welche Zeilen versendet werden sollen und welche nicht. Im Folgendem ein Überblick über mögliche IDs.

ID	**Datentyp**
fr	Paketzähler (frame counter)
ts	Zeitstempel (timestamp)
6d	Standard-Körper
6df/6df2	Flysticks
6dmt	Messwerkzeuge
gl	Fingertracking Handschuhe
3d	einfache 3-Winkel Marker
6dcal	zusätzliche Informationen

Tabelle 2.7: ASCII Datagramme

frame counter Datagramme fangen immer zuerst mit dem frame counter an. Dieser wird mit der Syncronisierungsfrequenz hochgezählt.

Beispiel: fr 21753

timestamp Der Zeitstempel ist optional, je nach Konfiguration des Systems. Es werden die Sekunden seit 00:00 UTC der System Uhr wiedergegeben.

Beispiel: ts 39596.024831

6d Messdaten für alle standard 6DOF Körper (6 degrees of freedom). Messdaten haben die folgende Form [id qu][sx sy sz $\eta\theta\phi$][b0 b1 b2 b3 b4 b5 b6 b7 b8] mit der Bedeutung:

1. ID Nummer (id, ab 0), Qualität (qu, bisher ungenutzt),
2. Position (si), Orientierungswinkel ($\eta\theta\phi$) und
3. Rotationsmatrix (bi) der Körperorientierung

Alle Zahlen sind durch Leerzeichen (hex 20) abgetrennt. Die Rotationsmatrix hat folgende Form:

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

Beispiel:

6d 1 [0 1.000][326.848 -187.216 109.503 -160.4704 -3.6963 -7.0913]
[-0.940508 -0.339238 -0.019025 0.333599 -0.932599 0.137735
-0.064467 0.123194 0.990286]

2.4.4 Binäre Datagramme

Binäre UDP Datagramme enthalten nur 6DOF und 3DOF Daten und werden als Struktur mit variabler Länge verschickt. Flysticks, Messwerkzeuge und Fingertrackinggeräte werden als Standardkörper behandelt, sodass bis auf Position und Orientierung keine weiteren Informationen vorhanden sind. Die Struktur in C:

```
1  struct{
2          unsigned long framenr;   /* frame number */
3          int nbodies;             /* number of tracked bodies */
4          struct{
5                  unsigned long id;/* id number */
6                  float quality;   /* quality */
7                  float loc[3];    /* location */
8                  float ang[3];    /* orientation angles(eta,theta,phi) */
9                  float rot[9];    /* rotation matrix */
10         } body[ nbodies ];       /* variable number of blocks */
11         int nmarkers;            /* number of tracked single markers */
12         struct{
13                 unsigned long id;/* id number */
14                 float quality;   /* quality */
15                 float loc[3];    /* location */
16         } marker[nmarker];       /* variable number of blocks */
17 };
```

2.5 Augmented-Reality

2.5.1 Augmented-Reality allgemein

Augmented Reality ist eine computergestützte Erweterung der Realitätswahrnehmung. Häufig werden Informationen zu Bildern oder Videos computergeneriert, mittels Einblendung/-Überlagerung visuell erweitert.

2.5.1.1 Definition

Häufig wird die Definition von Azuma zitiert. Sie besagt, dass virtuelle Realität und Realität teilweise überlagert sind, d.h. Interaktion erfolgt in Echtzet und virtuelle und reale Objekte stehen dreidimensional zueinander in Bezug. (de.a)

Hierbei wird nur ein Teilaspekt der erweiterten Realität beachtet und man beschränkt sich zudem auf technische Merkmale. Andere Definitionen ergänzen Augmented Reality erweitere die menschliche Wahrnehmng mit Hilfe von Sensoren um Umgebungseigenschaften, die der Mensch nicht wahrnehmen kann. (Bon)

2.5.1.2 Einsatzgebiete

Hier nur einige, mögliche Anewendungsbeispiele für Augmented Reality. (en.a)

- Konstruktion und Wartung: Beschriftung von Teilen
- Navigation: Einmalen der Route auf die Straße
- Militär: Anzeige von Freund/Feind
- Katastrophenmanagement: Anzeige von Bränden
- Architektur: wie sieht mein Haus aus, wenn es fertig ist
- Spiele
- Museen: Übersichtliche Anzeige von Informationen

2.5.2 Frameworks

Mittlerweile gibt es für Android eine Vielzahl von Frameworks für Augmented Reality Applikationen. Im folgenden sollen einige Frameworks mit ihren Vor- und Nachteilen aufgezählt werden, um anschließend zu evaluieren, ob eine Eigenentwicklung nötig ist oder ob auf bereits Vorhandenem aufgebaut werden kann.

2.5.2.1 Qualcomm AR SDK

Das Qualcomm SDK ist ein mächtiges Framework zur Erstellung von Augmented Reality Applikationen, die positionsunabhängig dreidimensionale Objekte einblendet, wo bestimmte Strukturen vom Programm erkannt werden. Das SDK ist in C geschrieben und benötigt das Android NDK um nativen C Code auf dem Zielgerät auszuführen. Dadurch ist das SDK verhältnismäßig schnell und kann so höhere Aktualisierungsgeschwindigkeiten des Bildes ermöglichen. Man könnte sich beispielsweise ein Spiel vorstellen, bei dem eine mit der Kamera aufgenommene Asphaltstrecke um die zu steuernden Rennwagen erweitert wird. D.h. ohne zusätzliches Gerät würde man auf dem Boden die Rennstrecke sehen, aber erst durch den Blick über den Bildschirm lässt sich das eigentliche Spiel sehen. Einen kleinen Überblick, wie eine solche Anwendung aufgebaut ist, zeigt folgendes Bild:

(Ince)

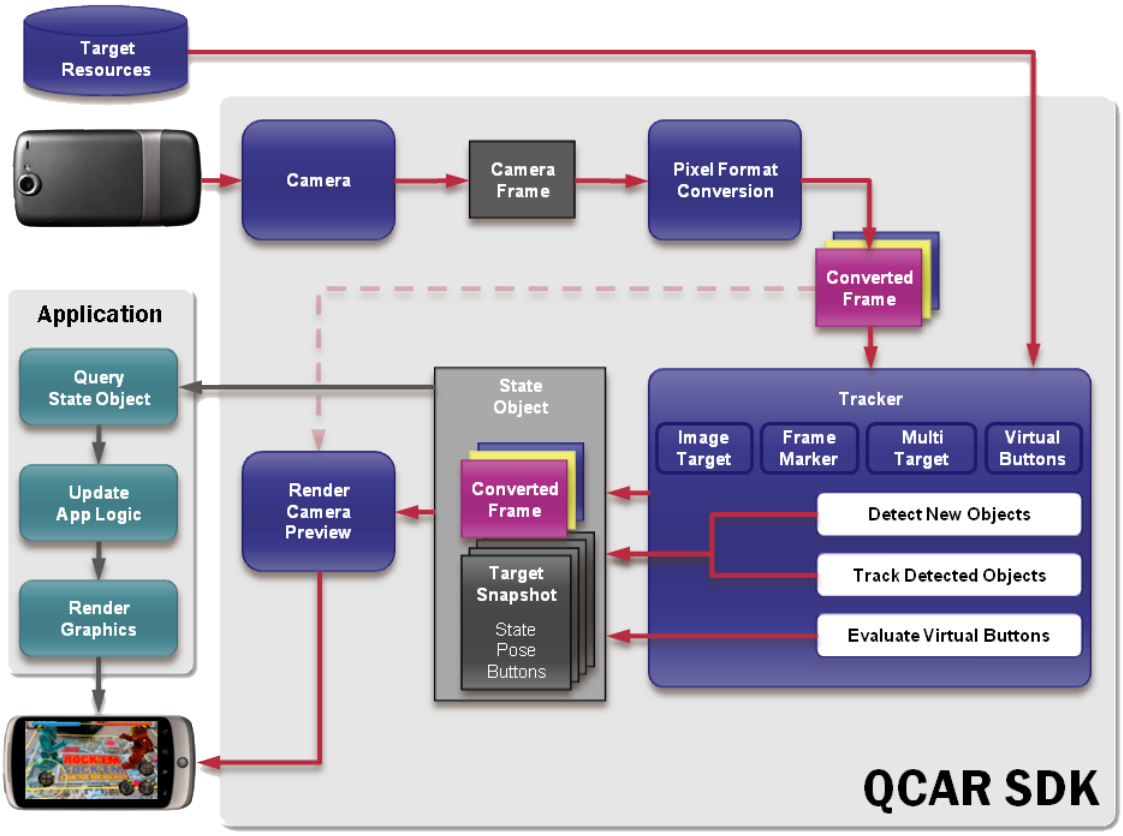


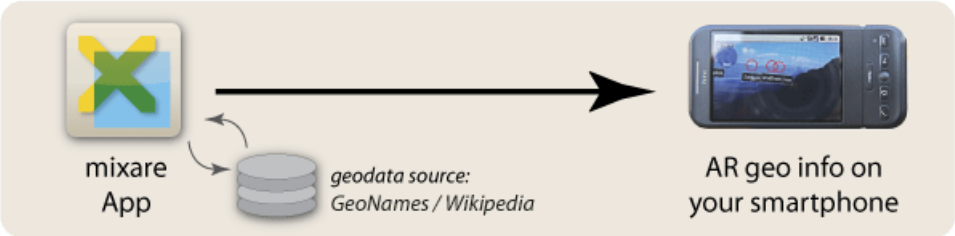
Abbildung 2.5: Qualcomm Anwendung

2.5.2.2 mixare

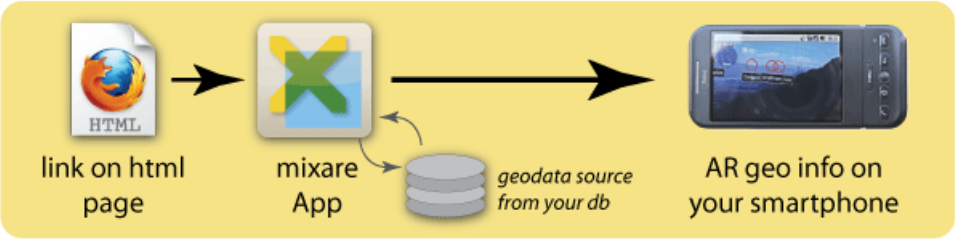
mixare (mix Augmented Reality Engine) ist ein freier open source Augmented Reality Browser, der unter der GPLv3 veröffentlicht wurde. Die Engine ist sowohl für Android als auch für iOS erhältlich. mixare ist ein bereits lauffähiges Programm und kann so einfach durch eigene Geolocation Daten erweitert werden oder aber für eigene Zwecke verändert und erweitert werden. (Gob)

Auf der mixare Homepage werden vier mögliche Einsatzgebiete vorgestellt:

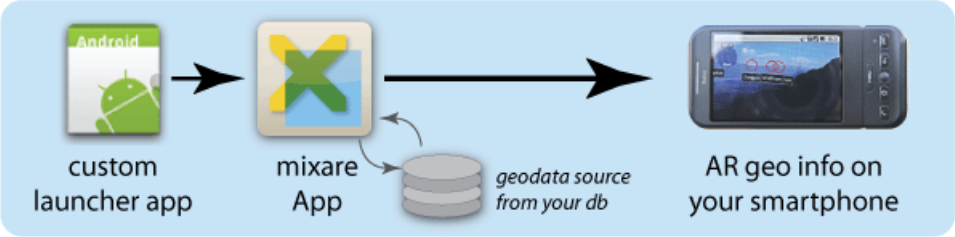
1. Als autonome Applikation, welche Wikipedia points-of-interests darstellt



2. durch klicken eines Links werden eigene Geoinformationen übertragen



3. Übertragung der Daten über eine eigene Startapplikation



4. Als Eigenentwicklung kann die Applikation frei erweitert werden

mixare kann gut für Applikationen genutzt werden, bei dem die Positionen von Objekten als geographische Längen- und Breitengrade angegeben sind. Will man Positionen, die als kartesische Koordinaten vorliegen, beispielsweise bei Indoor-Navigation verwenden, so müssten sämtliche Transformationen zur Darstellung auf dem Bildschirm überarbeitet werden.

2.5.2.3 AndEngine

AndEngine ist eine freie Android 2D OpenGL Spiele-Engine, die unter der GNU Lesser General Public License veröffentlicht wurde. Ein wesentlicher Vorteil dieser Engine ist, das sie durch Extensions leicht erweiterbar ist und es zudem bereits eine Augmented-Reality Extension existiert. Da Spiele oft auch mit kartesischen Koordinaten arbeiten, sollte diese Engine auch gut für Indoor-Navigation geeignet sein. Allerdings setzt das Arbeiten mit dieser Engine kenntnisse mit OpenGL voraus.

2.5.2.4 Eigenentwicklung

Die Eigenenticklung einer Augmented-Reality Engine könnte mehr Zeit beanspruchen, allerdings ist diese Engine dann vollständig auf das eigene Problem angepasst.

2.5.2.5 Fazit

Für die Entwicklung der Applikation soll eine eigene Augmented-Reality Engine entwickelt werden. Das Problem ist sehr speziell, da viele Frameworks davon ausgehen, das die Position durch GPS genügt. In diesem Fall wäre die Position viel zu ungenau, da GPS im Normalfall eine Genauigkeit von 20m bieten kann, die zudem auch noch durch das Gebäude selbst gestört wird.

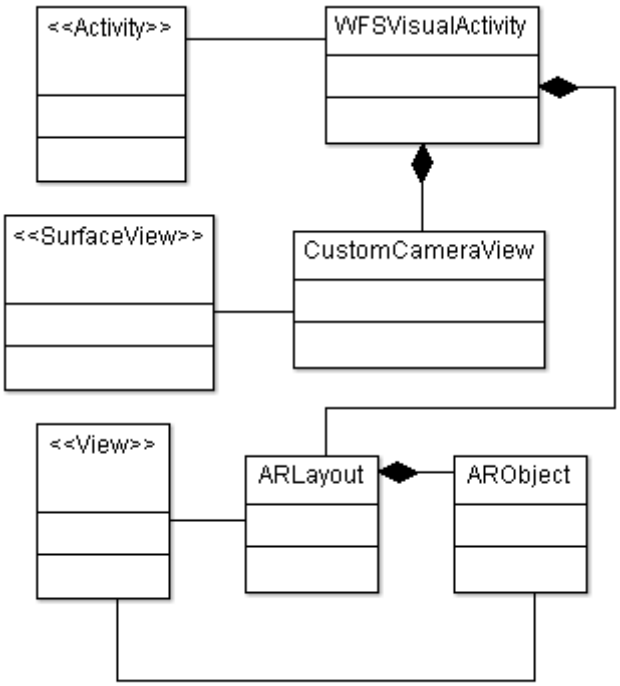
Kapitel 3

Design

3.1 Augmented-Reality Engine

Dieses Kapitel soll einen Überblick über das Design der Android-Applikation geben. Es zeigt auf, welche Klassen benötigt werden und wofür diese gebraucht werden.

Zunächst besteht jede Android-Applikation aus mindestens einer Activity. Activities implementieren die sichtbaren Bestandteile einer Anwendung und interagieren mit dem Anwender. (BP11) In einem gut strukturiertem Programm nach dem Model-View-Controller Pattern würden man das Layout in den dafür vorgesehenen XML Dateien anpassen. Für eine Augmented-Reality Engine benötigen wir allerdings kein Standard-Design, sondern ein auf die Kamerasicht aufbauendes Design. Das folgende UML-Diagramm soll zeigen welche grundlegenden Klassen benötigt werden und wie diese zueinander in Beziehung stehen.



Nachfolgend eine kurze Beschreibung der Klassen, sowie deren Aufgaben. Einen detailierteren Überblick gibt einem das Kapitel Implementierung (s. Kapitel 4).

3.1.1 WFSVisual

WFSVisual ist die Startactivity und in diesem Fall auch die einzige Activity. WFSVisual sorgt dafür, dass die CustomCameraView und das ARLayout gestartet werden. (Inca)

3.1.2 CustomCameraView

CustomCameraView ist eine von SurfaceView abgeleitete Klasse. SurfaceView ist ein dedizierte, eingebette Zeichenumgebung in einer View-Hierarchie. Vorteile dieser View sind, dass man Größe und Format selbst bestimmen kann, sowie die eigenständige, korrekte Positionierung auf dem Bildschirm. Für eine möglichst übersichtliche Anzeige der Soundquellen des Wellenfeldsynthese-Systems ist eine Darstellung im Landscape-Mode (Breitbild-Format) von Vorteil. Diese Oberfläche ist Z-ordered, d.h. sie liegt ganz unten in der View-Hierarchie und außerdem wird alles weitere auf ihr gezeichnet. Diese View ist folglich zur Darstellung der Umgebung sehr gut geeignet, da sie zudem auch noch ermöglicht, auf ihr die anzuzeigenden Quellen zu zeichnen. (Incc)

3.1.3 ARLayout

ARLayout ist vom Interface View abgeleitet, das für die Darstellung vorgesehen ist. ARLayout ist für das Zeichnen des Head-up-Displays (beispielsweise könnte die eigene Position angezeigt werden oder auch ein Radar, dass die Quellen zusätzlich anzeigt), sowie für das Zeichnen der einzelnen Augmented-Reality-Objekte zuständig. Zur Bestimmung der einzelnen Positionen auf dem Bildschirm muss ARLayout Zugriff auf Beschleunigungs- und Magnetische-Sensoren erhalten, da die Blickrichtung des Handys anhand der Lage bestimmt werden soll. (Incd)

3.1.4 ARObject

Auch ARObject ist von View abgeleitet und hat eine rein optische Funktion. Neben der eigentlichen Darstellung des Objekts, beispielsweise ein Kreis, ist Beschriftung, also Informationen zum Objekt selbst denkbar. (Incd)

3.1.5 Zusammenfassung

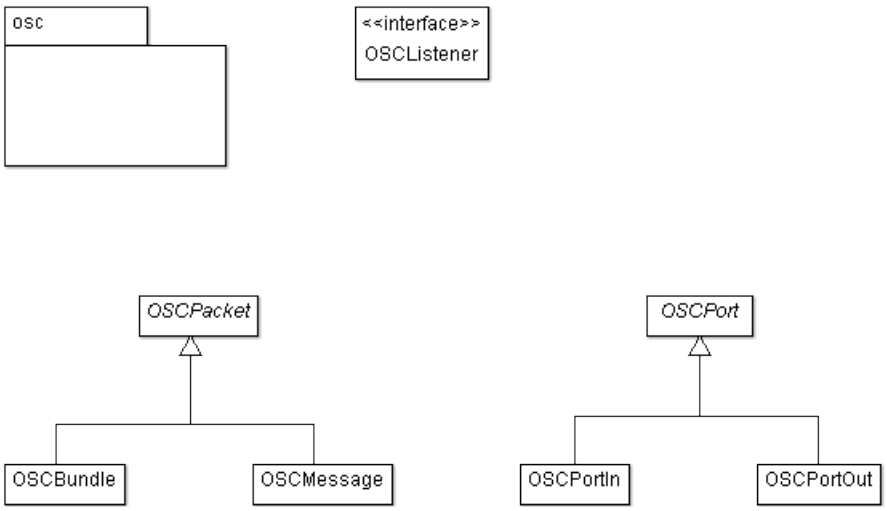
Durch die mächtige API von Android benötigt man also nicht viel mehr als eine View zur Darstellung des Realbildes, sowie den Views für die Erweiterungen.

3.2 Erhalt der Quellpositionen

Zur Darstellung von Augmented-Reality Objekten benötigt man logischerweise erst Daten, die angezeigt werden sollen. Hierfür wird das Open Sound Control Protokoll genutzt, um vom Wellenfeldsynthese-System Quellpositionen zu erhalten.

3.2.1 Open Sound Control

Zum verschicken von Nachrichten, sowie zum Erhalt der Positionsdaten genügt, die in Kapitel 2 vorgestellte Java OSC Implementierung (s. 2.3.4.2). Einen Überblick über die Implementierung soll folgendes UML-Diagramm geben, sowie darauffolgend eine kurze Beschreibung:



3.2.2 OSCPacket

Die OSCPacket-Klasse ist eine Kapselung des Bytestreams. Sie enthält also die Rohdaten der eigentlichen Nachricht.

3.2.2.1 OSCBundle und OSCMessage

Wie im Abschnitt über das Opensoundcontrol-Protokoll (s. 2.3) beschrieben, gibt es gewöhnliche Nachrichten und Bundles, die mehrere Nachrichten enthalten. Diese Klassen implementieren OSCPacket und machen die Rohdaten nutzbar. Für diese Arbeit ist das OSCBundle uninteressant.

3.2.3 OSCPort

Die abstrakte Klasse OSCPort ist eine Kapselung der DatagramSocket-Klasse. Sie hat zusätzliche Methoden zum Erhalt des Standard Supercollider Ports.

3.2.4 OSCPortIn und OSCPortOut

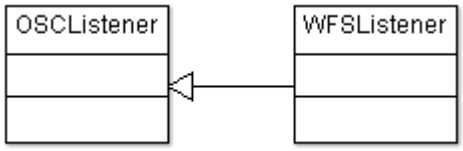
OSCPortIn und OSCPortOut implementieren OSCPort. Wie die Klassennamen bereits erahnen lassen, ist OSCPortIn zum Erhalt von OSC Nachrichten und OSCPortOut zum versenden von OSC Nachrichten zuständig.

3.2.5 OSCListener

OSCListener ist ein Interface, das implementiert werden kann, wenn man kontinuierlich Nachrichten erwartet. Die Quellpositionen des Wellenfeldsynthese-Systems können ständig aktualisiert werden, weswegen eine Implementierung dieser Schnittstelle für das Applet sinnvoll ist.

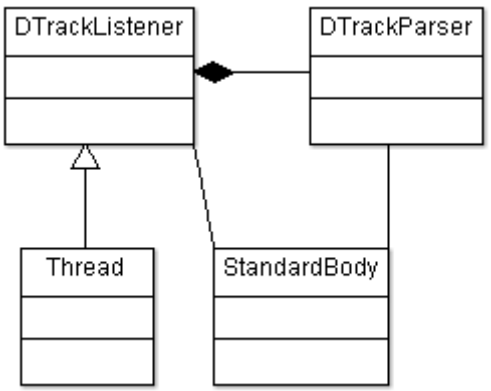
3.2.6 WFSListener

Um also durchgängig Nachrichten des Wellenfeldsynthese-Systems zu erhalten, wird das Interface OSCListener implementiert:



3.3 Erhalt der Eigenposition

Das A.R.T. Tracking-System verschickt über das lokale Netzwerk per Multicast Datagramme, wie in Kapitel 2.4.3 beschrieben. Dabei ist die Anbindung mit und ohne Kabel möflich. Ein Thread, der auf der entsprechend eingstellten Multicast IP lauscht, wird benötigt. Außerdem muss das ASCII Datagramm bearbeitet werden, weswegen ein Parser und eine Modelklasse, zur Kapselung der Datagramme in einer nutzbaren Form, benötigt werden. Nachfolgendes Diagramm und darauffolgende Erläuterungen, beschreiben jene Zusammenhänge:



3.3.1 DTrackListener

Der DTracklistener implementiert Thread und besteht unter anderem aus dem DTrackParser, der die StandardBody Modelklasse liefert, wenn ein Datagramm an den Listener übergeben wurde.

3.3.2 DTrackparser

Diese Klasse soll die ASCII Datagramme parsen und dem DTrackListener ein StandardBody zurückliefern.

3.3.3 StandardBody

StandardBody ist eine Modelklasse, die u.a. Position, ID und Ausrichtung einer Person enthält.

3.4 Gesamtkonzept

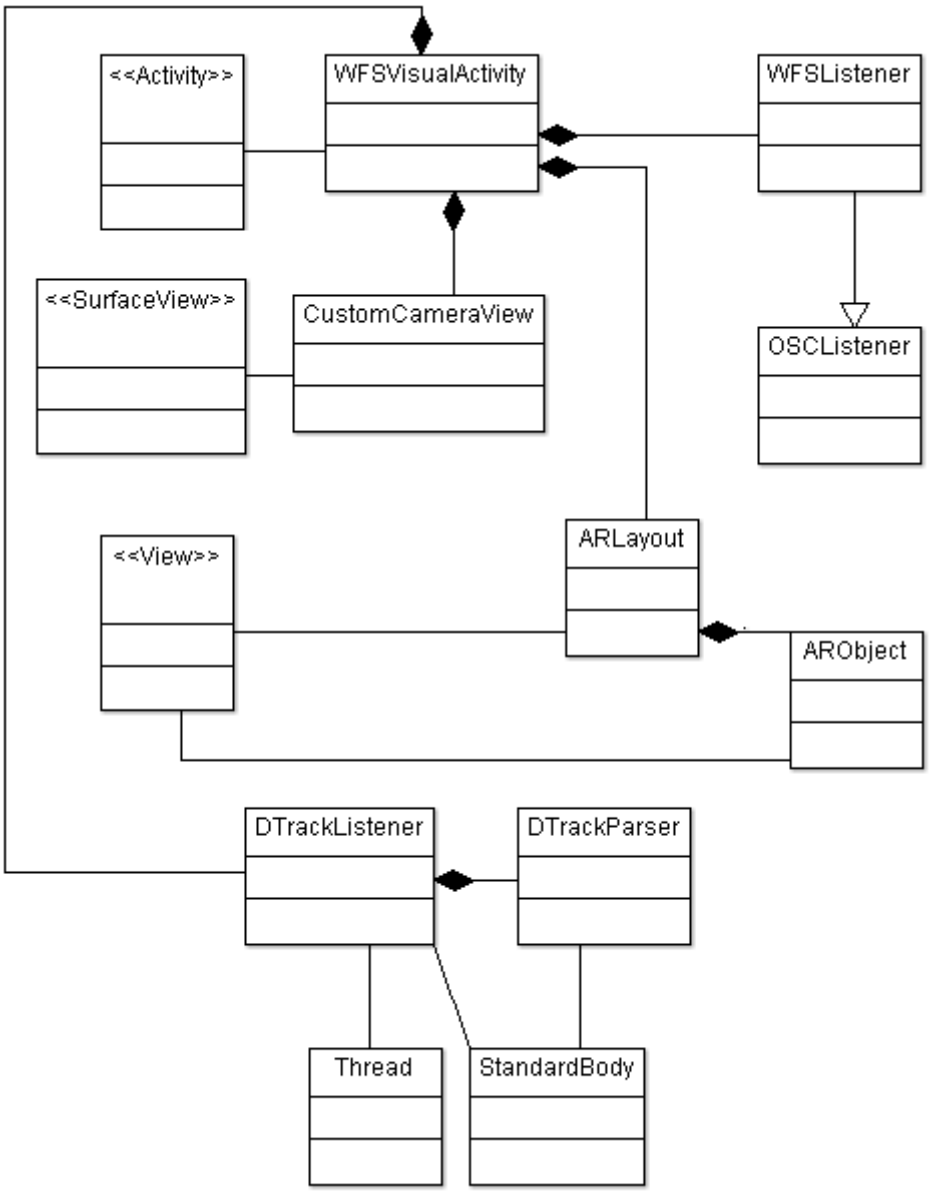
Die einzelnen Komponenten müssen nun noch miteinander verbunden werden. Nachfolgendes UML-Diagramm zeigt das Gesamtbild der Applikation, also Kopplung des Wellenfeldsynthese-Systems und des Tracking-Systems mit der Augmented-Reality Engine:

3.4.1 Verbindung zum WFSListener

Der WFSListener wird in der Main-Activity WFSVisualActivity instanziiert und gestartet. Die Activity kann dem Listener so eine Datenstruktur übergeben, die ebenfalls an das ARLayout übergeben werden kann.

3.4.2 Verbindung zum DTrackListener

Ähnlich wie beim WFSListener wird der DTrackListener auch in der Main-Activity instanziiert, sodass die Position, durch eine Referenz auf einen in der Main-Activity instanziierten StandardBody, immer bekannt ist.


<<Activity>>
WFSVisualActivity
WFSListener
<<SurfaceView>>
CustomCameraView
OSCListener
<<View>>
ARLayout
ARObject
DTrackListener
DTrackParser
Thread
StandardBody

Kapitel 4

Implementierung

Dieses Kapitel beschreibt die Implementierung der Android-Applikation. Es werden nacheinander, die bereits vorgestellten Klassen aus dem Kapitel 3, aufgezeigt und erläutert. Auf eine Optimierung der Algorithmen, sowie die strikte Einhaltung von Android-Code Konventionen wurde verzichtet. Ziel ist es zu prüfen, ob die begrenzten Ressourcen eines Smartphones verglichen zu einem Desktop-PC oder Notebook, für eine flüssige Darstellung der Szene ausreichen. Zum Abschluss des Kapitels wird in Abschnitt 4.7 die Ausführung, der hier genannten Implementierung, auf einem Testgerät beschrieben und der Eindruck zur Nutzbarkeit dargelegt.

4.1 WFSVisualActivity

Diese Klasse stellt den Einstiegspunkt der Applikation dar (Main-Klasse). In ihr werden alle Komponenten für den reibungslosen Betrieb gestartet.

```
1  public class WFSVisualActivity extends Activity {
2          CustomCameraView cv;
3          Context ctx;
4
5          /** Called when the activity is first created. */
6          @Override
7          public void onCreate(Bundle savedInstanceState) {
8                  super.onCreate(savedInstanceState);
9                  setRequestedOrientation(ActivityInfo.
                       SCREEN_ORIENTATION_LANDSCAPE);
10                 requestWindowFeature(Window.FEATURE_NO_TITLE);
11                 ctx = getApplicationContext();
12                 getWindow().setFlags(WindowManager.LayoutParams.
                       FLAG_FULLSCREEN, WindowManager.LayoutParams.
                       FLAG_FULLSCREEN);
```

```
13                  cv = new CustomCameraView(ctx);
14                  FrameLayout rl = new FrameLayout(ctx);
15
16                  WindowManager w = getWindowManager();
17                  Display d = w.getDefaultDisplay();
18                  int width = d.getWidth();
19                  int height = d.getHeight();
20                  ARLayout ar = new ARLayout(ctx);
21                  ar.screenheigth = height;
22                  ar.screenwidth = width;
23                  ar.arObjects = new HashMap<Integer, ARObject>();
24
25                  rl.addView(cv);
26                  rl.addView(ar, width, height);
27
28                  setContentView(rl);
29                  WFSListener wfs;
30                  try {
31                          wfs = new WFSListener();
32                          wfs.ctx = ctx;
33                          wfs.arObjects = ar.arObjects;
34                          wfs.start();
35                          DTrackListener dtl = new DTrackListener();
36                          dtl.layout = ar;
37                          dtl.start();
38                  } catch (IOException e) {
39                          e.printStackTrace();
40                  }
41                  Log.d("WFSVisual", "All started.");
42          }
43  }
```

Der Programmablauf mit Zeilenangaben:

- oncreate() wird sofort aufgerufen, sobald die Activity gestartet (7) wird
- ruft oncreate() der geerbten Activity-Klasse auf (8)
- veranlasst, dass das Bild nicht gedreht wird, wenn das gerät gedreht wird (9)
- eine Titelleiste wird nicht benötigt, sie zeigt nur den Namen an (10)
- Context wird nachfolgend weitergereicht, da dies allgemeine Programminformationen enthält (11)

- diese Flags bewirken, das Videos auf den gesamten Bildschirm abgespielt werden, in diesem Fall die Kamera (12)
- erstellt CustomCameraView (13)
- ein Layout, dass den gesamten Bildschirm füllt und mit z-Ebenen arbeitet (14)
- Schnittstelle zur Fensterverwaltung (16)
- Schnittstelle zum Bildschirm (17)
- Bildschirmauflösung in x-Richtung (18)
- Bildschirmauflösung in y-Richtung (19)
- erstellt ARLayout (20)
- ARLayout benötigt die Bildschirmauflösung (21)
- eine Map über ARObjects, gepaart mit ihrer ID für einen schnellen Zugriff (23)
- hinzufügen der CustomCameraView in das Layout (25)
- hinzufügen von ARLayout in das Layout (26)
- das Layout wird nun als aktive View gesetzt (28)
- erstellt FSListener (31)
- selbe Referenz zur Map, wie in Zeile 23. ARLayout und WFSListener greifen auf die selbe Map zu (33)
- WFSListener und DtrackListener sind Threads und werden gestartet (34-37)

4.2 ARLayout

ARLayout ist für das Zeichnen der Augmented-Reality-Objekte zuständig und benötigt deshalb die Sensordaten. Bevor die Methoden von ARLayout im Detail beschrieben werden, hier erst ein grober Überblick über die gesamte Klasse:

```
1 public class ARLayout extends View implements SensorEventListener {
2         private Context ctx;
3         private float[] magVals;
4         private float[] accVals;
5         private final float kfilteringFactor = 0.05f;
```

```
6         private final double radiansToDegree = 57.2957795;
7         private double azimuth;
8         private double pitch;
9         private double roll;
10        private final double az = 3;
11        private final double cz = 1;
12        public double cx, cy;
13        public SensorManager sensorMan;
14        public boolean debug = false;
15        public Map<Integer, ARObject> arObjects;
16        public int screenwidth;
17        public int screenheigth;
18        String zeile0 = "";
19        String zeile1 = "";
20        String zeile2 = "";
21
22        public ARLayout(Context context) {
23                ..
24        }
25        @Override
26        public void onSensorChanged(SensorEvent evt) {
27                ...
28        }
29        @Override
30        public void onDraw(Canvas c) {
31                ...
32        }
33        @Override
34        public void onAccuracyChanged(Sensor sensor, int accuracy) {
35                ...
36        }
37 }
```

Welche Bedeutung die Propertys im einzelnen haben, wird nachfolgend durch die Erläuterung der einzelnen Methoden, dargestellt.

4.2.1 Konstruktor ARLayout

Der Konstruktor der ARLayout Klasse hat bereits einige wichtige Aufgaben und sieht wie folgt aus:

```
1 public ARLayout(Context context) {
2         super(context);
3         ctx = context;
```

```
4           cx = 0;
5           cy = 0;
6           sensorMan = (SensorManager) ctx.getSystemService(Context.
                SENSOR_SERVICE);
7           sensorMan.registerListener(this, sensorMan.getDefaultSensor(
                Sensor.TYPE_MAGNETIC_FIELD), SensorManager.
                SENSOR_DELAY_FASTEST);
8           sensorMan.registerListener(this, sensorMan.getDefaultSensor(
                Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_FASTEST
                );
9   }
```

- Aufruf Konstruktor von View (2)
- der Applikationscontext wird für die Sensoren benötigt (3)
- Kameraposition in kartesischen Koordinaten, wobei cz nicht gesetzt wird, da die Z-Position durch das Wellenfeldsynthesesystem festgelegt ist (4-5)
- Schnittstelle zu den Sensoren (6)
- es wird angegeben, dass Sensordaten vom Kompass mit möglichst geringer Verzögerung benötigt werden (7)
- außerdem werden Beschleunigungssensordaten mit möglichst geringer Verzögerung benötigt (8)

Da ARLayout SensorEventListener implementiert, kann beim registrieren für die Sensoren die eigene Referenz übergeben werden.

4.2.2 onAccuracyChanged

Die Methode onAccuracyChanged muss für das SensorEventListener Interface imeplementiert werden, ist für die Applikation allerdings nicht von Interesse.

```
1   @Override
2   public void onAccuracyChanged(Sensor sensor, int accuracy) {
3   }
```

Sollte sich die Genauigkeit eines Sensors ändern, so wird diese Methode aufgerufen. Das kann beispielsweise passieren, wenn die Position nicht mehr über GPS übergeben wird, sondern über AGPS. Für die im Konstruktor erwähnten Sensoren ist dies aber irrelevant. (Incb)

4.2.3 onDraw

Sobald der WindowManager davon ausgeht, dass die View nicht mehr aktuell ist, wird diese Funktion aufgerufen. Die View kann auch manuell als ungültig markiert werden, um beispielsweise neu zu zeichnen, sobald neue Sensordaten vorhanden sind.

```
1  @Override
2  public void onDraw(Canvas c) {
3          Paint paint = new Paint();
4          paint.setColor(Color.WHITE);
5          paint.setAlpha(150);
6          paint.setTextSize(24);
7          c.drawText(zeile0, 20, 20, paint);
8          c.drawText(zeile1, 20, 40, paint);
9          c.drawText(zeile2, 20, 60, paint);
10         for (ARObject obj : arObjects.values()) {
11                 obj.draw(c);
12         }
13 }
```

- der WindowManager übergibt die korrekte “Leinwand” c zum Zeichnen (2)
- Paint wird für einfache Zeichenaufgaben benötigt (3)
- um den Alpha Wert zu ändern, muss zuerst eine Farbe ausgewählt werden, da die Farbe selbst ursprünglich auch einen Alpha Wert enthält (4-5)
- es sollen Angaben zu den Neigungen des Geräts in die obere, linke Ecke gezeichnet werden (7-9)
- alle Augmented-Reality-Objekte sollen neu gezeichnet werden (10-12)

4.2.4 onSensorChanged

Dies ist die wichtigste Methode der ARLayout Klasse. Sie berechnet die Neigungen des Geräts und bestimmt die einzelnen Bildschirmpositionen der Augmented-Reality-Objekte. Bevor die Methode im Detail beschrieben wird, wird zunächst aufgezeigt, wie die Umrechnung von Weltkoordinaten zu Bildschirmkoordinaten funktioniert.

4.2.4.1 perspektivische Projektion

Für die Umrechnung von Welt- zu Bildschirmkoordinaten kann u.a. die perspektivische Projektion genutzt werden. Diese Projektion lässt Punkte in Ferne kleiner wirken, als Punkte

die näher dran sind. Diesen Effekt nennt man Perspektive.Bei dieser Projektion wird davon ausgegangen, dass man ein Objekt durch ein Kameraobjektiv beobachtet. So haben Kamera und zu betrachtendes Objekt eine Position und die Kamera zusätzlich Winkel, die ihre Neigung beschreiben. Folgende Variablen werden im dreidimensionalem Raum benötigt:

- $a_{x,y,z}$ die Position des zu projizierenden Objekts
- $c_{x,y,z}$ die Position der Kamera
- $\theta_{x,y,z}$ die Orientierung der Kamera
- $b_{x,y}$ die 2D Projektion von a

Um die Projektion b zu berechnen, muss zunächst die Position der Kamera von der Position des Objekts subtrahiert werden und anschließend um $-\theta$ gedreht werden. Man spricht von einer Kamera-Transformation. (en.b) Die Kamera-Matrix lautet:

$$\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{pmatrix} \begin{pmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{pmatrix} \begin{pmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} - \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} \right)$$

Anschließend wird mit:
$b_x = d_x / d_z$ und
$b_y = d_y / d_z$ die Projektion berechnet. Der erhaltene Punkt liegt zwischen -1 und 1. Er muss noch auf die Bildschirmgröße skaliert werden:
$screen_x = (b_x * (screenwidth/2)$
$screen_y = (b_y * (screenheigth/2)$
Weiterhin muss überprüft werden, ob d_z negativ ist. Ist d_z negativ, liegt das zu projizierende Objekt hinter der Kamera und soll nicht auf dem Bildschirm erscheinen. Mit dem letzten Schritt hat man die Koordinaten für den Bildschirm und das Objekt kann gezeichnet werden. (AS)

4.2.4.2 Sensordaten filtern

Die Sensordaten sind auch im Ruhezustand sehr instabil. Um eine relativ einfache und zugleich effektive Filterung vorzunehmen kann man wie folgt vorgehen:

```
1 sensorWert = (neuerSensorWert * kfilteringFactor) + (sensorWert * (1 -
2 kfilteringFactor));
```

Es handelt sich hierbei um eine Bildung des Mittelwerts. Für kfilteringFactor setzt man nun beispielsweise einen Wert von 0.05. Dadurch ändert sich der genutzte Sensorwert mit jeder Messung maximal um 5%.

4.2.4.3 Implementierung onSensorChanged

Die Berechnungen und das Auslesen der Sensoren kann wie folgt umgesetzt werden:

```
1  @Override
2  public void onSensorChanged(SensorEvent evt) {
3  switch (evt.sensor.getType()) {
4  case Sensor.TYPE_MAGNETIC_FIELD:
5          this.magVals = evt.values.clone();
6          break;
7  case Sensor.TYPE_ACCELEROMETER:
8          this.accVals = evt.values.clone();
9          break;
10 }
11
12 if (this.magVals != null && this.accVals != null) {
13         float[] R = new float[9];
14         float[] actual_orientation = new float[3];
15         SensorManager.getRotationMatrix(R, null, this.accVals, this.
               magVals);
16         SensorManager.getOrientation(R, actual_orientation);
17
18         azimuth = (actual_orientation[0] * kfilteringFactor) + (azimuth *
                (1 - kfilteringFactor));
19         pitch = (actual_orientation[2] * kfilteringFactor) + (pitch * (1
               - kfilteringFactor));
20         roll = (actual_orientation[1] * kfilteringFactor) + (roll * (1 -
               kfilteringFactor));
21         zeile0 = String.format("Azimuth:␣%3d°", (int) (azimuth *
               radiansToDegree));
22         zeile1 = String.format("Pitch:␣␣␣␣␣␣␣%3d°", (int) (pitch *
               radiansToDegree));
23         zeile2 = String.format("Roll:␣␣␣␣␣␣␣␣␣%3d°", (int) (roll *
               radiansToDegree));
24
25         double sinOx = Math.sin(-pitch); // pitch
26         double cosOx = Math.cos(-pitch); // pitch
27         double sinOy = Math.sin(roll); // roll
28         double cosOy = Math.cos(roll); // roll
29         double sinOz = Math.sin(azimuth); // azimuth
30         double cosOz = Math.cos(azimuth); // azimuth
31
32         for (ARObject obj : arObjects.values()) {
33                 double ax = obj.posx;
```

```
34                  double ay = obj.posy;
35
36                  double x = ax - cx;
37                  double y = ay - cy;
38                  double z = az - cz;
39
40                  double dx = cosOy * (sinOz * y + cosOz * x) - sinOy * z;
41                  double dy = sinOx * (cosOy * z + sinOy * (sinOz * y +
                        cosOz * x)) + cosOx * (cosOz * y - sinOz * x);
42                  double dz = cosOx * (cosOy * z + sinOy * (sinOz * y +
                        cosOz * x)) - sinOx * (cosOz * y - sinOz * x);
43
44                  double distance = x * x + y * y + z * z;
45                  distance = Math.sqrt(distance);
46                  obj.distance = distance;
47
48                  if (dz > 0) {
49                          double bx = dx / dz;
50                          double bz = dy / dz;
51
52                          obj.x = (int) (bx * (screenwidth / 2));
53                          obj.y = (int) (bz * (screenheigth / 2));
54
55                          obj.visible = true;
56                  } else {
57                          obj.visible = false;
58                  }
59
60          }
61  }
62  postInvalidate();
63  }
```

- diese Methode wird aufgerufen, sobald ein Sensor neue Daten liefert. Je nachdem welcher Sensor sich meldet, sollen die Daten in eine 3x3 Matrix kopiert werden (2-10)
- sobald angemeldete Sensoren Daten liefern, können Daten berechnet werden (12)
- getRotationMatrix gibt eine 3x3 Rotationsmatrix wieder, die die Orientierung der Kamera wieder gibt (15)
- getOrientation gibt drei eulersche Winkel der Orientierung aus der Rotationsmatrix in Radiant wieder (16)

- diese Werte sollten gefiltert werden, da der Beschleunigungssensor relativ instabile Werte zurückgibt (18-20)
- die Winkel sollen auf dem Bildschirm angezeigt werden und werden dafür in einen String für onDraw vorbereitet (21-23)
- damit anschließende Rechnungen nicht mehrfach durchgeführt werden, werden einige Werte vorberechnet (25-30)
- anschließend muss für jedes Augmented-Reality-Objekt die Berechnung zur Darstellung auf dem Bildschirm durchgeführt werden (32)
- a_z ist fest, da die Quellen des Wellenfeldsynthese-Systems immer auf der selben Höhe sind (33-34)

Für die in 4.2.4.1 genannten Transformationen gibt es zwei Möglichkeiten zur Umsetzung. Zum Einen kann die Android API für Matrix-Multiplikationen genutzt werden. Zum Anderem, so wie hier verwendet, durch Ausmultiplizieren der Matrizen, da damit einige Berechnungen für alle Quellen im Voraus getätigt werden können. Aus 4.2.4.1 folgt:

$$d_x = cos\theta_y(sin\theta_z(a_x - c_x) + cos\theta_z(a_x - c_x)) - sin\theta_y(a_z - c_z)$$
$$d_y = sin\theta_x(cos\theta_y(a_z - c_z) + sin\theta_y(sin\theta_z(a_y - c_y) + cos\theta_z(a_x - c_x))) + cos\theta_x(cos\theta_z(a_y - c_y) - sin\theta_z(a_x - c_x))$$
$$d_z = cos\theta_x(cos\theta_y(a_z - c_z) + sin\theta_y(sin\theta_z(a_z - c_y) + cos\theta_z(a_x - c_x))) - sin\theta_x(cos\theta_z(a_y - c_y) - sin\theta_z(a_x - c_x))$$

- mutlipliziert man die Kamera-Matrix aus, kann d_x, y, z ohne Matrizenrechnung berechnet werden (40-42)
- da die Quellen des Wellenfeldsynthese-Systems punktförmig sind, muss die Distanz berechnet werden, damit ein Kreis skaliert gezeichnet werden kann (44-46)
- für negative d_z ist das Objekt nicht sichtbar und es müssen keine weiteren Berechnungen folgen (48,57)
- ansonsten muss auf die 2D-Ebene projeziert werden (49-50) und anschließend auf den Bildschirm skaliert werden (52-53)
- postinvalidate führt dazu, dass die View als ungültig markiert wird und somit neu gezeichnet werden muss (62)

Mit diesen Methoden ist ARLayout bereits lauffähig und übernimmt sowohl auslesen, als auch Transformationen für die Darstellung.

4.3 ARObject

ARObject ist das Modell für die Augmented-Reality-Objekte, die auf dem Bildschirm dargestellt werden. Diese Klasse kapselt alle Informationen, die durch Wellenfeldsynthese-System und von ARLayout kommen. Dadurch kann recht einfach Beschriftung und allgemeines Erscheinungsbild verändert werden. Die Alternative wäre, direkt in ARLayout die Objekte zu zeichnen. Dadurch würde der Code allerdings sehr überladen und unübersichtlich wirken. Dieser Ansatz ist Objektorientiert und zudem deutlich wartbarer. Hier nun die Klasse im Überblick:

```
1  public class ARObject extends View {
2          Paint p;
3          public boolean visible;
4          public double posx;
5          public double posy;
6          public int x;
7          public int y;
8          public int id;
9          public double distance;
10         private int radius;
11         private String textZeile1;
12         private String textZeile2;
13         private String textZeile3;
14         private String textZeile4;
15
16         public ARObject(Context context) {
17                 super(context);
18                 p = new Paint();
19         }
20
21         @Override
22         public void draw(Canvas c) {
23                 if (visible) {
24                         textZeile1 = "#" + id;
25                         textZeile2 = "␣x:␣" + posx;
26                         textZeile3 = "␣y:␣" + posy;
27                         textZeile4 = String.format("%2.2fm", distance);
28
29                         radius = (int) (2 / distance * 100);
30                         p.setColor(Color.BLUE);
31                         p.setAlpha(50);
32                         c.drawCircle(x, y, radius, p);
33                         p.setColor(Color.WHITE);
```

```
34                          p.setTextSize(20);
35                          c.drawText(textZeile1, x - radius - 50, y -
                                radius, p);
36                          c.drawText(textZeile2, x + radius, y + radius +
                                20, p);
37                          c.drawText(textZeile3, x + radius, y + radius +
                                40, p);
38                          c.drawText(textZeile4, x + radius, y - radius, p)
                                ;
39                  }
40          }
41
42  }
```

- der Konstruktor ruft, wie immer bei einer View, den Super-Konstruktor auf (17)
- Paint für grundlegende Zeichenaufgaben wird instanziiert (18)
- ist das Objekt sichbar, kann es gezeichnet werden (23)
- neben dem eigentlichem Objekt sollen ID, Position und Distanz angezeigt werden (24-27)
- radius gibt den Pixelradius des zu zeichnenden Kreises, von der Distanz abhängig, an (29)
- der Kreis soll blau und leicht durchsichtig gemalt werden (30-32)
- die Schrift hingegen weiß um den Kreis herum (33 - 38)

Da alle Berechnungen von ARLayout durchgeführt werden, muss diese Klasse nur mit den übergebenen Parametern zeichnen. Selbst der Aufruf von draw() wird von ARLayout übernommen.

4.4 CustomCameraView

CustomCameraView soll das Bild der Kamera darstellen. Dies ist dank der Android API relativ unkompliziert. Hier die dazugehörige Implementierung:

```
1  public class CustomCameraView extends SurfaceView {
2          Camera camera;
3          SurfaceHolder previewHolder;
```

```
4         SurfaceHolder.Callback surfaceHolderListener = new SurfaceHolder.
              Callback() {
5             public void surfaceCreated(SurfaceHolder holder) {
6                 camera = Camera.open();
7                 try {
8                     camera.setPreviewDisplay(previewHolder);
9                 } catch (Throwable t) {
10                    throw new RuntimeException();
11                }
12            }
13
14            @Override
15            public void surfaceChanged(SurfaceHolder surfaceHolder,
                  int format, int w, int h) {
16                Parameters params = camera.getParameters();
17                params.setPreviewSize(w, h);
18                params.setPictureFormat(PixelFormat.JPEG);
19                camera.setParameters(params);
20                camera.startPreview();
21            }
22
23            @Override
24            public void surfaceDestroyed(SurfaceHolder arg0) {
25                camera.stopPreview();
26                camera.release();
27            }
28        };
29
30        public CustomCameraView(Context ctx) {
31            super(ctx);
32            previewHolder = this.getHolder();
33            previewHolder.setType(SurfaceHolder.
                  SURFACE_TYPE_PUSH_BUFFERS);
34            previewHolder.addCallback(surfaceHolderListener);
35            setBackgroundColor(Color.TRANSPARENT);
36        }
37 }
```

- SurfaceHolder wird typischerweise zusammen mit SurfaceView genutzt, um Größe, Form und einzelne Pixel zu bestimmen (1,3)
- sollte sich SurfaceView ändern, beispielsweise durch drehen des Geräts, sollte ein

Callback-Handler angegeben werden, damit das Kamera-Bild darauf angepasst werden kann (4)

- dazu muss bei der Erstellung der View die Ressource Camera zuerst geöffnet werden (6)
- und anschließend das Bild auf die View gelegt werden (8)
- ändert sich SurfaceView, müssen die Parameter nur weitergereicht werden (15-21)
- beim Beenden der Applikationen sollten Ressourcen selbstverständlich wieder frei gegeben werden (24-28)
- im Konstruktor wird der Hintergrund Transparent gemacht, da durch das Kamerabild eh nicht sichtbar, außerdem muss der Callback-Handler angemeldet werden und typisiert werden, wobei SURFACE_TYPE_PUSH_BUFFERS normalerweise für Videos und Kamera-Vorschau genutzt werden (30-36)

Mit der Instanziierung der Klasse, sowie dem hinzufügen zum Layout ist die Kamera-Vorschau läuffähig. Solange keine weiteren Funktionen, wie beispielsweise eine Fotofunktion, hinzukommen, muss nichts weiter angepasst werden.

4.5 WFSListener

Um zu verstehen, wie Daten des Wellenfeldsynthese-Systems zum Android-Geräte gelangen, soll die Klasse WFSListener näher erläutert werden. Hier zunächst der Quellcode:

```
1  public class WFSListener extends Thread implements OSCListener {
2          private final String connect = "/WONDER/stream/visual/connect";
3          private final String position = "/WONDER/stream/visual/connect";
4          private final String ping = "/WONDER/stream/visual/connect";
5          private final String pong = "/WONDER/stream/visual/connect";
6          private OSCPortOut sender;
7          private OSCMessage msg;
8          public Context ctx;
9          private OSCPortIn receiver;
10         public Map<Integer, ARObject> arObjects;
11         public byte[] ip = { (byte) 172, (byte) 16, (byte) 0, (byte) 12
               };
12         public int port = 58100;
13
14         public WFSListener() throws IOException {
```

```
15                  sender = new OSCPortOut(InetAddress.getByAddress(ip),
                        port);
16                  String[] host = { "android" };
17                  OSCMessage msg = new OSCMessage(connect, host);
18                  sender.send(msg);
19                  receiver = new OSCPortIn(sender.getPortOut());
20          }
21
22          @Override
23          public void run() {
24                  receiver.addListener(position, this);
25                  receiver.addListener(ping, this);
26                  receiver.startListening();
27          }
28
29          @Override
30          public void acceptMessage(Date time, OSCMessage message) {
31                  if (message.getAddress().equals(ping)) {
32                          msg = new OSCMessage(pong, message.getArguments()
                                );
33                          try {
34                                  sender.send(msg);
35                          } catch (IOException e) {
36                          }
37
38                  } else if (message.getAddress().equals(position)) {
39                          int id = (Integer) message.getArguments()[0];
40                          float x = (Float) message.getArguments()[1];
41                          float y = (Float) message.getArguments()[2];
42                          if (arObjects.containsKey(id)) {
43                                  arObjects.get(id).posx = x;
44                                  arObjects.get(id).posy = y;
45                          } else {
46                                  ARObject obj = new ARObject(ctx);
47                                  obj.id = id;
48                                  obj.posx = x;
49                                  obj.posy = y;
50                                  arObjects.put(id, obj);
51                          }
52                  }
53          }
54  }
```

Das ist eine alles andere als optimale Implementierung. Die IP und der Port zum Wellenfeldsynthese-System sind hart codiert und somit zur Laufzeit nicht mehr konfigurierbar. Weiterhin wird bei Verlust der Verbindung nicht erneut mit dem System verbunden. Die WONDER-Software sendet nachdem sich verbunden wurde periodisch eine “Ping” Nachricht, die mit einer “Pong” Nachricht beantwortet werden muss, da die Applikation sonst keine Aktualisierungen mehr erhält. Wenn die Verbindung nun kurzzeitig unterbricht, muss die Applikation mit dieser Implementierung neu gestartet werden. Dennoch sind wesentliche Bestandteile bereits vorhanden und sollen näher erläutert werden:

- im Konstruktor werden ein Port zum Versenden und Erhalten von Nachrichten erstellt und anschließend ein connect Befehl gesendet, um stetig Aktualisierungen über die Positionen der Quellen zu erhalten. Zu beachten ist weiterhin, dass für sender und reciever der selbe DatagrammSocket genutzt werden muss, da WONDER beim erhalt des connect Befehls dessen Port und Adresse nutzt (14-20)
- ohne Angaben, werden alle Nachrichten herausgefiltert, es muss folglich erst angegeben werden, welche Nachrichten einen interessieren (24-25)
- anschließend wird der Listener scharf gestellt (26)
- bei Eingang einer gewünschten Nachricht wird acceptMessage aufgerufen (30)
- zuerst muss geprüft werden, um was für eine Nachricht es sich handelt (31,38)
- bei einem ping Befehl wird eine Sequenznummer mitgeliefert, die so mit dem pong Befehl zurückgeschickt werden muss (32-36)
- handelt es sich um eine Positions-Nachricht werden dessen Informationen, also ID und Position, einfach an die in WFSVisualActivity beschriebene Map weitergereicht (39-50)

4.6 DTrackListener und StandardBody

Zum Erhalt der eigenen, aktuellen Position wird eine Klasse benötigt, die diese Daten von der DTrack-Software erhält und in eine nutzbare Form bringt. Der DTrackListener soll diese Daten erhalten und die Modellklasse StandardBody kann erhaltene Daten, die als Byte-Array ankommen wieder in den erwarteten ASCI-String konvertieren und anschließend parsen. Nun die Klassen im Detail:

4.6.1 DTrackListener

```
1  public class DTrackListener extends Thread {
2          public String adress = "230.230.230.230";
3          public int port = 5230;
4          private StandardBody body;
5          public boolean running = true;
6          private final String hello = "Hello";
7          private int refreshingRate = 200;
8          public ARLayout layout;
9
10         @Override
11         public void run() {
12                 InetAddress group = null;
13                 MulticastSocket s = null;
14                 while (running) {
15                         try {
16                                 group = InetAddress.getByName(adress);
17                                 s = new MulticastSocket(port);
18                                 s.joinGroup(group);
19                                 DatagramPacket packet = new
                                       DatagramPacket(hello.getBytes(), hello
                                       .length(), group, port);
20                                 s.send(packet);
21                                 s.receive(packet);
22                                 body.parseAndSet(packet.getData());
23                                 layout.cx = body.x;
24                                 layout.cy = body.y;
25                                 Thread.sleep(refreshingRate);
26                         } catch (Exception e) {
27                         }
28                 }
29                 try {
30                         s.leaveGroup(group);
31                 } catch (IOException e) {
32                 }
33         }
34 }
```

Die Parameter für den Multicast-Port sind in dieser Implementierung hart-codiert, aber zumindest von außen änderbar. Dennoch sind diese Parameter so noch nicht zur Laufzeit änderbar. Zum Programmablauf:

- Initialisierung der Adresse (16)

- Initialisierung des Multicast-Sockets (17)
- anschließend muss die Adresse zum Beitreten der Gruppe übergeben werden (18)
- es wird hello an die Gruppe geschickt, da sonst keine Nachrichten empfangen werden können (19-20)
- mit den erhaltenen Daten kann der Parser nun die gewünschte Position herauslesen (21-22)
- und die entsprechenden Werte an das ARLayout übergeben (23-24)
- Die Aktualierungsrate der DTrack-Software ist relative hoch, weshalb die der Thread nur alle 200ms die Positionierung prüfen soll (25)

4.6.2 StandardBody

Die nachfolgende Implementierung zeigt, wie die ASCI-Pakete der DTrack-Software geparst werden können.

```
1  public class StandardBody {
2          private final static Pattern line = Pattern.compile("\\r\\n6d.*")
               ;
3          private final static Pattern number = Pattern.compile("-?\\d+[.\\
               d+]*\\s");
4          public int id;
5          public double quality;
6          public double x;
7          public double y;
8          public double z;
9
10         public boolean parseAndSet(byte[] datagram) {
11                 String toParse = new String(datagram, 0, datagram.length)
                       ;
12                 Matcher getLine = line.matcher(toParse);
13                 String line = "";
14                 if (getLine.find()) {
15                         line = getLine.group(0);
16                 } else {
17                         // line, describing the needed bodytype, not
                               found
18                         return false;
19                 }
20                 Matcher matcher = number.matcher(line);
```

```
21
22                  for (int i = 0; i < 7; i++) {// first 6 tokens only
                        needed
23                      if (matcher.find()) {
24                              setValue(matcher.group(0), i);
25                      } else {
26                              return false;
27                      }
28                  }
29                  return true;
30          }
31
32          private void setValue(String s, int index) {
33                  switch (index) {
34                  case 2:
35                          id = Integer.parseInt(s);
36                          break;
37                  case 3:
38                          quality = Integer.parseInt(s);
39                          break;
40                  case 4:
41                          x = Integer.parseInt(s);
42                          break;
43                  case 5:
44                          y = Integer.parseInt(s);
45                          break;
46                  case 6:
47                          z = Integer.parseInt(s);
48                          break;
49                  case 0:// bodytype
50                  case 1:// number of bodys
51                  default:
52
53                  }
54          }
55  }
```

- zuerst muss geprüft werden, ob der erwartete Bodytyp eingegangen ist (11-19)
- anschließend werden alle Zahlen heraus gesucht (20)
- die ersten 6 Tokens sind entscheidend und in ihrer Reihenfolge immer gleich,deshalb kann die setValue Methode mit der Anzahl der Schleifendurchläufe aufgerufen werden (22-29)

Mit den in diesem Kapitel genannten Klassen ist eine einfach Implementierung zur Szenenvisualisierung eines Wellenfeldsynthese-Systems auf einem Android-Gerät möglich.

4.7 Test der Applikation

In diesem Abschnitt soll die Applikation auf einem Android-Gerät installiert und geprüft werden. Für den Test werden alle Komponenten der Wonder-Software-Suite auf einem Rechner installiert und ausgeführt. Eine akustische Ausgabe ist damit nicht möglich, allerdings soll damit nur gezeigt werden, dass die Kommunikation zum Wellenfeldsynthese-System funktioniert. Weiterhin werden mit einer kleinen Testapplikation von einem anderem Rechner, im selben Netzwerk, testweise Datagramme, wie von der DTrack-Software, versendet um die Kommunikations zum Tracking-System zu simulieren. Bei dem Android-Testgerät handelt es sich um ein Google Nexus One mit folgender Spezifikation:

Prozessor	Qualcomm QSD8250, 1 GHz
Betriebssystem	Android 2.3.3 (Gingerbread)
Arbeitsspeicher	512 MB
Kamera	5.0 Megapixel Farbkamera mit Auto Fokus
Sensoren	u.a G-Sensor, digitaler Kompass und Lage Sensor

Tabelle 4.1: Spezifikation Nexus One

Nach Start der Applikation ist auf dem Bildschirm das Kamerabild, sowie in der oberen, linken Ecke Angaben zur Lage des Geräts zu sehen. Durch Ausrichten des Geräts in Richtung, der in der WONDER-Software bereits hinzugefügten Quellen, werden diese Quellen auf dem Bildschirm mit zusätzlichen Informationen angezeigt. Je nach Lage, Eigenposition und Position der Quellen soll nachfolgendes Bild zeigen, wie die Applikation aussieht:

Zu sehen sind 2 Quellen mit ihrer ID oben links, der Distanz oben rechts und den dazugehörigen Koordinaten unten rechts. Dieses Gerät hat keinen dedizierten Grafikprozessor, sodass sämtliche Berechnungen zu Lasten der CPU geht. Das ist in diesem Fall von Vorteil, da Anhand der CPU-Last nun leicht bestimmt werden kann, wie ressourcenhungrig die Applikation ist und ob diese überhaupt flüssig laufen kann. Das Android-SDK enthält viele nützliche Werkzeuge für die Entwicklung und das Debugging einer Applikation, u.a. die Android Debug Bridge (kurz: adb), mit dessen Hilfe eine Ash-Shell auf dem Gerät aufgerufen werden kann. Auf allen Geräten und auch auf Emulatoren ist beispielsweise das Konsolen-Werkzeug top installiert, dass die CPU-Last einzelner Andwendungen anzeigen kann. Die CPU-Last von WFSVisual lag bei konstant 32%. Die Anwendung arbeitet folglich flüssig und für Erweiterungen der Applikation sind noch genügend Ressourcen vorhanden.

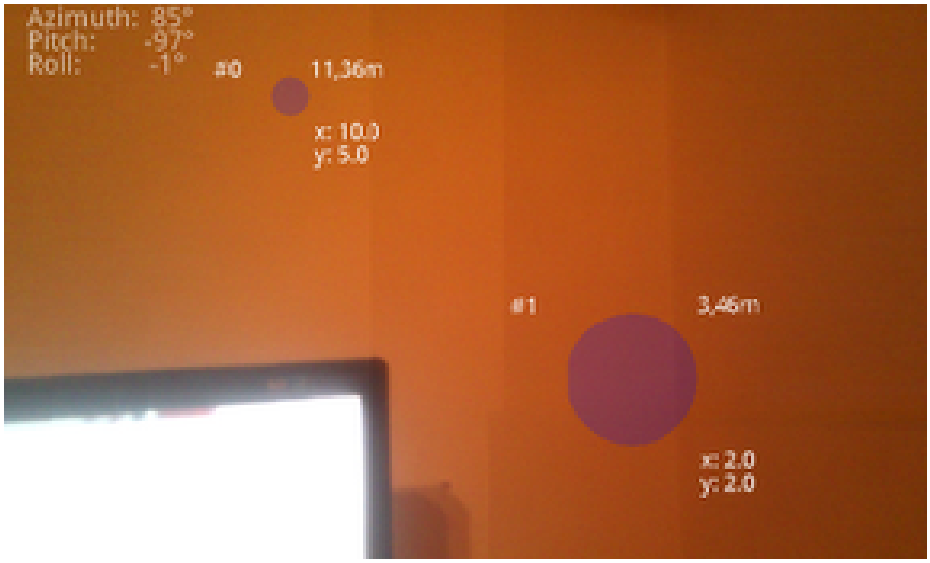


Abbildung 4.1: Beispielbild

Kapitel 5

Zusammenfassung und Ausblick

Dieses Kapitel gibt zum Abschluss wieder was nach dieser Arbeit erreicht wurde und was noch zu erwarten ist oder einfach zumindest denkbar ist.

5.1 Zusammenfassung

Die im Rahmen dieser Arbeit entwickelte Android-Applikation kann Szenen eines Wellenfeldsynthese-Systems visualisieren und über ein lokales Netzwerk kommunizieren. Zur Kommunikation wird das Open Sound Control-Protokoll genutzt, da die in dieser Arbeit eingesetzte Software zur in Betriebnahme des Wellenfeldsynthese-Systems, die WONDER-Softwaresuite, nur über dieses Protokoll nach außen kommuniziert. Dabei lassen sich Nachrichten auch an jedes andere externe System verschicken, wodurch eine Steuerung von Soundquellen des Systems denkbar ist. Für die Darstellung auf dem Android-Gerät wird das Kamerabild der Frontkamera auf dem Bildschirm angezeigt und durch Soundquellen und Informationen über die Quelle erweitert (Augmented Reality). Diese Informationen erhält das Android-Gerät durch das Open Sound Control-Protokoll und muss angekommene, kartesische Koordinaten erst durch eine perspektivische Projektion zu Bildschirmkoordinaten transformieren. Weiterhin erhält das Android-Geräte die Eigenposition über das lokale Netzwerk von der DTrack-Software. Das Kamera-Tracking-System verschickt ASCI Multicast Datagramme, die analysiert und aufgeteilt werden müssen (parsen), damit die Eigenposition in die Berechnungen der perspektivischen Projektion mit eingehen können. Damit ein für das Auge flüssiges Bild entsteht, müssen mehrfach in der Sekunde Daten des Kamera-Tracking-System verarbeitet werden, Quellpositionen verarbeitet werden und schließlich das Bild neu berechnet und angezeigt werden. Der Abschnitt 4.7 zeigte, dass die Szene mit heute gängigen Smartphones flüssig darstellbar ist. Mit einer geringen CPU-Auslastung von 32% auf einem Google Nexus One, das am 5. Januar 2010 erschienen ist, also einem relativ altem Smartphone, ist vorraussichtlich genügend Leistung vorhanden, um noch weitere Features einzubauen und sogar die graphische Gestaltung zu verbessern, vor allem da

gerade neu erschienene Tablets und Smartphones noch leistungsfähiger sind und teilweise sogar bereits mit Dual-Core CPUs ausgestattet sind.

5.2 Ausblick

Die hier entwickelte Android-Applikation ist zur Szenensteuerung noch nicht geeignet. Sie visualisiert die Szene bisher so einfach wie möglich und zeigt damit, dass eine solche Applikation bereits mit heutiger Hardware gut umsetzbar und nutzbar ist. Wie ein Test in Abschnitt 4.7 zeigt, sind noch genügend Ressourcen vorhanden, um die Applikation zu erweitern. Im folgenden Abschnitten soll darauf eingegangen werden, welche Erweiterungen denkbar sind und die Nutzbarkeit erhöhen.

5.2.1 Steuerung der Position der Quellen

Die Quellpositionen werden bisher angezeigt, jedoch lassen diese sich nicht über das Android-Gerät verschieben. Da die Höhe der Quellen durch die Höhe der Lautsprecher, des Wellenfeldsynthese-Systems vorgegeben ist, ist eine Transformation von Bildschirmkoordinaten (2D) zu Weltkoordinaten (3D) gut berechenbar. Für die Umsetzung wäre eine einfache Touchgeste denkbar. Die Quelle wird mit einem Finger per Drag'n'Drop an die gewünschte Position vorschoben, indem das Android-Gerät in die entsprechende Richtung geschwenkt wird. Weiterhin könnten zwei zusammenziehende und auseinanderziehende Finger, eine übliche Multitouch-Geste für eine Zoom-Funktion, dazu genutzt werden, um die Quelle vom und zum Nutzer zu bewegen.

5.2.2 Steuerung der Lautstärke

Um die Nutzbarkeit weiter zu erhöhen, könnte die Lautstärke über das Android-Gerät steuerbar gemacht werden. Für die Steuerung ist eine mit zwei Fingern durchgeführte Kreisbewegung um die Quelle herum vorstellbar. Diese Multitouch-Geste kann mit üblichen Lautstärkereglern verglichen werden, bei denen die Lautstärke im Uhrzeigersinn erhöht wird und gegen den Uhrzeigersinn gesenkt wird.

5.2.3 OpenGL

Die bisherige Anzeige ist einfach gehalten und diente bisher nur zur Analyse der Machbarkeit einer solchen Anwendung auf heutigen Android-Geräten. Die Umgestaltung mit OpenGL kann einige Vorteile bringen. Mit OpenGL können Quellpositionen dreidimensional dargestellt werden und mit Effekten ausgestattet werden. Dadurch könnte die Darstellung realistischer wirken, wodurch Realität und Virtualität näher zusammen rücken. Für die Nutzbarkeit

ist weiterhin ein Drahtgitter, das das Koordinatensystem darstellt, denkbar. So würde das Verschieben von Quellen erheblich erleichtert werden. Außerdem wäre eine Gesamtübersicht über die Szene nützlich. Dafür kann zum Einem ein kleines Radar mit dem eigenem Blickwinkel und den sichtbaren Quellen angezeigt werden und zum Anderem eine komplett neue Activity eingerichtet werden, die die Szene zweidimensional von oben zeigt.

Literaturverzeichnis

[AG] AG, Lawo: *Bregenzer Festspiele - Seebühne*. http://www.lawo.de/de/lawo-ag/referenzen/nach-produkten/mcseries/bregenzer-festspiele-seebuehne.html. – Stand: 22.07.2011

[A.R07] A.R.T. ; A.R.T. GMBH (Hrsg.): *ARTtrack DTrack User Manual*. 1.24.3. http://www.ar-tracking.de: A.R.T. GmbH, 2007. – Stand: 14.07.2011

[AS] ALT, Helmut ; SCHOLZ, Sven: *Die Kamera in eine 3D Umgebung*. http://www.inf.fu-berlin.de/lehre/WS06/19605_Computergrafik/doku/rossbach_siewert/camera.html. – Stand: 14.07.2011

[BBH+] BAALMAN, Marije ; BLASCHKE, Tobias ; HOHN, Torben ; KOCH, Thilo ; MOND, Hans-Joachim ; PLEWE, Daniel ; SCHAMPIJER, Simon: *WONDER - Wave field synthesis Of New Dimensions of Electronic music in Realtime*. http://swonder.sourceforge.net. – Stand: 14.07.2011

[Bon] BONSOR, Kevin: *How Augmented Reality Works*. http://www.howstuffworks.com/augmented-reality.htm. – Stand: 14.07.2011

[BP11] BECKER, Arno ; PANT, Marcus: *Android 2: Grundlagen und Programmierung*. dpunktVerlag, 2011. – ISBN 3898646777

[de.a] DE.WIKIPEDIA: *Erweiterte Realität*. http://de.wikipedia.org/wiki/Erweiterte_Realit%C3%A4t. – Stand: 14.07.2011

[de.b] DE.WIKIPEDIA: *Wellenfeldsynthese*. http://de.wikipedia.org/wiki/Wellenfeldsynthese. – Stand: 14.07.2011

[en.a] EN.WIKIPEDIA: *Augmented Reality*. http://en.wikipedia.org/wiki/Augmented_reality. – Stand: 14.07.2011

[en.b] EN.WIKIPEDIA: *Perspective Projection*. http://en.wikipedia.org/wiki/3D_projection#Perspective_projection. – Stand: 14.07.2011

[en.c] EN.WIKIPEDIA: *Wavefieldsynthesis*. http://en.wikipedia.org/wiki/Wave_field_synthesis. – Stand: 14.07.2011

[Gmb] GMBH eyePlorer: *Seefestspiele Mörbisch:Alle Fakten auf einen Blick*. http://de.vionto.com/show/me/Seefestspiele+M%C3%B6rbisch. – Stand: 27.07.2011

[Gob] GOBBETTI, Daniele: *mixare - Augmented Reality Engine*. http://www.mixare.org/usage/. – Stand: 14.07.2011

[Hel] HELMUTOELLERS: *Wellenfeldsynthese*. http://www.syntheticwave.de/Wellenfeldsynthese.htm. http://www.syntheticwave.de/Wellenfeldsynthese.htm. – Stand: 14.07.2011

[IDM] IDMT, Fraunhofer: *Wellenfeldsynthese*. http://www.idmt.fraunhofer.de/de/projekte_themen/wellenfeldsynthese.htm. – Stand: 27.07.2011

[Inca] INCORPORATED, Google: *Android SDK - Activity*. http://developer.android.com/reference/android/app/Activity.html. – Stand: 14.07.2011

[Incb] INCORPORATED, Google: *Android SDK - SensorListener*. http://developer.android.com/reference/android/hardware/SensorListener.html. – Stand: 14.07.2011

[Incc] INCORPORATED, Google: *Android SDK - SurfaceView*. http://developer.android.com/reference/android/view/SurfaceView.html. – Stand: 14.07.2011

[Incd] INCORPORATED, Google: *Android SDK - View*. http://developer.android.com/reference/android/view/View.html. – Stand: 14.07.2011

[Ince] INCORPORATED, QUALCOMM: *Qualcomm - Augmented Reality Kit*. http://developer.qualcomm.com/dev/augmented-reality. – Stand: 14.07.2011

[MBa] MUSIC, The Center For N. ; BERKELEY, Audio Technology (CNMAT) U.: *Introduction to OSC*. http://opensoundcontrol.org/introduction-osc. http://opensoundcontrol.org/introduction-osc. – Stand: 14.07.2011

[MBb] MUSIC, The Center For N. ; BERKELEY, Audio Technology (CNMAT) U.: *The Open Sound Control 1.0 Specification*. http://opensoundcontrol.org/spec-1_0. `http://opensoundcontrol.org/spec-1_0`. – Stand: 14.07.2011

[MBc] MUSIC, The Center For N. ; BERKELEY, Audio Technology (CNMAT) U.: *The Open Sound Control 1.0 Specification*. http://opensoundcontrol.org/implementation/liblo-lightweight-osc-api. `http://opensoundcontrol.org/implementation/liblo-lightweight-osc-api`. – Stand: 14.07.2011

[MBd] MUSIC, The Center For N. ; BERKELEY, Audio Technology (CNMAT) U.: *The Open Sound Control 1.0 Specification*. http://opensoundcontrol.org/implementation/java-osc. `http://opensoundcontrol.org/implementation/java-osc`. – Stand: 14.07.2011

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. August 2011	
Ort, Datum	Unterschrift