ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Medieninformatik

VR-Window: Interaktion zwischen Smartphone- und VR-Brillen-Nutzern

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Sebastian Schäf
sebastian.schaef@uni-ulm.de

Gutachter:
Prof. Dr. Enrico Rukzio

Betreuer:
Jan Gugenheimer

2015

„VR-Window: Interaktion zwischen Smartphone- und VR-Brillen-Nutzern“
Fassung vom 23. September 2015

Satz: PDF-LaTeX 2ε

Abstract

Kommerzielle *Virtual-Reality*-Systeme wie die *Oculus Rift* gewinnen seit ein paar Jahren wieder immer mehr an Beliebtheit. Und trotzdem besteht gerade bei solchen VR-Brillen noch immer das Problem, dass diese vollständig nach Außen abgeschirmt sind und dadurch keine Teilnahme von Außenstehenden erlauben. Da Smartphones sehr weit verbreitet sind, bietet es sich an eine Verbindung so zwischen diesen und VR-Brillen zu schaffen, dass ein Einblick von Außen ermöglicht wird. Ein Smartphone dient dann als Fenster in die virtuelle Umgebung und erlaubt ebenfalls Interaktionen mit dieser. Deshalb werden in dieser Bachelorarbeit Interaktionskonzepte zwischen VR-Brillen-Nutzern und Smartphone-Anwendern erarbeitet und diskutiert. Diese Konzepte beschäftigen sich mit Interaktionen um eine virtuellen Umgebung herum, die die beiden Anwender miteinander teilen. D.h. dass virtuelle Interaktionen innerhalb der künstlichen Umgebung, aber auch reale Interaktionen außerhalb dieser entwickelt werden. Das geschieht unter besonderer Beachtung der unterschiedlichen Benutzerschnittstellen der beiden Systeme. Die Interaktionskonzepte werden allgemein formuliert, damit sie als Basis für Weiterentwicklungen dienen. Außerdem werden sie benannt und strukturiert in Kategorien präsentiert, damit eine Übersicht geschaffen wird. Zusätzlich werden zu einigen ausgewählten Konzepten Beispiele für konkrete Anwendungen entworfen. Um diese zu präsentieren, wurde eine Demonstrationsapplikation im Zuge dieser Arbeit entwickelt, auf deren Implementierung und Umfang ebenfalls eingegangen wird. Es wird außerdem beschrieben, wie Teile dieser Applikation als Basis für neue Anwendungen dieser Art verwendet werden können.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Die Mensch-Computer-Interaktion bei *Virtual-Reality*-Systemen unterscheidet sich deutlich von der bei Smartphones. Durch die zusätzliche dritte Dimension und den immersiven Eindruck wird bei Forschungen zu VR-Benutzerschnittstellen vor allem auf die Natürlichkeit dieser Wert gelegt. D.h. dass Interaktionen in der virtuellen Umgebung möglichst der realen Welt nachempfunden werden. Auch bei Smartphones wird darauf Wert gelegt, jedoch unterscheiden sich deren Benutzerschnittstellen allein schon durch die Zweidimensionalität. Das Problem dabei ist, dass in beiden Feldern getrennt geforscht und dabei keine Verbindung zwischen diesen geschaffen wird. In dieser Arbeit wird diese Verbindung nun hergestellt und Interaktionskonzepte zwischen den beiden so entwickelt, dass die jeweiligen Vorteile der Benutzerschnittstellen der Systeme zur Geltung kommen. Damit wird eine Übersicht an Konzepten erstellt, die eine Basis zur Weiterentwicklung dieser schafft.

1.2 Aktuelle Situation

Im Moment sind kommerzielle VR-Geräte noch eher ein Nischenphänomen, aber gewannen in den letzten Jahren immer mehr an Bedeutung [25]. Vor allem nach der erfolgreichen Finanzierung der *Oculus Rift*, mit einem Volumen von 2.437.429 US-Dollar, begannen andere Firmen ebenfalls verstärkt Geräte in diesem Bereich zu entwickeln [31, 11]:
Google entwarf mit *Cardboard* [20] eine Komponente, die zu Smartphones hinzugefügt wird und diese damit zu einer vollwertigen VR-Brille werden lässt. Diese Variante ist auch sehr kostengünstig, da die Komponente nur aus einem faltbaren Karton, zwei Linsen und einem Magnetschalter besteht. Daher lässt sie sich gut für spontanere VR-Erlebnisse (Schule,

Besichtigung, Museum etc.) einsetzen, weil *Cardboard* um ein bereits mobiles Gerät herum entwickelt wurde. Ein weiterer Vorteil gegenüber Systemen wie der *Oculus Rift* ist, dass hier kein Zusatzsystem in Form eines schnellen Computers benötigt wird. Lediglich ein Smartphone, das aber bereits jeder Zweite in Deutschland besitzt [35].

1.3 Problemstellung

Um den dreidimensionalen Eindruck zu erschaffen, sind bei VR-Brillen die beiden Bildschirme jeweils direkt vor den Augen angebracht. Damit eine Immersion mit der virtuellen Umgebung, die darauf abgebildet wird, entsteht, sind die Brillen nach außen abgeschirmt. Speziell bei solchen VR-Systemen besteht deshalb das Problem, dass auch umgekehrt nicht von Außen hineingesehen werden kann. Das bedeutet, dass eine Außenstehende Person nicht nachvollziehen kann, was ein solcher VR-Anwender gerade erlebt.
Ein kleiner Schritt in die Richtung, diese Erfahrung auch für andere greifbar zu machen, besteht im Moment nur darin, die visuelle Wahrnehmung zu spiegeln (*Mirroring*) - d.h. den Bildschirminhalt, den der Spieler in der VR-Brille sieht, auf einem davon abgekoppelten anderen Bildschirm nochmals auszugeben. Dies löst zunächst auch das Problem, diesem von außerhalb genauere akustische Anweisungen geben zu können. Auch kann man den Träger der VR-Brille in eine bestimmte Richtung drehen, wenn dieser etwas bestimmtes nicht findet o.ä. Weitere Interaktionen, wie etwa ein aktives Eingreifen in das Geschehen oder eine Kommunikation innerhalb der virtuellen Welt, sind zwischen solchen Personen mit dieser Lösung aber nicht möglich. Insgesamt bleibt die Erfahrung des VR-Anwenders deshalb trotzdem noch eine sehr private.
Außerdem existiert gerade bei mobilen VR-Systemen wie *Google Cardboard* bezüglich des *Mirrorings* noch ein weiterer Nachteil: Da die Anwendung auf dem Smartphone selbst läuft, muss auch von dort aus der Bildschirminhalt gestreamt werden, was zu Resourceneinbußen und Bildverzögerungen führt. Letztere lassen sich dabei nur durch starken Bildqualitätsverlust umgehen.

1.4 Lösungsansatz

Da Smartphones sehr weit verbreitet sind, bietet sich hier die Möglichkeit eine Schnittstelle zwischen diesen und VR-Systemen zu schaffen. Die Idee ist es, mit dem Smartphone einen Einblick in die virtuelle Umgebung zu bekommen und so ein Fenster in die virtuelle Realität zu schaffen (vergl. *VR Window*).
Dazu soll dieselbe Anwendung sowohl auf dem Smartphone, als auch auf dem VR-Gerät ausgeführt werden. In Verbindung mit *Google Cardboard* muss dabei nur für ein einziges System entwickelt werden. Weil nun jedes Smartphone selbständig die Bildberechnungen durchführen kann, werden diese via Bluetooth gekoppelt und erhalten dadurch eine bidirektionale Verbindung, die es erlaubt Informationen auszutauschen. Es reicht hier aus, wenn dabei nur Positionskoordinaten und Rotationsvektoren des Avatars des VR-Nutzers in der virtuellen Umgebung an das zweite Smartphone (*Second-Screen*) übertragen werden. Dadurch kann das *Second-Screen*-Gerät die virtuelle Szene selbst darstellen. Der Vorteil gegenüber dem *Mirroring* ist, dass hier keine Leistungseinbußen auf dem VR-Smartphone entstehen und dass das *Second-Screen*-Gerät das Bild adaptiv für dessen Bildschirm berechnen kann. D.h., dass die Szene im Vollbild dargestellt werden kann, während beim *Mirroring* auch der Bildschirm des *Second-Screens* in die Darstellung für das jeweils linke und rechte Auge aufgeteilt und zusätzlich für die vorgeschobenen Linsen verzerrt ist (sh. 2.3). Durch die voneinander unabhängige Bildberechnung der Systeme entsteht ebenfalls die Möglichkeit für den *Second-Screen* die virtuelle Szene aus einem anderen Blickwinkel zu betrachten. In diesem Fall möchte man wiederum den VR-Anwender nicht im Unwissen lassen, von wo aus er beobachtet wird. Deshalb ist eine visuelle Darstellung der Position des *Second-Screens* als Avatar angebracht. Erlaubt man dem Zuschauer zusätzliche Interaktionen mit der virtuellen Umgebung, kann dieser aktiv in das Geschehen eingreifen

und wird somit zu einem vollständigen Teilhaber am VR-Erlebnis. Je nachdem wie diese Interaktionen angesetzt sind, kann der *Second-Screen* nun eine höhere Position durch Mehrwissen (z.B. Übersicht), niedrigere durch weniger Wissen oder eine gleichwertige gegenüber dem VR-Nutzer einnehmen. Eine gleichwertige Position entsteht, wenn alle dieselben Interaktionsmöglichkeiten besitzen oder aber unterschiedliche, die aber gleichwertig sind. Das könnten beispielsweise Interaktionen mit der virtuellen Umgebung sein, die jeweils an die speziellen Eingabe- und Ausgabeschnittstellen der Geräte angepasst und dafür etwa besser geeignet sind.

1.5 Einschränkungen

Neben den Nachteilen, dass die Anwendung auf beiden Geräten installiert und auch kompatibel sein muss, wird in dieser Arbeit auch der Fokus auf die Interaktion mit der virtuellen Welt und die Visualisierung in dieser gelegt. Außerdem soll in dieser Arbeit lediglich die Anwendung eines mobilen *Second-Screens* demonstriert werden, was bedeutet, dass keine VR- zu VR-Interaktion geschaffen wird.

1.6 Gliederung

Diese Arbeit ist wie folgt strukturiert:

Kapitel 2 **Hintergrund: Einführung in die virtuelle Realität:** In diesem Kapitel wird erläutert, was die virtuelle Realität ist und wie sie entstand. Außerdem wird die Funktionsweise aktueller VR-Systeme näher erläutert.
Kapitel 3 **Verwandte Forschung:** Dieses Kapitel beschäftigt sich mit Forschungen, die teilweise in den Bereich dieser Arbeit hineinreichen oder als Basis für Weiterentwicklungen dienen.
Kapitel 4 **Interaktionskonzepte:** In diesem Kapitel werden verschiedene Interaktionskonzepte erarbeitet, benannt und diskutiert.
Kapitel 5 **Implementierung:** Dieses Kapitel beschäftigt sich mit der verwendeten Technik und der Architektur bezüglich der Beispielanwendung. Außerdem wird erklärt, wie Teile

dieser für andere Projekte übernommen werden können.

Kapitel 6 **Beispielanwendung:** In diesem Kapitel werden die realisierten Konzepte benannt und die Funktionalität der Beispielanwendung erläutert.

Kapitel 7 **Fazit:** Dieses Kapitel fasst den Inhalt dieser Arbeit zusammen und präsentiert und diskutiert deren Ergebnisse.

2 Hintergund: Einführung in die virtuelle Realität

2.1 Allgemein

Die virtuelle Realität beschreibt eine von einem Computer in Echtzeit generierte künstliche Welt. Die Eingabe- und Ausgabeschnittstellen des Computersystems sind dabei darauf ausgelegt, die Interaktion und Wahrnehmung dieser virtuellen Welt möglichst real zu gestalten. Die Immersion entsteht beispielsweise durch die stereoskopische Bildwiedergabe in Verbindung mit einem gyroskopischen Sensor. D.h., dass entsprechend der realen Kopfausrichtung die Bilder für das linke und rechte Auge getrennt berechnet und ausgegeben werden, wie es in einer VR-Brille der Fall ist.

2.2 Geschichte

Die Geschichte der immersiven *Virtual Reality*, wie wir sie heute kennen, begann 1955 mit der Entwicklung der *Sensorama* durch Morton Heilig [23, 22]. Dieses System erinnert vom Design her an alte Spielhallenautomaten, wobei man hier aber seinen Kopf vollständig in einer Vorrichtung versenkt. Diese dient dazu, dass nicht nur Sinnesorgane wie die Augen, sondern auch Ohren, Nase und allgemein die Haut, angesteuert werden können. Der erste kommerzielle 4D-Film *The Sensorium*, in dem eine Fahrradtour durch Brooklyn gezeigt wird, konnte darin mittels stereoskopischem 3D und Stereo-Lautsprechern gezeigt werden. Selbst Gerüche, Wind und Vibrationen werden damit wiedergegeben.
Das erste VR-System, das am Kopf eines Nutzers angebracht ist und auf dessen Drehungen reagiert, ist das sogenannte *Schwert des Damokles*, welches von Ivan Sutherland entwickelt

wurde [36]. Das Gerät ist so schwer, dass es zusätzlich an der Decke des Testraums fixiert werden muss (daher der Name). Es besitzt jedoch dieselbe Funktionalität, wie wir sie von heutigen VR-Brillen wie der *Oculus Rift* kennen. Denn die Kopfausrichtung des Anwenders wird registriert und das angezeigte Bild auf den vor dem jeweiligen Auge angebrachten Bildschirm dementsprechend angepasst. Die genaue Funktionsweise einer solchen VR-Brille wird im folgenden Abschnitt 2.3 näher erläutert.
Im kommerziellen Sektor erschienen die ersten VR-Brillen in den späten 1980er Jahren, wie etwa das *EyePhone* von *VPL Research* [37]. Der breiten Masse waren Systeme wie diese wegen der hohen Preise von fast $10.000 aber nicht zugänglich. Jedoch wurde damit vor allem ein Anstoß in der Entwicklung von VR-Geräten im Computerspielsektor geschaffen: Unter anderem kündigte *Sega* eine VR-Brille für seine Konsolen an [33] und *Nintendo* zog wenige Jahre später mit dem *Virtual Boy* nach [10].
Diese Welle flachte im nachfolgenden Jahrzehnt aber fast vollständig ab, dessen Ursache wohl die zu dieser Zeit noch unausgereifte Technik ist. Erst ab den 2010er Jahren wurde wieder verstärkt in diesem Gebiet entwickelt. 2013 lieferte *Oculus* seine VR-Brille *Rift* an Entwickler aus, die hauptsächlich für den Einsatz am Heimcomputer entwickelt wurde [31]. Außerdem kündigte *Sony* ebenfalls sein VR-Gerät für die *PlayStation 4* an, das unter dem Namen *Project Morpheus* entwickelt wird [24] - *Google* probiert sich dagegen seitdem mit seinem kostengünstigen *Cardboard* an dem System des Smartphones aus, um dieses in eine vollwertige VR-Brille umzufunktionieren [20].
Auf die einzelnen hier genannten Techniken und Geräte wird im folgenden Abschnitt nun genauer eingegangen.

2.3 Systeme

Das Eintauchen in die virtuelle Realität wurde mit mehreren unterschiedlichen Systemen umgesetzt. Dieser Abschnitt beschäftigt sich mit den wichtigsten zwei davon und erklärt die Funktionsweisen dieser.

(a) Die CAVE am Electronic Visualization Laboratory der University of Illinois at Chicago

(b) Die *Vive* von *HTC*

Abbildung 2.1: Beispiele für die zwei wichtigsten VR-Systeme *CAVE* und *VR-Brille*

2.3.1 CAVE

Das *Cave Automatic Virtual Environment* (kurz: CAVE) entspricht einer stationären Realisierung. Denn hierfür wird mindestens ein kleiner Raum benötigt, in dem mehrere Projektionsleinwände platziert werden. Die virtuelle Umgebung wird auf diesen abgebildet, während sich ein oder mehrere Benutzer innerhalb dieser positionieren. Damit für jeden einzelnen darin ein räumlicher Eindruck der computergenerierten Welt entsteht, werden zusätzliche Techniken verwendet:
Allen voran stereoskopisches 3D mittels einer Brille, das ebenfalls durch unterschiedliche Methoden realisiert wird. Dazu gehört die *Anaglyphenbrille* [18], die verschiedene Farbspektren für das jeweilige Auge filtert, denn die zugehörigen Bilder werden bei dieser Technik im meist roten und blauen Farbspektrum übereinander gedruckt. Der Vorteil hierbei ist, dass sich ein 3D-Effekt auf herkömmlichen 2D-Fernsehern und Leinwänden ohne zusätzliche Technik erschaffen lässt. Ein Nachteil ist dagegen der Verlust von vielen Farbinformationen. *Shutter-Brillen* besitzen dieses Manko nicht, denn sie liefern die kompletten Originalbilder mit allen Farben. Dies ist möglich, da die jeweiligen Bilder für die Augen nacheinander auf einem Display gezeigt werden und die Brille in diesem Rhythmus das jeweilige Glas des anderen Auge abdunkelt. Letzteres geschieht durch Flüssigkristalle [21], was aber auch bedeutet, dass die Brille eine eigene Stromversorgung benötigt. Diese fällt wiederum bei der *Polarisationsfilterbrille* weg, denn die unterschiedlichen Bilder werden hier lediglich durch spezielle Polarisationsfilter projiziert, so dass diese mit denselben Filtern in der Brille im jeweils anderen Auge blockiert werden.

Ein räumlicher Eindruck in einem *CAVE* lässt sich aber auch ohne Brillen realisieren. Dazu werden Kameras im Raum bzw. an den Leinwänden angebracht, um die Position eines Nutzers zu bestimmen und dementsprechend das Bild aus dessen Blickwinkel zu rendern. Das bedeutet, dass das Bild nur in 2D ausgegeben wird und eine 3D-Täuschung nur dann entsteht, wenn der Spieler seinen Kopf bewegt. Bei einer statischen Haltung unterscheidet sich die Ausgabe also nicht von einer „normalen“ zweidimensionalen [30].
Insgesamt liegt der Vorteil von *CAVEs* darin, dass freie Bewegungen im Raum möglich sind und sich natürliche Interaktionen anbieten. Denn die Sicht auf den eigenen Körper ist nicht eingeschränkt und trügt somit nicht das natürliche Empfinden. Außerdem lassen sich auch zusätzlich reale Objekte innerhalb des CAVE platziern mit denen interagiert werden kann - zum Beispiel ein Cockpit eines Flugzeugs, das auch physikalisches Feedback zulässt.

2.3.2 VR-Brille

Diese sind mobilere Lösungen, da sie vom Nutzer direkt am Körper getragen werden. VR-Brillen wie die *Oculus Rift* [31], bestehen aus zwei Displays für je ein Auge in einer Vorrichtung vor dem Gesicht. Vor diesen sind zusätzlich jeweils Linsen angebracht, damit das Gesichtsfeld vergrößert und die Augen überhaupt das Bild scharf wahrnehmen können. Mit Hilfe eines integrierten Gyroskops [15] wird ständig der Roll-Nick-Gier-Winkel des Kopfes erfasst, um damit dementsprechend die zugehörigen Bilder der virtuellen Welt aus dem Blickwinkel des Anwenders berechnen zu können. Diese werden anschließend stereoskopisch auf den beiden Bildschirmen vor den Augen ausgeben.
Da die *Oculus Rift* nur Eingabe- (Kopfdrehung) und Ausgabegerät ist, benötigt diese zusätzlich einen Computer auf dem die VR-Anwendung ausgeführt wird. VR-Brillen wie *Google Cardboard* [20] sind dagegen vollständig mobil, weil hier das komplette VR-System dem Smartphone entspricht. Mit diesem lassen sich, verglichen mit einem schnellen Computer, natürlich keine hochauflösenden und aufwändigen Szenen in einer passablen Bildrate darstellen. Aber der Vorteil liegt hier darin, dass jeder sechste weltweit und sogar jeder zweite in Deutschland bereits ein Smartphone besitzt [34, 35] und sich ein solcher User nur zusätzlich die kostengünstige „Pappbrille“ (*Cardboard*) für ein vollwertiges VR-System zulegen muss. Ein Nachteil von VR-Brillen gegenüber dem *CAVE* ist etwa, dass die eingebauten Gyroskope lediglich den Roll-Nick-Gier-Winkel des Kopfes erkennen können, aber nicht dessen Position. Bewegt sich der Spieler im realen Raum oder versucht er mittels einer Neigung

des Körpers/Kopfes um eine Ecke zu schauen, wirkt sich das nicht auf die Kameraposition in der virtuellen Umgebung aus. Vice versa gelingt mit einem *CAVE* beispielsweise keine vollständige $360°$-Abdeckung und die Materialkosten und Platzaufwand dafür sind enorm gegenüber einer VR-Brille.

3 Verwandte Forschung

3.1 Fort Bragg VR-Simulator

Abbildung 3.1: Ein Squadleader behält die Übersicht über seine Truppe in der virtuellen Trainingsumgebung

Die US-Army trainiert verschiedene Squads seit 2012 in einem VR-Simulator [16]. Damit sollen vor allem Szenarien geübt werden, die sich nur selten in einer realen Militärübung umsetzen lassen. Für in etwa Luftangriffsübungen wird u.a. angemessenes Wetter benötigt oder für eine Übung für das Sicherstellen einer Zielperson z.B. ein größeres freies Areal. Solche Dinge lassen sich im Simulator zwar noch nicht komplett real umsetzen, dafür aber um einiges kostengünstiger und vor allem variabler, denn das Übungsareal lässt sich mit nur wenigen Klicks verändern oder anpassen.
Um an einem solchen Szenario teilzunehmen, beziehen die Soldaten auf einem sogenannten *module pad* Stellung, auf dem sie physikalisches Feedback zu ihren Aktionen im virtuellen Raum erhalten. Ein Squad-leader oder Trainingsleiter erstellt eine Mission, kann diese auch dynamisch verändern, während diese gerade im Gange ist und behält insgesamt den Überblick über das ganze Geschehen (Abb. 3.1). Zusätzlich gibt es noch

diverse andere Einrichtungen, die es u.a. ermöglichen bewegliche Objekte (wie Fahrzeuge) mittels Computer in der virtuellen Welt zu steuern oder bereits animierte (wie Vögel oder Hunde) darin zu platzieren.
Nach einer Übung kann diese auch wiedergegeben und damit ein umfassendes Feedback gegeben werden - auch für jeden einzelnen Soldaten individuell, weil man sich komplett in dessen Situation versetzen kann.
Aus diesem Projekt sind daher verschiedene Arten von stationären *Second-Screens* hervorgegangen, weshalb es für diese Arbeit relevant ist. Unter anderem das Interaktionskonzept des Squad-leaders, der mit einem Computer als *Second-Screen* den Überblick über seine Soldaten behält und deren VR-Erlebnisse manipuliert. Außerdem die Möglichkeit, dass weitere Akteure in die virtuelle Umgebung ohne ein VR-System einsteigen können und darin ein vollwertiger Teilnehmer werden.
Die Abgrenzungen zu den später in dieser Arbeit entwickelten Interaktionskonzepten bestehen darin, dass der *Fort Bragg VR-Simulator* nicht mobil einsetzbar ist, keine Anwendung eines Smartphones und deshalb auch keine Steuerung über ein Gerät mit Multi-Touch Bildschirm enthält. Hauptsächlich aber die Entwicklung des Systems für den Spezialfall einer Militärsimulation und nicht für eine Heimanwendung.

3.2 Haptic Turk

Abbildung 3.2: *Haptic Turk* mit *OculusRift*

Physikalisches Feedback für Aktionen in der virtuellen Realität wird hauptsächlich von stationären und meist größeren Geräten realisiert, wie z.B. in einem Fahr- oder Flugsimulator. Am Hasso-Plattner-Institut hat man deshalb nach mobileren und kostengünstigeren Alternativen gesucht - das Ergbnis war *Haptic Turk* [17]:
Die Grundidee dabei ist, dass stationäre Gerätschaften durch Menschen ersetzt werden, die letztendlich das physikalische Feedback mit Muskelkraft auf den Spieler umsetzen. Als Beispielanwendung wurde u.a. ein Drachenflug-Spiel entwickelt, in dem der Spieler durch eine Landschaft gleitet, die mit Windrädern, Zeppelinen und Wetterveränderungen ausgeschmückt ist. Dazu wird der Spieler, wie beim Drachenflug, in eine horizontale Lage versetzt, indem ihn vier andere Personen jeweils unterhalb der Arme und an den Füßen mittels Bändern halten (siehe Abb. 3.2). In dieser Haltung können die einzelnen *Aktoren* (Träger) z.B. mit wechselseitigem Anheben bzw. Absenken des Spielers Rollbewegungen realisieren. Damit ein solches physikalisches Feedback auch zum richtigen Zeitpunkt geschieht, bringen die Aktoren ihre Smartphones an den Bändern an und sehen darauf ihre Anweisungen. Letztere wurden ähnlich deren in *Guitar Hero* visualisiert - d.h. die Aktoren sehen 7 Sekunden in die Zukunft und können sich damit auf ein mögliches Ereignis vorbereiten.
Mit *Haptic Turk* wurde also ein Interaktionskonzept außerhalb der virtuellen Realität entwickelt. Denn die *Aktoren* können nicht direkt selbst in die virtuelle Umgebung blicken, jedoch durch das Minispiel erahnen was darin vorgeht. Dieses lässt sich ebenfalls auf einem *Second-Screen* darstellen, weshalb es für diese Arbeit von Bedeutung ist.
Jedoch wird dieses Konzept in dieser Arbeit noch erweitert, so dass ein *Second-Screen*-Anwender beispielsweise einen eigenen Blickwinkel in die virtuelle Welt erhält oder selbständig, d.h. ohne Anweisungen, handelt.

3.3 3D Positioning Techniques for Multi-touch Displays

Mit Smartphones und Tablets hat sich auch die Interaktion mittels Multi-Touchscreen etabliert. Dadurch ist diese Methode aber stark an die zwei Dimensionen der flachen Bildschirme angepasst und hauptsächlich in diesem Bereich erforscht. Deshalb hat Martinet et al. zwei verschiedene Interaktionsarten entwickelt und evaluiert, mit denen sich auch Objekte im dreidimensionalen Raum manipulieren lassen [28].
Zur Demonstration wurden mehrere Kugeln in einer virtuellen Umgebung platziert, die ein

Abbildung 3.3: Illustrationen der (i) *Z-Methode* und (ii)+(iii) *Multitouch-Viewport-Methode*

User später in dieser umpositionieren soll. Zum einen wurde ihm dazu die *Z-Methode* (Abb. 3.3(i)) bereitgestellt, die es erlaubt aus einer Perspektive heraus zu agieren: Der Benutzer tippt mit einem Finger auf ein Objekt, das durch einen *Raycast* ermittelt wird und kann dieses dann durch Wischbewegungen des Fingers in zwei Dimensionen verschieben (X und Y). Wird nun ein zweiter Finger auf das Display gelegt, dient dieser als „Regler“ für die Z-Positionierung, indem dieser nach unten oder oben auf dem Bildschirm bewegt wird.
Zum anderen wurde die *Multitouch-Viewport-Methode* (Abb. 3.3(ii)+(iii)) entwickelt, bei der mehrere Blickwinkel verwendet werden. Im ersten lässt sich, bei einer Fingerberührung auf ein Objekt, ebenfalls in X- und Y-Richtung verschieben. Der zweite entspricht dagegen einer seitlichen Ansicht des Objekts lässt somit die Z-Positionierung durch eine links-rechts-Verschiebung zu.
Einfache und intuitive Interaktionsmethoden wie diese sind von Bedeutung, wenn nicht nur ein passiver Einblick in eine virtuelle Welt bereitgestellt werden, sondern diese auch von Geräten wie Smartphones manipulierbar sein soll, die keine räumliche Wiedergabe zulassen. Wichtig für diese Arbeit war jedoch hauptsächlich das Steuerungsprinzip auf einem Multi-Touch Bildschirm. Dieses wird für ein in dieser Arbeit entwickeltes Interaktionskonzept übernommen, aber die zugehörige Manipulation in der virtuellen Umgebung erweitert, so dass mehr als nur eine Umpositionierung von Objekten ermöglicht wird.

3.4 Shared Screen

VR-Erlebnisse sind sehr persönlich - das liegt einerseits an dem eigenen Blickwinkel in die virtuelle Welt, aber auch daran, dass kein Außenstehender in das Geschehen hineinblicken kann. Unter anderem bei Computerspielmessen ist das aber ein Problem, weil anstehende

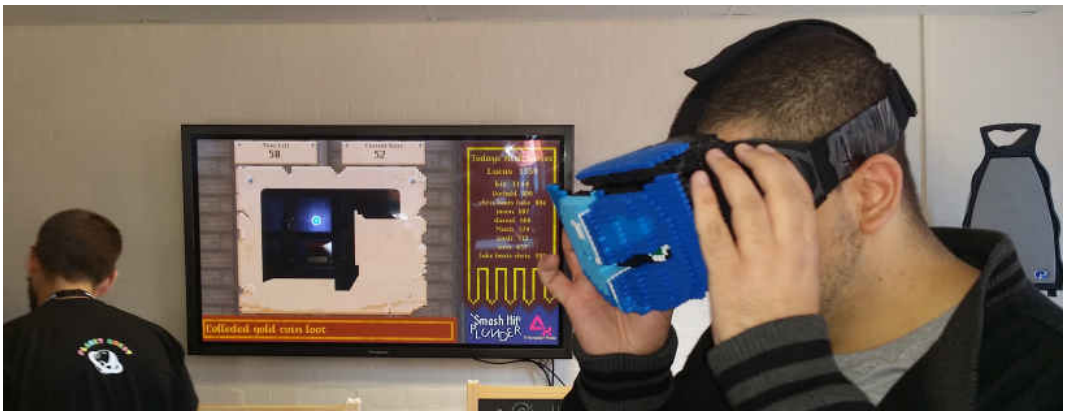


Abbildung 3.4: *GearVR*-Spieler mit seinen Statistiken und seiner Position im Hintergrund

Zuschauer keinen Eindruck davon bekommen können, was sie erwartet. Hauptsächlich aber, weil man den Testern keine Anweisungen geben kann, falls diese sich nicht zurechtfinden. Letzteres hat die Entwickler von *TriAngularPixels* dazu bewegt, den *Shared Screen* [12] zumindest für ihre Demotouren um ihr Spiel *Smash Hit Plunder* zu entwickeln.
Sobald also ein Tester mittels *GearVR* die virtuelle Umgebung erkundet, verbindet sich das Smartphone mit einem Server, der auf einem nebenstehenden Computer läuft. Zu diesem werden alle nötigen Informationen, wie Koordinaten, Drehung oder Highscore, übertragen. Auf dem Server läuft ebenfalls eine Version des Spiels, die sich aber von der grafischen Oberfläche her unterscheidet. Somit können mit Hilfe der übertragenen Spielerdaten u.a. ein Spielerklon auf dem Server erstellt und die Statistiken ausgegeben werden. Angezeigt wird dies z.B. auf einem Fernseher, wie in Abbildung 3.4 zu sehen ist.
Das Entwicklerteam hat einerseits durch das wegfallende Ratespiel, wo sich der Spieler gerade befindet, positive Erfahrungen gesammelt. Damit war es möglich, den Testern einfacher das Spiel näher zu bringen und auch ggf. diese zu unterstützen. Andererseits hatte der *Shared Screen* auch positive Auswirkungen auf die anstehenden Menschen, denn „[diese] bekommen eine allgemeine Ahnung davon, um was es in dem Spiel geht und falls sie es gerade gespielt haben, haben sie die Möglichkeit ihren Freunden zuzurufen, wenn diese eine höhere/niedrigere Punktzahl erreichen, als sie selbst“. Ein zusätzlich angezeigter Timer soll ebenfalls die anstehenden Spieler etwas beruhigt haben, weil dadurch wohl die Wartezeit besser eingeschätzt werden konnte.
Erweiterungsideen zum *Shared Screen* gibt es ebenfalls: Etwa die Ausgabe des Akkustandes, dass es dem Team erlaubt, rechtzeitig den Akku zu wechseln oder die Aufzeichnung

der Spielerdaten auf dem Server. Das erlaubt das spätere Generieren von Heatmaps, aber auch die komplette Wiedergabe eines Spielerlebnisses.
Interessant ist diese Entwicklung für diese Arbeit deshalb, weil der *Shared Screen* ebenfalls auf einem *Second-Screen*-Smartphone abgebildet werden kann. Dadurch ergibt sich auch die Abgrenzung zu dieser Arbeit, denn der *Shared Screen* wurde lediglich für Messen und Entwicklungszwecke entwickelt, jedoch nicht für Heimanwender und Smartphones.

3.5 Collaborative Mixed Reality Environments

(a) 3D-Modell

(b) Darstellung auf einem Desktop-Computer

(c) Darstellung auf dem Smartphone des Nutzers, der die reale Umgebung teilt

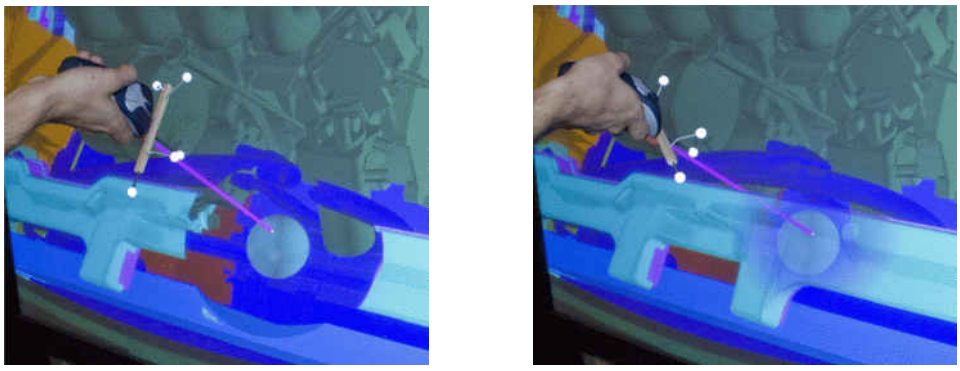
(d) Darstellung auf einem Smartphone eines Nutzers, der virtuell am Spiel teilnimmt

Abbildung 3.5: Das 3D-Modell der realen Umgebung und drei unterschiedliche Darstellungsweisen

Im Zuge dieses Forschungsprojekts wurde das *BEAMING*-System ins Leben gerufen, das es ermöglicht eine oder mehrere Personen virtuell an einen realen Ort zu versetzen inklusive Personen, die sich dort aufhalten [32]. Um diese Idee herum wurden mehrere diverse Anwendungsfälle demonstriert, wobei hier vor allem das dabei entwickelte Spiel *scavenger hunt* von Interesse ist. Dabei begibt sich ein Spieler mit einem Smartphone an einen realen Ort, wie zum Beispiel einer Bibliothek und startet die *BEAMING*-App. Das Display zeigt ihm nun seine Position auf einer Karte aus der Draufsicht (Abb. 3.5(c)). Da sich die Aufgaben im Spiel nicht allein lösen lassen, treten ihm nun mehrere andere Teilnehmer bei, die die verschiedensten Systemen dafür benutzen können. Unter anderem wurde hierfür ein *CAVE* [4] benutzt, das einen Spieler an denselben Ort versetzt, indem dieser virtuell dargestellt wird. Für diese Darstellung wurde ein *SecondLife* [26]-Client verwendet, der auch problemlos für andere Systeme wie einem Desktop-Computer (Abb. 3.5(b)) oder einem weiteren Smartphone (Abb. 3.5(d)) realisierbar war. Die einzelnen Spieler können sich nun gegenseitig in der virtuellen Welt sehen und interagieren - dazu werden diese als Avatare auf

den *SecondLife*-Clienten und als Punkte in der Draufsicht abgebildet. Alle Teilnehmer können sich nun frei in dieser Umgebung bewegen, wobei die Position des *Locals* (Person am realen Ort) mittels GPS in dessen Smartphone bestimmt wird. Ziel des Spiels ist es verschiedene Boxen zu finden, die in der realen Umgebung verteilt und mit QR-Codes versehen sind. Dementsprechend sind diese ebenfalls in der virtuellen Umgebung repräsentiert. Befinden sich alle gemeinsam in der Nähe einer Box, wird eine bildschirmfüllende Aufgabe angezeigt, die es nun gemeinsam zu lösen gilt.
Diese Idee der Interaktion zwischen verschiedenen Systemen soll es ermöglichen, mehrere Nutzer virtuell an einen realen Ort zu versetzen, an dem sich ein einziger aus der Gruppe befindet. Quasi ein virtuelles Fenster zu einem realen Ort, um dort gemeinsam etwas zu unternehmen. Wichtig für diese Arbeit ist zudem das Zusammenspiel verschiedenster Systeme und die spezielle Anpassung der Anwendung für diese.

3.6 Show-Through Techniques



(a) *Cutaway*-Methode (b) *Transparency*-Methode

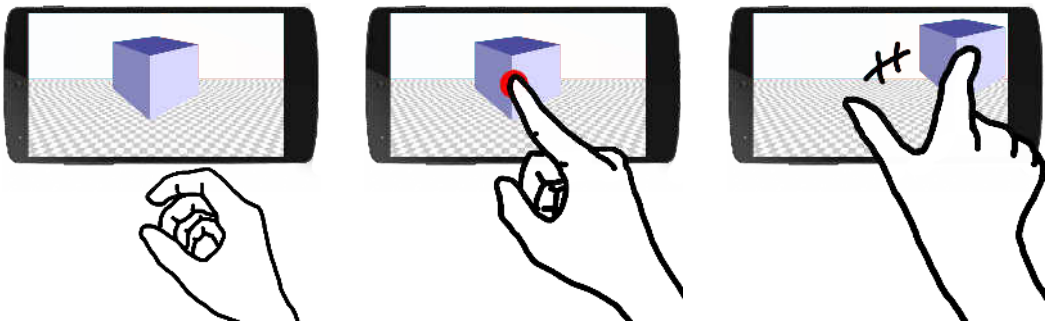
Abbildung 3.6: Die beiden Freistellungsmethoden jeweils auf dasselbe Objekt angewandt

Mehrere Benutzer in einem VR-System besitzen jeweils individuelle Blickwinkel auf die virtuelle Umgebung. Dadurch entsteht oft das Problem, dass für einen Nutzer ein Objekt im virtuellen Raum sichtbar ist, aber für einen weiteren verdeckt. Letzterer muss dann meist seine Position wechseln. Argelaguet et al. hat deshalb zwei verschiedene Konzepte entworfen, die es erlauben, gezeigte Objekte aus jeder Blickrichtung heraus sichtbar zu machen [13].

Zu Demonstrationszwecken und Evaluierung wurde hier ein Motorblock in der virtuellen Umgebung verwendet. Zeigt ein User mittels eines Pointers auf ein Objekt, wird dieses ermittelt und durch zwei verschiedene Techniken so „freigelegt“, dass die blockierende Gegenstände zwischen Betrachter und Objekt gar nicht mehr oder nur teilweise sichtbar sind. Dazu wird ein Raycast vom Betrachter zum Objekt verwendet, der die Texturen der zwischenliegenden Gegenstände verändert. Bei der *Cutaway*-Methode (Abb. 3.6(a)) werden diese in einem festgelegten Radius um den Raycast herum komplett ausgeblendet bzw. unsichtbar gemacht, so dass ein Fenster durch Blockierungen hindurch entsteht. Die *Transparency*-Methode (Abb. 3.6(b)) funktioniert größtenteils gleich, blendet aber die Texturen nicht komplett aus, sondern lässt diese halbtransparent mit einem äußeren Fade-Out erscheinen.
Die Evaluation hat keinen signifikanten Unterschied zwischen den beiden Techniken festgestellt, dagegen jedoch mehrere Vorteile gegenüber der Nichtverwendung dieser. U.a. dass verschiedene User nicht mehr nah aufeinander rücken müssen und damit einen komfortableren Abstand zueinander halten können.
Solche Lösungen sind für diese Arbeit relevant, weil die einzelnen Aktoren teilweise auch verschiedene Blickwinkel in der virtuellen Welt besitzen. Der Unterschied zu dieser Arbeit liegt jedoch darin, dass die Methoden für einen Einsatz vor einer Leinwand entwickelt wurden, wobei sich die Aktoren im selben Raum aufhalten. Die später entwickelten Interaktionskonzepte dieser Arbeit setzen aber einen Fokus auf einen mobilen Einsatz, bei dem der reale Abstand der Aktoren nicht zwingend dem virtuellen entspricht.

4 Interaktionskonzepte

Die Vision ist es umstehende Personen mit deren Smartphones in das Erlebnis eines VR-Nutzers zu integrieren. Dazu werden in diesem Kapitel verschiedenste Interaktionsarten in und außerhalb der geteilten virtuellen Umgebung (VU) erarbeitet und diskutiert. Diese Konzepte teilen wir zunächst in drei Kategorien ein:



(a) Zuschauer: keine Interaktion möglich

(b) Passiver Teilnehmer: Hinweisen auf Inhalte in der VU möglich

(c) Aktiver Teilnehmer: Manipulation der VU möglich

Abbildung 4.1: Einteilung der Interaktionskonzepte

Zuschauer Diese Kategorie bespricht Konzepte, die es einem Außenstehenden erlauben eine Einsicht in die VU zu erhalten. Dabei findet zwischen diesem und dem VR-Nutzer keinerlei Interaktion innerhalb der VU statt (Abb. 4.1(a)).

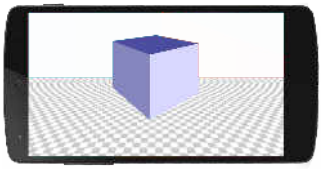
Passiver Teilnehmer Ein Außenstehender als *passiver Teilnehmer* kann mit verschiedenen Zeigewerkzeugen visuelle Hinweise geben oder etwas erläutern, jedoch nicht aktiv in das Geschehen eingreifen (Abb. 4.1(b)).

Aktiver Teilnehmer Ein *aktiver Teilnehmer* kann Objekte in der VU manipulieren, erschaffen oder zerstören. Er ist somit ein vollständiger Teilhaber an den Ereignissen innerhalb der virtuellen Welt (Abb. 4.1(c)).

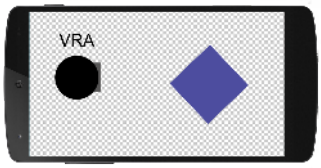
Die in dieser Arbeit entwickelten und benannten Konzepte wurden wie folgt in die drei Interaktionskategorien eingeteilt:

Zuschauer	Passiver Teilnehmer	Aktiver Teilnehmer
· Akustischer Anweiser	· Zeiger	· Aufgabenteilung
· Kommentator	· Objektmarkierung	· Mitspieler
· Regisseur	· Zeichenwerkzeuge	· Gottesmodus
· Physikal. Feedbackgeber		

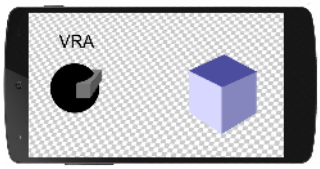
Es wird außerdem in den nachfolgenden Abschnitten, in denen diese Konzepte beschrieben werden, auf drei Blickwinkel-Modi des Außenstehenden referenziert. Diese Modi beschreiben wie der Außenstehende, der von nun an *Second-Screen-Aktor* (SSA) genannt wird, in die VU blickt in der sich der *Virtual-Reality-Aktor* (VRA) befindet.



(a) Klon: Derselbe Blickwinkel wie VRA



(b) Übersicht: Überblick über VU



(c) Spectator: Freie Blickwinkelauswahl

Abbildung 4.2: Drei verschiedene Blickwinkel in die VU auf dem *Second-Screen*

Klon Der *Klon*-Modus entspricht dabei der Hineinversetzung des SSA in den VRA, so dass beide exakt dieselbe Ansicht teilen (Abb. 4.2(a)).

Übersicht Dieser Modus erlaubt es dem SSA eine größere oder allgemein andere Perspektive auf die virtuelle Welt zu erhalten, damit dieser einen übergeordneten Überblick über das Geschehen darin besitzt. Beispielsweise eine Draufsicht auf ein Labyrinth, in dem sich der VRA aufhält (Abb. 4.2(b)).

Spectator Der *Spectator* ($\neq$ *Zuschauer*) beschreibt die Möglichkeit der freien Bewegung des SSA im virtuellen Raum. Dieser kann sich darin also möglicherweise in der VU vollständig frei bewegen oder verschiedene Blickwinkel auswählen. Letztere können statische oder dynamische Kameras im Raum sein oder aber an den VRA gebunden (Rückblick, Schulterblick, 3rd Person etc.) (Abb. 4.2(c)).

4.1 Zuschauer

4.1.1 Akustischer Anweiser

Der SSA besitzt als aktustischer Anweiser die Möglichkeit in die VU zu sehen. Weil er sieht was darin geschieht, kann er dem VR-Nutzer dementsprechend außerhalb der virtuellen Welt akustische Hinweise geben.
Das vereinfacht zunächst den Prozess einem VR-Nutzer Anweisungen geben zu können,

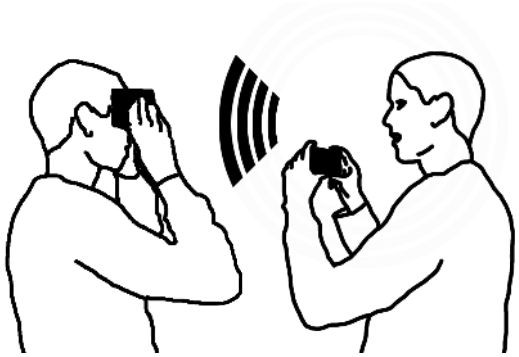


Abbildung 4.3: Ein VRA bekommt akustische Anweisungen von einem SSA

falls dieser noch nicht mit der VR-Anwendung vertraut ist. Denn dadurch, dass der SSA das Geschehen in der VU einsehen kann, muss er dieses nicht durch ständiges Nachfragen ermitteln. Dazu bietet sich vor allem der *Klon*-Modus an, weil hier auch etwa gesehen werden kann, welcher Menüpunkt vom VRA anvisiert wird oder welche Objekte außerhalb seines Blickwinkels liegen.
Letzteres kann ebenfalls als Basis für ein Spielprinzip dienen, bei welchem der SSA über mehr Informationen gegenüber dem VRA verfügt. Das ist vor allem dann der Fall, wenn der SSA den *Second-Screen* in der *Übersicht* verwendet. Als Beispiel dazu platzieren wir den VRA in einem Labyrinth. Der SSA bekommt durch eine Draufsicht einen kompletten Überblick über dieses und die darin herumlaufenden Monster. Dadurch erhält er die Möglichkeit den VRA vorzuwarnen oder absichtlich in die Falle zu locken. Baut man dieses Spiel noch weiter aus, indem man mehrere VRA darin platziert, die einen bestimmten Ort oder eine Person beschützen müssen, bietet sich zusätzlich an, dass ein SSA als Team-Leiter durch eine solche Ansicht die Übersicht behält und einzelne Aufgaben akustisch verteilen kann (vergl. 3.1).

4.1.2 Kommentator

Als Kommentator verfügt der SSA über erweiterte Informationen zum Geschehen in der virtuellen Welt. Ihm ist es hier möglich das VR-Erlebnis des VRA auf eine abstrakte Art nachzuvollziehen, weil der Fokus auf den Zusatzinformationen liegt.
Das können beispielsweise Spielstatistiken sein, die der VRA nicht selbst einsehen kann

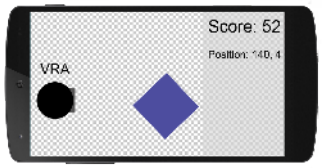


Abbildung 4.4: Spielstatistiken auf dem *Second-Screen*

(vergl. Abb. 4.1.2). Als Beispiel dient hier der *Shared Screen* (sh. 3.4), der als *Second-Screen* eine *Übersicht* über das aktuelle Level und zugehörige Statistiken wie Punktzahlen aller Spieler anzeigt. Das ermöglicht dem SSA die Informationen gegenüber dem VRA oder anderen Außenstehenden zu kommentieren. Beispielsweise private Kommentare, die nur den VRA und SSA betreffen, weil einer der beiden die Punktzahl des anderen übertrumpft [12]. Andererseits aber auch das Kommentieren des Spielgeschehens, wie es etwa in der E-Sport-Szene der Fall ist.

4.1.3 Regisseur

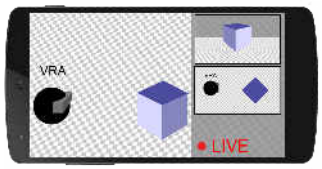


Abbildung 4.5: Der SSA besitzt eine Auswahl aus mehreren Kameraeinstellungen

Bei diesem Konzept macht der SSA vor allem Gebrauch von den unterschiedlichen Blickwinkeln in die VU. Durch unterschiedliches Aneinanderreihen dieser kann ein Film erstellt werden.
Als Beispiel wird der VRA wieder in ein Labyrinth mit Monstern versetzt, die den Spieler angreifen. Während sich der VRA allein gegen diese verteidigt, kann nun der SSA mit unterschiedlichen Regisseur-Werkzeugen gleichzeitig dabei einen kleinen Film vom Spielgeschehen auf dem Smartphone anfertigen. Denn das Multi-Touch Display eines Smartphones eignet sich gut für eine Interaktion mit solchen Werkzeugen. Zu diesen könnten kleine Vorschauen anderer Kameraperspektiven zählen, die zur Hauptperspektive ausgewählt werden, wenn diese mit dem Finger berührt werden (Abb. 4.1.3). Geeignete Perspektiven sind hier

vor allem die *Klon*-Ansicht und hauptsächlich diverse *Spectator*-Modi. Dazu könnten unter anderem eine Verfolgungskamera auf ein Projektil oder eine *First-Person*-Ansicht eines Monsters zählen, damit spektakulärere Szenen entstehen. Außerdem die Vogelperspektive, die auch eine freie Bewegung der Kamera von Seiten des SSA zulässt. Anschließend könnte der Film entweder lediglich gespeichert bzw. auf *YouTube* geteilt werden oder direkt an einen Stream weitergeleitet werden, der ihn für weitere Außenstehende zugänglich macht.

4.1.4 Physikalischer Feedbackgeber

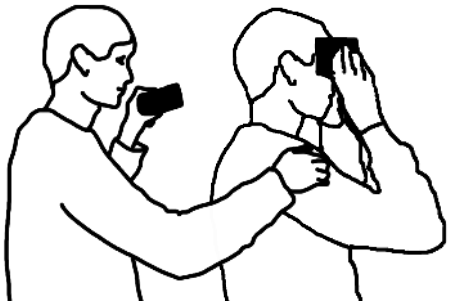


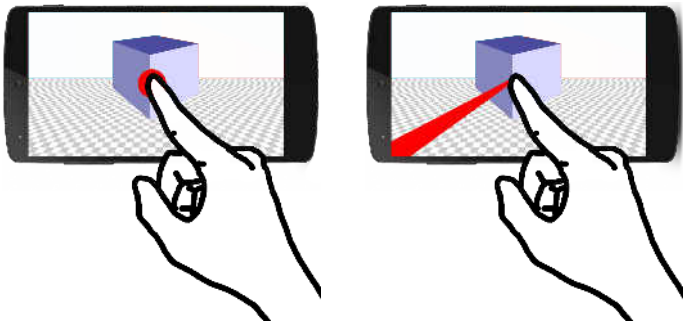
Abbildung 4.6: Ein SSA dreht einen VRA in eine andere Richtung

Der SSA reagiert bei diesem Konzept auf Geschehnisse in der virtuellen Umgebung, indem er den VRA physikalisch beeinflusst.
Das kann unterstützend zum *akustischen Anweiser* (vergl. 4.1.1) angewandt werden, wenn beispielsweise ein Objekt in der VU außerhalb des Blickfelds des VRA liegt. Denn hier kann der SSA zusätzlich den VR-Nutzer in die entsprechende Richtung drehen.
Ebenfalls lässt es sich in ein Spielprinzip umsetzen, wie es mit *Haptic Turk* (sh. 3.2) demonstriert wird. Dabei bekommen ein oder mehrere *Second-Screen-Aktoren* Anweisungen auf ihrem Smartphone, wie sie den Spieler physikalisch beeinflussen sollen. Sie reagieren entsprechend auf Ereignisse in der virtuellen Realität und geben dem VRA damit passendes physikalisches Feedback. Legt sich der VR-Spieler beispielsweise in einem Flugdrachen in der VU in eine Rechtskurve, heben zwei SSA seine linke Körperhälfte so an, damit auch ein realer Gefühlseindruck dazu entsteht. Anweisungen wie diese sind bereits 7 Sekunden vorher auf dem *Second-Screen* zu sehen, so dass die SSA rechtzeitig reagieren können. Dementsprechend müssen in der Zukunft liegende mögliche Ereignisse rechtzeitig Berech-

net werden. Im Beispiel des Flugdrachens, das der Demonstrationsanwendung von *Haptic Turk* entspricht, werden Raycasts in alle Richtungen vom Spieler aus „geschossen“, in die er sich bewegen kann. Treffen diese etwa auf ein Objekt, ist eine Kollision innerhalb der nächsten 7 Sekunden möglich und wird vom SSA angezeigt.

4.2 Passiver Teilnehmer

4.2.1 Zeiger



(a) Punktdarstellung (b) Stabdarstellung

Abbildung 4.7: Zwei mögliche Zeigerarten

Der SSA besitzt ein oder mehrere Zeigerwerkzeuge mit denen er auf Dinge in der VU deuten kann. Dabei ist der verwendete Zeiger ebenfalls für den VRA sichtbar.
In der *Klon*-Perspektive kann der SSA dazu den Bildschirm berühren, um auf eine vor dem VRA liegende Stelle zu zeigen. Die visuelle Darstellung dazu kann als Punkt auf dem Display des *Second-Screens* erfolgen (Abb. 4.7(a)). Weil dort aber kein räumlicher Eindruck der VU vorhanden ist, kann dieser Punkt über der VU als Element der grafischen Oberfläche gezeichnet werden. Da der räumliche Eindruck aber beim VR-System vorhanden ist, muss der Punkt im Raum gezeichnet werden. Würde der Punkt auf einem festen Abstand vor den Augen des VRAs in der VU schweben, wäre ein Schielen des VR-Anwenders die Folge, weil dieser entweder den Punkt an sich oder die dahinterliegende VU fokussiert. Deshalb ermittelt man mittels Raycast die entsprechende gezeigte Stelle im Raum und stellt den Punkt darauf dar (ähnlich einem Laserpointer) - das bewirkt, dass lediglich eine Stelle fokussiert werden muss.

Eine weitere Möglichkeit wäre die Darstellung einer Art Stock oder Stab vom Avatar des VRA ausgehend (Abb. 4.7(b)). Der Vorteil dieser Methode liegt bei weit entfernten Objekten, weil der Ausgangspunkt und die Richtung immer sichtbar sind, während eine Punktdarstellung möglicherweise zu klein sein könnte.
Blickt der SSA von anderen Stellen aus, wie etwa der *Übersicht* oder als *Spectator*, in die VU, können diese Zeigerwerkzeuge ebenso verwendet werden. Problematisch ist dabei aber, dass die Zeiger außerhalb des Sichtfelds des VRAs (und vice versa) liegen können. In solchen Fällen könnte man auf das Modell eines schwebenden Zeigestocks im Sichtfeld des VRA setzen. Dieser richtet sich entsprechend von der Position des VRA zum markierten Punkt in der virtuellen Welt und verschwindet, sobald sich dieser im Sichtfeld befindet.

4.2.2 Objektmarkierung

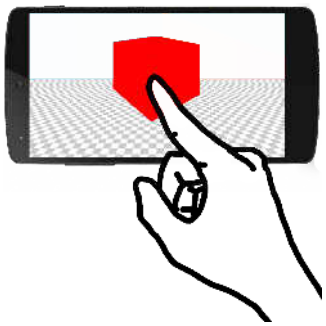


Abbildung 4.8: Ein Würfel wird vom SSA rot markiert

Der SSA kann Objekte in der VU markieren bzw. hervorheben, damit der VRA darauf aufmerksam wird.
Berührt der SSA den Bildschirm, wird ermittelt ob dieser ein Objekt in der VU angewählt hat. Um dieses für den VRA von den anderen Objekten abzuheben, käme eine auffällige farbliche Markierung (z.B. rot) in Frage (Abb. 4.2.2). Da das zu Problemen bei Farbenblinden führen kann, würde auch eine virtuelle physikalische Markierung sinnvoll erscheinen. Dazu könnte das Objekt beispielsweise vibrieren oder mit einem Schein aufblinken.
Liegt das Objekt außerhalb des Sichtfelds des VRA, kann auch hier ein schwebender Zeiger auf dieses verwendet werden (vergl. 4.2.1). Problematischer wird es dagegen, wenn es von anderen Objekten im Raum oder einer Wand verdeckt wird. Das kann dann der Fall sein, wenn sich der SSA nicht im *Klon*-Modus befindet und eine vom VRA unabhängige Sicht in

die VU besitzt. Argelaguet et al. hat für solche Vorkommnisse bereits zwei verschiedene Konzepte entwickelt: Bei beiden ist es dem VRA möglich durch blockierende Objekte (wie Wände) hindurchzusehen. Während dabei die *CutAway*-Methode „runde Fenster“ durch all diese einschneidet, werden bei der *Transparency*-Methode namensgebend die im Weg liegenden Objekte halbtransparent mit einem Überblendungseffekt dargestellt (vergl. 3.6) und ermöglicht damit eine direkte Betrachtung der Markierung.

4.2.3 Zeichenwerkzeuge

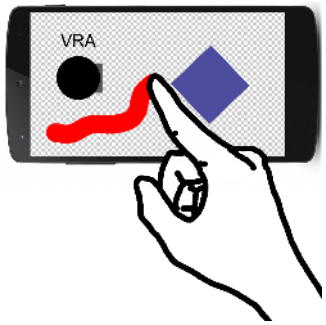


Abbildung 4.9: Ein SSA zeichnet einen Linienhinweis für den VRA

Dieses Konzept erlaubt dem SSA visuelle Hilfsmittel in die VU zu zeichnen. Diese besitzen jedoch keinen physikalischen Körper, so dass der VRA diese zwar sehen, aber nicht mit ihnen interagieren kann.
Als Einsatzbeispiel für Zeichenwerkzeuge dient das Spiel *Minecraft* [29]. Ein Teil des Spiels ist es, Gebäude oder sonstige Objekte mit Hilfe von Blöcken (ähnlich Legosteinen) zu bauen. Während der VRA hier das Bauen übernimmt, könnte der SSA visuelle Anweisungen zum Bauvorhaben geben. Es eignet sich beispielsweise aus der *Übersicht*-Perspektive Linien, Rechtecke oder Kreise auf den Boden der VU vor dem VRA zeichnen zu können (Abb. 4.2.3). Die Steuerungen für zweidimensionale Zeichenoperationen wie diese lassen sich dabei sehr einfach auf einem Multi-Touch Bildschirm realisieren und sind aus zahlreichen Apps wie *Sketchbook mobile* [14] bekannt.
Damit aber auch im dreidimensionalen Bereich durch Zeichenwerkzeuge etwas veranschaulicht werden kann, muss die Möglichkeit der Tiefenregulierung geschaffen werden. Ist diese vorhanden, lässt sich aus jeder beliebigen Perspektive des SSA in der VU zeichnen. Dazu können die von Martinet et al. entwickelten 3D-Positionierungstechniken für Multi-Touch

Bildschirme angewandt werden (sh. 3.3). Diese reichen zunächst für das Umpositionieren von Zeichnungen aus, sollten aber um einen Modus erweitert werden, der es erlaubt direkt in den dreidimensionalen Raum zeichnen zu können. Soll beispielsweise eine Linie gezeichnet werden, die eine Flugbahn eines Pfeils (auch Monster spielen in *Minecraft* eine Rolle) angibt, könnte das wie folgt mit einer erweiterten *Z-Methode* (sh. 3.3) realisiert werden: Der SSA setzt einen Finger am gewünschten Ausgangspunkt an und legt danach den zweiten Finger auf, der die Flughöhe des Pfeils angibt. Durch das gleichzeitige Verschieben beider Finger ergibt sich dann die dreidimensionale Flugbahn.
Zusätzlich zu den Zeichnungen wäre es auch der Einsatz von virtuellen Dummys denkbar. D.h., dass der SSA Spieler- oder Monsteravatare in der VU absetzen kann, um beispielsweise deren Größe einschätzen zu können. Diese besitzen dementsprechend auch keinen physikalischen Körper, haben keinen Einfluss auf das Spiel und sollten bestenfalls halbtransparent angezeigt werden.

4.3 Aktiver Teilnehmer

4.3.1 Aufgabenteilung

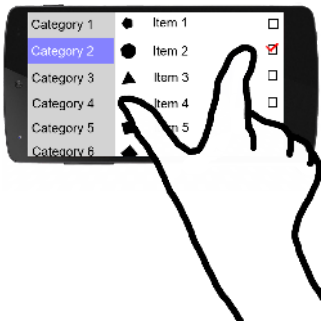


Abbildung 4.10: Ein SSA wählt Elemente für den VRA aus

Im Fokus dieses Konzepts liegen Interaktionen zwischen den beiden Aktoren, die ein gemeinsames Lösen von Problemen ermöglichen. Dabei sollen Teilprobleme so an die Aktoren verteilt werden, dass Vorteile der jeweiligen Plattform ausgenutzt werden.
Beispielweise soll in einer archäologischen VR-Anwendung ein zersplittertes Fundstück wieder zusammengesetzt werden. Dazu sind die mehreren tausend Einzelteile 3D-Modelliert

und kategorisiert. Da der VRA über eine räumliche Darstellung verfügt, ist er gegenüber dem SSA besser für die Zusammensetzung der Bruchstücke geeignet. Zum Problem wird für den VR-Anwender aber die vorige Auswahl dieser, denn in der virtuellen Umgebung sollten verschiedene Kategorien bzw. Ordnerstrukturen als Räume dargestellt werden. Der VRA müsste diese zunächst betreten und verliert damit Zeit. Der SSA ist dagegen besser für diese Aufgabe geeignet, denn eine zweidimensionale Abbildung einer Ordnerstruktur lässt sich schneller mit einem Multi-Touch Bildschirm als durch eine Blicksteuerung durchsuchen (Abb. 4.3.1). Dazu kommt der Vorteil einer Textsuche auf dem Smartphone.
Das lässt sich ebenfalls auf Szenarien für Heimanwender anwenden, wo vor allem VR-Spiele eine wichtige Rolle spielen. Analog zur Archäologie-Anwendung wäre ein Minispiel für den SSA denkbar, in dem sich dieser um die Organisation eines Spielerinventars kümmert (Anordnung der Elemente, Nachladen von Waffen etc.). Der VRA besitzt dagegen die Aufgabe sich beispielsweise mit Hilfe der Waffen, die er vom SSA bereitgestellt bekommt, gegen Monster zu wehren und diese überhaupt einzusammeln.
Ein weiteres Beispiel wäre etwa eine Steuerungsteilung. Etwa in einem Spiel, in dem gemeinsam ein Panzer gesteuert werden soll. Der SSA könnte dabei die Fahrtrichtung mit eingeblendeten Tasten auf dem Touchscreen halten, während er aus der *Third-Person*-Perspektive (Blickwinkel von außen hinter dem Fahrzeug) auf den Panzer blickt. Der VRA besitzt dagegen die Aufgabe das Geschütz und den Drehturm mittels Blickrichtung in der VR-Brille zu steuern.

4.3.2 Mitspieler

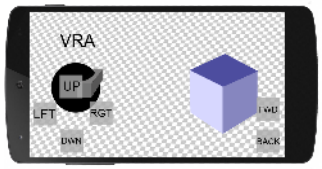


Abbildung 4.11: Ein SSA kann sich mittels Steuerungstasten selbst bewegen und mit der VU interagieren

Der SSA agiert als Mitspieler unabhängig vom VRA in der gemeinsamen VU.
Das bedeutet, dass die beiden Aktoren eventuell gemeinsam ein Problem lösen, jedoch nicht aneinander gebunden sind und getrennt voneinander handeln können. Als Beispiel

dazu dient die Demonstrationsanwendung *BEAMING* (sh. 3.5). Dabei initiiert der SSA als Smartphone-Anwender das Spiel an einem realen Ort seiner Wahl. Auf seinem Bildschirm wird dabei lediglich eine Kartenansicht (vergl. *Übersicht*-Modus) der realen Umgebung gezeigt. Der VRA kann diesem nun beitreten und wird virtuell an denselben Ort versetzt, so dass beide Aktoren nun innerhalb der VU miteinander interagieren können. Es gilt dabei gemeinsam Aufgaben zu lösen, die sowohl real als auch virtuell in der Umgebung verteilt sind. Hervorzuheben ist hierbei vor allem die für die einzelnen Systeme angepasste Darstellung und Interaktion des Spiels.
Weitere Möglichkeiten des Konzepts sind, dass die Aktoren gegeneinander arbeiten oder neutral zueinander stehen und dabei lediglich die VU miteinander teilen. Letzteres könnte einer sozialen Anwendung wie *Second-Life* [26] entsprechen, in der beide unabhängig die virtuelle Welt erkunden, aber auch miteinander mittels Sprachchat interagieren können.

4.3.3 Gottesmodus

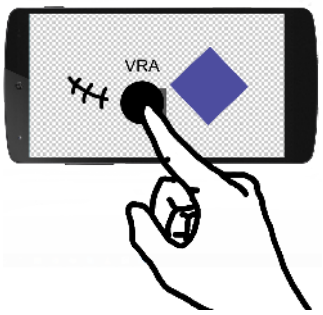


Abbildung 4.12: Ein SSA verschiebt einen VRA in der VU

Im Gottesmodus verfügt der SSA über übergeordnete Fähigkeiten oder Mehrwissen gegenüber dem VRA. Der VRA kann dabei nicht mit dem SSA interagieren - vice versa ist das aber möglich.
Der *Fort Bragg VR-Simulator* (sh. 3.1) ist ein Beispiel für dieses Interaktionskonzept. Mehrere VRA halten sich dabei in einer VU auf und versuchen eine Mission zu erfüllen, die vom SSA erstellt wurde. Der SSA kann dabei zusätzlich Ablenkungsfaktoren wie Tiere in der virtuellen Welt zur Laufzeit absetzen. Die VRA bekommen dabei von der Anwesenheit des SSA nichts mit, weil dieser weder ein Avatar besitzt, noch mit ihm interagiert werden kann. Dieses Prinzip lässt sich ebenfalls auf Heimanwendungen und speziell in VR-Spielen umset-

zen. Ein Beispiel ist eine veränderte Version der *Haptic Turk* Demonstrationsanwendung (sh. 3.2), in der der VRA mit einem Flugdrachen durch die VU fliegt. Der SSA könnte hier Umgebungsvariablen wie das Wetter manipulieren. Beispielsweise das Erzeugen von Turbulenzen durch einen aufkommenden Wind, indem mit dem Finger auf dem Touchscreen gewischt wird. Auch das Auslösen von Schäden am Flugdrachen ist denkbar, da sich eine Auswahl solcher Effekte aus Kontextmenüs auf einem zweidimensionalen Bildschirm anbietet.
In anderen Kontexten könnte auch eine direkte Manipulation des Spielers wie z.B. mittels *Drag & Drop* realisiert werden. Dabei kann der SSA den VRA in der *Übersicht* anwählen und ihn mit einer Fingerbewegung andernorts absetzen (Abb. 4.3.3).

5 Implementierung

Die Demonstrationsanwendung, die Beispiele zu sechs Interaktionskonzepten implementiert, wurde für *Android*-Smartphones realisiert. Die Anwendung wird gleichermaßen auf dem VR- als auch auf dem *Second-Screen*-Smartphone ausgeführt. D.h., dass die Spiellogik auf beiden Geräten dieselbe ist, jedoch nur die Eingabe- und Ausgabeschnittstellen unterschieden werden. Die restlichen Fallunterscheidungen innerhalb der Anwendung entsprechen einer normalen Mehrspieler-Anwendung.

5.1 Verwendete Technologien

Für die Umsetzung der Demonstrationsanwendung wurde *Unity* [8] in Verbindung mit *MonoDevelop* [6] und zeitweise *Sublime Text* [7] verwendet. Die darin geschriebenen Skripte sind in der Programmiersprache *C#* realisiert. Ein Teil der Szene wurde auf Basis des *Cardboard SDK for Unity* [3] erstellt und zusätzlich verschiedenste vorgefertigte Objekte und Animationen wie Zombies [9] und mittelalterlichen Gebäuden [5] (kostenlos im August 2015) aus dem *Asset-Store* benutzt. Außerdem ebenfalls ein $360°$-Panorama von *easypano* [19]. Da das Zielsystem *Android* ist und sich zwei Geräte mittels *Bluetooth* austauschen, wurde zusätzlich das Unity-Plugin *Android Bluetooth Multiplayer* [1] verwendet. Mit Hilfe des *Android SDK* wurde die Anwendung kompiliert und *Android Studio* zur Log-Ausgabe gebraucht [2].

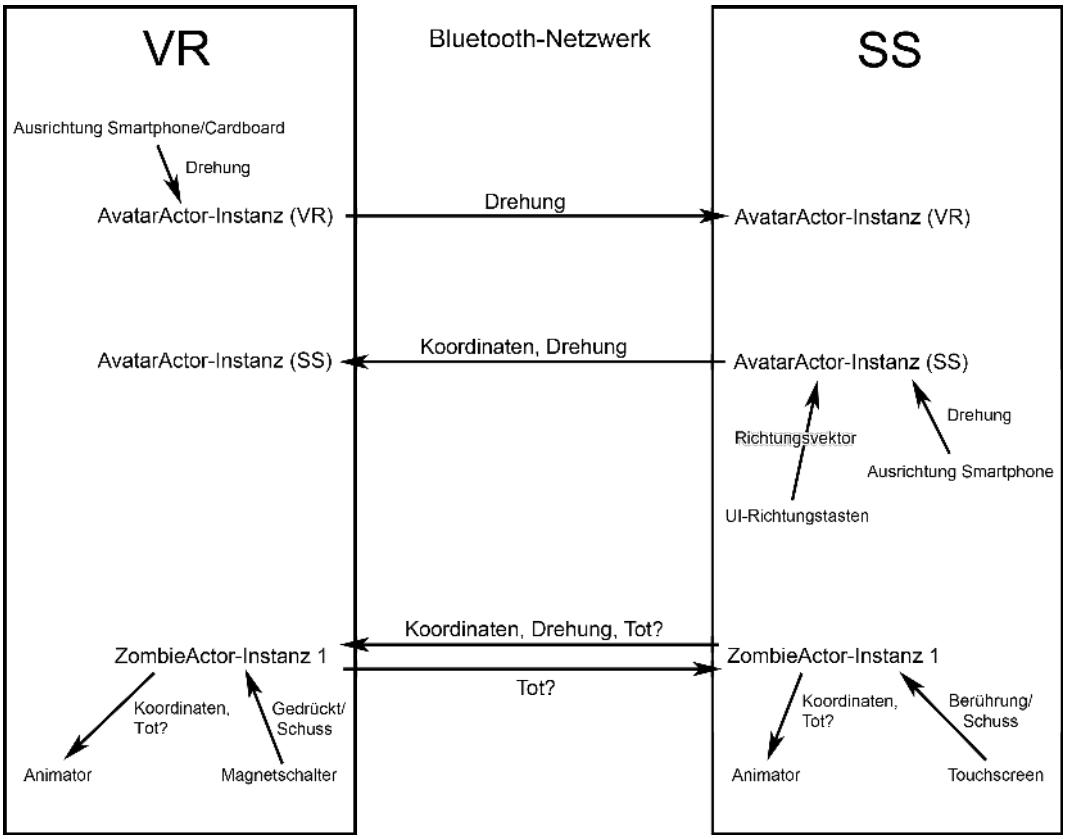


Abbildung 5.1: Datenaustausch über das Bluetooth-Netzwerk

5.2 Realisierung

Die Basis der `DemoScene` bildet die in der Beispielanwendung des *Cardboard SDK for Unity* enthaltene Kamerakonfiguration. Das `StereoController.cs`-Skript kümmert sich dabei um eine dreidimensionale Darstellung, wenn es an eine Kamera angehängt wird. Diese lässt sich auch deaktivieren, so dass eine zweidimensionale Darstellung mit nur einer Kamera möglich ist, wie es im späteren *Spectator*-Modus des SS verwendet wurde.
In der Szene wurden außerdem mehrere statische Objekte, wie mittelalterliche Gebäude, die Panorama-Kugel oder Kameras für den SS platziert. Ebenfalls das `Main`-Objekt, an welches das gleichnamige Skript angehängt ist, das die Hauptlogik der Anwendung implementiert: Hier werden die verschiedenen Zustände verwaltet, in denen sich die Anwendung gerade befindet. Zudem hält die `Main`-Klasse Referenzen zu relevanten Objekten in der Szene, die

im späteren Verlauf manipuliert (etwa Kameras) oder erstellt (Avatare) werden und verschiedene Konstanten (etwa Startpositionen). Die `OnGUI()`-Methode zeichnet je nach Zustand der Anwendung die entsprechende grafische Benutzeroberfläche auf das Display und enthält die Programmierlogik, die ausgeführt wird, wenn ein bestimmter Button gedrückt wird. Dazu gehört beispielsweise das Öffnen des Bluetooth-Dialogs. `Update()` implementiert das Reagieren auf Fingerberührungen außerhalb eines Buttons, so dass etwa ein Zombie in der Welt erstellt wird, aber auch die Darstellung des dreidimensionalen *Head-Up-Displays*, das z.B. Auskunft über den Zustand des Spielers gibt.
Sind die beiden Geräte mittels Bluetooth miteinander verbunden (diese Aufgabe übernimmt das Plugin *Android Bluetooth Multiplayer*), werden zu Beginn die jeweiligen Avatare der Spieler im Netzwerk instanziiert. D.h., dass mit der Referenz auf das `ActorPrefab` das Objekt in den jeweiligen Szenen erstellt und angezeigt wird. An dieses Prefab ist das Skript `AvatarActor.cs` gekoppelt, das sich um den Austausch der Positions- und Rotationskoordinaten speziell dieses Objekts zwischen den Geräten kümmert. Wird also beispielsweise das Avatar des VRA erstellt, werden auf dem VR-Gerät die Rotationskoordinaten im `AvatarActor` der Drehung des Geräts angepasst - Auf dem SS-Gerät werden diese Rotationskoordinaten dagegen vom VRA über das Netzwerk bezogen und im Skript angepasst (Abb. 5.1). Dieselbe Methode wird ebenfalls bei all den anderen im Netzwerk erstellten Objekten verwendet. Siehe dazu `PanoramaPointerActor.cs` und `ZombieActor.cs`. Letzerer kommuniziert zusätzlich noch mit einem *Animator*, der eine bestimmte Animation auslöst, wenn ein bestimmter Zustand erreicht wird. Wird etwa `isDead` auf `true` gesetzt, fällt der jeweilige Zombie auf seinen Rücken.
Für die Anzeige des Panoramas wurde der spezielle Shader `InsideOut` erstellt, der es erlaubt, eine Textur auf der Innenseite einer Kugel zu zeichnen und eine gleichmäßige „Beleuchtung“ ermöglicht. Ein weiteres Problem war dabei, dass ein Raycast von innerhalb der Kugel keine Kollision mit dieser erkannt hat (Stichwort `GazePointer`). In diesem Fall musste ein Punkt auf dem Strahl außerhalb der Kugel gewählt und die Richtung umgekehrt werden, so dass eine Kollision erkannt und benutzt werden konnte. Wie das Problem konkret im Quelltext gelöst wurde, ist im Anhang A zu sehen. Dies war vor allem nötig, weil vom SS-Gerät aus ein Raycast nicht immer vom tatsächlichen Mittelpunkt der Kugel kommt.

5.3 Verwendung

Um das Grundgerüst dieser Anwendung als Basis für eine neue dieser Art zu verwenden, müssen lediglich die anwendungsspezifischen Objekte außen vor gelassen werden. D.h. dass die Ordner `Medieval Buildings`, `Zombie` und `Cardboard/Legacy/Resources` entfernt werden können. Außerdem alle Skripte mit der Endung `Actor`, mit der Ausnahme von `AvatarActor.cs`, falls die Aktoren mit Avataren in der VU visualisiert werden sollen. Exportiert man alle zurückbleibenden Dateien in einem `unitypackage`, erhält man ein Asset, das nun zu jeder beliebigen Anwendung hinzugefügt werden kann um diese mit der *Second-Screen*-Funktionalität zu erweitern.
Im ersten Schritt muss dazu das `CardboardMain`-GameObject die Hauptkamera ersetzen. Danach muss die Hauptklasse der bestehenden Anwendung wie folgt angepasst werden:

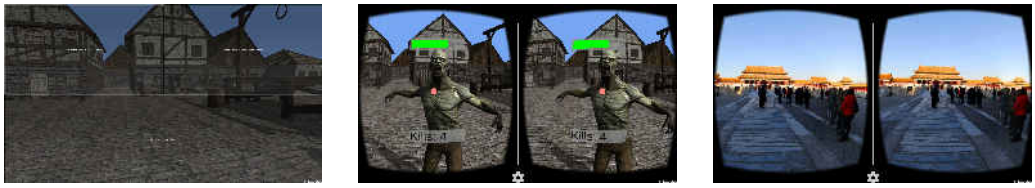
Bluetooth-Methoden übernehmen:
Es müssen hierbei nur Inhalte aus `Main.cs` aus der Beispielanwendung kopiert werden. Dazu zählen Attribute die im Block *Bluetooth* definiert wurden, alle Bluetooth-Methodenaufrufe in den Methoden `Awake()` und `OnDestroy()`. Außerdem den gesamten Inhalt, der als `#region bluetooth events` markiert wurde.

Bluetooth-Verbindung aufbauen:
Wie der Aufbau von beiden Seiten funktioniert, kann in der Methode `OnGUI()` oder in der Dokumentation zu *AndroidBluetoothMultiplayer* [27] eingesehen werden. Nach dem Aufbau werden entsprechend die Methoden `OnServerInitialized()` und `OnConnectedToServer()` aufgerufen.

Szene über Netzwerk synchronisieren:
Dazu wird das standardmäßige Unity-Network verwendet. Das bedeutet, dass alle neu in der Szene erstellten Objekte mit `Network.Instantiate()` instanziiert werden müssen und ein `NetworkView` besitzen müssen.

Modus setzen:
Der Eingabe- und Ausgabemodus (stereoskopisch oder normal) kann durch Setzen des statischen `bool`-Attributs `Cardboard.SDK.VRModeEnabled` verändert werden.

6 Beispielanwendung



(a) Hauptmenü (b) Zombie-Spiel (c) $360°$-Panorama

Abbildung 6.1: Das Hauptmenü der Anwendung und die beiden Hauptmodi

Die Android-App realisiert konkrete Beispiele zu insgesamt sechs der im Kapitel 4 entworfenen Konzepte. Diejenigen, die in die Bereiche *Zuschauer* und *Aktiver-Teilnehmer* fallen, sind im *Zombie-Spiel* realisiert. während *Passiver-Teilnehmer*-Konzepte in der Panorama-Anwendung zu finden sind.
Als Benutzer des Smartphones im *Google Cardboard* wählt man zunächst einen der beiden Modi im Hauptmenü aus und gelangt damit in die stereoskopische Ansicht. Dies entspricht einem der beiden oberen Buttons, wie in Abb. 6.1(a) zu sehen ist. Beide Modi lassen sich theoretisch ebenfalls allein vom VRA benutzen, jedoch ist der automatische Start im Zombie-Modus hier deaktiviert und startet erst, wenn der SSA beitritt.

Ziel im *Zombie-Spiel* (Abb. 6.1(b)) ist es, sich gegen so viele Zombies wie möglich zu wehren, die einen angreifen. Kommt ein Zombie dem Spieler zu nahe, attackiert er diesen und zieht ihm Lebenspunkte ab. Im Gegenzug kann der Spieler jeden Zombie mit einem Schuss auf dessen Kopf oder Oberkörper töten - gezielt wird dabei mit der Blickrichtung, dessen Ziel mit einer roten Lasermarkierung angezeigt wird. Der Schuss löst sich beim betätigen des Magnetschalters am *Cardboard* oder bei einer Berührung des Bildschirms.

Der *Panorama*-Modus (Abb. 6.1(c)) erlaubt es dem VR-Benutzer sich frei in einem $360°$-

Panorama umschauen zu können. Auch hier ist der Mittelpunkt des Blicks mit einer roten Markierung versehen.

6.1 Zuschauer

(a) Spectator (b) Übersicht (c) Klon

Abbildung 6.2: Die drei verschiedenen Kameraperspektiven im Zuschauer-Modus

Tritt der *Second-Screen*-Anwender dem Spiel bei, startet dieser zunächst wenige Meter rechts vom VR-Spieler in der Welt. Mit den nun auf dem Display angezeigten Richtungstasten und der Drehung des Geräts kann sich der Zuschauer frei in der virtuellen Umgebung bewegen. Damit kann das Avatar des VRA betrachtet werden und wie dieser sich gegen die ankommenden Zombies verteidigt (Abb. 6.2(a)). Diese werden zufällig an festgelegten Startpunkten abgesetzt und laufen anschließend auf den VR-Spieler zu. Eine vollständige Übersicht erhält man als SSA, wenn in die Draufsicht mit dem Button *Map* gewechselt wird (Abb. 6.2(b)). Mit *VR Clone* kann dagegen in dieselbe Ansicht gesprungen werden, die der VR-Spieler vor sich hat, jedoch nicht im stereoskopischen Modus (Abb. 6.2(c)). Damit werden zwei Konzepte gleichzeitig umgesetzt:

akustischer Anweiser (sh. 4.1.1):
Vor allem in den Blickwinkel-Modi *Übersicht* und *Spectator* kann der SSA auch ankommende Zombies im Rücken des VRA sehen. Dadurch kann er diesen entweder akustisch vorwarnen oder absichtlich falsch informieren.

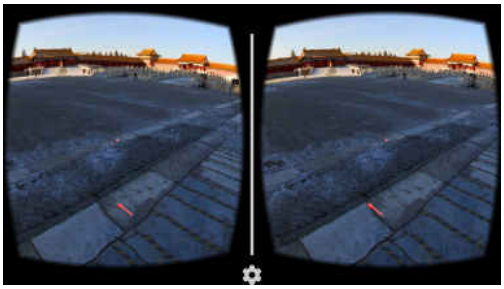
physikalischer Feedbackgeber (sh. 4.1.4):
Dieses Konzept ist zunächst unterstützend zum *akustischen Anweiser* umgesetzt. D.h., dass der SSA den VRA in eine bestimmte Richtung drehen kann, so dass dieser auf einen an-

greifenden Zombie aufmerksam wird. Eine andere Möglichkeit ist das Packen und Schütteln des VRA als physikalisches Feedback zu einer Verletzung durch einen Zombie.

6.2 Passiver Teilnehmer

(a) Zeiger des VR-Nutzers (Mitte) und des passiven Teilnehmers (Rechts)

(b) *FishEye*-Ansicht des passiven Teilnehmers



(c) Zeiger des passiven Teilnehmers außerhalb des Blickfelds

Abbildung 6.3: Zwei *Second-Screen*-Ansichten und die stereoskopische Ansicht des Panoramas

Diese Konzeptkategorie wurde im *Panorama*-Modus umgesetzt. Dabei wird der VR-Anwender in ein $360°$-Panorama versetzt und kann sich frei umschauen. Seine Blickrichtung wird ihm

und dem *Second-Screen*-Nutzer mit einem kleinen roten Zylinder dargestellt (Abb. 6.3(a)). Auch der passive Teilnehmer kann sich durch Drehung seines Geräts im *Spectator*-Modus frei umschauen und mit einer Fingerberührung auf eine Position im Panorama zeigen. Diese Position wird mit einer größeren roten Kugel markiert (siehe Abb. 6.3(a), rechts). Wird diese Markierung außerhalb des Blickfelds des VR-Anwenders gesetzt, bekommt dieser einen dreidimensionalen Pfeil vor sich eingeblendet, der auf die Position der Markierung zeigt (Abb. 6.3(c)). Der *FishEye*-Modus entspricht hier der *Übersicht*, da mit dieser Kameraeinstellung ein Großteil des Panoramas auf einmal abgedeckt wird. Beim *VR Clone* wird dagegen wieder die Blickrichtung des VR-Anwenders übernommen und dementsprechend kann ein gesetzter Zeiger nicht mehr außerhalb des Blickwinkels landen. Damit wurde folgendes Interaktionskonzept realisiert:

Zeiger (sh. 4.2.1):
Da es dem SSA möglich ist mit einer roten Kugel eine Stelle im Panorama zu markieren, wird hier die Punktdarstellung verwendet. Außerdem wurde zusätzlich die vorgestellte Lösung zum Problem eines Zeigers außerhalb des Blickfelds eingesetzt, indem ein schwebender Pfeil auf die Position der Markierung hinweist.

6.3 Aktiver Teilnehmer

Von allen Zuschauer-Modi im *Zombie-Spiel* kann in einen jeweiligen aktiven Modus gewechselt werden, der verschiedene Interaktionen erlaubt. Im *Spectator* wird man etwa wieder an die Seite des VR-Spielers gestellt und kann sich nun nicht mehr frei bewegen, sondern nur noch drehen. Als *Second-Screen*-Spieler besitzt man nun eine eigene Lebens- und Abschussanzeige und wird dabei ebenfalls von Zombies angegriffen (Abb. 6.3). Man verteidigt gemeinsam eine Position gegen die zufällig gespawnten Untoten. Diese greifen dabei immer den nächstgelegenen Spieler an.
Wechselt man dagegen in der *Übersicht* in den aktiven Modus, ist man nun als *Second-Screen*-Spieler in der Lage die Zombies selbst mit einer Berührung auf dem Display abzusetzen (vergl. Abb. 6.2(b)). Diese laufen dann wie gewohnt auf den VR-Spieler zu und attackieren diesen.
Beim *VR Clone* verschmilzt man wiederum zu einer Einheit, indem der VR-Spieler mit seiner

Abbildung 6.4: Aktiver Teilnehmer bekämpft Zombies, die den VR-Spieler angreifen

Blickrichtung zielen und der *Second-Screen* mit Druck auf den Touchscreen schießen muss (vergl. Abb. 6.2(c)). Man steuert hier also gemeinsam ein Avatar und besitzt ein gemeinsames Leben.
Insgesamt drei Konzepte wurden mit diesen Spielmodi umgesetzt:

Aufgabenteilung (sh. 4.3.1):
Befindet sich der SSA im *Klon*-Modus, besitzt er die Aufgabe die angreifenden Zombies abzuschießen indem er auf seinen Bildschirm tippt. Das Anvisieren übernimmt gleichzeitig der VRA mit seiner Kopfbewegung. Dadurch ist eine *Aufgabenteilung* in Form einer Steuerungsteilung realisiert, die die Vorteile der jeweiligen Systeme nutzt: VRA mit natürlicher Kopfbewegung und SSA durch erhöhte Schussfrequenz durch schnelles Tippen (gegenüber dem langsamen Magnetschalter am *Cardboard*).

Mitspieler (sh. 4.3.2):
Der SSA besitzt im aktiven *Spectator*-Modus eine eigene Lebens- und Abschussanzeige. Durch seine separate Position, Darstellung mit Avatar und weil er ebenfalls von Zombies

attackiert wird, ist er ein vollwertiger Mitspieler der unabhängig vom VRA agiert.

Gottesmodus (sh. 4.3.3):
In der aktiven *Übersicht* kann der SSA nach Belieben Zombies in der VU absetzen, die dann den VRA angreifen. Er besitzt damit eine volle Kontrolle über eine Umgebungsvariable, die ihm auch nicht beschränkt wird. Der VRA kann sich gegen den SSA nicht zur Wehr setzen, sondern lediglich die Zombies ausschalten.

7 Fazit

In dieser Arbeit wurde zunächst das Problem angegangen, dass bei der Nutzung von VR-Brillen außenstehende Personen vollständig aus der *Virtual-Reality*-Erfahrung ausgeschlossen werden. Es wurde festgestellt, dass sich Smartphones durch ihre weite Verbreitung anbieten ein Fenster in die virtuelle Umgebung zu schaffen. Dazu wurde eine direkte Verbindung zwischen diesen angesetzt, damit keine Verbindungen zu zusätzlichen Servern notwendig sind. Allein die Möglichkeit einen Einblick in die virtuelle Umgebung zu besitzen, lässt die *Second-Screen*-Nutzer am Geschehen teilhaben und steigert deren Interesse. Da eine solche Anwendung gleichermaßen auf dem VR-System und dem Smartphone läuft, werden zusätzlich Interaktionen innerhalb der virtuellen Umgebung möglich. Deshalb wurde in dieser Arbeit ein Katalog an allgemeinen Interaktionskonzepten zwischen VR-Brillen-Nutzern und Smartphone-Anwendern erarbeitet. Diese Konzepte errichten eine Verbindung zwischen den beiden Systemen unter Beachtung der unterschiedlichen Benutzerschnittstellen. Um eine Übersicht zu schaffen, wurden die zehn entwickelten Konzepte in die drei Kategorien *Zuschauer*, *passiver Teilnehmer* und *aktiver Teilnehmer* eingeteilt. Diese Kategorien entsprechen einer Abstufung des Interaktionsgrades des Smartphone-Anwenders in der virtuellen Umgebung. Außerdem wurden verschiedene Lösungsvorschläge zu möglichen entstehenden Problemen innerhalb einzelner Konzepte vorgestellt. Die Interaktionskonzepte sollen als Basis zur Weiterentwicklung dienen, deshalb sind diese auch sehr allgemein formuliert. Um aber zusätzlich eine mögliche konkrete Anwendung ausgewählter Konzepte zu demonstrieren, wurde im Zuge dieser Arbeit die Applikation *VR-Window* entwickelt. Diese implementiert Beispiele zu insgesamt sechs Interaktionskonzepten. Bei der Entwicklung zeigte sich u.a. das Problem, dass eine solche 3D-Anwendung schnell eine große Menge an Speicher benötigt. Das wird vor allem auf Smartphones problematisch und wirkt sich kontraproduktiv auf den spontanen Einsatz aus, wenn ein *Second-Screen*-Nutzer erst eine große Datenmenge aus dem App-Store herunterladen muss. Umgehen lässt sich das mit einer speziellen kleineren Applikation für Smartphones, die beispielsweise nur Statistiken

anzeigt oder eine Draufsicht, so dass keine großen 3D-Modelle benötigt werden. Dadurch entsteht jedoch der Nachteil, dass Entwickler auch die Anwendungslogik und damit die gesamte Anwendung speziell für zwei Systeme entwickeln müssen. Das wäre aber ohnehin bei einer Verbindung zwischen einem Computer mit *Oculus Rift* und einem Smartphone nötig. Ist die VR-Anwendung aber für *Google Cardboard* oder *Gear VR* angesetzt, bietet sich eine Verbindung mit anderen Smartphones an, da die Anwendung ebenfalls schon auf einem zweiten Smartphone lauffähig ist und nur die Benutzerschnittstellen angepasst werden müssen. Wie das in Verbindung mit *Unity* aussehen kann, wurde mit der Beispielapplikation zu dieser Arbeit demonstriert. Die entsprechenden Teile der Anwendung können, wie beschrieben wurde, ebenfalls in neue oder bereits bestehende Projekte übernommen werden, um in diesen eine Teilnahme von *Second-Screen*-Smartphones zu ermöglichen. So soll sie als Anstoß für weitere Anwendungen auf Basis eines oder mehrerer Konzepte und für die Entwicklung neuer allgemeiner Konzepte dienen.

A Quelltexte

```
1  if(Input.GetTouch(0).phase == TouchPhase.Began){
2          Ray TouchPoint = currCam.ScreenPointToRay(
3                  new Vector3(
4                          Input.GetTouch(0).position.x,
5                          Input.GetTouch(0).position.y,
6                          0f
7                  )
8          );
9          /*
10          * Getting a point on the ray outside of the
11          * panorama-sphere and inverting the
12          * direction that the Raycast will register
13          * a hit with the sphere.
14          * From inside an hit wouldn't be registered.
15          *
16          * The sphere got only a radius of 5, but we
17          * need in FishEyeMode at least a point >10
18          * on the ray obviously.
19          */
20         TouchPoint.origin = TouchPoint.GetPoint(15f);
21         TouchPoint.direction = -TouchPoint.direction;
22
23         RaycastHit hit;
24
25         if(PanoramaSphere.GetComponent<Collider>().Raycast(
26                         TouchPoint,
27                         out hit,
```

```
28                          15f)
29                  ){
30                  /*
31                   * Just in case. But the Raycast should
32                   * always hit the sphere, because the
33                   * cameras are inside it.
34                   */
35                  PanoramaPointerInstance = (GameObject)
                        Network.Instantiate(
36                          PanoramaPointerActorPrefab,
37                          hit.point,
38                          Quaternion.identity,
39                          0
40                  );
41          }
42  }
```

B Glossar

Second-Screen	Das Smartphone, das als zweiter Bildschirm (neben der VR-Brille) zum Einblick in die virtuelle Umgebung dient.
VRA	*Virtual-Reality*-Aktor: Der Anwender, der mittels VR-Brille in die virtuelle Umgebung eintaucht.
SSA	*Second-Screen*-Aktor: Der Anwender, der mittels Smartphone am Geschehen in der virtuellen Umgebung teilnimmt.

Literaturverzeichnis

[1] *Android Bluetooth Multiplayer (Basic)*. Unity-Plugin, . – https://www.assetstore.unity3d.com/en/#!/content/20928

[2] *Android SDK und Android Studio*. SDK und Entwicklungsumgebung, . – https://developer.android.com/sdk/index.html

[3] *Cardboard SDK for Unity*. SDK, . – https://developers.google.com/cardboard/unity/

[4] *Cave Automatic Virtual Environment*. – https://de.wikipedia.org/wiki/Cave_Automatic_Virtual_Environment

[5] *Medieval Buildings*. Unity-Asset, . – https://www.assetstore.unity3d.com/en/#!/content/34770

[6] *MonoDevelop*. C-Sharp Editor, . – http://www.monodevelop.com/

[7] *Sublime Text*. Editor, . – http://www.sublimetext.com/

[8] *Unity*. Engine und Entwicklungsumgebung, . – http://unity3d.com/

[9] *Zombie*. Unity-Asset, . – https://www.assetstore.unity3d.com/en/#!/content/30232

[10] Introduction by Nintendo. In: *New York Times* (1955). – http://www.nytimes.com/1995/08/22/business/introduction-by-nintendo.html

[11] *Oculus Rift: Step Into the Game*. Kickstarter-Kampagne, 2012. – https://www.kickstarter.com/projects/1523379957/

oculus-rift-step-into-the-game

[12] *The „Shared Screen"*. Blog-Post, 2014. – http://www.triangularpixels.net/cms/development/the-shared-screen/

[13] ARGELAGUET, Ferran ; KUNERT, André ; KULIK, Alexander ; FROEHLICH, Bernd: *Improving Co-located Collaboration with Show-Through Techniques*. Paper, 2010

[14] AUTODESK: *Sketchbook mobile*. – https://www.sketchbook.com/mobile

[15] BOHNENBERGER, Johann G. F.: Beschreibung einer Maschine zur Erläuterung der Gesetze der Umdrehung der Erde um ihre Axe, und der Veränderung der Lage der letzteren. In: *Tübinger Blätter für Naturwissenschaften und Arzneikunde* 3 (1817), S. 72–83

[16] BYMER, Maj. L.: *Virtual reality used to train Soldiers in new training simulator*. Blog-Post, 2012. – http://www.army.mil/article/84453/

[17] CHENG, Lung-Pan ; LÜHNE, Patrick ; LOPES, Pedro ; STERZ, Christoph ; BAUDISCH, Patrick: *Haptic Turk: a Motion Platform Based on People*. Paper, 2014

[18] D'ALMEIDA, Joseph: *Nouvel appareil stéréoscopique*. 1858

[19] EASYPANO: *Panoramabild*. – http://www.easypano.com

[20] GOOGLE: *Cardboard*. 2014. – https://www.google.com/get/cardboard/

[21] HAMMOND, Laurens: *Stereoscopic motion picture*. http://www.google.com/patents/US1435520. Version: 1922. – US Patent 1435520

[22] HEILIG, Morton: *Cinema of the future*. Paper, 1955

[23] HEILIG, Morton: *Sensorama*. US-Patent 3050870, 1961. – http://www.mortonheilig.com/SensoramaPatent.pdf

[24] JOYCE, Kevin: *Project Morpheus PlayStation Vita Integration confirmed*. Blog-Post, 2014. – http://vrfocus.com/archives/5059/project-morpheus-playstation-vita-integration-confirmed/

[25] KZERO: *Prognose zum Umsatz mit Virtual Reality weltweit in den Jahren 2014 bis 2018 (in Millionen US-Dollar)*. 2015. – http://de.statista.com/statistik/daten/studie/318536/umfrage/prognose-zum-umsatz-mit-virtual-reality-weltweit/

[26] LINDEN-LAB: *Second Life*. Spiel, . – http://secondlife.com/

[27] LOSTPOLYGON: *Android Bluetooth Multiplayer Dokumentation*. – http://lostpolygon.com/unity-assets/android-bluetooth-multiplayer/

[28] MARTINET, Anthony ; CASIEZ, Géry ; GRISONI, Laurent: *The Design and Evaluation of 3D Positioning Techniques for Multi-touch Displays*. Paper, 2010

[29] MOJANG: *Minecraft*. – https://minecraft.net/

[30] NINTENDO: *Visually Convincing Depiction of Object Interactions in Augmented Reality Images*. US-Patent 20150130790, 2013. – http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=3&f=G&l=50&co1=AND&d=PG01&s1=Nintendo.AS.&OS=AN/Nintendo&RS=AN/Nintendo

[31] OCULUS: *Rift*. 2013. – https://www.oculus.com/en-us/rift/

[32] OYEKOYA, Oyewole ; STONE, Ran ; STEPTOE, William ; ALKURDI, Laith ; KLARE, Stefan ; PEER, Angelika ; WEYRICH, Tim ; COHEN, Benjamin ; TECCHIA, Franco ; STEED, Anthony: *Supporting Interoperability and Presence Awareness in Collaborative Mixed Reality Environments*. Paper, 2013

[33] SEGA: *VR*. Video, 1991. – https://www.youtube.com/watch?v=yd98RGxad0U

[34] STATISTA ; EMARKETER: *Prognose zur Anzahl der Smartphone-Nutzer weltweit von 2012 bis 2018 (in Milliarden)*. 2015. – http://de.statista.com/statistik/daten/studie/309656/umfrage/prognose-zur-anzahl-der-smartphone-nutzer-weltweit/

[35] STATISTA ; SCHMIDT, Holger: *Anzahl der Smartphone-Nutzer in Deutschland in den Jahren 2009 bis 2015 (in Millionen)*. 2015. – http:

//de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/

[36] SUTHERLAND, Ivan: The Sword of Damocles. In: *Proceedings of AFIPS 68* (1968), S. 757–764

[37] VPL-RESEARCH: *EyePhone*. – https://vrwiki.wikispaces.com/VPL+EyePhone

Name: Sebastian Schäf Matrikelnummer: 751913

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Sebastian Schäf