Institut für Visualisierung
Abteilung für Mensch-Computer Interaktion
Universität Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart

Diplomarbeit Nr. 3391

Framework für natürliche Interaktion mit 3D Displays

Dirk Uwe Coehne

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Albrecht Schmidt
Betreuer:	Dr. Dipl.-Inf. Florian Alt M. Sc. Thomas Kubitza
begonnen am:	1. Oktober 2012
beendet am:	3. März 2013
CR-Klassifikation:	H.1.2, H.5.2, I.3.7

Kurzfassung

In dieser Arbeit wird das System *Kinetic Action using Smartphones to Interact with Animations* (kurz KASIA) vorgestellt. Dieses System ermöglicht einem Kinobesucher über sein Smartphone direkt mit einer stereoskopischen Animation zu interagieren.

Nach einer Literaturrechersche zu den Themen Gestenerkennung durch 3D-Beschleunigungssensoren, Interaktion mit großen Displays, sowie Implementierung von Client-Server-Systemen wurde ein Konzept für eine Anwendung entwickelt, das aus drei Teilen besteht. Diese Teile wurden prototypisch wie folgt implementiert:

- Einen Smartphone-Client, also eine Android-Applikation, die die Beschleunigungssensoren des Mobiltelefons ausliest und dadurch Hand-Gesten erkennt.
- Ein Animations-Framework, das eine komplette stereoskopische Animation erstellt und deren Verhalten und Aussehen beeinflussen kann.
- Einen zentralen Server, der ortsunabhängig die Kommunikation zwischen den einzelnen Clients und der Animation regelt.

Alle Bestandteile wurden in der Programmiersprache Java implementiert und durch eine Benutzerstudie evaluiert.

Die Arbeitsthese, dass natürliche Handgesten von Benutzern besser angenommen werden als andere Interaktionsmethoden, konnte in der Studie jedoch nicht bestätigt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In dieser Arbeit wird das Framework *Kinetic Action using Smartphones to interact with Animations* – Kasia vorgestellt. Für die Grundlage einer Evaluation wurde mit Kasia eine interaktive Animation erstellt, in der Benutzer mit ihrem Smartphone auf sie zufliegende Gegenstände fangen müssen. Ein Anwendungsfall für das System ist zum Beispiel ein Spiel vor dem Hauptfilm eines Kinos, bei dem die Benutzer durch das Fangen von Gegenständen Preise gewinnen können. Es soll erforscht werden wie Benutzer mit Hilfe ihres Smartphones mit einer stereoskopischen Animation interagieren. Es wird davon ausgegangen, dass ein System, mit dem in einer Kinoumgebung direkt mit der Animation interagiert werden kann, von den Benutzern sehr positiv aufgenommen wird und dass sie eine (natürliche) Handgeste einer anderen Interaktionsart vorziehen. Diese These soll am Ende mit einer Studie geprüft werden. Es wurden zwei Aspekte untersucht. Zum einen welche Geste die Benutzer zur Interaktion bevorzugen, zum anderen wie durch visuelle Indikatoren angezeigt werden kann, ob ein Gegenstand in Fangreichweite ist.

In diesem Kapitel wird zunächst in Abschnitt 1.1 verdeutlicht, wo der Anreiz zu dieser Arbeit liegt. In Abschnitt 1.2 ist der Ausschreibungstext und die Aufgabenstellung zu dieser Arbeit zu finden. Abschnitt 1.3 auf Seite 13 präsentiert Herausforderungen der Arbeit und Lösungsansätze, um ihnen zu begegnen. Abschließend wird die Gliederung dieser Arbeit erläutert(siehe Abschnitt 1.4 auf Seite 15).

1.1. Motivation

Filme in 3D zu produzieren ist keine Erfindung des 21. Jahrhunderts. Schon im Jahre 1922 lief der US-Amerikanische Stummfilm *The Power of Love*[1] in den Kinos. Doch steckte das Kino und vor allem die 3D-Technik noch weit in den Kinderschuhen, der 3D-Effekt wurde damals noch mit Rot-Grün-Brillen erzeugt. In den 50er Jahren des 20. Jahrhunderts gab es den zweiten 3D-Boom, doch auch dieser hielt etwa nur zehn Jahre an.

Heute, im Zeitalter von Computern, aufwändiger Grafik, hoher Auflösung und ausgereifterer Technologie, gibt es einen neuen Hype rund um dreidimensionales Kino. Spätestens mit dem Film *Avatar*[2] hat der 3D-Film wieder Einzug in die Kinos gehalten. Lichtspielhäuser und Projektorhersteller übertreffen sich mit den unterschiedlichen Verfahren, die ein Eintauchen

[1] http://www.imdb.com/title/tt0013506/
[2] http://www.imdb.de/title/tt0499549/

in die schöne 3D-Welt ermöglichen. Sogar im heimischen Wohn- und Spielzimmer ist die 3D-Technik angekommen. War der 3D-Film früher noch die Domäne ganz weniger Spezialkinos, kann sich heute jeder das 3D-Kino-Erlebnis nach Hause holen. Ermöglicht wird das durch immer größere Speichermedien, geschickte Kodierungsverfahren und immer günstiger werdenden 3D-Hardware. So sind inzwischen 3D-fähige Monitore und sogar 3D-Beamer in HD-Qualität auf dem Markt, die nicht nennenswert teurer sind als „normale“. Selbst Computeranimationen und -spiele können nun dreidimensional wahrgenommen werden.

Eine ähnlich rasante Entwicklung, aber in einem ganz anderen Technikzweig, ist die der Mobiltelefone. Moderne Smartphones, was nichts anderes als kleine, mit Sensoren vollgepackte Computer mit Telefonoption sind, erleben derzeit einen riesigen Boom. Für Weihnachten 2012 ging der Bundesverband Informationswirtschaft (BITKOM) davon aus, dass 20% der Deutschen sich zum Fest ein neues Smartphone wünschen [Pup12].

Die Herausforderung besteht nun darin, die Technologie, die in heutigen Smartphones steckt, mit stereoskopischen Animationen oder sogar Filmen zu verbinden. Wer hat nicht schon einmal beim Sehen eines 3D-Films davon geträumt, der Figur, die gerade genüsslich ihr Eis verspeist, die Tüte zu stehlen und selbst zu essen? Schließlich erscheint das Eis in der 3D-Welt zum Greifen nah.

Diese Idee, mit 3D-Bildern nur durch eine Handbewegung zu interagieren, ist der Ansatzpunkt dieser Arbeit. Das hier vorgestellte System soll es dem Zuschauer eines 3D-Abenteuers ermöglichen, direkt mit der Animation zu interagieren. Das bedeutet, dass der Benutzer die Animation aktiv durch Gesten steuern kann. Doch wie gut ist bei so einem System die *User Experience*? Wird ein solches System von Kinobesuchern überhaupt akzeptiert? Diese Fragen sollen in einer Benutzerstudie getestet werden.

1.2. Aufgabenstellung

Durch die Produktion von immer mehr Filmen und Sendungen in 3D ist in den letzten Jahren ein Massenmarkt für 3D-fähige Ausgabegeräte wie Projektoren und Displays entstanden. Als Folge sind heute nahezu alle Kinos mit 3D-Projektoren ausgestattet und viele Displays, vor allem im Home-Entertainment Bereich, verfügen über die Möglichkeit dreidimensionale Bilder auszugeben.

Parallel dazu ist ein Trend hin zu interaktiven Displays zu beobachten, welche mehr und mehr in der Werbebranche eingesetzt werden. Die Verwendung von Sensoren wie Kameras (z.B. Microsoft Kinect) erlaubt es mit Inhalten auf Displays zu interagieren (z.B. interaktive Spiele) und Informationen mit dem Display auszutauschen. Auf diese Weise können Inhalte von Displays für den Benutzer interessanter gestaltet werden sowie aufregende und nachhaltige Erfahrungen geschaffen werden. Weitere Möglichkeiten der Interaktion mit dem Display bieten Touch-Oberflächen und Mobiltelefone.

In Zukunft wird es dem Benutzer möglich sein, auch mit 3D-Inhalten zu interagieren. Eine Herausforderung ist hierbei heute noch die Verwendung eines geeigneten Eingabegerätes. Computermäuse erlauben im Moment lediglich Bewegungen im zweidimensionalen Bereich

und die Bewegung auf der z-Achse kann nur durch die Kombination von Klicken und Ziehen oder die Verwendung der Tastatur erreicht werden.

Diese Diplomarbeit beschäftigt sich mit dem Mobiltelefon als alternatives, natürliches Eingabegerät, welches neuartige Möglichkeiten der Interaktion durch Verwendung von Beschleunigungssensoren, Gyroskopen und dem Kompass ermöglicht. Der Hauptfokus liegt dabei darauf zu verstehen, wie Benutzer mit diesem Eingabegerät interagieren und welche Arten der Eingabe sinnvoll sind. Dazu soll eine mobile Anwendung entwickelt werden, welche es dem Benutzer erlaubt mithilfe von Gesten mit dreidimensionalem Inhalt zu interagieren. Diese Gesten sowie das Benutzerverhalten sollen in einer Studie evaluiert werden.

Aufgaben:

- Entwicklung eines Display-Clients, welcher interaktive, stereoskopische Inhalte rendern, animieren und in eine beliebige Szene einbetten kann
- Entwicklung eines Mobiltelefon-Client, welcher die Sensordaten des Geräts ausliest und Gesten erkennen kann.
- Entwicklung einer Client-Server-Architektur, welche die Kommunikation zwischen Display-Client und der mobilen Applikation ermöglicht.
- Durchführen einer Benutzerstudie, welche verschiedene Gesten untersucht und ihre Eignung für die Interaktion mit stereoskopischen Daten evaluiert.

1.3. Herausforderungen und Lösungsansätze

Ein System, das eine stereoskopische Animation anzeigen und mit der durch viele Menschen interagiert werden soll, bietet viele Möglichkeiten, allerdings auch einige Einschränkungen.

Das System soll in einer Kinoumgebung zum Einsatz kommen. Das bedeutet, dass viele Menschen zeitgleich mit der Animation interagieren werden. Das setzt voraus, dass alle Teilnehmer die gleichen Chancen haben, mit den Objekten zu interagieren. Daher wurde entschieden, das System als Client-Server-Architektur aufzubauen. Der Server soll hierbei die Anfragen der Smartphone-Clients aggregieren und an die Animation zur Auswertung weiterleiten. Das erleichtert auch die Rückmeldung der Animation an den Client, wenn zum Beispiel das Spiel gewonnen wurde.

Der Ansatzpunkt war, dass Benutzer mit einer stereoskopischen Animation interagieren sollen. Dazu muss erst einmal ein Framework und die nötige Hardware gefunden werden, die eine solche Animation überhaupt darstellen. Es wurde entschieden, für diesen Zweck Java3D zu verwenden. Nähere Gründe für diese Entscheidung befinden sich im Kapitel 5 Implementierung. Künstliche Stereoskopie hat die Eigenschaft, dass alle Menschen, die eine dreidimensionale Welt betrachten, das selbe Bild sehen. Dieses Bild kann vom jeweiligen Standpunkt relativ zum Display oder zur Leinwand etwas verzerrt sein. Dennoch ist der Eindruck, ob ein Objekt auf einen Benutzer zukommt oder nicht, für alle anwesenden Benutzer gleich. Das kann ein Vor- und ein Nachteil sein. Der Vorteil ist, dass man bei der

Entwicklung keine Rücksicht nehmen muss, wo sich der Benutzer relativ zur Leinwand befindet. Das macht es einfacher, ein System zu entwickeln, das alle Benutzer gleichermaßen anspricht. Der Nachteil ist, dass man dadurch mit den Möglichkeiten sehr eingeschränkt ist. So ist es zum Beispiel nicht möglich, eine Art „Pong"-Spiel zu entwickeln, bei dem gezielt Objekte auf einzelne Benutzer zukommen und diese die Objekte dann weiterschlagen können. Aus diesem Grund wurde entschieden, ein einfaches Spiel zu entwerfen, bei dem alle Benutzer mit einem Objekt interagieren können, sobald dieses in Reichweite ist.

Eine weitere Herausforderung an eine stereoskopische Animation ist das Abschätzen von Entfernungen. Die Welt ist sehr künstlich und es wurde davon ausgegangen, dass der Zeitpunkt, zu dem ein Objekt nah genug ist, von den Benutzern sehr subjektiv wahrgenommen wird. Um dem zu begegnen wurde eine Zone definiert, in der Objekte tatsächlich „fangbar" sind. Um diese Zone zu verdeutlichen gibt es mehrere Möglichkeiten. Ein erster Ansatz, eine semitransparente Ebene parallel zur Sichtebene zu verwenden, die anzeigte, wo die Zone anfing. Dieser Ansatz zerstörte allerdings den stereoskopischen Effekt, weswegen andere Indikatoren verwendet wurden. Am Ende entschied man sich für ein Glühen der Objekte und dafür, Objekte semitansparent werden zu lassen, sobald sie nah genug sind. Zusätzlich wird weiterhin eine transparente Ebene, die parallel zum Boden der Animation dargestellt wird und die tatsächliche Lage der Fanzone abbildet. In der Benutzerstudie soll dann herausgefunden werden, welche Indikatoren am besten geeignet sind, die Fangbarkeit darzustellen. Mögliche andere Indikatoren werden für die Zukunft verwendet.

Eine zentrale Frage ist ebenfalls, ob das System „natürlich" bedienbar ist. Hier wird von dem Ansatz ausgegangen, dass eine einfache Hand-Geste eine natürliche Interaktionsart ist. Komplizierte Handbewegungen sind nicht natürlich, sondern müssen erlernt werden. Beispiele sind Zeichensprachen. Daher wird auch hier eine möglichst einfache, wenn auch eindeutige Hand-Geste verwendet, um mit dem System zu interagieren. Dafür muss die Hand lediglich mit erhöhter Geschwindigkeit in eine Richtung bewegt werden, in die der Benutzer ein Objekt schlagen möchte. Um einen Vergleich mit einer „unnatürlichen" Geste zu haben, wird ebenfalls eine Touch-Geste verwendet. Bei dieser Touch-Geste wird einfach nur ein Knopf auf dem Touch-Display des Smartphones gerückt, um diese Geste auszuführen. In der Benutzerstudie soll dann verglichen werden, welche Art der Interaktion von Benutzern bevorzugt wird.

Die Gestenerkennung stellt ebenfalls eine Herausforderung dar. Komplexe Gestenerkennungen sind zwar inzwischen sehr gut erforscht, diese nachzuimplementieren hätte hier den Rahmen der Arbeit gesprengt. Zuverlässige Systeme, die man hätte einbauen können, wurden ebenfalls nicht gefunden. Daher wird hier eine einfache Schwellwertmessung durchgeführt. Die Richtung, in die die Geste ausgeführt wird, darf ebenfalls keine Rolle spielen. Wie erwähnt ist der Grund dafür ist, dass der Richtungsvektor dazu verwendet werden soll, das gefangene Objekt in die Richtung abzulenken, in die die Geste ausgeführt wurde.

Da das System komplett neu entwickelt wurde, wurde nicht auf eine gute Darstellung geachtet, sondern Wert darauf gelegt, dass man die Möglichkeit hat, bestehende Modelle einerseits in das System aus externen Systemen zu integrieren, andererseits sollte es auch möglich sein, diese zu verändern und die veränderten Modelle zusammen mit dem Zustand der anderen

Einstellungen lokal zu speichern. Dieses Problem wurde am meisten unterschätzt, da Java3D keine Möglichkeit bietet, die das Speichern vereinfachen.

Die verwendeten Objekte sollen Gegenstände darstellen, die aus dem Kino bekannt sind: Popcorntüten, Getränkedosen, Schokoriegel und Chipstüten. 3D-Modelle dieser Objekte sind nur kommerziell beziehbar. Die eigene Erstellung über ein 3D-Modellierungsprogramm wie zum Beispiel *Blender* hätte ebenfalls zu viel Zeit gekostet. Daher konnte für das Design der Modelle Maaret Posti von der Universität Oulu gewonnen werden, die großartige Arbeit geleistet hat. Blender-Modelle haben den Vorteil, dass sie in das WaveFront-Format exportiert werden können, das mit wenig Arbeit auch von Java3D eingelesen werden kann.

Das System soll auch in einer realen Umgebung, also in einem Kinosaal getestet werden. Für diesen Zweck wurde erfolgreich eine Kooperation mit dem Traumpalast in Esslingen aufgebaut. Um die Stabilität und Leistungsfähigkeit des Systems zu gewährleisten wurde die Benutzerstudie allerdings zunächst im Labor des Instituts durchgeführt.

1.4. Gliederung

Die Arbeit ist folgendermaßen gegliedert:

Kapitel 2 – Hintergrund: Visuelle Wahrnehmung und Interaktion durch Gesten erläutert zunächst die Grundlagen dieser Arbeit. Dabei werden zwei Bereiche behandelt: Einerseits wird das räumliche Sehen erklärt und verschiedene Techniken diskutiert, mit denen künstliche stereoskopische Welten erschaffen werden. Andererseits wird die Bedeutung von Gesten behandelt.

Kapitel 3 – Verwandte Arbeiten stellt Arbeiten aus den Bereichen der Interaktion mit stereoskopischen Projektionen oder der Gestenerkennung vor.

Kapitel 4 – Verwendete Hardware führt die während der Bearbeitung verwendeten Geräte ein und erläutert, welche Aufgaben sie in der Zeit übernommen haben.

Kapitel 5 – Implementierung: In diesem Kapitel werden detailliert die Architektur, die Bestandteile des Kasia-Systems und deren Zusammenspiel erläutert.

Kapitel 6 – Benutzerstudie gibt Aufschluss über den Versuchsaufbau, die Durchführung und die Ergebnisse der Benutzerstudie, die der Evaluation des Kasia-Systems diente.

Kapitel 7 – Zusammenfassung und Ausblick fasst die Arbeit und die Ergebnisse noch einmal zusammen und gibt einen Ausblick, wie zukünftige Entwicklungen im vorgestellten Bereich aussehen könnten.

Im Anhang A befinden sich die Spezifikationen der verwendeten Geräte aus Kapitel 4 auf Seite 41.

Im Anhang B befinden sich die Fragebögen der Benutzerstudie, sowie detaillierte Resultate nach der Ausewertung dieser Fragebögen.

2. Hintergrund: Visuelle Wahrnehmung und Interaktion durch Gesten

In dieser Arbeit wird die visuelle Wahrnehmung Aspekte angesprochen und die menschliche Gestik verwendet, um mit dem hier vorgestellten System zu interagieren. Die stereoskopische Bilderzeugung wird benutzt, um den 3D-Eindruck zu erzeugen, dass Gegenstände scheinbar auf den Benutzer zukommen. So soll erreicht werden, dass der Benutzer in diese künstliche Welt eintauchen kann. Für die direkte Interaktion mit der Szene werden zwei verschiedene Gesten eingesetzt: Eine Geste, die mit der Hand ausgeführt wird und eine einfache Drück- oder Touch-Geste, bei der der Benutzer lediglich im richtigen Moment auf ein Feld auf dem Touch Bildschirm seines Smartphones drücken muss.

In diesem Kapitel wird auf die Grundlagen der räumliche Wahrnehmung und der Gesteninteraktion eingegangen. Im ersten Teil wird erklärt, wie Menschen überhaupt ihre Umgebung räumlich wahrnehmen (vergleiche Abschnitt 2.1). Dabei spielt Tiefenwahrnehmung (Abschnitt 2.2.5 auf Seite 19) die Hauptrolle. In Abschnitt 2.3 auf Seite 21 wird dann erläutert, welche Möglichkeit es gibt, die Wahrnehmung auszunutzen, um künstliche, räumliche Welten zu erschaffen, wie es zum Beispiel in modernen 3D-Filmen geschieht.

Im zweiten Teil wird kurz darauf eingegangen, warum Gesten so wichtig für Menschen sind (vergleiche Abschnitt 2.4 auf Seite 27). Dort werden zudem die unterschiedlichen Arten von Gesten erläutert.

Als Referenz für die Abschnitte der visuellen Wahrnehmung sei durchgängig auf die Fachbücher *Das ABC der Optik* [MFK61] und auf *Augenheilkunde* [Rei96] verwiesen, die als Quellen dienten.

2.1. Visuelle Wahrnehmung

Anders als die meisten Tierarten verfügt der Mensch über echtes dreidimensionales (oder echtes räumliches) Sehen. Diese Art der Wahrnehmung ist in der Natur verhältnismäßig selten. Doch nur so gelang es Menschen mit Hilfe unserer Hände sehr präzise Arbeiten auszuführen. Ohne Tiefenwahrnehmung wäre es unseren Vorfahren sicherlich schwer gefallen, auf Distanz mit einem Speer oder mit Pfeil und Bogen zu jagen, oder anschließend das gejagte Tier mit großer Handfertigkeit auszunehmen.

Tiere verwenden hingegen andere Techniken um Entfernungen abzuschätzen. Bei Vögeln spielt beispielsweise die Bewegungsparallaxe (siehe Abschnitt 2.2.4) und das Größenverhältnis der Beute eine große Rolle, um Entfernungen abzuschätzen. Andere Tiere verlassen sich

gar nicht mehr auf die Augen und identifizieren ihre Beute beispielsweise durch Ultraschall-Echos (wie Fledermäuse), hörbaren Schall (wie Eulen), elektromagnetische Ströme (wie der Zitteraale), oder Geruch (wie Hunde und die meisten Raubkatzen).

2.2. Tiefenwahrnehmung: Grundlagen

Tiefeneindrücke sind nicht nur auf das stereoskopische Sehen beschränkt. Es gibt viele Faktoren, die uns einen Eindruck von räumlicher Tiefe geben. Dieses monokulare Sehen wird dementsprechend auch von Menschen beherrscht, die über kein echtes stereoskopische Sehen verfügen oder gar auf einem Auge blind sind.

In diesem Abschnitt soll erläutert werden, welche Faktoren zu Tiefeneindrücken führen. Neben dem eigentlichen binokularen oder stereoskopischen Sehen (Abschnitt 2.2.5), sollen zuerst in den Abschnitten 2.2.1 bis 2.2.4 auf der nächsten Seite die monokularen Techniken vorgestellt werden.

2.2.1. Statische Perspektive, Größenverhältnisse

Von vielen Objekten kennen wir aus Erfahrung ziemlich genau die jeweilige Form und Größe. Gute Beispiele dafür sind Straßen, Autos, Bäume, Häuser, Menschen und viele mehr. Sehen wir nun ein Objekt, wie zum Beispiel ein Auto, das sehr viel kleiner ist als das Bild, das wir in unserer Vorstellung haben, schließen wir daraus, dass sich dieses Objekt in viel größerer Entfernung befindet. Sehen wir zwei unterschiedliche Objekte mit Größenverhältnissen, von denen wir wissen, dass sie nicht zueinander passen, können wir sogar Aussagen über die Abstände der Objekte machen. Ein anderer Effekt ist die visuelle Konvergenz von parallelen Linien, wie bei einer Straße, die in am Horizont verschwindet. Von unserem Standpunkt aus läuft dieses rechteckige Objekt zu einem Trapez zusammen. Dieser Effekt wurde von Künstlern schon vor Hunderten von Jahren verwendet, um Landschaften mehr Tiefe zu geben. (Vergleiche auch Heinecke, Kapitel „Statische Perspektive“ und „Größentäuschung“ [Hei12])

Optische Täuschungen entstehen dann, wenn das Beobachtete nicht mit der Erfahrung übereinstimmt. Ein Beispiel hierfür ist eine Kamerafahrt durch eine Modelllandschaft, die uns im Film durch die Nachbildung der Größenverhältnisse real erscheint. Auf diesem Prinzip basieren die meisten Spezialeffekte in Filmen, da hier meistens Modelle anstelle echter Fahrzeuge oder Gebäude verwendet werden. Zudem würde es auch zu viel kosten, wegen eines Films ein komplettes Raumschiff zu bauen. Interessant ist, dass die Täuschung erkannt wird, wenn das Modell zu lange im Bild ist.

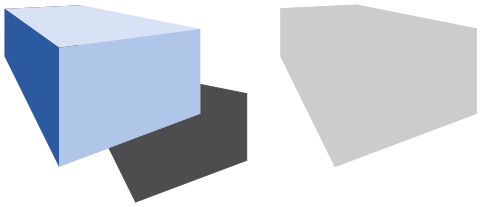

Abbildung 2.1.: Links: Objekt mit Beleuchtung und Schatten. Rechts: Ohne Beleuchtung

2.2.2. Verdeckung

Verdeckung gibt dem Menschen einen Hinweis über die Entfernung von Objekten zueinander. Ein näheres Objekt überdeckt aus der Sicht des Beobachters ein Objekt das weiter hinten liegt. Dabei spielt die Größe der Objekte keine allzu große Rolle, außer, dass größere Objekte im Vordergrund die Objekte im Hintergrund vollständig überdecken und diese daher nicht mehr sichtbar sind. Möglich wird die Erkennung einer Überdeckung dadurch, dass wir die die Form der meisten Objekte kennen und für uns unvollständige Objekte schließen können. (Vergleiche Heinecke Abschnitt „Gestaltgesetze: Gesetz der Schließung“ [Hei12])

2.2.3. Schattenwurf und Beleuchtung

Licht spielt bei der visuellen Wahrnehmung natürlich eine zentrale Rolle. Ohne Licht würden wir nichts sehen. Allerdings würden wir ohne eine direktionale Lichtquelle Formen nicht als plastisch sondern als flach wahrnehmen (siehe Abbildung 2.1). Mit Hilfe von Schatten lassen sich zusätzlich Aussagen treffen über die Größe des Objekts, der Stärke und die Position der Lichtquelle(n), sowie über die Lage der Objekte im Verhältnis zum Boden [Hei12].

2.2.4. Bewegungsparallaxe

Die Bewegungsparallaxe gehört zu den wichtigsten monokularen Tiefenwahrnehmungen. Viele Tiere, wie z.B. Raubvögel schätzen so die Entfernung zu ihrer Beute ab. Bei der Bewegungsparallaxe wird nur der Kopf etwas nach links oder nach rechts verschoben. Dadurch bewegt sich die bewegte Szenerie ebenfalls ein bisschen: Die Objekte, die nahe am Betrachter liegen bewegen sich stärker als Objekte im Hintergrund. Objekte in sehr großer Entfernung bewegen sich gar nicht mehr. So können Entfernungen verhältnismäßig genau ermittelt werden [Hei12].

2.2.5. Binokulares (stereoskopisches) und zweiäugiges Sehen

Mütze et al. [MFK61] verweist auf die Norm, in der zwischen binokularem und zweiäugigem Sehen unterscheiden wird.

Definition 1 (*Binuklares und zweiäugiges Sehen*)

[...] ***Binokulares*** *Sehen liegt vor, wenn es möglich ist, einen sterischen Tiefeneindruck (→ Tiefenqualität) wahrzunehmen, auch wenn der Beobachter selbst nicht in der Lage hierzu ist. Die beiden von*

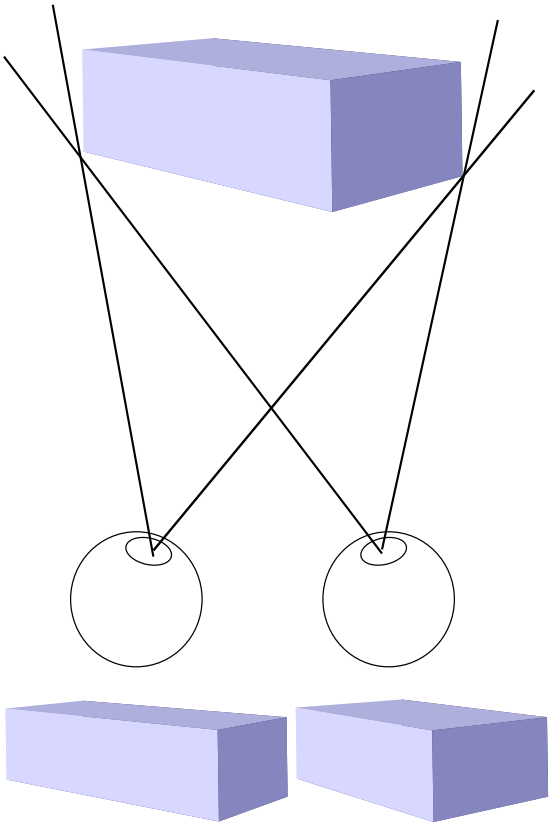

Abbildung 2.2.: Das binokulare (stereoskopische) Sehen

den Augen empfangenen Bilder des betrachteten Objektes unterscheiden sich etwas voneinander infolge der durch den Augenabstand bedingten Lage der beiden Aufnahmezentren; ***zweiäugiges*** *Sehen liegt vor, wenn beiden Augen das gleiche Bild dargeboten wird, so daß ein sterischer Tiefeneindruck nicht möglich ist, z.B. bei der Betrachtung einer Photographie.*

In dieser Arbeit wird somit binokulares mit stereoskopischem Sehen gleichgestellt und der letztere Begriff verwendet. Da sich die Arbeit hauptsächlich mit stereoskopischer Wahrnehmung beschäftigt, wird das zweiäugige Sehen nicht betrachtet.

Doch wie funktioniert nun das stereoskopische Sehen? Wichtig ist dabei die Tatsache, dass wir über zwei Augen verfügen. Durch den Abstand der Augen ergeben sich zwei leicht unterschiedliche Bilder. Im Gehirn werden diese Bilder übereinandergelagert und die Unterschiede berechnet. In Abhängigkeit der Abstände der Objekte im Verhältnis zum Hintergrund wird daraus der Tiefeneindruck generiert (s. Abbildung 2.2). Ob sich die Objekte näher oder entfernter vom Betrachter befinden, wird dadurch ermittelt, wie groß der Unterschied zwischen den Einzelbildern ist. Allerdings konnte bisher noch nicht geklärt werden, wie das Gehirn korrespondierende Objekte auf beiden Halbbildern erkennt und miteinander verknüpft.

Aufgrund unseres Augenabstandes verliert sich in der Realität der stereoskopische Effekt etwa nach acht Metern. Alle Tiefeneindrücke, die wir bei größerer Entfernung erfahren, haben oben genannte monokulare Effekte als Ursache.

2.3. Stereoskopie

Mütze et al. [MFK61] definieren Stereoskopie wie folgt:

Definition 2 (*Stereoskopie*)

*[**Stereoskopie:**] Verfahren zur Untersuchung des räumlichen Sehens unter künstlichen Bedingungen durch die Darbietung zweier Halbbilder, von denen das jeweils entsprechende dem rechten und dem linken Auge getrennt dargeboten wird (→ Stereoskopisches Sehen). Stereoskopie ist die Technik der Erzeugung dieses Raumeindrucks durch entsprechende Bilder, die Herstellung der Bilder auf zeichnerischem oder photographischem Wege (Stereoaufahmegerät → **Stereophotographie**) und die Darbietung der Bilder in unmittelbarer Betrachtung (→ **Stereoskop**) oder durch Projektion (**Stereoprojektion** → Stereophotographie). In der Kinematographie hat die S. Eingang durch die Schaffung des → **plastischen Films** gefunden.*

Damit umfasst die Stereoskopie den gesamten Themenkomplex der künstlichen räumlichen Bilderzeugung. Stereoskopie wird in vielen unterschiedlichen Bereichen verwendet: In der Computergrafik, in der Film- und Kinotechnik, sowie in der Fotografie. Auf einige dieser Techniken soll im Folgenden eingegangen werden. Wir unterscheiden hier nochmals zwischen Stereogrammen, bei denen es sich um einzelne stereoskopische Bilder handelt, und zwischen stereoskopischen Filmen.

2.3.1. Stereofotografie

Stereofotografien sind verhältnismäßig lange bekannt und sind die einfachsten Stereogramme (siehe Abbildung 2.3 auf der nächsten Seite). Von einer Szenerie, einem Objekt oder einer Landschaft werden zwei Bilder mit der gleichen Brennweite aufgenommen, aber mit einem horizontalen Abstand, der in etwa dem Abstand der menschlichen Augen entspricht. Diese Bilder können nun mit Hilfe eines Stereoskops oder mit etwas Übung durch einfaches Übereinanderlegen der beiden Bilder betrachtet werden. Das Gehirn generiert daraus wiederum einen Tiefeneindruck. Stereobilder sind seit Mitte des 19. Jahrhunderts bekannt.

2.3.2. Stereogramme durch Zufallspunkte

Das *Magische Auge* [Bac98] ist ein Buch, in dem computergenerierte Zufallspunkte enthalten sind, die durch forciertes Schielen zu einem dreidimensionalen Bild konvergieren. Ein Beispiel für ein solches Bild findet sich in Abbildung 2.4 auf der nächsten Seite. In den 1990er Jahren erfuhr diese Art der Kunst einen großen Boom. Der Grundstein für diese Art Bilder wurde bereits 1960 von Julesz [Jul60] gelegt. Dieser hat den Effekt beobachtet, dass zwei sich

Abbildung 2.3.: Stereofotografie: Through Project Native [ano]. Um einen Tiefeneindruck zu erhalten muss versucht werden, die beiden Punkte durch Schielen übereinanderzulegen.

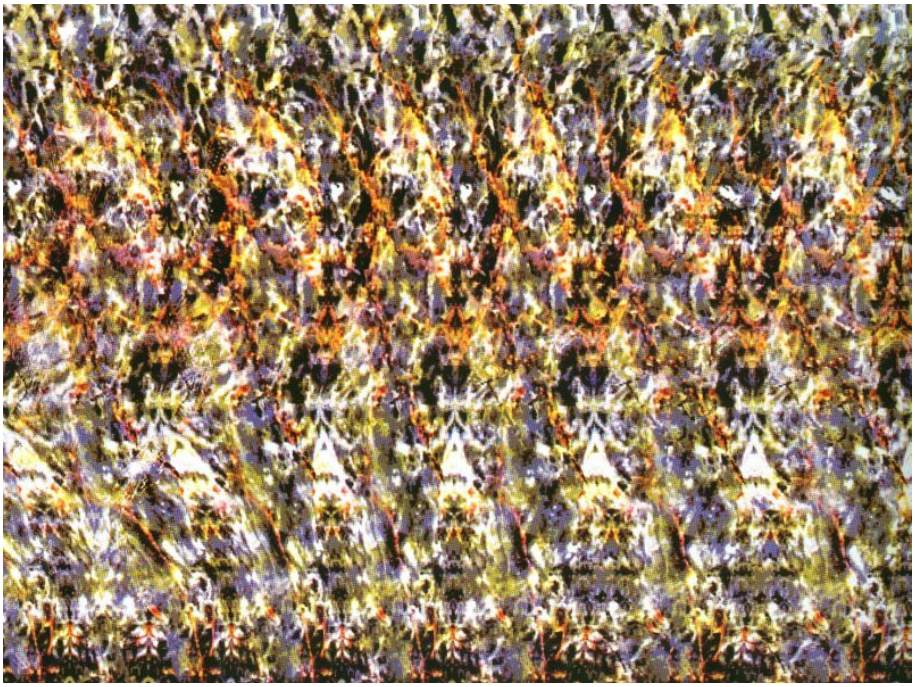

Abbildung 2.4.: *Das Magische Auge*: Beethoven [Bac98]

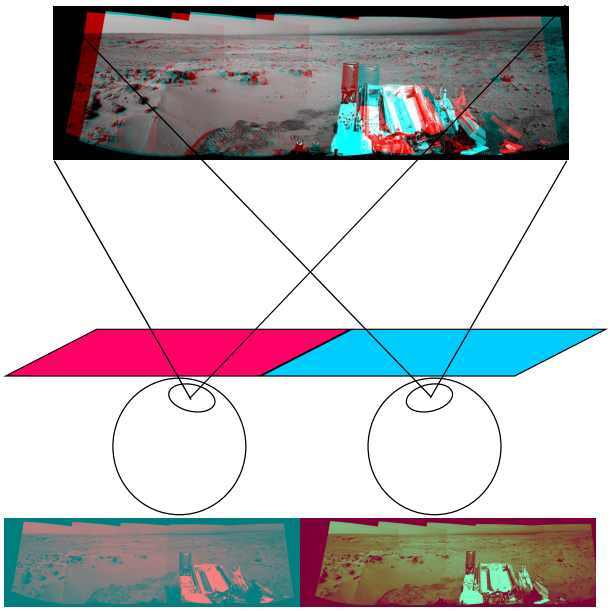

Abbildung 2.5.: Anaglyph 3D: durch Cyan- und Magenta-Filter werden die jeweiligen Farben aus dem Bild genommen und übrig bleiben die Halbbilder

leicht unterscheidende Bilder mit Zufallspunkten bei Überlagerung zu einem dreidimensionalen Effekt führen. Diese Technik wurde dann Tyler und Clarke [TC90] weiterentwickelt und in ein einziges Bild integriert.

2.3.3. (Farb-)Anaglyphe Verfahren

Anaglyphe Verfahren machen sich zunutze, dass zwei Halbbilder auf einem Bild übereinandergelegt sind. Anaglyph kommt aus dem Griechischen und bedeutet so viel wie *aufeinander darstellen*. Somit sind alle Verfahren, die zwei Bilder übereinander legen, anaglyphe Techniken. Allerdings hat sich diese Bezeichnung landläufig nur die farbanaglyphe Verfahren eingebürgert.

Bei farbanaglyphen Verfahren ist das eine Halbbild mit einer Farbe kodiert, das andere mit einer anderen. Durch spezielle 3D-Brillen, die mit einem entsprechenden Filter die Farbe für das jeweilige Auge herausfiltern, werden die Halbbilder wieder getrennt und im Gehirn richtig zusammengefügt. So entsteht der Tiefeneindruck. Ursprünglich wurden dafür die typischen Rot-Grün-Filter und -Bilder verwendet. Doch mit der Zeit stellte sich heraus, dass sich Rot-Cyan-Filter besser eignen. Daneben gibt es noch Techniken die Cyan und Bernstein, Magenta und Cyan, bzw. Gelb und Cyan verwenden. Allerdings haben sich diese noch nicht durchgesetzt.

Diese Farbanaglyphtechnik wurde schon in den 1920er Jahren für das Kino verwendet, allerdings erfuhr das 3D-Kino erst in den 1950er Jahren den ersten großen Boom. Da sich aber der Farbfilm immer mehr durchsetzte wurde das 3D-Kino mit farbanaglyphen Techniken

mehr und mehr uninteressant. Der Grund dafür dürfte sein, dass Filme mit dieser Technik selbst keine oder unnatürliche haben können. Durch die verwendeten Farbfilter wird der mögliche Farbraum für Filme erheblich eingeschränkt, da sonst der Tiefeneindruck verloren gehen würde.

2.3.4. Polarisation

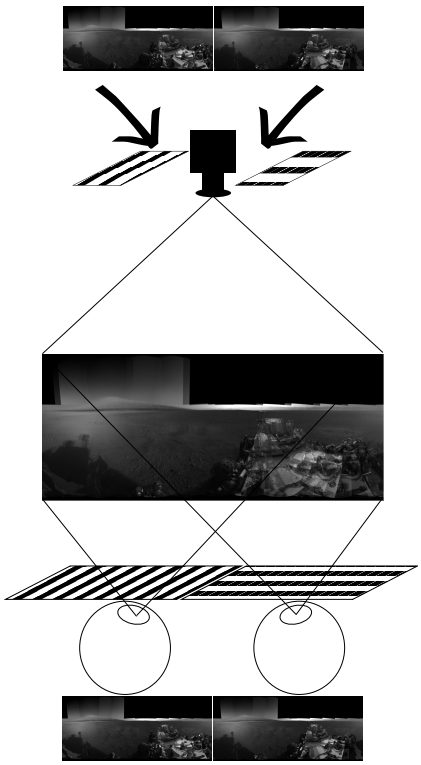

Abbildung 2.6.: Polarisation

Die lineare Polarisation ist eine Technik, bei der Licht durch Filter gelenkt wird, so dass nur noch Wellen mit einer bestimmten Polarisierung durchgelassen werden. Dadurch dass Lichtwellen nur eine einzige Drehrichtung haben, kann es durch einen zweiten Filter entweder hindurch gelassen (bei gleicher Ausrichtung der Filter) oder blockiert werden (bei anderer Ausrichtung des zweiten Filters. Projiziert man nun zwei Bilder mit unterschiedlicher Polarisation auf eine Leinwand, die die Polarisation des Lichts beibehält, kann man durch Brillen mit den jeweiligen Filtern pro Auge die Bilder wieder trennen.

Diese Technik hat sich Anfang der 1990er Jahren etabliert und fand lange Zeit nur in speziellen IMAX-3D-Kinos Verwendung. Erst mit der Weiterentwicklung der Kinotechnik und der Ausrüstung der Kinos wurde auch die Polarisationstechnik um das Jahr 2010 zum Standard.

Die Polarisationstechnik hat den Vorteil, dass die benötigten Brillen sehr leicht sind und dass Filme in der gewohnten Bildfrequenz gezeigt werden können. Allerdings muss der Betrachter den Film immer aus der selben Haltung betrachten, das heißt, er darf den Kopf nicht drehen, da sonst die Bilder nicht mehr richtig getrennt werden und die übereinandergelagerten Bilder in beiden Augen wahrgenommen werden.

2.3.5. Interferenz

Neben der linearen Polarisation gibt es auch noch andere optische Möglichkeiten, Bilder zu trennen und dabei nicht auf die Farben zu verzichten. Die Firma Dolby[1] bedient sich Interferenz-Brillen. Bei der Interferenztechnologie werden je Auge aus dem gesamten Farbspektrum abwechselnd einzelne Bänder herausgefiltert. Dadurch dass die Bereiche sehr nahe beieinander liegen, bemerkt der Zuschauer nichts von den fehlenden Farbbereichen je Auge. Beim Zusammenfügen der Bilder im Gehirn entsteht wieder ein normaler Farbeindruck.

Dieses Verfahren hat leichte Vorteile gegenüber der Polarisation und der unten beschriebenen Shuttertechnik. Dadurch, dass das Licht nicht polarisiert wird, muss nicht darauf geachtet werden, in welcher Position sich der Kopf im Verhältnis zur Projektion befindet. So können Paare einen 3D-Kinofilm auch dann ohne Einschränkung erleben, wenn der eine Partner dem anderen in den Armen liegt[2].

Diese Technologie wird beispielsweise im Kooperationskino in Esslingen [Tra] verwendet.

2.3.6. Shuttering

Beim Shuttering wird in sehr kurzen Abständen zuerst das Bild für das eine Auge und dann das Bild für das andere Auge projiziert. Getrennt werden diese Bilder beim Betrachter durch eine aktive Brille, die in der gleichen Geschwindigkeit die jeweiligen Brillengläser öffnet und schließt. Da sich die Bilder mit einer Geschwindigkeit von 50 Hz abwechseln, vereinigen sich die Halbbilder pro Auge zu einem einzigen Bild. Die Trennung ist dabei so gut, dass das andere Bild für das andere Auge nicht wahrgenommen werden kann. Bei dieser Technik kommt es darauf an, dass die Brille mit den Bilder perfekt synchronisiert werden. Ohne die Synchronisation sieht der Betrachter in etwa dasselbe wie Ohne Brille: Zwei übereinander gelagerte Bilder, die Unscharf zu sein scheinen. Für jeden Schaltvorgang der Brille wird daher zusätzlich zum Bild ein Infrarotimpuls vom Projektor gesendet.

Im Gegensatz zur Polarisation ist diese Technik unabhängig von der Kopfdrehung, so lange die Brille den Infrarottaktgeber empfangen kann. Zudem kann diese Technik auch verwendet werden, um Bildschirme, die über keine spezielle Polarisationstechnik verfügen, 3D-fähig zu machen. Voraussetzung dafür ist, dass die Grafikkarte die Technik unterstützt und ein

[1] www.dolby.com/de/de/index.html
[2] vorausgesetzt, er sieht dann noch die Leinwand

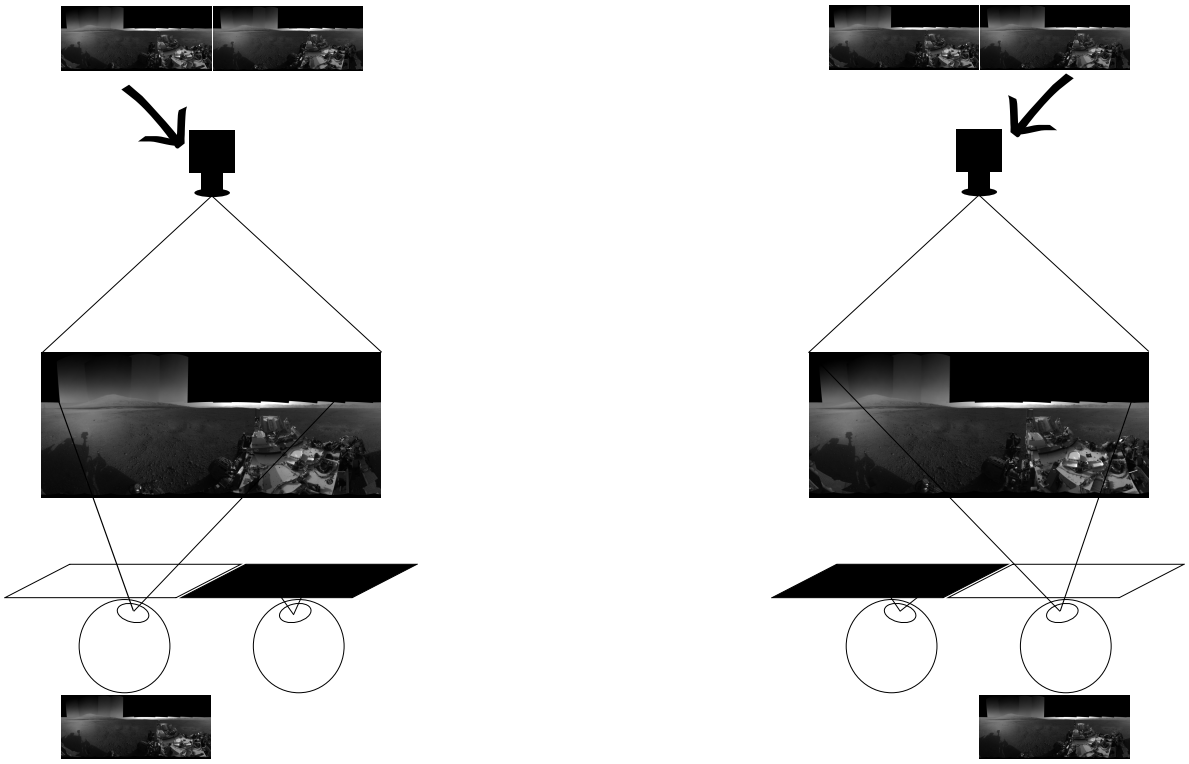

Abbildung 2.7.: Shutterbrillen. Links zum Zeitpunkt $2t$: Nur das linke Auge kann das linke Bild sehen. Rechts zum Zeitpunkt $2t + 1$: analog

Abbildung 2.8.: Epson Shutter Brillen

Infrarotsender für die Taktung angeschlossen ist. Allerdings nehmen viele Menschen das Öffnen und Schließen der Brillengläser als Flackern wahr, was zu Unwohlsein führen kann.

2.3.7. Verwendung in Kinos und am Institut

Derzeit sind viele unterschiedliche Systeme für die Erzeugung eines 3D-Effekts im Einsatz. Am Institut wird auf dem 3D-Fernseher (vergleiche Abschnitt 4.4.1 auf Seite 45) die Polarisationstechnik genutzt. Der institutseigene 3D-Beamer (siehe Abschnitt 4.4.2 auf Seite 46) verwendet eine Shuttertechnik mit aktiven 3D-Brillen. Auch in Kinos werden beide Technologien verwendet. Eine Besonderheit stellt das eher seltene Interferenz-Verfahren dar, das nur von Kinos mit Dolby-Systemen verwendet wird. Farbanaglph-Filme werden inzwischen nur noch im Zusammenhang mit 3D-Filmen aus der Vergangenheit gezeigt.

2.4. Gesten und Natürliche Interaktion

Laut dem Brockhaus für Psychologie ist die Gestik folgendermaßen definiert [SP01]:

Definition 3 (*Gestik*)

[zu lat. gestus „Gebärdenspiel"]: Gesamtheit der zielgerichteten Ausdrucksbewegungen des Körpers (Gesten), besonders der Hände und des Kopfes, die häufig begleitend als Untermalung oder zur Hervorhebung der sprachlichen Kommunikation eingesetzt werden. Ausdrucksbewegungen des Gesichts, die in der Kommunikation vorwiegend der Übermittlung emotionaler Zustandsinformationen dienen, werden als Miene oder →Mimik bezeichnet. Systeme konventionalisierter Gesten erlauben auch die Übermittlung komplexer Sachverhalte und können die gesprochene Sprache ersetzen (→Gebärdesprache).

Auf Gesten treffen wir überall im Leben. Wir winken unseren Freunden auf der anderen Straßenseite zu, wir nicken oder schütteln den Kopf während eines Gesprächs, um unsere Zustimmung oder Ablehnung zu untermalen. Und wir stellen etwas pantomimisch dar, wenn unser Gegenüber mit einem Begriff nichts anfangen kann. Gesten zu verwenden ist natürlich.

Gesten sind sehr vielseitig. Dort, wo wir nicht sprechen können, wie unter Wasser, im Gefecht oder in der Jagd, wenn man den Gegner oder die Beute nicht wissen lassen will, dass man sich in der Nähe aufhält, auch in Sportarten wie dem American Football, beim Fallschirmspringen und vielen anderen auch, haben sich eigene Zeichensprachen entwickelt. Diese Art der Sicht-Kommunikation ist allerdings auf ein sehr kleines Vokabular beschränkt.

Im Gegensatz zu den oben genannten Spezialanwendungen ersetzen echte Gebärdesprachen die komplette vokale Sprache. So können sich auch taubstumme Menschen mitteilen und sich unterhalten. Im Laufe der Zeit haben sich sehr viele Zeichensprachen entwickelt. Auch die Kommunikation zwischen Mensch und Schimpanse wurde so ermöglicht. Dem 2007 verstorbenen Schimpansenmädchen Washoe hatte das Wissenschaftlerpaar Gardner

die Amerikanische Zeichensprache (American Sign Language - ASL) beigebracht [GG88]. Innerhalb von 51 Monaten hatte sie bereits 132 Zeichen gelernt und konnte diese auch willentlich einsetzen um sich mitzuteilen. Mehr noch, sie konnte diese Gesten zum Teil ihren Artgenossen ohne menschliche Hilfe beibringen.

Menschenaffen sind auch sonst ein gutes Beispiel dafür, dass sich nicht nur Menschen durch Gesten verständigen. So wurde von Tomasello, Gust und Frost über Jahre eine Gruppe Schimpansen beobachtet, die tatsächlich Gesten für die Kommunikation benutzen [TGF89]. Die verwendeten Gesten ändern sich mit dem Alter, so dass es typischere Gesten der erwachsenen Individuen gibt und typische der noch jungen. So gibt es spezielle Gesten, wenn junge Schimpansen gestillt oder gekratzt werden wollten. Diese Gesten werden später nicht mehr verwendet, wenn sie nicht mehr notwendig sind.

Das legt den Schluss nahe, dass Menschen, bevor sie sprechen konnten, ähnlich wie Menschenaffen heute, sich ebenfalls nur über Gesten unterhalten haben. Auch heute sind Gesten in jeder Kultur verankert. Und diese haben von Kulturkreis zu Kulturkreis recht unterschiedliche Bedeutungen [Nat12]. So hat z.B. das „Ringchenzeichen“ viele Bedeutungen. In Italien und Deutschland hat es eine vulgäre Bedeutung, in Frankreich bedeutet es „Wertlos“, in Japan so viel wie „Bezahl mich!“, in den Vereinigten Staaten einfach „Okay“ und bei Tauchern bedeutet es „Alles in Ordnung“. Richtig deutlich wird das bei einfachen und vermeintlich eindeutigen Gesten, wie dem Nicken und dem Kopfschütteln. Ersteres bedeutet in westlichen und orientalischen Ländern „Ja“ und die zweite „Nein“. In Griechenland und in osteuropäischen Ländern ist die Bedeutung beinahe invertiert. Molnar-Szakacs et al. zeigten, dass es einen Unterschied macht, ob jemand, der aus derselben Kultur stammt wie der Beobachter, eine Geste macht, oder ob dies ein Fremder tut [MSWRI07]. Dabei wurden Probanden Gesten gezeigt, die von Mitgliedern der eigenen Kultur und von Mitgliedern einer anderen Kultur durchgeführt wurden. Dabei stellte sich heraus, dass selbst wenn die Geste, die aus der fremden Kultur stammt, ähnlich oder identisch (evtl. mit anderer Bedeutung) zu der der eigenen Kultur ist, das Gehirn dennoch weniger angeregt wird, als wenn ein Mitglied der eigenen Kultur dieselbe Geste ausführt. Die Forscher machen dafür sogenannte Spiegelneuronen[3] verantwortlich, die sonst auch das Mitgefühl für andere Menschen steuern.

Trotzdem dieser unterschiedlichen Interpretationen mancher abstrakten Gesten ist es möglich, sich interkulturell nur mit „Händen und Füßen“ zu unterhalten. Das Zeichen für „essen“, bei dem man pantomimisch Nahrung in den Mund steckt, oder in eine Richtung zu zeigen wird wahrscheinlich überall verstanden. Gerade solche einfachen Gesten sind universell einsetzbar und können auch von Computern verstanden werden. In dieser Arbeit wird mit eine schnelle Hand-Geste verwendet, um ein Objekt zu fangen. Hier wird davon ausgegangen, dass diese Geste als natürlich empfunden wird, da sie auch im Alltag verwendet wird, wenn etwa Dinge gefangen werden sollen, die auf jemanden zugeworfen werden. Zudem hat die Geste Ähnlichkeit mit der Schlagbewegung in einem Tischtennisspiel.

[3]Spiegelneuronen weisen beim Beobachten eines Vorgangs dieselbe Aktivität auf, als wäre das Beobachtete nicht nur passiv sondern aktiv erlebt worden

3. Verwandte Arbeiten

Diese Arbeit befasst sich mit verschiedenen Bereichen der Mensch-Computer Interaktion. Dabei steht die Multi-User Interaktion mit einer stereoskopischen Animation im Vordergrund. Viele andere Veröffentlichungen beschäftigen sich mit der Interaktion durch unterschiedliche Gesten. Auf diese wird noch genauer eingegangen. Zunächst werden Systeme präsentiert, die im Kern den Fokus auf die Interaktion mit stereoskopischen Anwendungen legen.

Das Kapitel ist folgendermaßen gegliedert: In Abschnitt 3.1 wird auf die in den Arbeiten verwendete Hardware eingegangen. Anschließend wird in Abschnitt 3.2 auf der nächsten Seite beschrieben, warum haptisches Feedback wichtig für die Interaktion mit stereoskopischen Umgebungen ist. In Abschnitt 3.3 auf Seite 31 wird eine Technik beschrieben, die eine mögliche direkte Interaktion mit stereoskopischen Displays präsentiert. Abschnitt 3.4 auf Seite 31 beschäftigt sich mit einer zentrale Frage bei der Multi-User-Interaktion, nämlich wie mehreren Benutzern Touch-Gesten zugeordnet werden können. Besondere Herausforderungen stellen sich bei der Interaktion mit großen Displays, wie z.B. Kinoleinwänden, da dort keine Touch-Gesten verwendet werden können. In Abschnitt 3.5 auf Seite 32 werden alternative Interaktionsmöglichkeiten vorgestellt, die z.B. die Nutzung eines Smartphones vorsehen. Ein kritischer Punkt bei der Interaktion von Menschen mit Displays jeder Art ist die Erkennung von Gesten, die der Benutzer durchführt. Abschnitt 3.6 stellt verschiedene mögliche Techniken vor um Gesten zu erkennen, sowie Studien darüber welche Gesten Benutzer als natürlich empfinden. Zum Schluss wird in Abschnitt 3.7 auf Seite 40 ein kommerzielles System vorgestellt, das sich mit der Interaktion zwischen Kinobesuchern und der Leinwand auseinandersetzt.

3.1. Verwendete Hardware

Die meisten der folgenden Arbeiten sind bereits mehrere Jahre alt. Mobiltelefone, die neben einem leistungsfähigen Prozessor auch über Beschleunigungssensoren und Touchscreens verfügen, gibt es erst seit dem Jahr 2005. Das Smartphone *Samsung SCH-S310* wird von Vajk et al. als das erste Mobiltelefon genannt, das über Beschleunigungssensoren verfügt [VCBE08]. In dieser Zeit wurden Beschleunigungssensoren zum ersten Mal auch in der Öffentlichkeit interessant. Der Spielehersteller Nintendo schaffte mit seiner *Wii*-Spielekonsole 2006 den Durchbruch [inc13] und revolutionierte die Interaktion mit modernen Konsolenspielen. Kurze Zeit nach Erscheinen der Konsole wurde deren Steuerung, die *WiiMote* (von Wii Remote abgeleitet), nach und nach auch für die Wissenschaft interessant. Das Besondere an der WiiMote ist, dass sie neben den Beschleuigungs- und Lagesensoren auch über eine Infrarotkamera verfügt. Durch zwei stationäre Sender, die über oder unter dem Bildschirm

angebracht werden, kann so die Lage der WiiMote relativ zum Bildschirm ermittelt werden. Durch diese Technik lässt sich die Zeigeposition sehr präzise ermitteln. Durch einen eingebauten Vibrator und einen einfachen Lautsprecher an der WiiMote verfügt das Gerät auch über haptisches und akustisches Feedback für den Benutzer. Moderne Smartphones verfügen inzwischen über ähnliche Ausstattungen, mit Ausnahme der Infrarotkamera: Die meisten integrierten Kameras können bisher nur sichtbares Licht empfangen.

Noch bevor es Smartphones oder die Wii gab, haben sich Forscher der Universität Oulo einen eigenen Prototypen gebaut: Die (SoapBox) (Sensing, Operating and Activating Peripheral Box) [TY002]. Auch dieses Gerät war mobil, das heißt batteriebetrieben, verfügte über eine Bluetooth-Anbindung und besaß erstmals Beschleunigungssensoren, ein Gyroskop und einen elektronischen Kompass. Optional konnte man ebenfalls Temperatur- und Helligkeitssensoren integrieren. Außerdem besaß es Knöpfe, mit denen man interagieren konnte. Da es sich hierbei um einen reinen Prototypen handelte, war die Verbreitung und Vergleichbarkeit mit anderen Geräten sehr beschränkt.

Heute sind in den meisten Smartphones Beschleunigungssensoren vorhanden. Der Absatz von Smartphones wächst jedes Jahr. So wurden 2012 in Deutschland 22,9 Mio neue Smartphones verkauft, was einer Steigerung von 43% zum Vorjahr entspricht. Damit liegt der Anteil von Smartphones im Verhältnis zu allen verkauften Mobiltelefonen bei 70% [SP12]. Das bedeutet wiederum, dass die Möglichkeit, mithilfe eines Smartphones andere Geräte zu steuern, sei es durch Gesten oder andere Verfahren, immer mehr Leuten zur Verfügung steht.

3.2. Haptisches Feedback bei stereoskopischen Interaktionen

Die Interaktion mit einer virtuellen Welt stellt die die Menschen vor Herausforderungen. Eine davon ist die Anstrengung der Augen, wenn man lange Zeit auf eine künstliche, dreidimensionale Welt starrt. Die andere, auf die zunächst eingegangen werden soll, ist das fehlende Feedback an den Benutzer. Wie im folgenden Abschnitt 3.3 erläutert wird, fehlt in der virtuellen Welt das Berühren und das Gewicht des zu greifenden oder bewegenden Objekts. Um die Illusion, völlig in die virtuelle Welt einzutauchen, ist eine solche Rückkopplung durchaus nicht unerheblich.

Einem haptischen oder akustischen Feedback bei der Interaktion durch Gesten messen auch Mäntyjärvi et al. [MKKK04] und später Kela et al. [KKM+06] einen hohen Stellenwert bei, wenngleich in ihren Prototypen dieses nicht implementiert wurde. Ihrer Meinung nach soll das Feedback durch Töne oder Vibration gelöst werden und das Feedback an den Benutzer soll in späteren Studien erforscht werden [MKKK04].

3.3. Interaktion mit stereoskopischen Displays

Valkov et al. beschreiben die Verwendung von Touch-Displays, bei denen der Benutzer direkt mit dem Finger die stereoskopischen Umgebung manipuliert [VGH12]. Dabei steht im Zentrum der Forschung das haptische Feedback für den Benutzer. Das Problem bei der Interaktion mit virtuellen Objekten ist, dass es für einen Benutzer schwer ist herauszufinden, ob er ein Objekt gerade berührt oder nicht. In der realen Welt ist das kein Problem, da man spürt, wenn man einen Gegenstand berührt. In der virtuellen Welt ist dies allerdings nicht so trivial. In der Luft schwebende, virtuelle Objekte sind nicht greifbar, da sie nicht physisch vorhanden sind. Würde man nur die Oberfläche des Bildschirms berühren, ohne die stereoskopischen Effekte in Betracht zu ziehen, so würde einer von zwei Effekten eintreten: Im Falle der positiven Parallaxe würden Benutzer das Objekt nicht erreichen, da die Bildschirmoberfläche zu weit von dem „dahinter liegenden“ Objekt befindet. Im Falle der negativen Parallaxe würde der Finger durch das schwebende Objekt hindurchgehen, da dieses sich vermeintlich „vor“ dem Bildschirm befindet. Dadurch würde der immersive Effekt gestört werden, da Menschen im allgemeinen davon ausgehen, dass sie ein Objekt spüren wenn sie es berühren anstatt es nicht zu erreichen oder gar durch es hindurchzugreifen.

Der von Valkov et al. beschriebene Ansatz geht davon aus, dass sich ein Benutzer, der sich auf die Interaktion mit einem stereoskopischen Objekt konzentriert, nicht darauf achtet, ob dieses Objekt die Lage verändert, so lange sich diese Bewegung durch eine Größenänderung neutralisiert wird [VGH12]. Mit dieser Vorstellung wurde von den Autoren ein System entwickelt, das den Abstand des Fingers relativ zur Bildschirmoberfläche verfolgt und in dem Maße, wie sich der Finger darauf zubewegt, wird das ausgewählte Objekt auf den Finger zubewegt. Gleichzeitig wird die Größe perspektivisch verändert, so dass der Benutzer immer dieselbe Größe des Objekts wahrnimmt. Für das Daraufzubewegen eines Fingers wird mit Hilfe einer Kinect die Entfernung des Fingers zum Bildschirm verfolgt.

Valkov et al. zeigen, dass Benutzer tatsächlich von der Veränderung kaum abgelenkt werden und dass das haptische Feedback im Moment des tatsächlichen Berührens als „motivierend“ empfunden wird. Einer ihrer Eingangsthesen war, dass die Parallaxe keinen Einfluss auf die Entfernungswahrnehmung hat, wenn Objekte wie dort beschrieben manipuliert werden. Allerdings stellte sich in der Benutzerstudie heraus, dass Objekte, die bei einer positiven Parallaxe starten und sich auf den Finger zubewegen, von den Probanden als statisch wahrgenommen werden, weswegen die Teilnehmer der Studie die Entfernung zu den Objekten unterschätzten.

3.4. Multiuser Interaktion mit Touch-Display

Für die Interaktion mit Displays durch mehrere Benutzer seien Rofouei et al. erwähnt [RWBT12]. Diese verwenden für die Interaktion ein einfaches Touch-Display, sie sind allerdings in der Lage, die Berührung den einzelnen Benutzern zuzuordnen. Dafür wird ebenfalls eine Kincet verwendet, die Bewegungen der Benutzer aufzeichnet. Zudem wird ein Mobiltelefon, das über Beschleunigungssensoren verfügt, ausgelesen. Die Beschleunigungsdaten der

Telefone der Benutzer werden dann mit den Beschleunigungsdaten, die eine Kincet extern von den Benutzern aufgezeichnet hat, verglichen und den jeweiligen Benutzern zugeordnet. Wurde eine Berührung auf dem Touch-Display erkannt, so lässt sich mit hoher Wahrscheinlichkeit ermitteln, von welchem Benutzer diese Bewegung ausging. Allerdings ist diese Möglichkeit der Interaktion auf eher kleinere Displays mit wenigen Benutzern beschränkt, da die verwendete Kinect technische Grenzen setzt. Zudem setzt das System voraus, dass ein Benutzer die Hand, die das Mobiltelefon hält, immer mitbewegt, wenn er das Display berührt. Daher ist dieses Verfahren für das Anwendungsfeld Kino eher ungeeignet. Dennoch könnte diese Technik mit der Arbeit von Valkov et al. [VGH12] kombiniert werden, um eine Interaktion mit stereoskopischen Anwendungen für mehrere Benutzer zu ermöglichen. Dazu müsste die Interaktion allerdings auf eine Person zur gleichen Zeit beschränkt sein, damit sich der Effekt, das berührte Objekt auch zu fühlen, auch von der agierenden Person richtig wahrgenommen wird.

3.5. Interaktion mit großen Displays

In den Ansätzen von Valkov et al. [VGH12], beziehungsweise von Rofouei et al. [RWBT12], geht es generell darum, wie ein oder mehrere Benutzer mit einem Touch-Display interagieren können. Nimmt man allerdings die Operationen hinzu, die man für gewöhnlich in stereoskopischen Umgebungen durchführen will (z.B. Navigation in der Szene, Auswählen, Drehen, Bewegen und Skalieren von 3D-Objekten), stößt man schnell an die Grenzen der Leistungsfähigkeit von zweidimensionalen Eingabegeräten wie eines Touch-Displays. Steinecke et al. [SHSK08] geben hierfür eine Reihe von Lösungsansätzen, die eine Kombination eines Touch-Displays mit einem mobilen Gerät vorsehen. Ihrer Meinung nach lassen sich dadurch einige der Unzulänglichkeiten bei der Manipulation stereoskopischer Szenen beseitigen.

Einen spannenden Ansatz zur Multi-User Interaktion mit potentiell großen Displays stellen Hyakutake et al. vor [HOKK10]. In ihrer Arbeit geht es nicht zwangsläufig um stereoskopische Anwendung, dafür aber um die Interaktion mit mobilen Geräten im öffentlichen Raum. In diesem Ansatz kommen transparente Marker zum Einsatz, die auf einer Polarisationsfolie aufgebracht sind. Diese Marker lassen nur polarisiertes Licht hindurch. Für den Betrachter werden diese Marker nicht wahrgenommen solange dieser keine Polarisationsbrille trägt (vergleiche 2.3.4 auf Seite 24). Die Kamera eines mobilen Geräts, vor der ein Polarisationsfilter angebracht wurde, kann diese Marker jedoch mit sehr hoher Zuverlässigkeit erkennen. Durch das Erkennen der Marker kann das System genau bestimmen, welcher Teil des Displays gerade betrachtet wird und es kann diese Informationen für die Interaktion nutzen. Dadurch dass das System die Größe und Lage der Marker kennt, kann durch Zurückrechnen der Transformation zudem auch die Lage der Kamera im Raum relativ zum Display bestimmt werden.

Für eine mögliche Anwendung in einem Kino wäre eine solche Lösung nahezu ideal, da neben Gesten auch der Bildausschnitt der Leinwand bestimmt wird und somit die direkte Interaktion mit Teilen der stereoskopischen Szene möglich wäre. Allerdings wäre diese Art der Interaktion nicht mehr „natürlich“ da man bewusst mit dem Mobiltelefon auf das

Display zeigen muss. Die praktische Umsetzung würde zudem einige Probleme aufwerfen: Die von Hyakutake et al. vorgestellte Technik basiert auf der Polarisation von Licht, um die Marker transparent zu machen. In den meisten Kinos wird jedoch gerade die Polarisation verwendet, um die stereoskopischen Bilder voneinander zu trennen. Eine alternative Technik zur Polarisation für das Anzeigen stereoskopischer Bilder wäre Shutterbrillen zu verwenden. Ein weiteres Problem ergibt sich daraus, dass die vorgestellte Technik nur schlecht bei einem dunklen Hintergrund der Animation funktioniert, da für die Kamera die Marker bereits als schwarz wahrgenommen werden. Als Lösung dieses Problems schlagen Hyakutake et al. vor, von Zeit zu Zeit einen kurzen Lichtblitz zu senden und verweisen auf die inzwischen existierenden Mobiltelefone mit Hochgeschwindigkeitskameras [HOKK10]. In einem Kino, bei dem Menschen die ganze Zeit auf die Leinwand sehen, werden solche Blitze mit hoher Wahrscheinlichkeit als extrem störend empfunden. Um die Marker auch von der letzten Kinoreihe erkennen zu können, müssen die Marker entsprechend groß sein. Das sollte kein Problem sein. Jedoch wird die Auflösung der Erkennung durch die geringere Anzahl an Markern vermutlich etwas eingeschränkt. Zuletzt muss das Kino mit einer solchen Technik ausgerüstet sein, was mit hoher Wahrscheinlichkeit kein Kinobetreiber tragen würde. Gerade aus dem hohen technischen Aufwand haben wir uns dagegen entschieden, diesen Ansatz weiter zu verfolgen.

3.6. Natürliche Interaktion und Gestenerkennung

Neben den in den vorherigen Kapiteln eingeführten Interaktionsmöglichkeiten soll nun auf die Gestenerkennung und deren Anwendung eingegangen werden. Gestenerkennung gilt schon seit langem als Grundlage der Interaktion. Im Falle der Interaktion mit anderen Menschen, was für dieses Paradigma die Grundlage darstellt, schon sehr viel länger als dass es Computer gibt (siehe Kapitel 2.4 auf Seite 27). Gesten werden überall, zu jeder Zeit und bei jeder Gelegenheit verwendet. Auch in der Interaktion mit Computern und anderen technischen Geräten wird schon seit langem auf unterschiedliche Gesten gesetzt.

In diesem Abschnitt sollen Techniken beschrieben werden, wie Gesten erkannt werden. Für die Einführung dient der Abschnitt 3.6.1. Anschließend werden die Hidden Markov Models erläutert. Dieses Verfahren wird von vielen verschiedenen Techniken verwendet und soll daher hier kurz erklärt werden. Anschließend werden die verschiedenen Möglichkeiten zur Gestenerkennung vorgestellt. Zuerst wird in Abschnitt 3.6.3 auf Seite 35 die Erkennung und der Einsatz von Hand- und Zeigegesten vorgestellt. Dann wird in Abschnitt 3.6.4 die zweidimensionale Gestenerkennung auf PDAs und Touch-Displays erläutert bevor in Abschnitt 3.6.5 auf Seite 37 und allen darauf folgenden Abschnitten um die Erkennung von dreidimensionalen Gesten geht.

3.6.1. Gestenerkennung: Ein Überblick

Gerade in der modernen Informationswelt wird in vielen Bereichen mit Gesten und deren Erkennung gearbeitet. Im weiteren Zusammenhang könnte man sogar die Bewegung einer

Maus als Geste auffassen. Allerdings soll diese Art der Interaktion hier nicht behandelt werden. In diesem Abschnitt wollen wir nur auf die Erkennung von Hand- und Körpergesten eingehen, die tatsächlich auch interpretiert werden können.

Es gibt unterschiedlichste Ansätze für die menschliche Gestenerkennung. Weit verbreitet ist, direkt die Geste durch Kameras und durch Bilderkennung zu realisieren. Die für die Spielekonsole *Xbox 360* von Microsoft entwickelte Gestenerkennung *Kinect* [Xbo13], die bereits erwähnt wurde, bedient sich dieser Möglichkeit. Dieses Gerät enthält neben einer optischen Kamera auch einen Infrarot Tiefenlaserscanner, mit dem man die Bewegungen von Menschen im dreidiminsionalen Raum verfolgen. Durch die eingebaute Bilderkennung wird innerhalb der Kinect ein einfaches Skelett-Modell erstellt, dessen Gliedmaße bekannt sind und die in Anwendungen separat ausgelesen werden können. Hierfür stehen unterschiedliche APIs zur Verfügung, was die Entwicklung eigener Software sehr vereinfacht. Die Spielekonsole Playstation 3 (kurz PS3) [Pla13] von Sony erkennt Gesten ebenfalls durch eine Kamera. Allerdings werden hier große Marker verwendet, mit denen die Benutzer interagieren und die, die vom System erkannt werden durch ihre Farbe und Bewegung erkannt werden.

Von den Spielekonsolen abgesehen gibt es es ein paar sehr spezielle Anwendungen für Gestenerkennungen. Wie in Abschnitt 3.6.3 beschrieben wird, können selbst Gesten der Gebärdensprache mit Hilfe eines Spezialhandschuhs erkannt werden.

3.6.2. Hidden Markov Models

Die meisten anderen Verfahren wenden Hidden Markov Models an (HMM). Markov Modelle sind ein statistisches Verfahren, das von Andrei A. Markov Anfang des 20. Jahrhunderts entwickelt wurde. Ziel war es, die „Buchstabensquenzen in Werken russischer Literatur zu modellieren [...] Diese wurden aber dann zu statistische Werkzeuge weiterentwickelt“ [MS99].

Markov Modelle beruhen au der Annahme, dass Zustände nur von einem anderen in der kürzeren, aber nicht aus der entfernteren Vergangenheit abhängig sind und man anhand dieses Zustandes Aussagen über zukünftige Zustände ableiten kann. So kann man Markov Modelle als einen Nichtdeterministischen Endlichen Automaten ansehen. Dabei sind die Kanten zwischen den Zuständen die Übergangswahrscheinlichkeiten. Bei einem *sichtbaren* (visible oder vanilla) Markov Modell (VMM) kann man die Zustände als Ausgaben betrachten. Dieses Modell heißt sichtbar, da man zu jeder Zeit weiß in welchem Zustand man sich gerade befindet. Zustände können beliebig viele ein- und ausgehende Kanten haben. Allerdings ist die Summe der Wahrscheinlichkeiten aller ausgehenden Kanten je Zustand gleich 1.

Bei *versteckten* Markov Modellen (HMM) kennt man die Zustandsfolge nicht. Das Modell ist *versteckt*. Man kennt zwar die Zustände und die Übergangswahrscheinlichkeiten. Man kennt auch die Wahrscheinlichkeiten der Ausgaben abhängig vom Übergang eines Zustands in einen anderen Zustand. Aber anhand der Ausgaben muss man auf mögliche Zustandsfolgen und deren Wahrscheinlichkeiten schließen.

Markov Modelle werden vorwiegend in der Erkennung von natürlicher Sprache verwendet [MS99] und tritt im Zusammenhang des *Maschinellen Lernens* auf. Auch für die Erkennung von Mustern und Gesten wird es sehr gerne herangezogen. Im Falle der Gesten muss es eine Abbildung von Gesten Sensordaten auf eine endliche Anzahl an Zustände geben (z.B. eine Teilgeste bewegt sich nach rechts oben). Verschiedene Folgen von Zuständen ergeben eine Geste. In der Trainingsphase werden die Zustände miteinander verbunden und mit Übergangswahrscheinlichkeiten versehen. Dadurch kann durch mehrere Zustandsfolgen und sogar durch ähnliche auf die jeweilige Geste geschlossen werden.

3.6.3. Handgestenerkennung

Ein Beispiel für die Gestenerkennung durch Bilder ist der Ansatz von Lee und Hong, welche eine *Bumblebee* Stereokamera verwenden [LH10]. Anstatt den gesamten Körper zu erkennen beschränken sie sich auf die Erkennung der Hand und deren Bewegung. Dabei werden die beiden Bilder der Stereokamera verglichen und aus dem Abstand der Objekte zueinander mit Hilfe der *Summe absoluter Unterschiede* (sum of absolute differences - SAD) eine Tiefenkarte erstellt. Dabei wird „das Ding, das sich bewegt, als Hand erkannt“ [LH10]. Um den Rechenaufwand zu minimieren wird alles andere außer der Hand entfernt und nur was sich in geringer Entfernung der erkannten Hand befindet als Handgeste interpretiert. Lee und Hong konnten somit ein Vokabular von acht Gesten mit einer Genauigkeit von 85% erkennen und dieses System für ein chinesisches Schachspiel verwenden.

Um reine Zeigegesten geht es in der Arbeit von Guan und Zheng [GZ08]. Auch sie verwenden eine Stereokamera, um die Geste und deren Richtung zu erkennen. Dabei wird der obere Teil des Körpers erkannt: Kopf und Arme. Anhand der Kopfposition, der Entfernung und dem Winkel zum ausgestreckten Finger lässt sich so die Zeigeposition bestimmen, ohne das Gesicht und die Blickrichtung erkennen zu müssen. In einer Benutzerstudie wurde das Verfahren einmal im Labor und einmal in einem Stadtplanungsmuseum mit einer großen unebenen Bodenfläche getestet. Im Labor wurden auf dem Boden zwischen 3,5 m und 12 m im Abstand von 3,5 m 12 Flächen verteilt: Je Abstand eine in der Mitte und je links und rechts im Abstand 1,4 m. Flächen hatten eine Maximalgröße von 20 cm mal 25 cm. Mit dieser Technik erreichen sie eine Zeigegenauigkeit von durchschnittlich 95,6%. In einem Abstand von 11 m lag die Erkennung im Durchschnitt noch 93,3%.

Einen anderen Ansatz um Handgesten zu erkennen, verwendeten Fels und Hinton [FH95]. Die Idee hinter dieser frühen Arbeit war es, Handgesten in Laute zu übersetzen. Diese Arbeit stammt bereits aus dem Jahr 1995 und soll vor allem der Kommunikation von sprachbehinderten Menschen dienen. Dazu benutzen sie einen *Cyberglove*, einen Sensorhandschuh, der an 18 Stellen die Spannung der Hand misst, zusätzlich einen Lage- und Drehsensor, einen *ContactGlove*, welches ebenfalls ein Sensorhandschuh ist und der die Berührungspositionen der Finger ermittelt, sowie ein Fußpedal. Das System bedient sich neuronaler Netzwerke, um Gesten zu lernen und zu festigen. Wurde eine Geste erkannt, wird der korrespondie-

rende Formant[1] synthetisiert und als Laut ausgegeben. Fels und Hinton sind der Ansicht, dass die Trainingsdaten eines Benutzers für die Erkennung anderer Benutzer verwendet werden können, jedoch muss jeder Benutzer mehr als 100 Zeitstunden aufwenden, um mit dem System zurechtzukommen. Die Komplexität und Fingerfertigkeit für das Erlernen des Systems wird mit der eines Instruments verglichen.

Eine jüngere Arbeit hat eher die Steuerung eines Computers und Handgesten als Mausersatz im Zentrum der Forschung. Pandit et al. [PDM+09] arbeiteten ebenfalls an einem Handschuhsystem, mit dessen Hilfe viele Aufgaben bewältigt werden sollen: Neben dem erwähnten Maus- und Tastaturersatz, sollte das System z.B. auch für die Hauselektronik und für die Interaktion mit dreidimensionalen Objekten dienen. Hier werden einfache Baumwollhandschue verwendet, die mit Tastern, 3-Achsen-Beschleunigungssensoren und einem Microcontroller und einer *XBee*-Sensoreinheit verbunden sind und welche über einen Funksender verfügt. Die Signale werden am Computer ausgewertet und in Mausbewegungen übersetzt. Die Gestenerkennung an sich ist sehr rudimentär und beschränkt sich hier auf wenige festgelegte Gesten, wobei eher das Zusammenspiel der Handschuhe ausgelesen wird, als dass tatsächlich Gesten erkannt werden. Trotz der zahlreichen Probleme, die vorwiegend mit drahtlosen Übertragung, der Skalierung und der Interpretation der Daten zu tun haben, stellen die Autoren das System als ein recht praktisches Alternativkonzept zum bisherigem Maus-/Tastaturparadigma dar. Leider fehlt in der Publikation eine Benutzerstudie.

3.6.4. Einfache 2D Gestenerkennung

Gesten werden im Kontext mit Computern und Mobiltelefonen sehr heterogen verwendet. Ein Ansatz für die Erkennung von 2D-Zeichengesten, wie z.B. einem Strich von links unten nach rechts oben, auf einem Touch-Display oder ähnlichem verfolgen Wobbrock, Wilson und Li mit ihrem *$1 Recognizer for User Interface Prototypes* [WWL07]. Dieser verwendet nur geometrische Operationen in seinem sehr einfachen Vier-Schritt-Algorithmus:

Schritt 1: Neuaufbau des Punktepfads Sensordaten werden als Menge von Einzelpunkte in die Anwendung gegeben. Dabei sind die Abstände zwischen den Punkten unterschiedlich, was aus der Geschwindigkeiten zwischen Eckdaten der Geste hervorgerufen wird. Diese sollten jedoch äquidistant sein und dieselbe Länge haben, um sie vergleichen zu können. So werden die Sensordaten auf N Punkte (mit $N = 64$) projiziert.

Schritt 2: Einmaliges Drehen auf Basis des *indikativen Winkels* Der *indikative Winkel*, wie ihn Wobbrock, Wilson und Li ihn nennen, ist der Winkel zwischen der X-Achse und der Verbindungslinie zwischen dem Zentrum der Geste und dem Anfangspunkt. Dieser Winkel wird durch Drehen auf 0 gesetzt, wodurch die ursprüngliche Drehung der Geste keine Rolle mehr spielt.

[1] Formant: Ein Laut mit einem bestimmten Frequenzbereich. Formante machen unter anderem menschliche Stimmen charakteristisch und unterscheidbar.

Schritt 3: Skalierung und Verschiebung Die Geste wird nun nicht-uniform skaliert, um sie an ein Referenzquadrat anzupassen. Dadurch wird erreicht, dass auch die Größen vergleichbar bleiben. Anschließend wird die Geste an den Koordinatenursprung verschoben.

Schritt 4: Finde den optimalen Winkel für das beste Ergebnis Dabei wird die Geste mit jeder Vorlage verglichen und eine Punktezahl für die Ähnlichkeit ermittelt.

Durch diesen Basisalgorithmus können Erkennungsraten bis zu 98% erzielt werden bei gerade einmal 100 Codezeilen. Dieser Ansatz soll hier allerdings nur als Grundlage für andere Arbeiten vorangestellt werden.

3.6.5. Gestenerkennung mit Beschleunigungssensoren

Eine „Kategorisierung und Eigenschaften von auf Bewegungssensoren basierenden Benutzerschnittstellen“ stellen Mäntyjärvi et al. vor [MKKK04]. Diese befindet sich in Tabelle 3.1

Art der Schnittstelle	**Arbeitsprinzip**	**Anpassbarkeit**	**Komplexität**
1. Messung und Kontrolle	Direkte Messung von Kippen, Drehen und Amplitude	-	Sehr niedrig
2. Diskrete Gestensteuerung	Gestenerkennung	Maschinelles Lernen, frei anpassbar	Hoch
3. Kontinuierliche Gestensteuerung	Kontinuierliche Gestenerkennung	Maschinelles Lernen, frei anpassbar	Sehr hoch

Tabelle 3.1.: Kategorisierung und Eigenschaften auf Bewegungssensoren basierender Benutzerschnittstellen [MKKK04]

Demnach können auf Bewegungsgesten basierende Steuerungen grob in die Kategorien einfache Messungen mit Schwellwertbeobachtung und in tatsächliche Gestenerkennung unterteilt werden.

Während die meisten Arbeiten sich mit der Gestenerkennung beschäftigen und versuchen, möglichst hohe Erfolgsraten zu erzielen, hat das System *Cicero*, eine spätere Arbeit von Mäntyjärvi et al. zum Ziel, eine einfache und intuitive Steuerung für mobile Geräte zu implementieren [MPSS06]. Diese soll in Museen in einem elektronischen Museumsführer zum Einsatz kommen. Als Gerät wurde ein PDA verwendet, das zusätzlich mit einem Gyroskop ausgestattet wurde. Die einfache Steuerung, nur durch Kippen des Geräts durch das Menü zu navigieren und durch Scannen eines RFID-Tags an einem Exponat, Informationen zu diesem auf den Bildschirm zu bekommen, mag einfach erscheinen, soll jedoch ihren Zweck erfüllen. Zwar hatten drei Viertel der Probanden Probleme mit der ungewohnten Steuerung, bewerteten die meisten mit 4 bis 5 von 5 möglichen Punkten im Schnitt als klar verständlich.

Die Seite des Marmormuseums, an dem das System getestet wurde legt den Schluss nahe, dass das System weiterhin im Einsatz ist [Mar].

3.6.6. Einfache 3D Gestenerkennung

Kratz und Rohs greifen nun die Arbeit von Wobbrock, Wilson und Li auf [WWL07] und entwickeln ihn zu ihrem *3$ Gesture Recognizer* [KR10] weiter. Dieser einfache Ansatz erkennt Gesten ebenfalls nur mit Hilfe von trigonomischen und geometrischen Berechnungen. Ziel dieser Arbeit ist es, möglichst ohne besonders großen Aufwand eine Gestenerkennung in Anwendungen zu implementieren. Diese kann später durch eine bessere Erkennung ersetzt werden.

Um Gestendaten zu erhalten verwenden Kratz und Rohs einen WiiMote-Controller. Im Gegensatz zu den genauen Punkt-Koordinaten von Wobbrock, Wilson und Li [WWL07], erhalten Kratz und Rohs auf diese Weise Beschleunigungsvektoren in die jeweilige Richtung. Um diese Werte weiterzuverarbeiten, wird nur die Differenz eines Beschleunigungswerts zum vorhergehenden berechnet. Die Verarbeitung läuft ähnlich zu der von Wobbrock, Wilson und Li ab. Zunächst wird die Anzahl auf $N = 150$ Punkte skaliert und der Abstand zwischen den Messpunkten äquidistant verteilt. Anschließend wird die Geste um den indikativen Winkel gedreht, der ebenfalls definiert ist als die Verbindungslinie des ersten Punkts p_0 der Geste und dem Zentrum c. Im nächsten Schritt wird die Geste so skaliert, so dass sie in einen normalisierten Würfel von 100^3 Einheiten passt. In der Erkennungsphase wird die so vorbereitete Geste leicht gedreht und überprüft, bei welchem Winkel die Ähnlichkeit dieser Geste zu den Trainingsgesten am höchsten ist. Dabei wird zu jeder Referenz-Geste eine Punktezahl ermittelt und in eine Tabelle eingetragen. Ist die Punktezahl über einem Schwellwert ($> 1.1\epsilon$), wird der dazu gehörende Gestenname ausgegeben.

Mit diesem Verfahren lassen sich im Durchschnitt Erkennungsraten von 80% erzielen. Dabei reichen die Einzelwerte von 58% bis zu 98%. Dennoch ist die Erkennung stark von der Geste und dem Benutzer abhängig. Ein weiterer Punkt ist, dass die Erkennung sehr rechenintensiv ist, da die Geste mit allen Trainigsgesten verglichen werden müssen. Dadurch entsteht eine Laufzeit von $O(N * M)$, wobei N die Anzahl der Messwerte pro Geste und M die Anzahl der Gesten in der Trainigsbibliothek ist. Daraus folgt, dass dieser Ansatz nur für 10-15 Gesten praktikabel ist. Das alles führt dazu, dass ein solches System für die Praxis eher ungeeignet ist. Werden zu viele Gesten falsch oder gar nicht erkannt, so wird ein Benutzer das System nicht sehr lange verwenden wollen [KKM+06].

Von Katz und Rohs gibt es zudem eine Implementierung für Android-Smartphones. Allerdings hatte dieser Prototyp bei der Evaluation für diese Arbeit erhebliche Softwareprobleme, weswegen er nicht für die Erkennung verwendet wurde.

3.6.7. 3D Gestenerkennung mit Maschinenlernverfahren

In der Arbeit von Kela et al. [KKM+06] wurden zwei Benutzerstudien durchgeführt: Bei der ersten Studie sollten die Probanden verschiedene Gesten zur Steuerung unterschiedlicher

Geräte einstudieren (z.B. Videorekorder, Hi-Fi-Anlage, Fernseher, Licht ...). Dabei hatten die Probanden die volle Freiheit sich eine eine Geste auszudenken und zu verwenden, da herausgefunden werden sollte, welche Art von Gesten überhaupt in Frage kommen. Dabei stellte sich heraus, dass verschiedene Menschen oft für dieselbe Aufgabe ähnliche Gesten verwenden (z.B. Einschalten, bzw. Ausschalten des Videorekorders) und ein einzelner Proband meist dieselbe Geste für eine ähnliche Aufgabe bei einem anderen Gerät verwenden will.

Die Studie zeigte ebenfalls, dass Menschen Gesten meist im zwei- und nur selten im dreidimensionalen Raum ausführen, das heißt, eher eine auf-ab-, oder eine Wischbewegung anstatt einer komplexen Raumbewegung wie z.B. eine Spirale. 67% empfanden die Steuerung von Haushaltsgeräten durch Gesten als natürliche Art der Steuerung [KKM+06].

Für die zweite Benutzerstudie wurde ein Framework aus zwei Servern und der SoapBox realisiert (vergleiche 3.1 auf Seite 29). Der erste Server diente der Signalverarbeitung der mobilen Geräte (neben SoapBox u.a. auch PDA, Tablet PC, Mikrofon). Die vorverarbeiteten Daten wurden dann per TCP/IP an einen zweiten Server gesendet, der das Design-Programm steuerte und die Ausgabe auf einer Videowand realisierte. Dieser Aufbau ähnelt in etwa der Architektur der hier vorgestellten Arbeit, die einen Server für die Kommunikation und je ein Client-Programm für die Sensordaten und eines für die Animation verwendet.

In einer zweiten Benutzerstudie von Kela et al. [KKM+06] wurden verschiedene Aspekte eines Design-Programms durch unterschiedliche Interaktionsmöglichkeiten (Gesten- und Spracherkennung, Steuerung über eine Browser-Applikation auf einem PDA, Physical Tangible Objects (PTO) - einer RFID-Tag-Tracking Variante) ermöglicht. Die Benutzer hatten die freie Auswahl an Steuerungsmöglichkeiten und konnten sich eine Kombination davon verwenden. Es zeigte sich hierbei, dass Menschen, mit unterschiedlichen Schwerpunkten verschiedene Interaktionsmöglichkeiten nutzen. So wird die Spracherkennung zum Beispiel hauptsächlich für die Navigation in Menüs verwendet. PTO wurde in einer Art Präsentationsmodus für 3D-Objekte benutzt und am häufigsten wurde das PDA verwendet, da die Steuerung mit dem Browser scheinbar die meiste Funktionen bot. Die Steuerung über Gesten wurde dennoch verwendet, um sich durch den dreidimensionalen Raum zu bewegen. So konnte man durch Gesten die Szene drehen, herein- und herauszoomen und sich frei bewegen.

Das Problem bei diesen Arbeiten war, dass beide eine Geste „markieren“ mussten. Das heißt, der Benutzer muss jedes mal einen Knopf drücken, solange er eine Geste ausführen will. Das resultiert aus der Tatsache, dass eine Geste aus einer festen Form erkannt wird. Das heißt, bevor eine Geste erkannt wird, werden die Datenpunkte gesammelt, vorverarbeitet und erst dann durch unterschiedliche Verfahren erkannt. Wird hingegen ein kontinuierliches Verfahren angewandt, muss bei jedem erkannten Symbol eine neue Erkennung gestartet werden, was z.B. bei 100 Datenpunkten einen enormen Rechenaufwand bedeutet.

Eine einfache Möglichkeit, den Beginn einer Geste zu bestimmen, ist es, erst ab einem gewissen Schwellwert mit der Erkennung zu beginnen. Davon abgesehen, dass in dieser Arbeit eher ein Ausschlag verwendet wird als eine richtige Gestenerkennung, wird die Aufzeichnung der Bewegungsdaten erst dann begonnen, wenn die Werte den Grenzwert von

$\Delta = 10.0$ überschreiten, was bei einem Maximalausschlag von $+/-24.0$ etwa der Hälfte entspricht. Allerdings ist durch diese Technik die Trennung von Gesten in einer schnellen Abfolge nicht mehr einfach zu erkennen.

3.7. Kinowerbung in Kanada

Zum Schluss sei noch die kanadische Firma TimePlay Inc. erwähnt. Diese haben ein marktreifes, kommerzielles System entwickelt, das der Interaktion zwischen Kinobesuchern und der Leinwand dient [Vle11]. Ähnlich zu dem System dieser Arbeit stehen bei TimePlay Spiele im Vordergrund, die Werbezwecke dienen sollen. Nach Angaben des Herstellers werden mit diesem System erfolgreich Werbekampagnen in ganz Toronto und Vancouver, Kanada durchgeführt. Leider ist die Applikation für das Android-Smartphone nicht in Deutschland verfügbar, weswegen es auch nicht mit dem hier vorgestellten System verglichen werden konnte. Zudem hat der Hersteller keine Beschreibungen der Programme, oder die Ergebnisse aus der Praxis veröffentlicht. Dennoch scheint es, dass sich dieses System und jenes der Firma TimePlay in mindestens zwei Punkten unterscheiden, obwohl eine Ähnlichkeit durchaus vorhanden ist. Erstens werden in dieser Arbeit Beschleunigungsdaten verwendet, um mit dem System zu interagieren. Bei Timeplay scheint es, dass die Interaktion ausschließlich auf den Touchscreen begrenzt ist. Und zweitens erscheint es, als seien die Spiele auf zweidimensionalen Inhalt begrenzt. Dennoch hat sich die Firma das Patent für *Motion picture theater interactive gaming system* [RLC99] bereits 1999, obwohl das System erst 2011 zum ersten Mal in der Praxis getestet wurde [Vle11].

4. Verwendete Hardware

Diese Arbeit verwendet, vom heutigen Standpunkt gesehen, sehr moderne Technik. Die stereoskopische Bilderzeugung ist gerade dabei, sich im Privatbereich zu etablieren, hoch leistungsfähige Smartphones, wie das hier verwendete, sind erst seit wenigen Jahren auf dem Markt erhältlich und auch der HDMI-Standard, der die Übertragung von stereoskopischen Inhalten an einen Bildschirm oder einen Projektor überträgt, stammt ebenfalls aus dem Jahr 2011[1].

In diesem Kapitel wird die verwendete Hardware vorgestellt und es wird beschrieben, welchen Zweck diese während der Bearbeitung erfüllte. Diese Daten werden hier vorgestellt, um eine bessere Vergleichbarkeit zur zukünftigen Arbeiten und Studien zu gewährleisten und die in der Studie erhobenen Daten nachvollziehbar und reproduzierbar zu machen.

4.1. Smartphone: Samsung Galaxy Nexus

Als mobiles Gerät wurde ein Samsung Galaxy Nexus (GT-I9250) [Sam] gewählt (siehe Abbildung 4.1). Die Wahl fiel auf dieses Smartphone, da es sich hierbei um das Referenzgerät für die 2011 veröffentlichte Android Version 4.0 handelt. Da es sich um das offizielle Gooogle-Smartphone handelt, wurde es vom Hersteller Samsung in keiner Weise modifiziert (gebrandet) und erhält immer sehr früh die neuesten Betriebssystem-Updates und Patches.

4.1.1. Betriebssystem: Android 4.2.1

Die Client-Applikation für das Smartphone wurde für das Android Betriebssystem geschrieben. Android ist ein Betriebssystem für mobile Geräte und wurde von der Firma Google entwickelt. Auf dem hier verwendeten Mobiltelefon läuft derzeit die Android Version 4.2.1 (Jelly Bean). Das Client-Programm ist ab der Android Version 2.3.3 und aufwärts kompatibel, wodurch das System derzeit auf rund 89,4% aller Android Geräte lauffähig ist (siehe Abbildung 4.2) [And]. Der Grund dafür, erst ab der Version 2.3.3 kompatibel zu sein liegt in den Erleichterungen, die diese Version im Bezug auf das Auslesen der Sensordaten bringt. Es sprachen viele Gründe dafür, ein mit diesem Betriebssystem betriebenes Telefon zu verwenden:

[1] http://www.hdmi.org/press/press_release.aspx?prid=120

Abbildung 4.1.: Galaxy Nexus GT-I9250

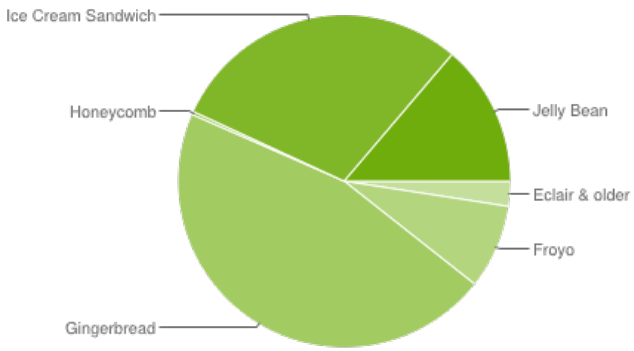

Abbildung 4.2.: Verteilung der Android Versionen am 4.2.2013 [And]

Android basiert auf Linux Android ist eine spezielle Linux Variante. Das bedeutet für die Entwickler unter anderem, dass der Quellcode des Betriebssystems, im Gegensatz zu Windows, frei und frei einsehbar ist. Dadurch ist es möglich, Modifikationen am System vorzunehmen.

Getrennte Java Virtual Machines Zwar ist der Kernel in C geschrieben, aber alle Applikationen werden jeweils in einer eigenen Java Virtual Machine ausgeführt. Das bedeutet, für die Implementierung der Applikationen wird reines Java verwendet. Dadurch lassen sich so gut wie alle Pakete und -klassen der Standard Java API auch für Android verwenden. Zwar bringt die Entwicklung von Android-„Apps"noch etwas Mehraufwand mit sich, z.B. bei der Entwicklung der Benutzeroberfläche oder dem Konzept der Activities, dennoch macht das verwendete Java die Entwicklung um einiges flexibler

und einfacher. Außerdem wird dadurch die Kompatibilität zwischen verschiedenen Smartphones über viele Android-Versionen sichergestellt. Dadurch dass die Applikationen in separaten virtuellen Maschinen laufen, ist es möglich, eine „App“ ohne weiteres zwangsweise zu beenden, ohne dass es negative Auswirkungen auf den Rest des Systems hat.

Verbreitungsgrad Inzwischen wurde die Anzahl der Geräte, auf denen das von Apple entwickelte Betriebssystem *iOS* läuft, von Android-Geräten auf den zweiten Platz verwiesen. Laut einer Studie von IDC [LRS12] besaßen im ersten Quartal 2012 bereits 59,0% aller neu verkaufen Smartphones das Android Betriebssystem.

4.1.2. Verwendung

Das Smartphone verfügt über eine kabellose Netzwerk-, eine Bluetooth- und eine 3G-Mobilfung-Verbindung. In den hier unternommenen Versuchen wurde allerdings nur das Wireless LAN verwendet. Die Lagesensoren werden nur dazu verwendet, um intern die auf das Mobilfunkgerät einwirkende Erdanziehungskraft herausrechnen zu können. Die Beschleunigungssensoren selbst werden für die Gestenerkennung ausgelesen und die Kamera dient dem Scannen von QR-Codes, welche eine einfache Verbindung zum Server ermöglichen. Insgesamt wurde das Galaxy Nexus als Referenzsystem und Entwicklungsgerät genutzt.

4.2. Server

Der Server wird derzeit in einer virtuellen Maschine auf einem EQ4-Server der Firma Hetzner (für technische Details siehe A.3 auf Seite 96) ausgeführt.

Dieser Server ist physikalisch nicht erreichbar, da es sich hierbei um einen im Internet gehosteten Rootserver handelt. Allerdings verfügt er mit einer 100 MBit/s Standleitung über eine sehr schnelle Verbindung zum Internet, wodurch sehr geringe Latenzzeiten erreicht werden können.

Der Server ist so konfiguriert, dass jedes Mal, wenn eine neue Programm-Version in das Versionierungsssystem (SVN) eingecheckt wird, die Applikation des mobilen Clients an die entsprechende Stelle kopiert wird, so dass sie über die URL `http://www.kaffeezombie.net/kasia/Kasia3D.apk` heruntergeladen werden und installiert werden kann. Das Serverprogramm wird ebenfalls bei diesem Vorgang so weit vorbereitet, so dass es nur neu gestartet werden muss. Sonst müssten die Dateien jedes Mal manuell an den richtigen Platz kopiert, diese dann kompiliert und der Server neu gestartet werden.

4.3. Animations-Client

Der Animations-Client wurde auf verschiedenen Computern entwickelt und getestet. Diese sollen hier nun vorgestellt werden. Der Grund hierfür war, dass dadurch die ebenfalls die Performanz der unterschiedlichen Hardware getestet werden konnte.

4.3.1. Laptop: ThinkPad T520

Auf diesem Laptop wurde die meiste Zeit getestet und der gesamte Quellcode und die Ausarbeitung geschrieben. Er wurde ebenfalls über lange Zeit als Referenzmaschine verwendet, da er über einen schnellen Prozessor, eine geeignete Grafikkarte, ausreichend Speicher und ein Betriebssystem verfügte, welches die Grafikhardware und deren Treiber entsprechend ansteuern kann. Die Hardwaredetails befinden sich in Abschnitt A.2 auf Seite 95.

Für das Ansteuern des Displays wird ein Adapter von Displayport auf HDMI benötigt. Für das Anzeigen der stereoskopischen Bilder auf dem Beamer oder auf dem Display wird ein spezieller Treiber vorausgesetzt (siehe Abschnitt 5.4 Konfiguration eines Computers).

4.3.2. Desktop-PC: HP Compaq 8200 Elite CMT PC

Dies ist der Computer, an dem das System hauptsächlich auf dem 3D-Display (vergleiche Abschnitt 4.4.1 auf der nächsten Seite) getestet wurde. Für die Spezifikationen der Hardware siehe Abschnitt A.4 auf Seite 97. Der Computer befindet sich im Labor und ist dort direkt mit dem 3D-Display verbunden. Er verfügt über einen direkten Netzwerkzugang und kann mit seiner Grafikkarte stereoskopische Bilder erzeugen. Da diese Maschine zwischenzeitlich nicht zur Verfügung stand, wurde auf einem nahezu baugleichen Computer versucht, das 3D-Display anzusteuern, der allerdings nur über eine NVidia NVS 290 verfügte. Dies schlug allerdings fehl, weshalb wieder auf diesen Computer zurückgegriffen wurde. Wie beim ThinkPad (siehe oben) wird für das Verwenden der stereoskopischen Displays ebenfalls ein Spezialtreiber benötigt.

4.3.3. Laptop: Sony Vaio VPCF22C5E

Für die Benutzerstudie wurde ein Sony Vaio des Instituts verwendet. Dieses verfügt, im Gegensatz zur restlichen Computer-Hardware über eine schnelle GeForce Grafikkarte, einen starken Prozessor und 8 GB Arbeitsspeicher. Die Hardwarespezifikation befindet sich ebenfalls im Anhang (siehe Abschnitt A.5 auf Seite 97).

Abbildung 4.3.: Philips 3D Fernseher

4.4. Display

Neben den Computern, in denen die Grafikkarten und die Netzwerkanschlüsse integriert sind, gibt es eben noch eine Reihe von Ausgabegeräten, Displays und Projektoren, auf denen die stereoskopische Projektion dargestellt wird.

4.4.1. Philips 3D Fernseher: 55PFL7606H

Dieser Fernseher diente während der meisten Zeit als Referenz-Display für die stereoskopische Anzeige der Animation. Auch dieses Gerät befindet sich im Labor des Instituts (vergleiche Abbildung 4.3). Bilder werden über ein HDMI-1.4a-Kabel übertragen und können dort mit guter Qualität dargestellt werden. Der HDMI-Standard erlaubt allerdings nur zwei Modi für die Übertragung von 3D-Bildern: Entweder bei einer 1920x1080 Pixeln (HD 1080p) und einer Bildfrequenz von 24 Hz, oder bei einer Auflösung 1280x720 Pixeln (HD 720p) und einer Frequenz von 60 Hz. Durch die hohe Auflösung wird zwar das Bild detailliert, aber durch die geringe Frequenz erscheint das Bild nicht flüssig zu sein. Daher wurde in den Versuchen die geringere Auflösung verwendet, da hier durch die höhere Frequenz eine angenehmeres Bild zustande kommt.

Um die Halbbilder zu trennen verwendet der Fernseher eine zeilenweise Polarisationstechnik und dementsprechend passive Polarisationsbrillen. Zudem verwendet der Fernseher das Philipps Ambilight, eine Lichtprojektion zur Wand hin, die den Bildausschnitt scheinbar vergrößern soll. Genaue technische Details befinden sich in Abschnitt A.6 auf Seite 98.

Abbildung 4.4.: Epson 3D Beamer. Quelle: www.epson.de

4.4.2. Epson EH TW-6000Beamer

Für die Benutzerstudie wurde der institutseigene Epson 3D-Beamer verwendet (vergleiche Abbildung 4.4). Dieser befindet sich dort im Labor und fest an das Deckengestänge montiert. Er verfügt über eine Full HD (1080p) Auflösung. Angesteuert wird er über HDMI. Der Beamer nutzt Shutterbrillen (siehe Abbildung 2.8 auf Seite 26, um die Halbbilder zu trennen. Für die Benutzerstudien mussten allerdings zuerst Brillen geordert werden, da die vorhandenen Exemplare nicht für die Probandenzahl ausgereicht hätte.

5. Implementierung

In diesem Kapitel wird die Implementierung des Kasia-Systems beschrieben. Dabei wird zunächst in Abschnitt 5.1 Schritt für Schritt erklärt, wozu das System überhaupt dient. In Abschnitt 5.2 auf der nächsten Seite wird dann darauf eingegangen, aus welchen Gründen das System in Java implementiert wurde. In Abschnitt 5.3 auf Seite 49 werden die Bestandteile von Java3D beschrieben, und wie die einzelnen Teile zusammenspielen, damit eine stereoskopische Animation erstellt wird. In einem separaten Abschnitt 5.4 wird beschrieben, wie die jeweiligen Computer konfiguriert werden müssen, um das System ausführen zu können. In Abschnitt 5.5 auf Seite 55 wird dann auf den Aufbau und das Zusammenspiel der einzelnen Teile eingegangen. Hier wird ebenfalls die Paket-Hierarchie erklärt. Anschließend werden die einzelnen Untersysteme erklärt: In Abschnitt 5.6 der Server, in 5.7 der Smartphone Client und in 5.8 das Animationsprogramm. Zum Ende wird in Abschnitt 5.10 auf Seite 71 auf die Erstellung und Verarbeitung des verwendeten QR-Codes eingegangen und in 5.11 der Datenaustausch zwischen den einzelnen Systemen erklärt.

5.1. Anforderungen an das System

Das Kasia-System bietet die Möglichkeit, mit einer stereoskopischen Animation zu interagieren. Dafür wurde in den Smartphone-Client zwei unterschiedliche Gesten integriert: Eine Hand-Geste, die durch Auslesen der internen Beschleunigungssensoren erkannt wird, und eine „Touch"-Geste, für die lediglich ein Knopf gedrückt werden muss. Diese Gesten dienen dem Fangen von fliegenden Objekten, die in der Animation dargestellt werden. Die Animationsseite muss eine stereoskopische Szene anzeigen, in die fliegende Objekte geladen, animiert und verändert werden können. Außerdem muss die Animation feststellen können, ob eine Geste, die vom Smartphone-Client gesendet wurde, auch ein fliegendes Objekt gefangen hat. Für die Auswertung der Benutzerstudie muss die Animation zudem über eine Schnittstelle zu einer Datenbank verfügen. Damit eine Kommunikation zwischen Smartphone und Animation zustande kommen kann wird ein Server benötigt, der die Anfragen der Smartphone-Clients bündelt und so Last von der Animation nimmt.

5.1.1. Szenario: Jasmin und Holger im Kino

An einem schönen Abend beschließen Holger und Jasmin in ihr Stammkino zu gehen. Das letzte Mal ist schon lange her und daher freuen sie sich auf den lange ersehnten 3D-Film. Zudem ist das Kino mit diesem neuartigen Kasia-System ausgerüstet, das sie schon lange

mal testen wollten. Dafür brauchen sie eine „App“ für ihr Handy. Holger hat schon auf der Kinohomepage den Client heruntergeladen und installiert. Jasmin muss das noch tun. Durch den QR-Code auf der Eintrittskarte kann sie das Programm direkt aus dem Kinonetzwerk herunterladen und installieren.

Nach den ersten 5 Minuten Werbung werden dann die Besucher noch einmal darauf hingewiesen, dass es gleich etwas zu gewinnen gibt und dass die Besucher noch einmal die Möglichkeit haben, das Programm zu installieren, bevor es in wenigen Minuten richtig los geht.

Nach einigen Werbespots ist auf der Leinwand der große QR-Code zu sehen, und eine Stimme fordert die Teilnehmer auf, die 3D-Brillen aufzusetzen und sich über das Scannen des Codes mit Hilfe des in der „App“ integrierten QR-Lesers, mit dem System zu verbinden. Die letzten Kurzentschlossenen haben nun noch einmal die Chance das Programm herunterzuladen. Der QR-Code für die Verbindung ist derselbe wie zur Installation des Programms.

Doch schon geht es los: Wie aus dem Nichts tauchen „Fliegende Objekte“ auf und scheinen auf die Besucher zuzukommen: Da kommt eine Popcorntüte, hier ein Eis der Firma *I Scream*, dort ein Schokoriegel von *Hard Candy*, da kommt eine Dose *Black Soul*-Cola und dort eine Tüte *Knoosprig*-Chips alles in sehr schneller Folge. Kurz bevor die Objekte links und rechts an den Zuschauer vorbeifliegen, leuchten sie noch einmal auf. Das muss der Zeitpunkt sein, an dem sie gefangen werden können. Wie der Großteil des Publikums auch sitzen Jasmin und Holger in ihren Sesseln und schwenken ihre Smartphones. Durch ihre Bewegungen können sie dann die Objekte „fangen“ das heißt, die Objekte tragen auf der Leinwand den Namen der Fangenden und die Objekte landen als kleine Grafik auf dem Handy. Natürlich achten beide darauf, nicht alles zu fangen, denn schließlich wollen sie ja nachher einen speziellen Snack gewinnen. Nach fünf Minuten Spielen ist alles vorbei.

Holger hat dieses Mal Pech, aber Jasmin schaut strahlend auf das Gerät und winkt dem Mitarbeiter zu, der ihr ihren Gewinn, eine Tüte *Knoosprig*-Chips übergeben wird. Zumindest bis jetzt hat sich der Kinobesuch in jedem Fall gelohnt.

Nach dem Film gehen die beiden gut gelaunt nach Hause: Zuerst beim Spielen gewonnen und der Film hat auch gehalten was er versprochen hat.

5.2. Warum Java und Java3D für eine stereoskopische Anwendung?

Das Kasia-System ist als Client-Server-Architektur implementiert. Grundlage hierbei die Programmiersprache Java für alle Bestandteile. Zwar haben sich in der Vergangenheit Implementierungen von dreidimensionalen Animationen in den Sprachen C oder C++ etabliert, aber man hat sich bewusst dagegen entschieden. Für die Wahl Java gibt es mehrere Gründe:

Plattformunabhängig Java gilt als sehr Plattformunabhängig. Da das Server-Programm auf einem reinen Linux-Server läuft, die Animation auf Windows-Treiber zugreift und die Smartphone-Applikation auf einem Android-Betriebssystem ausgeführt wird, war es naheliegend, alle Teilprogramme in der selben Sprache zu schreiben, um einen möglichst einfachen Datenaustausch zu ermöglichen.

Volle Unterstützung von Android Googles Betriebssystem Android hat im Kern getrennte Java-Virtual-Machines, genannt Dalvik[1]. Diese werden mit einem eigenen Dialekt von Java programmiert, der zu nahezu 100% kompatibel mit reinem Java ist. Daher können auch die meisten Java-Bibliotheken für die Netzwerkunterstützung eins zu eins übernommen werden.

Höhere Abstraktion Im Falle von Java3D, welches für die Animation verwendet wurde, gibt es bereits eine höhere Abstraktion, als reine Grafik-Programmierung wie beispielsweise OpenGL (siehe vrefsec:OpenGL und DirectX), welches hauptsächlich in über die Programmiersprachen C und C++ angesprochen werden, oder die Java Bindings für OpenGL (JOGL) (siehe ebenfalls 5.3.3), die ebenfalls sehr hardwarenah sind. So kann man mit bereits bestehenden Datenstrukturen arbeiten und muss nicht jede Einzelheit selbst implementieren.

Integration externer 3D-Modelle Java3D bietet von sich aus die Möglichkeit an, 3D-Objekte, die im WaveFront-Format vorliegen, direkt in den Szenegraph zu importieren. Dadurch konnten bereits vorhandene oder eigens für dieses System geschaffene Modelle direkt verwendet werden. Hierfür müssen die Objekte nur aus dem Modellierungsprogramm Blender in das Wavefront-Format exportiert werden.

5.3. Java3D

Java3D wurde seit 1997 von der der Firma Sun Microsystems entwickelt, bevor das Framework 2004 unter die freie BSD-Lizenz gestellt wurde. Java3D kapselt sowohl plattformunabhängige OpenGL-Bibliothek, als auch das von Microsoft entwickelte DirectX, welche beide direkt auf die Grafikhardware zugreifen. Im Fall von OpenGL verwendet Java3D die Bindings von JOGL, welches einen Wrapper zur Sprache C darstellt.

Für diese Arbeit wurde Java3D verwendet, da dieses Framework, genau wie der Rest des Systems, in Java implementiert ist, eine native Möglichkeit hat, bestehende 3D-Modelle im Wavefront-Format[2] zu importieren und außerdem den NVIDIA-Stereotreiber anzusprechen.

[1] http://developer.android.com/guide/appendix/glossary.html
[2] http://www.martinreddy.net/gfx/3d/OBJ.spec

5.3.1. Der Szenegraph

Eine Szene in Java3D wird durch einen einzigen Szenegraph (Universe) dargestellt. Dabei werden in den Wurzelknoten, der Canvas3D, alle anderen Elemente eingehängt und animiert. Mehrere Einzelelemente (Shapes) werden in einer Gruppe (Branchgroup) zusammengefasst, die selbst auch Untergruppen enthalten können. Gruppen werden im Raum durch Tansformationen positioniert und so auch animiert.

Universe Das „Universum“ entspricht dem Raum, in dem sich sowohl Benutzer, als auch die Animation befinden. Hier werden die Dimensionen des Raums (s. View), die Position des Benutzers (s. PhysicalBody) und die Anforderungen an den Raum definiert, wie beispielsweise, ob die Szene stereoskopisch sein soll und von welcher Natur das Display ist.

View Die View beschreibt praktisch den Ausschnitt in die virtuelle Welt. Eine View beschreibt den Abstand der Projektionsebene, die im Prinzip die Bildschirmoberfläche darstellt, die Ebene, in der sich der Benutzer bewegt mit einer negativen Parallaxe und der Bereich „hinter“ dem Bildschirm mit positiver Parallaxe.

Canvas3D Die Canvas entspricht der digitalen „Leinwand“ auf der die Szene abgebildet wird. Sie hängt direkt am Universum und gibt an, wie eine Szene dargestellt wird. Sie ist auch dafür verantwortlich, Objekte stereoskopisch darzustellen, da sie auch als Vermittler zum Gerät für die Grafikausgabe dient.

BranchGroup Eine BranchGroup dient als einfaches Gruppierungselement. Hier können alle Knoten, wie Shapes, weitere BranchGroups, Beleuchtungselemente oder Transformgroups zusammengefasst werden. Das hat den Vorteil, dass Änderungen für einen ganzen Zweig gemacht werden können und nicht jedes für einzelne Element.

TransformGroup In einer TransformGroup wird ein Element durch eine Transform3D verändert. Dabei kann immer nur ein einziges Unterelement modifiziert werden.

Transform3D Dieses Element dient der Veränderung von anderen Knoten. So können durch Translation Elemente positioniert, die Größe und Proportionen verändert und die Elemente um beliebige Achsen gedreht werden. Mit Hilfe der Transformationen lassen sich auch Animationen umsetzen: Hierfür werden Transformationen entlang eines vordefinierten Wegs angewendet. Zudem können Transformationsmatrizen angegeben werden, durch die beliebige Transformationen ermöglicht werden.

Shape3D Diese Form entspricht einem dreidimensionalen Objekt oder Teilobjekt. Jede Shape3D-Element besteht aus Geometrien, aus denen die Form aufgebaut ist und aus einer Appearance, die das Aussehen definiert.

Background Das Background-Element entspricht dem Hintergrund der Szene und wird dem Wurzel-BranchGroup-Element übergeben, das den Rest des Szenegraphen enthält. Die Größe des Hintergrunds wird durch den Durchmesser einer Kugel (BoundingSphere) definiert. Objekte, die außerhalb dieser Zone positioniert werden, befinden sich außerhalb der sichtbaren Szene.

Vector3/4d/f, Point3/4d/f, Color3/4d/f Diese Elemente (Vector3d, Vecor3f, usw.), erben alle vom Datentyp Tupel3/4d/f. Dabei steht die Zahl, anders als beispielsweise bei Shape3D, für die Dimension des Elements und die Buchstaben f für den Float-Datentyp, beziehungsweise d für Double. Vector und Point dienen der Positionierung im Raum, Color der Farbdefinition. Vierelementige Farben haben zusätzlich zu den Anteilen von Rot, Grün und Blau auch einen Alpha-Wert, der die Transparenz dieser Farbe regelt.

Appearance Die Appearance beschreibt das Aussehen eines Shape3D-Elements. Dadurch kann die Materialeigenschaft, die Textur, die Rahmendarstellung und die Transparenz eingestellt werden.

Material Das Material setzt sich aus vier Farbwerten zusammen, die die Materialeigenschaften repräsentieren:

Ambient Dieser Farbanteil gibt an, wie viel ambientes Licht der Umgebung von diesem Objekt reflektiert wird.

Diffuse Die hier definierte Farbe wird von dem Objekt diffus reflektiert, wenn eine Lichtquelle darauf leuchtet. Im Prinzip handelt es sich hierbei um die Materialfarbe an sich.

Specular Falls Licht auf eine spiegelnde Oberfläche trifft und man die Lichtquelle quasi auf der Stelle des Objekts erkennen kann, gibt dieser Farbanteil an, in welcher Farbe die Reflektion aufleuchtet.

Emissive Diese Farbe gibt an, in welcher Farbe das Objekt von innen heraus leuchtet. Beispiele sind Schirme von Schirmlampen oder eine glühende Herdplatte. In diesem System wird dieser Farbanteil verwendet, um ein Glühen in der Fangzone zu erzeugen.

Zudem kommt ein *Shininess*-Wert hinzu, der den Anteil an Reflexion angibt (Werte zwischen 1.0 und 128.0).

Texture Bei der Textur handelt es sich um die Grafik, die über das Objekt gezogen werden kann und ihm ein charakteristisches Aussehen gibt. Zum Beispiel das Logo des Getränkeherstellers auf einer Dose, oder die Maserung von Holz. Texturen haben viele Optionen, die in diesem System nicht nur durch Laden von externen Modellen verwendet werden. So können die Eckpunkte des Objekts angegeben werden, die mit Punkten der Grafik übereinstimmen müssen, wenn die Grafik über das Objekt gelegt wird. Es können auch dreidimensionale Texturen aufgebracht werden, also definiert werden, ob die Textur selbst eine Oberfläche hat. Was allerdings hier verwendet wird sind die Optionen, die einem die, kann zusätzlich angegeben werden, wie die oben genannten Materialwerte mit der Textur kombiniert werden. Hier gibt es drei Möglichkeiten: Nur die Textur wird verwendet und die Materialeigenschaften werden nicht in Betracht gezogen, das Material scheint an den schwarzen Stellen der Textur hindurch, oder die Farben des Materials werden auf die der Textur aufmoduliert.

TransparencyAttributes Diese Attribute geben an, in welchem Maße das Objekt transparent ist und welches Verfahren für die Transparenz verwendet wird.

PolygonalAttributes Hier hat man die Möglichkeit, das Objekt entweder normal anzuzeigen, die Linien, oder nur die Eckpunkte der Vertices anzuzeigen.

Beleuchtung Es gibt drei verschiedene Arten der Beleuchtung in Java3D: Das ambiente Licht kommt aus dem Nichts und beleuchtet die Szene gleichmäßig. Das direktionale Licht kommt aus einer Richtung parallel, ohne auf einen einzigen Punkt zurückzugehen. Die punktförmige Lichtquelle hat einen definierten Punkt im Raum, einen Öffnungswinkel und eine Scheinrichtung. Allerdings sieht man die Lichtquelle in der Szene nicht. Will man eine punktförmige Lichtquelle tatsächlich in der Szene sichtbar machen, muss man einen Scheinwerfer als Objekt kreieren, das Leuchtmittel als Kugel mit einem hohen Eigenleuchten definieren und die Lichtquelle in die Mitte der Konstruktion setzen. Auch Licht kann einfach in den Szenegraph eingefügt werden.

5.3.2. Stereoskopie

Java3D bringt von Haus aus die Möglichkeit mit, stereoskopische Bilder zu Erzeugen. Dafür muss der Betrachter in der Szene genau definiert werden: Der Augenabstand, der Abstand des Betrachters zur Leinwand und den Schalter, der Referenzpunkt für das Einäugige Sehen. Und es muss der Schalter umgelegt werden, damit die Szene auch stereoskopisch gerendert wird.

Gelöst wird das in Java3D mit Konfigurationsdateien, die einfach bei der Erstellung des Universums übergeben wird und alle dort definierten Einstellungen umsetzt. In dieser Datei wird auch der Ausgabebildschirm angegeben, auf dem die Animation ablaufen soll. Das Problem ist, dass man an den Einstellungen im Nachhinein nichts mehr verändern kann, wie z.B. die Ausgabebildschirm wählen, weswegen dieses Konzept etwas starr ist. Allerdings wurde keine andere Möglichkeit gefunden, Stereoskopie zu erzeugen ohne diese Konfigurationsdateien zu verwenden.

5.3.3. Schnittstellen zu OpenGL und zu DirectX

OpenGL und Direct3D sind die beiden führenden Sprachen für hardwareunterstützte Grafikprogrammierung. So ist es möglich, Befehle direkt auf der Grafikkarte auszuführen. So können alle Funktionen, die die Grafikkarte bereitstellt direkt angesprochen werden und da moderne Grafikkarten über leistungsfähige Prozessoren verfügen, ist die Ausführung auch sehr schnell.

OpenGL [Ope] (Open Graphics Library) ist eine offene, plattformübergreifende Bibliothek. Es gibt Schnittstellen zu vielen anderen Programmiersprachen. Allerdings werden diese meist über Wrapper angesprochen, die auf die C-Schnittstelle zugreift. So ist es auch im Fall von JOGL [Jog] (Java Bindings for OpenGL). JOGL ist ein Wrapper zwischen eben diesen C-Methoden und dem Äquivalent in Java. So ist es möglich, mit etwa derselben Syntax in

Java dieselben Ergebnisse wie in C zu erreichen. Allerdings hat die getestete Hardware große Probleme mit der stereoskopischen Darstellung, was von der Sprache durchaus gewährleistet wird.

DirectX [Mic] ist, im Gegensatz zu OpenGL, eine proprietäre Sprache der Firma Microsoft. DirectX wurde speziell für Computer mit dem Windows-Betriebssystem, bzw. für die Spielekonsole XBox [Xbo13] entwickelt. Auch DirectX spricht die Grafikhardware direkt an. Der Nachteil von DirectX ist, dass es eben nur kompatibel zu Microsoft-Prdoukten ist. Will man die Animation auf einem Linux-Computer ausführen, benötigt man entweder einen Wrapper wie WINE[3], der für die Kompatibilität zu Windows-Programmen sorgt, oder man muss die OpenGL-Variante verwenden.

Java3D unterstützt beide Techniken, ohne großen Aufwand zu betreiben: Dem Programmaufruf wird lediglich ein Kommandozeilenargument übergeben. Im Fall der Animation hat sich jedoch gezeigt, dass die verwendete Hardware (siehe Kapitel 4 auf Seite 41) große Probleme hat, Stereoskopie zu erzeugen, wenn OpenGL verwendet wird. Die Erfahrung zeigt, dass die Grafikkarte sehr viel besser von DirectX als von OpenGL angesprochen wird. Ein Grund hierfür dürfte sein, dass DirectX als Grundlage für viele 3D-Computerspiele dient. Daher wird eine Unterstützung gerade der Stereoskopie auch von der Spiele-Wirtschaft forciert.

5.4. Konfiguration eines Computers

Um die Animation auf einem Computer zu verwenden, benötigt man einige Unterprogramme und Treiber, die es überhaupt ermöglichen, dreidimensionale Bilder zu erzeugen. Nachfolgend werden alle benötigten Untersysteme beschrieben, die für die Ausführung der Animation benötigt werden:

Windows 7 oder höher Im getesteten Fall wurden die Schnittstellen zu DirectX verwendet. Für die Ausführung bietet sich ein modernes Windows-Betriebssystem an.

Java Für die Ausführung des Programms wird ein Java-Laufzeitsystem benötigt (Java Runtime Environment - JRE). Dieses erhält man auf der Seite des Herstellers[4]. Verwendet wird hier die aktuelle 64-Bit-Version 1.7.0-01.

Java3D Diese Bibliothek wird verwendet, um eine dreidimensionale Animation darzustellen[5]. In diesem Fall wurde die 64-Bit-Version 1.5.2 verwendet. Die 64-Bit-Version ist zur entsprechenden 64-Bit-Version von Java kompatibel.

OpenGL Java3D basiert zum Großteil auf OpenGL. Allerdings werden die nötigen Bibliotheken von Java3D ebenfalls mitinstalliert.

[3] http://www.winehq.org/
[4] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[5] http://java3d.java.net/binary-builds.html

DirectX Für die Animation wird DirectX verwendet. Bei aktuellen Windows-Versionen (z.B. Windows 7) ist bereits eine DirectX-Version installiert. Doch sollten nach der Installation der neuesten NVidia-Treiber stets darauf geachtet werden, dass ebenfalls die aktuellste DirectX-Version installiert ist. Treten Probleme auf kann die aktuelle Version auch vom Hersteller[6] bezogen werden.

NVidia Treiber Für die Animation sollten die neuesten Grafiktreiber installiert sein. Wie sich gezeigt hat können auch innerhalb der Bearbeitungszeit dieser Arbeit durchaus mit einem Update Probleme gelöst werden. Die aktuellste Version erhält man ebenfalls über die Homepage des Herstellers [7] bezogen werden.

NVidia 3DTV Play und TriDef Moderne Grafikkarten und -Treiber bringen alle Funktionalitäten mit sich, um stereoskopische Bilder auch auf 3D-Bildschirmen, -Beamern und -Fernsehgeräten anzuzeigen. Stereoskopie über HDMI ist inzwischen ein allgemeiner Standard, den die meisten modernen Grafikkarten auch umsetzen. Allerdings werden diese Funktionen nur durch einen Spezialtreiber freigeschaltet. Das ist im Falle der hier vorgestellten Animation die einzige Möglichkeit, dreidimensionale Bilder auf einem externen Fernseher darzustellen. Neben dem hier verwendeten NVidia 3DTV Play[8], welches gegen Entgeld bezogen werden kann, gibt es für Grafikkarten der Firma AMD [9] den Treiberhersteller TriDef [10]. Auch dieser Treiber sorgt dafür, dass dreidimensionale Objekte stereoskopisch ausgegeben werden und auch dieser ist nur käuflich zu erwerben. Da allerdings ausschließlich auf NVidia-Karten getestet und evaluiert wurde, kann nicht mit Sicherheit gesagt werden, ob das Programm auch über eine AMD-Grafikkarte korrekt dargestellt wird.

j3dcore-ogl.dll Diese Datei befindet sich bei der Installation von Java3D direkt im Verzeichnis `\bin` des Installationsverzeichnisses. Der Einfachheit halber sollte diese Datei ebenfalls in das Windows-Verzeichnis kopiert werden.

j3dcore-d3d.dll Diese Datei kann in der 32-Bit-Version direkt über die Java3D-Seite erhalten werden (siehe Java3D oben). Für die 64-Bit-Version dieser Bibliothek muss man entweder diese selbst aus dem Quellcode generieren oder man ist auf die Hilfe der Internetgemeinde angewiesen. Zum diesem Zeitpunkt ist der Link, über den die passende Datei heruntergeladen wurde bereits veraltet. Lässt sich diese Datei nirgends finden muss das System (Java und Java3D) komplett auf die 32-Bit-Version umgestellt werden. In jedem Fall muss die Datei ebenfalls in das Windows-Verzeichnis kopiert werden.

Für das Verwenden von Java3D in anderen Programmen und im Webbrowser sollten nach der Installation von Java3D noch folgende Systemvariablen angepasst werden. Für die reine Verwendung des Programms, sollten die oben genannten Schritte ausreichen.

[6] http://www.microsoft.com/de-de/download/details.aspx?id=35
[7] http://www.nvidia.de/Download/index.aspx
[8] http://www.nvidia.de/object/3dtv-play-buy-de.html
[9] http://www.amd.com/de/pages/amdhomepage.aspx
[10] http://www.tridef.com/

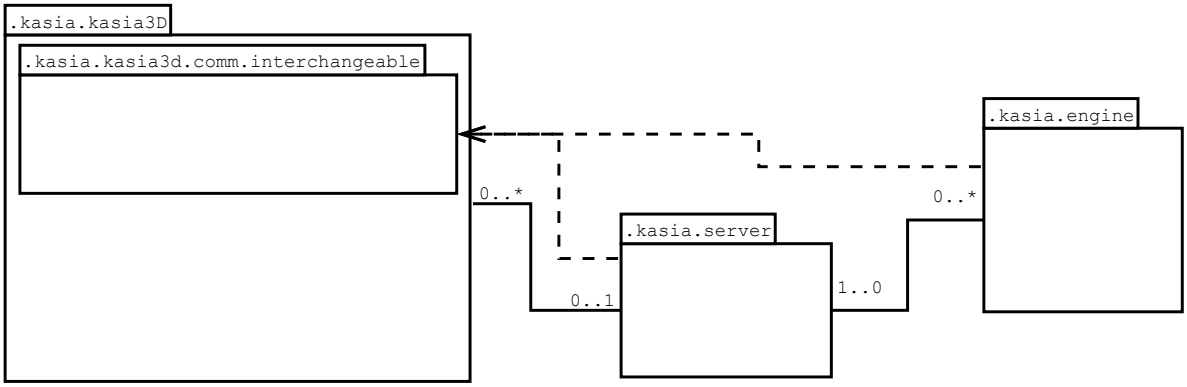

Abbildung 5.1.: KASIA: Übersicht der Pakete

Sei `C:\Program Files\Java\Java3D\1.5.2\` der Pfad zum installierten Java3D-System und `C:\Program Files\Java\jre7` der Pfad der installierten Java Laufzeitumgebung:

%CLASSPATH% Keine Änderung notwendig.

%JAVA_HOME% `C:\Program Files\Java\Java3D\1.5.2\bin`

%LD_LIBRARY_PATH% `C:\Program Files\Java\jre7\lib; C:\Program Files\Java\Java3D\1.5.2\lib\ext\`

5.5. Architektur

In diesem Abschnitt wird die Architektur des KASIA-System beschrieben.

Als Implementierungspattern wurde eine normale Client-Server-Architektur gewählt. Diese besteht aus einem zentralen Server (Kasia3D Server), der über eine Internet-, bzw. Netzwerkverbindung verfügt, und aus mehreren Clients. Als Client werden in diesem Zusammenhang sowohl die Applikation auf dem Smartphone (Kasia3D (Smartphone-Client)), als auch die Animation (Kasia3D Engine (Animationsclient)) bezeichnet (vergleiche Abbildung 5.1).

Der Server steht dabei als Bindeglied zwischen den einzelnen Clients und hat die einzige Aufgabe, Pakete von einem Client an den anderen zu senden. Dafür hält er eine Routingtabelle bereit und verwaltet jeden verbundenen Client in einem eigenen Thread. Sendet ein Client eine Nachricht an einen anderen, wird in der Routingtabelle nach dem Eintrag des Ziel-Clients gesucht und die Nachricht einfach an den Entsprechenden Socket des Ziels gesendet.

Hauptelement ist jedoch der Animations-Client. Dieser lädt Models als „Fliegende Objekte“ in die Szene, kann ihr Aussehen und ihre Größe nachträglich noch verändern und natürlich die komplette Animation steuern. Außerdem erkennt der Animations-Client selbst, ob durch eine Geste tatsächlich ein Objekt gefangen wurde und teilt dem zur Geste gehörenden Smartphone-Client dies im positiven Fall mit.

Der Smartphone-Client hat die Aufgabe, Gesten als solche zu erkennen und diese dann über den Server an den Animations-Client zu senden. Der Smartphone-Client dient auch der Interaktion mit dem Benutzer. So gibt er auch ein Akustisches Signal aus, wenn eine Geste erkannt, ebenso wenn ein Objekt gefangen wurde oder der Benutzer am Ende gewonnen hat.

Der Datenaustausch wird über verbindungsorientierte TCP/IP-Kanäle realisiert. Die Daten werden dabei als serialisierte Objekte ausgetauscht, was ein Standardverfahren in Java ist.

5.5.1. Namensraum

Für die Implementierung war sehr früh klar, dass alle Teilprogramme in derselben Programmiersprache (Java) geschrieben werden, weswegen alle Teilsysteme parallel in einem Projekt erstellt wurden. Sie befinden sich daher im selben Namensraum und daher auch in der selben Pakethierarchie. Dabei liegt das Standard Namensschema von Java zugrunde, wonach zuerst die URL des Herausgebers invers am Anfang steht. Das bedeutet, dass die zuerst die Topleveldomain, dann die Domain und anschließend der eigentliche Paketname den Namensraum definieren. In diesem wird aus der URL `http://kaffeezombie.net` der Paketname `net.kaffeezombie.kasia`. Dementsprechend sind auch die Namen für die Teilprojekte:

net.kaffeezombie.kasia.kasia3d enthält den Smartphone Client Kasia3D (siehe Abschnitt 5.7).

net.kaffeezombie.kasia.kasia3d.interchangeable enthält Klassen für den Datenaustausch (siehe Abschnitt 5.11).

net.kaffeezombie.kasia.engine enthält die stereoskopische Animation Kasia3D*Engine* (siehe Abschnitt 5.8).

net.kaffeezombie.kasia.server enthält die Server-Applikation Kasia3D*Server* (siehe Abschnitt 5.6).

5.6. Kasia3D Server

Der Server ist komplett in Java 1.7 implementiert und besitzt keine grafische Benutzeroberfläche. Direkt nach Programmstart lauscht er auf einem angegebenen Port um Verbindungen entgegenzunehmen.

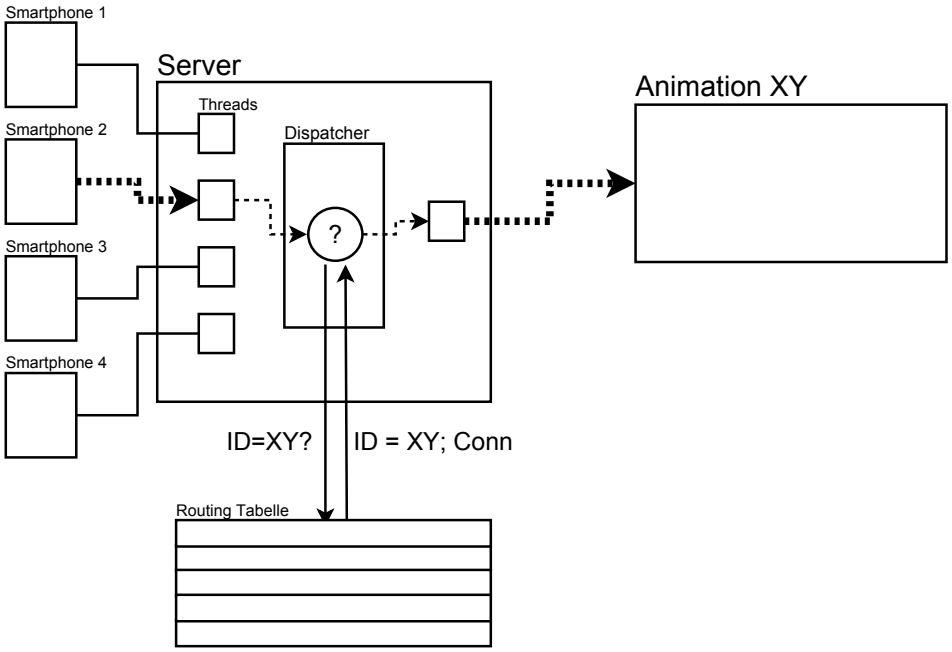

Abbildung 5.2.: Client-Server-Client-Kommunikation

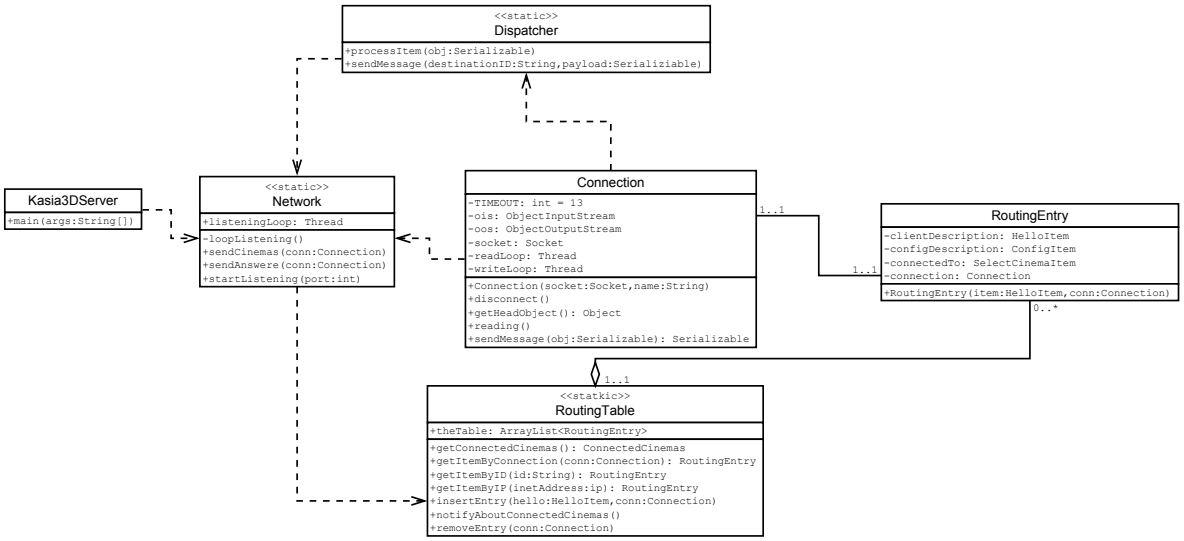

Abbildung 5.3.: Kasia3D Server: Klassendiagramm

5.6.1. Klassen

In Abbildung 5.3 werden die Klassen des Servers dargestellt.

Kasia3DServer Diese Klasse ist der Einstiegspunkt für das Programm. Dieses startet die Netzwerkkommunikation.

Network Diese statische Klasse dient dem Datenaustausch mit den Clients. Kern dieser Klasse ist die Methode *loopListening()*, die von der Methode *startListening(int port)* gestartet wird. *loopListening()* befindet sich in einem eigenen Thread und wartet auf eingehende Verbindungen. Wird eine eingehende Verbindung registriert und der Start-Handshake war erfolgreich (siehe 6.2 auf Seite 80), so wird diese in ein *Connection*-Objekt ausgelagert und ein Eintrag in der Routingtabelle erstellt. Die Methode *sendAnswer()* ist Teil des Verbindungsaufbaus und sendet eine Antwort, die den Namen und die ID des Servers enthält. *sendCinemas()* sendet im Falle eines verbundenen mobilen Clients, eine Liste aller registrierten Kinos zurück. Diese Methode wird ebenfalls aufgerufen, wenn sich ein neues Kino registriert und sendet eine Nachricht an alle mobilen Clients.

RoutingTable Die Hauptklasse des Servers. Auch sie ist statisch. In dieser Routing-Tabelle werden alle aktiven Verbindungen registriert und wird durch *RoutingEntry*-Objekte realisiert. Über diese Tabelle ist es möglich, über die ID, ein *Connection*-Objekt, oder eine IP-Adresse nach einem Eintrag zu suchen und empfangene Objekte weiterzureichen.

RoutingEntry Ein Eintrag in der *RoutingTable*. Er enthält zwei Attribute: Einerseits die *clientDescription*, also der Beschreibung des verbundenen Clients, das durch ein *HelloItem*-Objekt realisiert ist (siehe Abschnitt 5.11 auf Seite 73). Andererseits die *Connection*, die aktive Verbindung zu diesem Client.

Connection In den Verbindungsobjekten befindet sich ein Großteil der Kommunikationslogik. Jede *Connection* enthält den Eingabe-Strom (Attribut *ObjectInputStream ois*) und einen Ausgabestrom (Attribut *ObjectOutputStream oos*). Die Kommunikation wird als separater Lese-Thread implementiert, in welchem die Methode *reading()* ausgeführt wird. *reading()* prüft, ob auf dem Eingabestrom Daten hereinkommen. Werden diese korrekt erkannt, werden diese anhand der Ziel-ID über den *Dispatcher* an den jeweiligen Client weitergeleitet.

Dispatcher Der statische Dispatcher dient als Mittler zwischen den Verbindungen und der Routing-Tabelle. Wird in einem *Connection*-Objekt eine Nachricht empfangen, die für einen anderen Client bestimmt wird, so wird die Methode *processItem(Serializable obj)* aufgerufen. Anhand der Ziel-ID wird dann über die Methode *sendMessage(String destinationID, Serializable obj)* in der Routing-Tabelle nach der Ziel-Verbindung gesucht. Daraufhin wird die Methode *sendMessage(Serializable obj)* der Zielverbindung übergeben, die wiederum die Nachricht dem Empfänger zustellt.

5.6.2. Programmablauf

Nach dem Starten des Servers horcht dieser auf einem vorgegebenen Port. Inzwischen hat sich der Port 42424 aus zahlenmystischen Gründen etabliert. Verbindet sich ein Client zum Server wird die Verbindung in ein *Connection*-Objekt ausgelagert. Sobald ein *HelloItem*-Objekt eintrifft, wird ein Eintrag in der Routing-Tabelle erzeugt. Dieser Eintrag enthält neben der Verbindung später auch den vom Client ausgewählten Animations-Client (*CinemaSelectItem*), bzw. die Konfiguration für die Smartphne-Clients (*ConfigItem*). In den Austauschobjekten, die für den gerichteten Datenaustausch vorgesehen sind, wie z.B. die Gesten-Objekte (*GestureItem*), stehen jeweils auch die Empfänger-ID, an die das Objekt weitergeleitet werden soll. Trifft ein solches Objekt bei der eingehenden Verbindung in Thread ein, wird über den *Dispatcher* die Funktion *processItem(Serialziable)* aufgerufen. Diese verarbeitet das Objekt weiter. Wenn das Objekt weitergeleitet wird, ruft diese Funktion *sendMessage(String, Serializable)* auf. Diese Funktion wiederum sucht in der Routing-Tabelle nach den Eintrag, der zum gesuchten Ziel-Client gehört. Wie erwähnt enthält der Eintrag auch eine Referenz auf das Verbindungs-Objekt, über das das Objekt an den Ziel-Client weitergeleitet werden kann.

Trennt ein Client die Verbindung oder wird die Verbindung aufgrund eines Fehlers beendet, wird der entsprechende Client automatisch aus der Routingtabelle entfernt.

5.6.3. Technische Herausforderungen

Die einzige technische Schwierigkeit besteht darin, dass das System viele Anfragen gleichzeitig beantworten sollte, daher fiel die Wahl auf eine Lösung mit einem eigenen Thread pro Verbindung. Diese Technik hat das Problem der Endlosschleife im Dispatcher gelöst, die die Rechnerleistung sehr vermindert hat. Diese Endlosschleife sollte ursprünglich Verbindungen in einer Liste verwalten und in jedem Schleifenzyklus prüfen, ob ein neues Paket am offenen Socket angekommen ist. Allerdings wurde durch die Kommunikation durch Threads das Problem der Race-Conditions sehr spät erkannt, wenn zwei Threads gleichzeitig versuchen, entweder ein neues Element in der Routing-Tabelle anzulegen oder gleichzeitig über den Dispatcher eine Nachricht an den selben Client zu senden.

5.7. Kasia3D (Smartphone-Client)

Dieses Programm wird auf dem Smartphone installiert und erkennt Handgesten über die integrierten Beschleunigungssensoren des Geräts. Alternativ werden Press-Gesten durch Drücken einer Schaltfläche auf den Smartphone an den Server übermittelt. Die Datenübertragung erfolgt über den Server, der die Pakete entsprechend an den Animations-Client weiterleitet. Die Einstellungen des Smartphone-Client erhält dieser über Austausch-Objekte, die vom Animations-Client an alle verbundenen Geräte gesendet werden. Dem Benutzer wird über Vibration, über akustische Signale und über entsprechende Symbole mitgeteilt, ob eine Geste erkannt wurde und ob ein Objekt gefangen wurde.

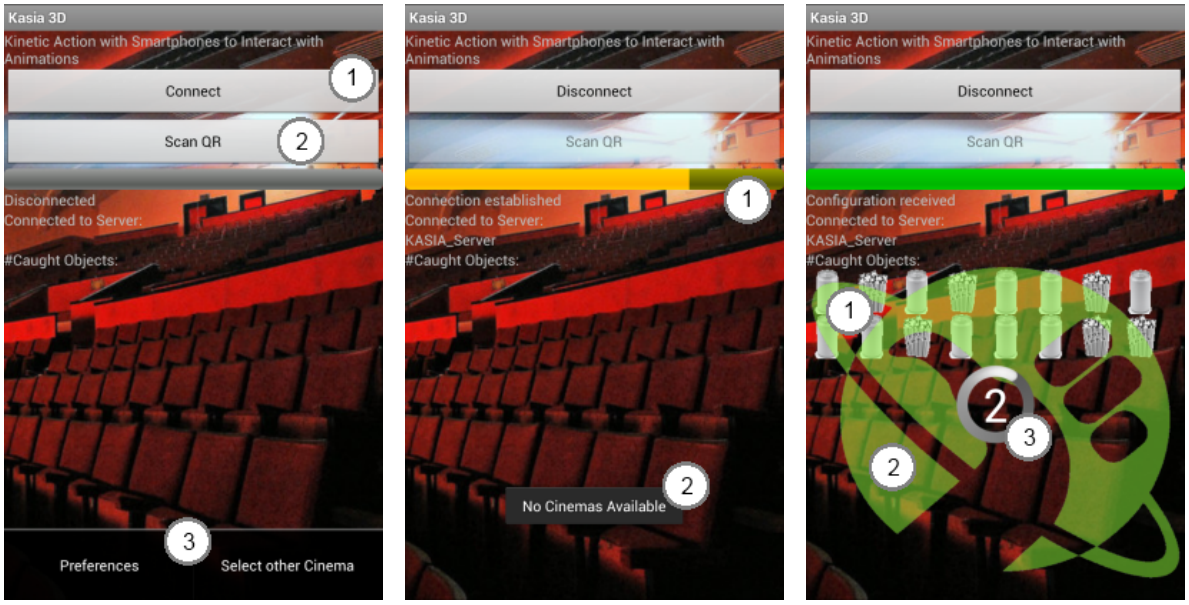

Abbildung 5.4.: Kasia3D Benutzeroberfläche *Links:* Nicht verbunden mit Kontextmenü im unteren Teil. *Mitte:* Mit Server verbunden, noch auf Animations-Clients wartend. *Rechts:* Verbunden und die Press-Geste ist aktiv. Der Countdown bis zum nächsten Versuch und einige gefangene Objekte werden angezeigt.

5.7.1. Grafische Benutzeroberfläche

Die Grafische Benutzeroberfläche wird unter Android komplett in XML[11] definiert. Über verschiedene Layout-Container werden diese dann auf auf dem Display angeordnet.

In Kasia3D werden sämtliche Elemente nur auf dem Startbildschirm angezeigt. Das erleichtert das Finden der unterschiedlichen Optionen. In einer früheren Version wurde die Touchgeste in einem anderen Ansicht angezeigt, zu der Gewechselt wurde, sobald man sich zum entsprechenden Kino verbunden hatte. Allerdings musste ein Weg gefunden werden, um auf die Ansicht zurückzukehren, wenn man aus Versehen auf den „zurück“-Button gedrückt hatte. Das erwies sich als sehr unpraktisch, weswegen auf eine zweite Ansicht verzichtet wurde. Alle Programmeinstellungen, wie die ID des Animations-Clienten, die Server-Adresse oder den Benutzernamen zu ändern Kontextmenü verlagert. Neben diesen Programmeinstellunen befindet sich hier auch die Möglichkeit, das Kino zu wechseln. Eine Beschreibung der Einzelnen Elemente befindet sich in Abschnitt 5.7.2 auf der nächsten Seite.

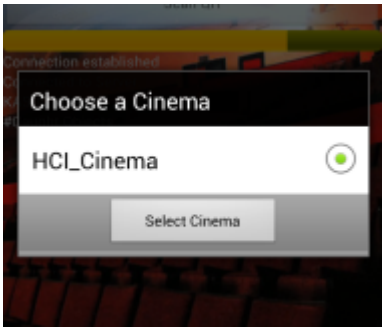

Abbildung 5.5.: Kasia3D: Auswahl eines Animations-Clients

5.7.2. Programmablauf

Sobald die Applikation heruntergeladen und installiert wurde, kann sie über ein Tippen im App-Bereich von Android gestartet werden. Wird das Programm zum ersten Mal gestartet wird intern eine eindeutige ID erzeugt und der Benutzer zur Eingabe eines Benutzernamens aufgefordert. Es öffnet sich der Startbildschirm (vergleiche Abbildung 5.4 Links). Anschließend werden beide Informationen gespeichert. Durch *Scan QR* (1) lässt sich ein QR-Code scannen, in dem einerseits das Installationspaket, andererseits die Verbindungsinformationen kodiert sind (siehe Abschnitt 5.10 auf Seite 71). Dieser Code kann entweder auf der Eintrittskarte abgebildet sein oder durch den Animations-Client an die Leinwand geworfen werden. Anschließend wird die Verbindung aufgebaut. Die Verbindung kann ebenfalls durch Drücken des Buttons *Connect* (2) hergestellt werden. Müssen Änderungen am Benutzernamen vorgenommen werden oder lässt sich der QR-Code nicht scannen, können über das Kontextmenü die Einstellungen vorgenommen werden (3). Ebenfalls im Kontext-Menü gibt es die Möglichkeit, den Animations-Client zu wechseln, vorausgesetzt es sind mehrere Clients mit dem Server verbunden.

Wurde der QR-Code gescannt oder wurde durch Drücken auf *Connect* eine Verbindung aufgebaut, verändert sich die Ansicht (vergleiche Abbildung 5.4 Mitte). Der Fortschrittsbalken (1) wechselt zur Farbe gelb und verharrt bei 75%, solange sich kein Animations-Client verbunden hat und es erscheint eine entsprechende Nachricht (2). Hat sich zuvor schon ein Animations-Client verbunden, wird der Benutzer aufgefordert, einen davon auszuwählen (vergleiche Abbildung 5.5).

Nach Auswahl des Animations-Clients wird der Fortschrittsbalken auf grün und auf 100% gesetzt (vergleiche Abbildung 5.4 Rechts). Sobald die Animation gestartet wurde sendet sie ein *ConfigItem*, durch das das Verhalten (Abkühlzeit der Animation, Art der Geste: Handgeste oder Press-Geste) festgelegt wird. Von nun an können die Elemente gefangen werden. Handgesten werden durch eine Bewegung mit dem Smartphone ausgeführt. Um eine Press-Geste auszuführen muss die große Schaltfläche in der Mitte gedrückt werden, die zusätzlich erscheint und unsichtbar ist, wenn diese nicht aktiv ist (1). Nach jeder korrekten

[11] Extensible Markup Language

Geste, egal ob Hand- oder Press-Geste, wird dies dem Benutzer durch ein kurzes Vibrieren und ein akustisches Signal signalisiert. Zudem erscheint in der Mitte ein Countdown, der dem Benutzer anzeigt, wann er wieder eine Geste ausführen darf (2). Wurde ein Objekt gefangen so wird das durch ein langes Vibrieren und durch ein anderes akustisches Signal mitgeteilt. Zudem erscheint in der Mitte des Displays ein kleines Symbol, das das gefangene Objekt repräsentieren soll (3).

Am Ende der Animation werden die Gewinner benachrichtigt. Auch das erfolgt über ein akustisches Signal. Anschließend werden alle mit der Animation verbundenen Clients beendet und so die Ressourcen für den Beschleunigungssensor und die Netzwerkverbindung wieder freigegeben.

5.7.3. Benutzer-Feedback

In Kasia gibt es drei Arten des Benutzer-Feedbacks: Vibration, über Akustische Signale und über visuelles Feedback.

Wie in Abschnitt 3.2 auf Seite 30 beschrieben, ist auch hier haptisches Feedback in Form von Vibration vorhanden. Vibration wird dann verwendet, wenn ein Objekt durch den Benutzer gefangen wurde. Akustische Signale werden verwendet um dem Benutzer mitzuteilen, dass er entweder eine korrekte Geste ausgeführt, ein Objekt gefangen, oder das Spiel gewonnen hat. Durch dieses akustische Signal soll die Nachbarschaft um den Benutzer dazu animiert werden, ebenfalls an dem Spiel teilzunehmen. Visuelle Rückkopplung wird nur verwendet, um dem Benutzer zu zeigen, wie viele Objekte er gefangen hat, wie lange es dauert, bis er wieder eine Geste ausführen kann, oder einfach nur, wo sich die Schaltfläche für die Press-Geste befindet.

Die Abkühlzeit, also die minimale Zeit, zwischen Geste auszuführen, wird hier auch dem Benutzer-Feedback zugeordnet. Diese Zeit soll den Benutzer dazu anhalten zu überlegen, wann es sich lohnt, wieder eine Geste auszuführen. Dass es überhaupt eine Abkühlzeit gibt liegt auch daran, dass Benutzer sonst versuchen würden, die ganze Zeit wild Gesten auszuführen. In diesem Fall wäre es eher Zufall, dass ein Benutzer ein Objekt fängt, anstatt er mit gezielte Absicht eine Geste ausführt um etwas zu fangen.

5.7.4. Gestenerkennung

Die Gestenerkennung ist verhältnismäßig rudimentär implementiert. Es wird geprüft, ob die Beschleunigungswerte über einen bestimmten Schwellwert steigen und wenn das zutrifft, wird der erkannte Beschleunigungsvektor zusammen mit der Geste an den Animation-Client geschickt und dort entsprechend das gefangene Objekt in diese Richtung geschleudert. Zudem werden die 15 Messwerte vor und nach der Geste mitgesendet.

Press-Gesten werden dadurch erkannt, dass der Benutzer die Schaltfläche gedrückt hat.

Was für die Auswertung interessant ist die Tatsache, dass auch ungültige Gesten an den Animations-Client gesendet werden. Als ungültig wird hier eine Geste bezeichnet, die vom Benutzer innerhalb der Abkühlzeit ausgeführt wurde. Diese Gesten haben allerdings keinen Einfluss auf die gefangenen Objekte und dienen nur der Statistik.

5.8. Kasia3D Engine (Animationsclient)

Die größte Teil der Arbeit steckt in der *Engine*, also in dem System zur Durchführung einer Simulation. Auch dieses System ist in Java implementiert und verwendet für die stereoskopische Animation Java3D.

5.8.1. Model-View-Controler

Dieses Programm wurde streng nach dem MVC-Paradigma entwickelt. Dabei wird die Programmlogik sowohl von der Speicherhaltung, als auch von der Benutzeroberfläche getrennt.

Model Paket: `net.kaffeezombie.kasia.engine.model`. Hier befinden sich alle Klassen, in denen Daten gehalten werden oder die Daten repräsentieren. Hier ist auch die Hauptklasse *FlyingObject* enthalten, die ein animiertes Objekt repräsentiert.

View Paket: `net.kaffeezombie.kasia.engine.gui`. Dieses Paket enthält die grafische Benutzeroberfläche (GUI). Hier befinden sich alle Klassen, mit denen der Benutzer die Animation einstellen und beeinflussen kann.

Control Paket: `net.kaffeezombie.kasia.engine.control`. Alle Abläufe, die entweder die Animation oder das Programm selbst bestimmen, befinden sich in diesem Paket. Dazu zählt die Programmsteuerung selbst, die Animation, das Szenen-Rendering und die Netzwerksteuerung. Im Unterpaket `.observer` befindet sich die Überwachungslogik.

5.8.2. Klassenübersicht

Hier werden die wichtigsten Klassen des Animations-Client vorgestellt. Die Kommunikation zwischen den Hauptklassen ist in Abbildung 5.6 dargestellt

Paket: `net.kaffeezombie.kasia.engine`

Kasia3DEngine Diese Klasse enthält die *main()*-Methode und ist somit Einstiegspunkt für das Programm. Über diese Klasse hat man auch Zugriff auf den Haupt-Thread. Nur über diesen kann Java3D eine Animation korrekt anzeigen. Der Splash-Screen (*CinemaSplash*), der einen Barcode enthält, muss ebenfalls auf diesem Thread ausgeführt werden.

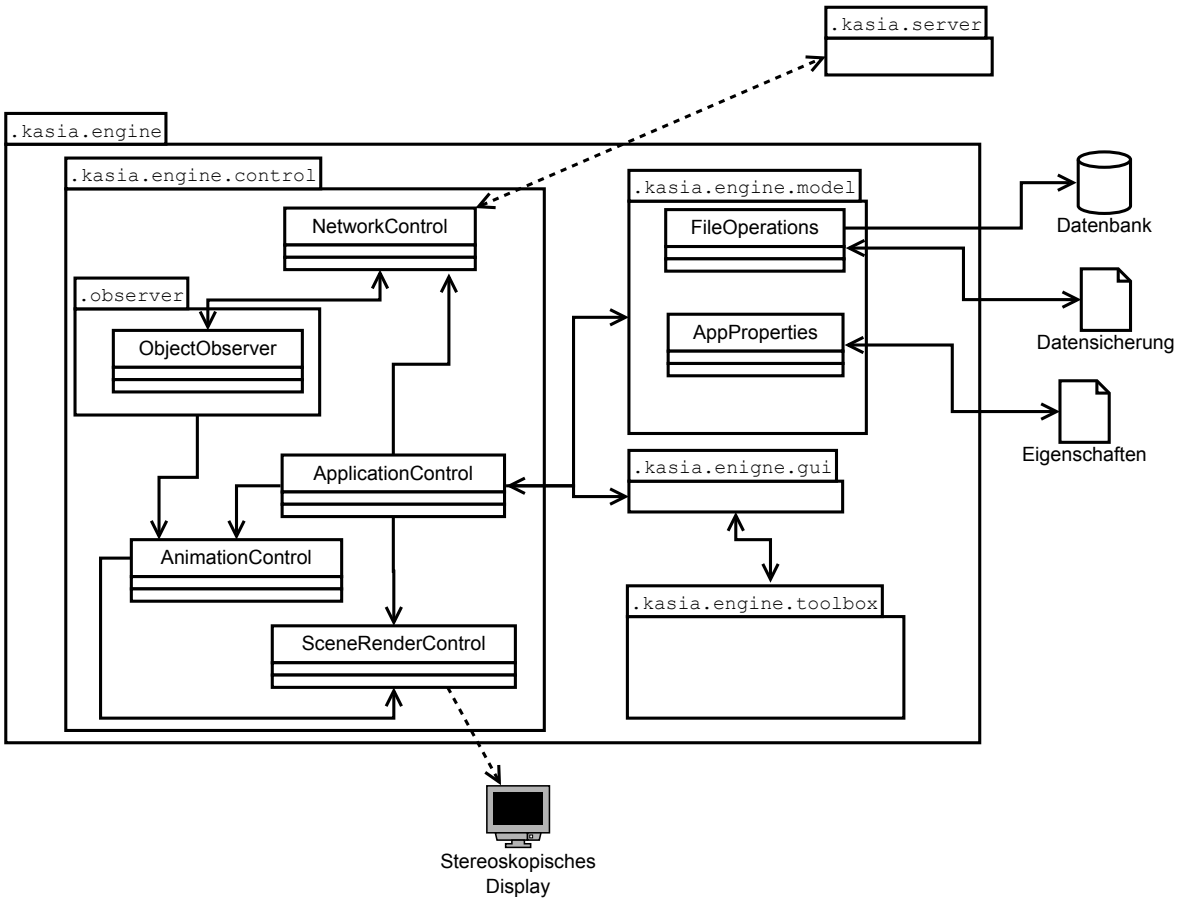

Abbildung 5.6.: Kasia3D Engine: Klassenkommunikation

Paket: `net.kaffeezombie.kasia.engine.control`

ApplicationControl Diese statische Klasse überwacht die komplette Programmsteuerung. Hier wird die Benutzeroberfläche geladen, diese mit Inhalten gefüllt und andere Methoden aufgerufen, wie die Animation, sowie das Laden und Speichern. Hier werden auch die optionalen Kommandozeilenargumente verarbeitet.

NetworkControl Diese statische Klasse stellt die Netzwerkkommunikation mit dem Server her. Die Verbindung wird in einen eigenen Thread ausgelagert, wodurch der Rest des Programms nicht beeinflusst wird. Das *NetworkControl* dient ebenfalls dazu, Nachrichten zu versenden, empfange Nachrichten zu sichten und diese gegebenenfalls an die zuständige Klasse weiterzuleiten. Realisiert wird das durch Queues: Empfangene Objekte werden von *readStream()* in die *incomingQueue* geschrieben, die von anderen Klassen mit der Methode *readMessage()* ausgelesen werden kann. Durch die *outgoingQueue* können andere Klassen via *sendObject(Object obj)* Nachrichten versenden. Entsprechende Objekte werden an diese Queue gesendet, die von *writeStream()* ständig kontrolliert wird. Diese Methode sendet die Objekte weiter an den Server.

SceneRenderControl Auch *SceneRenderControl* ist statisch und ist die Klasse, die sich um die Darstellung der dreidimensionalen Umgebung kümmert. Hier wird der Szenen-Graph aufgebaut, die Fliegenden Objekte eingefügt, der Hintergrund verändert, die physische Umgebung manipuliert (hintere und vordere Schnitt-Ebene der virtuellen Umgebung) und auf Tastatureingaben reagiert. In dieser Klasse befinden sich ebenfalls Hilfsfunktionen, um die zu animierende Objekte zu modifizieren und zu bewegen.

AnimationControl *AnimationControl* dient letzten Endes des Bewegung der Modelle. Vor dem Starten der Animation werden zuerst alle Modelle vorbereitet. Anschließend werden die Modelle und ihre Bewegungen nach und nach in die Animation eingefügt und Schritt für Schritt bewegt und geschaut, ob sie „fangbar“ sind. (siehe Abschnitt 5.9 auf Seite 69).

Paket: `net.kaffeezombie.kasia.engine.control.observer`

ObjectObserver Der statische *ObjectObserver* dient als Schnittstelle zwischen Netzwerkkommunikation (*NetworkControl*) und der Animation (*AnimationControl*). Er überwacht Nachrichten, die vom Server hergeleitet werden und prüft, ob es sich dabei um eine (gültige) Geste handelt. Wurde eine Nachricht als Geste erkannt, prüft er weiter, ob sich ein Modell in „Fangnähe“ befindet. Ist dies der Fall, so animiert er dieses Modell entsprechend der Geste und signalisiert dem Sender, über *NetworkControl.sendObject(Object obj)*, dass der Sender dieses Objekt gefangen hat.

Paket: `net.kaffeezombie.kasia.engine.gui`

IGui Dieses Interface ermöglicht es, alle Benutzeroberflächen mit *refresh()* zu aktualisieren: Jede Klasse, die diese Schnittstelle implementiert, kann somit einfach dazu veranlasst werden, die angezeigten Daten erneut einzulesen und darzustellen.

MainWindow Titel: Kasia3D*Engine*. Im *MainWindow* wird das Programm ausgeführt. Es enthält eine *MenuBar*, über die derzeitigen Einstellungen gespeichert, bzw. vorhandene Einstellungen geladen werden können, sowie eine *TabbedPane*, in der alle Reiter der Anwendung enthalten sind. Diese sind im Folgenden:

TabDisplayConfiguration Angezeigter Name: *Display Configuration*. Dieser Reiter kümmert sich um die Darstellung der virtuellen im Bezug zur physikalischen Welt. Hier kann der Benutzer Einfluss nehmen auf die *SceneRenderControl.frontClipDistance*, die vorderste Schnittebene der Animation die sich bei Abstand 0 im Benutzer selbst befindet, der *SceneRenderControl.backClipDistance*, der hinteren Ebene, hinter der keine Modelle mehr angezeigt werden und die *SceneRenderControl.userDistance*, dem virtuellen Abstand des Benutzers zum Bildschirm.

TabFlyingObjects Angezeigter Name: *Flying Objects*. In diesem Tab werden alle sichtbaren Modelle geladen und bearbeitet. Hier kann ein Hintergrund und die Geschwindigkeit der Animation definiert werden. Um Objekte zu laden und zu bearbeiten, wird durch die Schaltflächen *New...* und *Edit...* ein *DialogFlyingObject*-Dialog gestartet. Durch *Remove* wird das Objekt gelöscht und durch *Duplicate* das ausgewählte Objekt geklont.

TabClientRemoteControl Angezeigter Name: *Client Control*. In diesem Element kann der Benutzer gesteuert werden. Hier wird definiert, wie lange die Abkühlzeit dauern soll und ob der Benutzer mit einer Press- oder einer Handgeste interagieren kann. Das Entsprechne *ConfigItem* wird allerdings erst bei Animationsstart gesendet.

TabMonitoring Angezeigter Name: *Monitoring*. Hier werden die Ergebnisse der derzeitigen Animation gezeigt: Name und ID der vebundenen Clients sowie die Anzahl der von ihnen gefangenen Objekte.

TabShowRoom Angezeigter Name: *Show Room*. Hier wird die Form und das Aussehen der Fangzone definiert. Es ist ebenfalls möglich, eine sich bewegende Fang-Zone zu definieren.

TabNetworking Angezeigter Name: *Network Configuration*. In diesem Reiter können verschiedene Einstellungen getroffen werden: Der Name des Animations-Clients, die IP, bzw. die URL und der Port des Servers. Hier kann geprüft werden, ob der Server im Netzwerk erreichbar ist und es wird auch die automatisch generierte ID des Animations-Clients angezeigt.

DialogColors Angezeigter Name: *Colors*. Mit Hilfe dieses Dialogs lassen sich die Rot-, Grün- und Blauwerte einer Farbe einstellen. Optional lässt sich ebenfalls der Alphawert einer Farbe einstellen.

DialogFlyingObject Angezeigter Name: *Flying Object*. Dieser Dialog dient der Repräsentation von animierten Modellen. Hier kann eingestellt werden, welche Erscheinungsform (*Appearance*) ein Teilobjekte des Modells haben soll. Zudem kann festgelegt werden, ob ein Modell in der Animation eine Zufällige Farbe haben soll, transparent ist oder überhaupt kein visueller Effekt auftreten soll. Auch die Größe der Fangzone kann hier festgelegt werden.

DialogFlobAppearance Angezeigter Name: *Appearance*. Dieser Dialog dient der Darstellung eines Modells. Hier kann festgelegt werden, welche Farben das „Material“ ein Objekt haben soll, welche Eigenschaften die Polygone haben, wie die Transparenz aussieht, oder welche Textur ein Modell haben soll. Dabei steht jeder Reiter für eine andere Eigenschaft der Darstellung.

TabDialogAppearanceMaterial Angezeigter Name: *Material*. Hier können die vier Teilfarben (Ambient, Specular, Emmissive und Diffuse), das ein Material definiert, eingestellt werden. Zudem kann auch der Shininess-Wert festgelegt werden.

TabDialogAppearanceTexture Angezeigter Name: *Texture*. Hier kann eine beliebige Grafik geladen und als Textur festgelegt werden. Zudem kann eingestellt werden, wie die Farben des Materials mit der Textur zusammenwirken.

TabDialogAppearanceTransparency Angezeigter Name: *Transparency*. Dieser Reiter regelt, welche Funktion für die Transparenz verwendet werden, und wie stark das Objekt transparent sein soll.

TabDialogAppearancePolygonAttributes Angezeigter Name: *Polygon Attributes*. Hier wird festgelegt, wie die einzelnen Polygone, aus denen das Modell aufgebaut ist, dargestellt werden: Ob die Flächen normal dargestellt werden, ob nur die Rahmenlinien, oder ob nur die Eckpunkte auftauchen sollen.

TabAnimationStart Angezeigter Name *Start It!*. In dieser Ansicht werden die letzten Einstellungen getroffen, um eine Animation anzuzeigen. So kann definiert werden, wie der QR-Code aussieht, ob er beim Starten der Animation angezeigt und ob beim Starten der Animation ebenfalls eine Netzwerkverbindung aufgebaut werden soll. Hier wird ebenfalls die Animation über den Knopf *Start Animation* in Gang gesetzt.

CinemaSplash Dieser Splash-Screen enthält einen QR-Code und einen Countdown, der angibt, wie lange der Code noch angezeigt wird.

ImagePanel Das *ImagePanel* wird verwendet, um einfach Grafiken in der Benutzeroberfläche anzuzeigen. Mit *setImage(BufferedImage Image)* wird einfach ein Bild übergeben, das an der Stelle dieses *Panel*s angezeigt wird.

Paket: `net.kaffeezombie.kasia.engine.model`

AppProperties Hier werden die unterschiedlichen Eigenschaften (*Properties*) Systems gespeichert. Die Eigenschaften enthalten grundlegende Einstellungen, wie der zuletzt verwendete Suchpfad, Name des Clients, ID des Clients, der zuletzt ausgewählte Server und die letzte Fensterposition des Hauptfensters.

FileOperations Diese statische Klasse übernimmt das Laden und Speichern von Systemzuständen. So kann eine komplette Animation und sämtliche Einstellung sowohl der Animation als auch der Fliegenden Objekte gespeichert werden. In dieser Klasse wird auch auf die Datenbank zugegriffen, welche für die Auswertung der Gesten- und Objektdaten von großer Bedeutung ist.

FlyingObject Ein Fliegendes Objekt entspricht einem animierten Objekt in der Animation. Um die verschiedenen Ansprüchen an so ein Modell Genüge zu leisten, wurden alle Methoden um ein geladenes Modell zu verändern in diese Klasse vereinigt. So kann über diese Klasse ein Modell transformiert werden (vergrößern, verkleinern, drehen und an einen anderen Punkt setzen), mit Hilfe von *IMotion*-Klassen in der Szene bewegt, das Aussehen (*Appearance*) verändert, mit Text zu überlaget und sogar komplett geklont werden. Intern wird ein *FlyingObject* als eine *BranchGroup* dargestellt, die wieder in Unterzweige geteilt ist und die gemeinsam oder separat manipuliert

werden können. Von außen jedoch, erscheint ein *FlyingObject* als eine Einheit, deren komplexe interne Struktur so sehr abstrahiert wird. Wird ein *FlyingObject* animiert, wird lediglich der damit verbundene *mainBranch* in die Animation eingehängt. Die mit dem Objekt verbundene *IMotion*-Objekt kümmert sich daraufhin um die Bewegung des Objekts in der Szene.

FlyingObjectContainer Diese statische Klasse dient zur Verwaltung der *FlyingObject*-Instanzen. Hier können Modelle abgelegt, gesucht und gelöscht werden.

IMotion Dieses Interface stellt das Gerüst einer Bewegung dar. Es dient dazu, für verschiedene Bewegungs-Szenarien eine einzige Schnittstelle bereitzustellen, über die jede Bewegung in derselben Weise auf die Implementierungen zugegriffen werden kann.

ObjectMotion Diese Implementierung des Interface *IMotion* dient der normalen Bewegung von Modellen. Dabei wird in einer definierten hinteren Entfernung zufällig ein Punkt bewegt, der sich dann auf den Benutzer mit der konstanten Geschwindigkeit $z = 1$ zubewegt. Befindet sich das Verbundene Modell in Schlagweite, wird dieses entsprechend animiert.

HitMotion Diese Bewegung wird einem Objekt in dem Moment zugewiesen, in dem das Modell gefangen wurde. Im Normalfall, das heißt, das Objekt wird durch eine Handgeste getroffen, wird dabei das Objekt in die Richtung der Geste geschleudert. Dabei wird der Name des fangenden Clients über das Objekt gelegt. Wird das Objekt nur durch eine Handgeste getroffen, bewegt sich das Objekt in die Richtung zurück, aus der es gekommen war.

CatchZoneMotion Diese Bewegung wird verwendet, um die Fangzone zu animieren. So ist es möglich, die Schwierigkeit dadurch zu erhöhen, dass manche Objekte nur kurz oder gar nicht gefangen werden können.

Paket: `net.kaffeezombie.kasia.engine.toolkit`

GuiTools Diese Klasse ist ebenfalls statisch. Hier befinden sich zahlreiche Hilfsmethoden, wie zum Beispiel *createQRCode(String content, int qrSize)*, welches einen String mit Hilfe der Bibliothek `com.google.zxing.qrcode.*`[12] in eine zweidimensionale Barcode-Grafik umwandelt. Zudem gibt es einige Methoden, die sich um die Positionierung der Anwendung auf dem Bildschirm kümmern.

Log In diesem Logger werden Statusnachrichten aus dem gesamten Programm formatiert auf der Konsole ausgegeben.

Strings Diese statische Klasse dient dazu, die Übersetzung in andere Sprachen zu vereinfachen. Alle Elemente der Benutzeroberfläche haben einen einen eindeutigen Schlüssel. Die Klasse *Strings* sucht nach diesem in der Datei `Strings_XX.txt` (wobei `XX` für das

[12] http://code.google.com/p/zxing/

Abbildung 5.7.: Beispiel für eine Animation

Landeskürzel steht) und gibt den damit verbundenen Wert zurück. Sollte ein Schlüssel nicht in dieser Datei vorhanden sein, wird das durch auffällige Schrift in der Benutzeroberfläche angezeigt.

5.9. Animationsablauf

Eine Animation läuft in nicht immer klar trennbaren Schritten ab. Es wird aber hier versucht, den Ablauf möglichst genau zu schildern.

Bei Programmstart wird zunächst der *observationThread* des *ObjectObserver* mit Hilfe der Methode *startObservation()* gestartet. Dieser dient dazu, auf dem Netzwerkkanal zu lauschen und eingehende Gesten zu verarbeiten.

Nun können die Modelle in der Benutzeroberfläche definiert werden. Zudem kann der Hintergrund eingerichtet (anderes Hintergrundbild, Art der Skalierung), die Fangzone eingestellt (Form, Position, Aussehen), das Aussehen der Objekte in der Fangzone (Glühen, Transparenz oder nicht unternehmen), und die Abstände der vorderen und hinteren Schnittebene festgelegt werden. Modelle werden geladen, indem über den Reiter *Flying Objects* ein neues Objekt erstellt und dieses im folgenden Dialog angepasst wird. In diesem Dialog kann ein Objekt aus dem Datei-System geladen, wenn es sich hierbei um ein Wavefront-Objekt handelt. Wurde es erfolgreich geladen, kann entweder das vorhandene Aussehen übernommen, die Teil-Geometrien verändert werden. Auch hier ist es möglich, die Farbe, die Transparenz und die Textur der einzelnen Teile zu verändern. Möchte man eher etwas Abwechslung in der Animation haben, kann man den Objekten auch zufällige Farben zuweisen, die erst in der Vorbereitung der Animation zugewiesen werden. Neben dem reinen Aussehen der Geometrien kann das Modell als Ganzes auch verkleinert oder vergrößert werden. Die meisten erprobten Objekte waren sehr groß und mussten deutlich verkleinert werden.

Im nächsten Schritt kann das Verhalten der verbundenen KASIA-Clients beeinflusst werden. Über den Reiter *ClientControl* kann die Art der Geste eingestellt und die sogenannte Abkühlzeit definiert werden. Wie bereits erwähnt ist das der zeitliche Abstand, zwischen dem keine Gesten ausgeführt werden können.

Bevor man die Animation startet, muss noch im Tab *Network Configuration* ein Name für diesen Animations-Client und die IP, bzw. die URL des Servers angegeben werden. Nun kann die Netzwerkverbindung durch *Ping Server* getestet oder gleich gestartet werden. Für den Verbindungsaufbau hat man zwei Möglichkeiten: Entweder in diesem Reiter auf *Connect* klicken, oder im Reiter *Start It!* das Häkchen bei *Establish Connection on Start* setzen. Wird keine Netzwerkverbinung etabliert, können sich auch keine mobilen Clients mit der Animation verbinden.

Im Tab *Start It!* ist es ebenfalls möglich, vor dem Starten der Animation noch einen Splash-Screen anzuzeigen. Dabei handelt es sich um ein Vollbildfenster, das einen Countdown mit der restlichen Anzeigedauer desselben, sowie einen QR-Code anzeigt. Der QR-Code wird in Abschnitt 5.10 auf der nächsten Seite erklärt. Dafür muss das Häkchen *Show QR-Code on Start* gesetzt und eine Anzeigedauer in Sekunden eingestellt werden.

Durch Klicken des Knopfes *Start Animation* wird die Methode *ApplicationControl.startScene()* aufgerufen und im Hintergrund werden folgende Schritte ausgeführt:

Ist das Häkchen zum Netzwerkaufbau gesetzt, so wird eine Verbindung zum Server aufgebaut. Besteht bereits eine Netzwerkverbindung, wird diese genutzt. Trifft keines der beiden zu, so befindet sich die Animation in einer Art Testmodus, in der die Objekte zwar korrekt animiert, jedoch nicht ihnen interagiert werden kann.

Im nächsten Schritt werden die Modelle vorbereitet. Dabei werden aus den N definierten 3D-Modellen M gezogen (mit zurücklegen), die Bewegung (*ObjectMotion*) des Modells berechnet (mit Hilfe der Methode *ObjectMotion.initAnimation()*), diese dem Modell zugewiesen und die Bewegung in eine Warteschlange (*AnimationControl.preloadedMotionList*) eingefügt.

Dabei wird die Methode *FlyingObject.prepare()* des Modells aufgerufen: Wurde beim Erstellen der Modelle eine zufällige Farbe angegeben, so wird diese nun den Objekten zugewiesen. Anschließend wird das Modell kompiliert. Dadurch kann das Objekt bei der Animation zum richtigen Zeitpunkt in den Szenegraphen eingefügt werden, ohne dass die Animation durch den damit verbundenen Rechenaufwand gestört wird.

Ist das Häkchen für den Splash-Screen gesetzt, so wird dieser nun erzeugt und für die eingestellte Dauer angezeigt.

Dann wird die Szene mit *SceneRenderControl.prepareScene()* vorbereitet: Die Konfigurationsdatei `j3d1x1-stereo.cfg`[13] wird geladen und der Szenengraph daran angepasst. Diese Datei übernimmt grundlegende Einstellungen, zum Beispiel dass die Szene stereoskopisch ist und welcher Bildschirm für die Animation verwendet wird. Dieser Mechanismus ist Teil

[13] http://download.java.net/media/java3d/javadoc/1.4.0/com/sun/j3d/utils/universe/ConfiguredUniverse.html

von Java3D. Sobald das Universum erstellt wurde, wird dieses angezeigt. Nun wird die restliche Szene angepasst: Die Werte für den Abstand des Benutzers, die vordere und die hintere Wand der Animation (*userDistance*, *frontClipping* und *userDistance*) werden aus der Benutzeroberfläche übernommen. Anschließend wird die Beleuchtung in den *BranchGroup*-Wurzelknoten eingefügt und dieser dann an das Universum angehängt. Im letzten Schritt wird ein *KeyListener* an der Szene registriert, über den der Benutzer der Animation die Möglichkeit hat, die Szene ebenfalls zu manipulieren.

Anschließend wird die Animation gestartet (*AnimationControl.startAnimation()*, welches hier der Taktgeber ist, der für die Animation jedes Modell alle 10 Millisekunden um einen individuellen Faktor bewegt. Diese Bewegung wird ebenfalls in einem eigenen Thread ausgeführt und wird in diesem Fall über ein *Timer*-Objekt realisiert: Anstatt den Thread mit *Thrad.sleep(int millis)* für *n* Millisekunden zu unterbrechen, so wie es an anderer Stelle des Programms geschieht, wird dem Objekt hier ein *ActionListener* und die Frequenz übergeben, die angibt, in welchem Abstand in Millisekunden dieser Listener aufgerufen und die Anweisungen dort ausgeführt werden.

Nun werden die Modelle nach und nach eingefügt. Dabei werden die bereits vorbereiteten Bewegungen gestartet, das heißt auf der vordefinierten Bahn Schritt für Schritt weitergezogen. Gelangt das Modell in die Fangzone, wir dies vom *ObjectObserver* registriert. Dieser fügt das Modell in die List *inRangeItems*. Wird ein Modell nun gefangen, bevor es aus der Fangzone wieder hinaus gelangen konnte, wird es entsprechend der Einstellungen animiert. Dafür wird das *IMotion*-Objekt durch ein anderes ersetzt, welches zuvor ebenfalls mit *.initAnimation()* vorbereitet wird. Dadurch bewegt sich das Modell entsprechend anders. Wurde das Modell gefangen, wird über *FileOperations* ein entsprechender Eintrag in die Datenbank eingefügt. In jedem Fall, egal ob gefangen oder nicht, werden Gesten, die von der Animation empfangen wurden, in die Datenbank eingefügt.

Hat ein Objekt die Fangzone wieder verlassen, ohne gefangen worden zu sein, wird es aus der Liste *inRangeItems* des *ObjectObserver*s wieder gelöscht und setzt seinen Weg fort.

Wird die Animation beendet wird der Gewinner zum Schluss mit einem *WinnerItem* benachrichtigt und die sämtliche verbundenen Smartphone-Clients mit einem *ConfigItem* getrennt. Auch hier wird das Ergebnis zuvor noch in der Datenbank gespeichert, die Eigenschaften wie die Postion dies Hauptfensters, die Adresse des Servers und der Name des Animations-Clients in den *Properties* gespeichert und auch sämtliche Einstellungen, die für diese Animation vorgenommen wurden, in der Datei `LastPreferences.ksa` gespeichert.

5.10. QR-Code im Kasia-System

Bei einem QR-Code (was für Quick Response steht)[14] handelt es ich um einen zweidimensionalen Barcode, der inzwischen von den meisten Smartphones über deren eingebaute Kamera erkannt werden kann.

[14] http://www.qrcode.com/en/index.html

Abbildung 5.8.: Ein typischer QR-Code im Kasia System

Im Kasia-System hat der QR-Code zwei Aufgaben. Erstens ist es möglich, über den Code die Webseite zu finden, auf der der mobile Client zu finden ist. Und zweitens wird in dem Code kodiert, wie die Adresse des Kasia-Servers lautet. Ein Typischer Kasia-QR-Code ist in Abbildung 5.8 dargestellt und hat den Inhalt: `http://www.kaffeezombie.net/kasia/Kasia3D.apk?kasia=KASIA;Cinema_@12345;HCI_Cinema;178.63.194.227;42424`. Dieser kann dann vom mobilen Client geparst werden. Die einzelnen Bestandteile haben folgende Bedeutung:

`http://www.kaffeezombie.net/kasia/Kasia3D.apk` Das ist der Ort im Netzwerk oder im Internet an dem das Programm heruntergeladen werden kann. Handelt es sich nicht um den Android Market muss zuvor in den Einstellungen des Smartphones die Berechtigung aktiviert werden, Applikationen auch aus „unsicherer“ Quelle zu installieren.

`?kasia=KASIA` Hierbei handelt es sich nur um einen Marker, damit der Kasia-Client weiß; ob dies ein gültiger „KASIA-String“ ist.

`Cinema_@12345` Das ist die eindeutige ID des Animations-Client, der diesen QR-Code erzeugt hat. Nur über diese ID wird die Verbindung über den Server ermöglicht.

`HCI_Cinema` Das ist der Name des Kinos, bzw. des Animations-Clients.

`178.63.194.227` **und** `42424` Das ist die IP-Adresse und der entsprechende Port des Kasia-Servers, über den die Kommunikation laufen soll.

Das Erzeugen und Scannen von QR-Codes wird über die *ZXing*[15]-Bibliotheken ermöglicht, die frei verfügbar sind. Auf der Smartphone-Seite wird lediglich ein *Indent* eingefügt, über das die zuvor installierte Scan-Applikation gestartet wird. Bei einem *Indent* handelt es sich um einen Android-Baustein, über den andere Applikationen aufgerufen und genutzt werden können.

[15] http://code.google.com/p/zxing/

5.11. Datenaustausch

Der Datenaustausch in diesem System erfolgt über verschiedene serialisierbare Java-Objekte. Dabei werden die bereits bestehenden Objekte dem Datenstrom einer offenen Socketverbindung übergeben. Auf der Empfängerseite kann das empfangene Objekt direkt im Programm wieder verwendet werden.

Die Socketverbindung verwendet das TCP/IP-Protokoll. Dieses stellt sicher, dass IP-Pakete garantiert und in der richtigen Reihenfolge beim Empfänger ankommen, was wiederum der Fairness während einer Simulation dient.

Im Gegensatz zum JSON-Austauschformat, zu XML oder gar zu reinem Text, hat diese Technik den Vorteil, dass viel weniger Zeit dafür verwendet werden muss, um Objekte aus Text zu generieren. Dadurch, dass Objekte bereits in Binärform vorliegt und nicht erst Text geparst, interpretiert und nur aus Schlüsselk-Wert-Paaren Objekte aufgebaut werden müssen, ist der Overhead für die Verarbeitung sehr gering. Um allerdings in Zukunft kompatibel zu anderen mobilen Plattformen, wie einem Windows- oder einem iOS-System, zu sein, würde sich aber in Zukunft ein Wechsel zu einem anderen Format anbieten.

Die Klassen, die für den Datenaustausch verwendet werden, befinden sich im Paket `net.kaffeezombie.kasia.kasia3d.interchangeable`. Es befindet sich also in einem Unterpaket des mobilen KASIA-Clients. Das ist deswegen notwendig, da Android keine Objekte akzeptiert, die außerhalb des eigenen Namensraum erstellt wurden. Dabei spielt es bei der Kompilierung keine Rolle, ob das externe Paket ordnungsgemäß eingebunden wurde. Auf der anderen Seite ist es aber weder beim Server, noch bei der Animation ein Problem, das `.interchangeable`-Paket einzubinden und dessen Klassen zu verwenden.

Für die Kommunikation werden folgende Klassen verwendet (vergleiche Abbildung 5.9):

HelloItem Das *HelloItem* dient der Identifizierung der Kommunikationspartner und wird von einem Client direkt nach dem Verbindungsaufbau gesendet. Es enthält den Name, die eindeutige ID des Clients, seine IP und den *ClientType*, also ob es sich um eine Simulation (*CINEMA_Client*), um ein Smartphone (*MOBILE_CLIENT*), oder um einen Server (*BALLANCE_SERVER*) handelt. Wurde ein *HelloItem* von einem Server empfangen, trägt er die Client-Informationen in seine Routing-Tabelle ein. Als Antwort sendet der Server sein *HelloItem* um sich dem Client gegenüber mit dessen ID und Namen zu identifizieren.

ConnectedCinemas Diese Klasse enthält alle Animationen, bzw. Kinosäle, die an diesem Server derzeit verbunden sind. Die Liste enthält sowohl den jeweiligen Namen, als auch die ID der Animation. Ein *ConnectedCinemas*-Objekt wird vom Server entweder dann an ein mobiles Gerät gesendet, wenn sich dieses zum ersten Mal bei einem Server anmeldet. Oder wenn sich eine neue Animation beim Server registriert wird eine aktualisierte Liste an alle verbundenen mobilen Geräte gesendet.

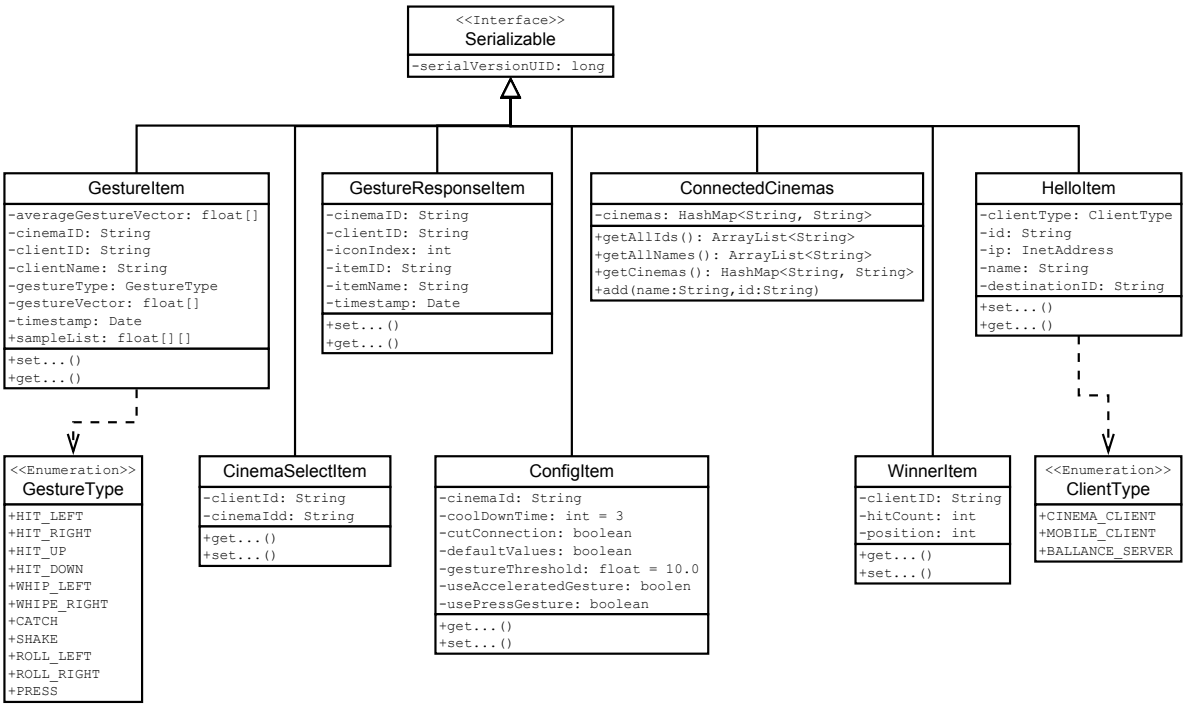

Abbildung 5.9.: Klassen für den Datenaustausch

GestureItem Das *GestureItem* enthält Informationen einer Geste: Einen Durchschnittsvektor (*averageVector*), den Vektor, der beim Fangen erkannt wurde, den Typ der Geste (*GestureType*), die ID des Clients und der Animation (*clientID* und *cinemaID*), den Namen des Clients und den den Erkennungs-Zeitpunkt der Geste. Ein Objekt dieser Klasse wird dann an den Server gesendet, unmittelbar nachdem eine Geste in einem Mobilen Gerät erkannt wurde. Da das Objekt bereits die ID der Animation enthält, wird das Paket vom Server direkt an diese weitergeleitet.

GestureResponseItem Diese Klasse dient als Antwort, wenn durch eine Geste ein Objekt getroffen wurde. Sie enthält die ID der Animation, die ID des gefangenen Objekts (*itemID*), der Index des Bildes (*iconIndex*), das als Repräsentation auf dem mobilen Gerät dient und den Zeitstempel, wann das Objekt gefangen wurde.

CinemaSelectItem Dieses Objekt wird von einem Smartphone-Client gesendet, wenn sich dieses fest mit einem Kino verbinden will. Es wird dann in der Routingtabelle des Servers eingetragen, sodass alle Clients einer Animation auf einmal über Veränderungen benachrichtigt werden können.

ConfigItem Dieses Item dient der Animation als Fernbedienung für seine verbundenen Smartphone-Clients. Über dieses Objekt kann dem Smartphone-Client mitgeteilt werden, welche Gesten-Art akzeptiert werden soll und wie lange die Abkühlzeit dauern soll. Zudem kann über ein „KillItem“, also einem *ConfigItem* mit der Option *cutConnection* alle verbundenen Clients zu veranlassen, die Verbindung zu trennen.

WinnerItem Diese Objekte enthalten nur die ID des Clients, der das Spiel gewonnen hat. Zudem kann das Objekt dem Nutzer mitteilen, auf welchem Platz der Rangliste er sich gerade befindet und wie viele Objekte er gefangen hat.

5.11.1. Abläufe:

In diesem Abschnitt werden die Protokolle beschrieben, die für den Datenaustausch verantwortlich sind.

Verbindungsaufbau: Animations-Client zu Server

Vorbedingung: Der Server wurde gestartet und horcht auf einen vorgegebenen Port (Beispiel: 42424).

1. In einer Animation wird eine TCP/IP-Verbindung zu diesem Server aufgebaut (Drücken von *Connect* / Verbindungsaufbau beim Start).
2. Der Server erzeugt ein *Connection*-Objekt und lagert die Verbindung dort in einen eigenen Thread aus.
3. Die Animation sendet sofort ein *HelloItem*.
4. Der Server trägt die Animation in die Routing-Tabelle ein.
5. Der Server sendet ein *HelloItem* mit seinen Daten zurück an die Animation.
6. Der Server sendet an alle verbundenen Clients die aktualisierte Liste von erreichbaren Animationen (*ConnectedCineams*-Objekt).
7. Die Verbindung bleibt bestehen.

Nachbedingung: Der Animations-Client ist mit dem Server verbunden. Der Server hält die Daten des Animations-Clients in der Routing-Tabelle.

Verbindungsaufbau: Mobiler Kasia3D-Client zu Server

Vorbedingung: Der Server wurde gestartet und horcht auf einen vorgegebenen Port (Beispiel: 42424).

1. Vom mobilen Client wird eine TCP/IP-Verbindung zum Server aufgebaut.
2. Der Server erzeugt ein *Connection*-Objekt und lagert die Verbindung dort in einen eigenen Thread aus.
3. Der mobile Client sendet ein *HelloItem*.
4. Der Server trägt den Client in die Routing-Tabelle ein.
5. Der Server sendet ein *HelloItem* mit seinen Daten zurück an den Client zurück.

6. Der Server sendet an diesen Client eine Liste von erreichbaren Animationen (*ConnectedCineams*-Objekt).
7. Die Verbindung bleibt bestehen.

Nachbedingung: Der mobile Client ist mit dem Server verbunden. Der Server hält die Daten des mobilen Clients in der Routing-Tabelle. Der Benutzer des mobilen Clients hat nun die Möglichkeit, ein Kino auszuwählen.

5.11.2. Verteilen von Konfigurations-Daten: Animations-Client zu Server und weiter zu mobilen Clients

Vorbedingung: Der Animations-Client ist mit dem Server verbunden. Der Benutzer des Animations-Clients hat die Animation und alle Einstellungen für den mobilen Client vorgenommen.

1. Der Benutzer des Animations-Clients startet die Animation.
2. Der Animations-Client sendet ein *ConfigItem* an den Server.
3. Der Server speichert das *ConfigItem* in dessen Routing-Tabelle.
4. Der Server prüft in der Routing-Tabelle, welche mobilen Clients sich mit diesem Animations-Client verbunden haben.
5. Der Server sendet an alle mit dem Animations-Client verbundenen mobilen Clients das *ConfigItem*.
6. Die mobilen Clients ändern ihre Einstellungen entsprechend dem *ConfigItem*.

Nachbedingung: Alle mit dem Animations-Client über den Server verbundenen mobilen Client haben nun die selben Einstellung.

5.11.3. Auswahl eines Kinos: Mobiler Kasia3D-Client

Vorbedingung: Der mobile Client hat sich bereits beim Server authentifiziert und der Server hat dem mobilen Client eine Liste mit Animations-Clients gesendet.

1. Der Benutzer wählt aus der Liste einen Animations-Client aus.
2. Der Mobile Client sendet ein *CinemaSelectItem* an den Server, welches die ID des mobilen Clients enthält und die ID des Animations-Clients.
3. Der Server speichert das *CinemaSelectItem* im Routing-Tabellen-Eintrag des Clients ab.
4. Der Server sucht in der Routing-Tabelle den entsprechenden Eintrag des Animations-Clients und leitet, sofern vorhanden, das zum Animations-Client gehörende *ConfigItem* an den mobilen Client weiter.
5. Der mobile Client ändert seine Einstellung entsprechend dem *ConfigItem*.

Nachbedingung: Der mobile Client ist nun mit dem ausgewählten Animations-Client verbunden und hat die entsprechenden Einstellungen vorgenommen.

Schlagen eines Objekts - gültige Geste und erfolgreicher Fang

Vorbedingung: Sowohl der mobile Client, als auch der Animations-Client sind am Server verbunden, der mobile Client hat den Animations-Client ausgewählt und die Einstellungen wurden untereinander ausgetauscht.

1. Der Benutzer des mobilen Clients führt eine gültige Geste aus (Hand oder Press).
2. Diese wird zusammen mit den Sensor-Daten der Geste als *GestureItem* an den Server gesendet. Dieser sucht aus der Routing-Tabelle den Eintrag für den Animations-Client und leitet diesem das *GestureItem* weiter.
3. Der Animations-Client überprüft, ob der mobile Client zuvor schon eine Geste gesendet hat. Ist dies der Fall wird ein neuer Eintrag in der Datenbank angelegt.
4. Der Animations-Client schreibt die Gesten-Daten in die Datenbank und prüft, ob ein Objekt in fangnähe ist. Das Ergebnis ist positiv.
5. Der Animations-Client animiert das getroffene Objekt entsprechend der Einstellung.
6. Der Animations-Client erstellt ein *GestureResponseItem* mit der Ziel-ID und dem Bild-Index für die Repräsentation des gefangenen Objekts für das Display des mobilen Clients und sendet dieses weiter an den Server.
7. Der Server sucht in der Routing-Tabelle den Eintrag des mobilen Clients und leitet ihm das *GestureResponseItem* weiter.
8. Der mobile Client empfängt das *GestureResponseItem* und benachrichtigt den Benutzer entsprechend durch Ton, Vibration und dem Bild-Index entsprechenden Abbildung des gefangenen Objekts.

Nachbedingung: Sowohl Animations-Client als auch mobiler Client haben den Treffer registriert.

Schlagen eines Objekts - ungültige Geste

Eine ungültige Geste entsteht, wenn der Benutzer die Geste ausführt, obwohl er sich noch in der Abklingzeit befindet. Diese Gesten werden nur aus statistischen Gründen gesendet. Der Ablauf hier ist dem der gültigen Geste sehr ähnlich, allerdings wird die Geste als *tooEarly* markiert und vom Animations-Client nur in die Datenbank geschrieben, ohne zu prüfen, ob ein Objekt in Reichweite war.

Schlagen eines Objekts - gültige Geste und erfolgloser Fang

Der Ablauf ist der selbe wie beim erfolgreichen Fang. Allerdings wird nicht auf die Geste reagiert und auch kein Objekt animiert.

Spielende, Benachrichtigung des Gewinners

Vorbedingung: Alle Objekte wurden Animiert und das Spiel ist vorbei.

1. Der Animations-Client ermittelt aus seinem Speicherstand den Gewinner.
2. Der Animations-Client erzeugt für den Gewinner ein *WinnerItem*, das die ID des gewonnenen mobilen Client enthält.
3. Der Animations-Client sendet das *WinnerItem* an den Server.
4. Der Server sucht in der Tabelle nach dem Eintrag des mobilen Clients und sendet ihm das *WinnerItem* weiter.
5. Der mobile Client, der das *WinnerItem* empfängt benachrichtigt den Benutzer entsprechend.
6. Der Animations-Client erzeugt ein „Kill“Item, ein *ConfigItem* mit der Option *cutConnection=true* und sendet dieses an den Server.
7. Der Server sucht in der Routing-Tabelle nach allen mit dem Animations-Client verbundenen mobilen Clients und leitet ihnen das *ConfigItem* weiter.
8. Alle mobilen Clients, die das *ConfigItem* empfangen trennen ihre Verbindung und schalten ihre Beschleunigungssensoren aus.
9. Der Animations-Client fügt letzte Einträge über die Animation in die Datenbank und beendet sich.

Nachbedingung: Alle mobilen Clients haben die Verbindung unterbrochen und der Animations-Client ist beendet.

6. Benutzerstudie

Um die Verwendbarkeit des Systems zu evaluieren wurde eine Benutzerstudie durchgeführt. Für den Versuch sind zwei Gesten definiert, eine Hand- und eine Touch-Geste, mit denen und der Hilfe eines Smartphones man ein Fliegendes Objekt fangen konnte. Um anzuzeigen, dass sich das Objekt in der Fangzone befindet, wurd als Indikator das Objekt zum Glühen gebracht oder halbtransparent dargestellt. Als weiteren Hinweis wird eine Ebene in die Szene gelegt, die die Ausmaße der Fangzone hat. Diese Parameter, Gestenart, Objekt-Indikator und Vorhandensein einer Ebene, können in Kombination evaluiert.

In diesem Kapitel wird die Benutzerstudie und deren Ergebnisse vorgestellt. Zunächst wird in Abschnitt 6.1 die angestrebte Kooperation mit einem nahegelegenen Kino vorgestellt. In Abschnitt 6.2 auf der nächsten Seite wird der Aufbau der Studie beschrieben. Die Durchführung und die tatsächlich verwendeten Kombinationen aus Geste, Ebene und Indikator wird in Abschnitt 6.3 auf der nächsten Seite beschrieben. Die Auswertung der Fragebögen und der Datenbank-Messdaten

6.1. Kooperation

Für die Durchführung der Benutzerstudie wurde ein Kooperation mit einem örtlichen Kino angestrebt. Infrage kamen dafür die Kinos im Großraum Stuttgart.

Der am besten geeignete Kandidat waren die Innenstadtkinos in Stuttgart [Inn]. Dabei handelt es sich um einen Zusammenschluss mehrerer Kinos: EM, Metropol, Gloria und Cinema. Durch die Tatsache, dass es sich hierbei nicht nur um ein einziges, sondern gleich um mehrere Kinos handelte und somit die mögliche Auswahl bei 10 Kinosälen mit Kapazitäten zwischen 67 und 530 Sitzplätzen lag, erschien hier eine Kooperation am wahrscheinlichsten. Allerdings wurde nie auf die Anfrage einer Kooperation geantwortet.

Daraufhin wurde versucht eine Kooperation mit dem Traumpalast [Tra] in Esslingen anzustreben. Am 20. März 2013 fand ein erstes Treffen wischen uns und der Kinoleiterin Frau Sandra Walter, dem Vertreter des Marketings, Herrn Andreas Hofmann, und dem Chef-Vorführer statt. Die Kinoleiter waren von dem Projekt gegenüber sehr positiv eingestellt. Jedoch war die Zeit für eine tatsächliche Kooperation sehr weit fortgeschritten, so dass Evaluation in einer realen Kinoumgebung erst kurz nach Abgabe dieser Arbeit erfolgen wird.

		Anz. Objekte	*Zeit zw. Objekten*	*Ebene*	*Indikator*	*Geste*
Probedurchläufe:	Durchlauf 1	42	5, 10*s*	ja	Glühen	Touch
	Durchlauf 2	42	5, 10*s*	nein	Transp.	Hand
Gruppe 1	Durchlauf 1	30	5, 10*s*	ja	Glühen	Touch
	Durchlauf 2	30	5, 10*s*	nein	Transp.	Hand
Gruppe 2	Durchlauf 1	30	5, 10*s*	ja	Glühen	Touch
	Durchlauf 2	30	5, 10*s*	ja	Glühen	Hand
	Durchlauf 3	30	5, 10*s*	nein	Transp.	Touch
	Durchlauf 4	25	5, 10*s*	nein	Glühen	Hand
Gruppe 3	Durchlauf 1	30	5, 10*s*	ja	Transp.	Hand
	Durchlauf 2	30	5, 10*s*	nein	Glühen	Touch
	Durchlauf 3	30	5, 10*s*	nein	Transp.	Touch
	Durchlauf 4	30	5, 10*s*	nein	Glühen	Hand
Gruppe 4	Durchlauf 1	42	5, 10*s*	ja	Transp.	Hand
	Durchlauf 2	30	5, 10*s*	nein	Transp.	Touch
	Durchlauf 3	30	5, 10*s*	nein	Transp.	Hand
	Durchlauf 4	30	5, 10*s*	ja	Glühen	Touch

Tabelle 6.1.: Einstellung von KASIA

6.2. Versuchsaufbau

Die Benutzerstudie fand in den Laborräumen des Gebäudes für Simulation Technology (SimTech) statt. Hier befand sich ein stereoskopischer Projektor (siehe Abschnitt 4.4.2 auf Seite 46), auf dem die Animation ablief. Die Mobiltelefone mit Android-Betriebssystem wurden entweder selbst von den Probanden mitgebracht, oder von der Abteilung für Mensch-Computer Interaktion zur Verfügung gestellt. Auf den privat mitgebrachten Mobiltelefonen wurde die Software via QR-Code heruntergeladen. Auf den Geräten des Instituts war die Software bereits heruntergeladen. Der Abstand der Teilnehmer zur Leinwand betrug 4, 80 Meter, der Abstand des Projektors zur Leinwand 3, 50 Meter. Die Beleuchtung wurde für die Zeit der Durchgänge auf ein Minimum reduziert.

Für die Netzwerkverbindung zwischen Server und den Clients diente ein eigens dafür eingerichtetes Wireless Network, zu dem nur die Probanden Zugang hatten und über das ebenfalls der Server und die Animation erreichbar waren.

6.3. Durchführung

Für die Durchführung der Studie wurden 19 Teilnehmerinnen und Teilnehmer eingeladen. Diese wurden in 4 Gruppen eingeteilt. Die Gruppen 1, 3 und 4 hatten jeweils 5 Teilnehmer, Gruppe 2 hatte 4 Teilnehmer.

Vor der Durchführung wurde jeweils die Datenbank, die der die Messdaten aufgezeichnet werden, geleert. Sobald die Teilnehmer vollzählig anwesend waren, wurde ihnen zuerst der komplette Ablauf erklärt und sie wurden darüber aufgeklärt, dass jederzeit die Möglichkeit besteht, den Versuch zu verlassen. Anschließend wurden die Eingangsfragebögen verteilt, die von den Teilnehmern ausgefüllt wurden.

Nach dem Ausfüllen des ersten Fragebogens wurden auf den privat mitgebrachten Mobiltelefonen über den QR-Code die Kasia3D-Software installiert. Dabei tauchte das Problem auf, dass der Großteil dieser Teilnehmer nicht die dafür notwendige Scannersoftware installiert hatten, die Kasia3D voraussetzt. Eine manuelle Installation war immer erfolgreich.

Nach der Installation und dem erfolgreichen Verbinden zum Kasia-Server wurden die Teilnehmer aufgefordert, die 3D-Brillen aufzusetzen und zu prüfen, ob diese ordnungsgemäß funktionierten.

Im nächsten Schritt wurde die erste Proberunde durchgeführt und auf Probleme der Teilnehmer ohne Erfahrung mit Smartphones eingegangen. Anschließend wurde die zweite Proberunde durchgeführt, welche die jeweils zweite Interaktionsmethode verwendete. Die Parameter der Proberunden waren jedes mal die gleichen (vergleiche Tabelle 6.1 auf der vorherigen Seite). Während den Proberunden wurde den Probanden nicht gesagt, wie sie das System zu bedienen hatten, bzw. welche Geste sie ausführen konnten. Stattdessen mussten sie die Funktion durch eigenes Handeln herausfinden. Erst als einige Zeit vergangen war und einzelne Teilnehmer frustriert zu sein schienen, wurde ihnen indirekte Hinweise gegeben.

Sowohl in den Probe- als auch in den Wertungsrunden betrug die Abklingzeit 3 Sekunden. Das ist die Zeitspanne zwischen zwei Gesten, in der keine neue Geste ausgeführt werden kann. Für die Statistik wurden diese Gesten dennoch in der Datenbank aufgezeichnet und als „ungültig“ markiert.

Nach der zweiten Proberunde wurde dann der erste richtige Durchlauf mit einer der beiden Gestenarten durchgeführt. Wichtig für die Auswahl der Parameter war, dass in jeder Gruppe beide Interaktionsarten zweimal verwendet wurden und dass dass die Indikatoren in unterschiedlichen Kombinationen mit der transparenten Ebene zum Einsatz kam (vergleiche Tabelle 6.1 auf der vorherigen Seite). Nach dem ersten gültigen Durchlauf wurde der erste *Fragebogen zu Interaktionsmethoden* mit dem Hinweis ausgehändigt, dass er sich auf die gerade verwendete Interaktionsmethode bezieht. Nach dem Ausfüllen des Fragebogens wurde der zweite Durchlauf durchgeführt, in dem die zweite Interaktionsmethode verwendet werden sollte. Anschließend wurde der zweite *Fragebogen zu Interaktionsmethoden* von den Teilnehmern ausgefüllt.

Bei der ersten Gruppe gab es noch erhebliche Probleme mit dem Server, da dieser mit den Race Conditions nicht richtig umgehen konnte und sich immer wieder abstürzte. Dieses Problem wurde allerdings erst in den richtigen Durchläufen offenbar, da die Teilnehmer anscheinend in den Testrunden noch sehr zaghaft Gesten ausgeführt haben. Das führte dazu, dass der erste Versuch bereits nach zwei von ursprünglich acht geplanten Konditionen abgebrochen wurde. Das Server-Problem wurde jedoch schnell gelöst. Die Gruppen 2 bis 4 konnten daher alle Durchgänge durchführen.

Die Anzahl der Durchläufe wurde nach der ersten Gruppe von geplanten 8 deswegen generell auf 4 heruntergesetzt, da sich schon bei dieser Gruppe trotz der Probleme abzeichnete, dass schon bei 4 Durchläufen eine Dauer von rund 90 Minuten erreicht werden würde. Eine längere Studie wollte man den Teilnehmern nicht zumuten.

Alle Teilnehmer der vier Gruppen mussten zum Schluss den Abschlussfragebogen ausfüllen. Die meisten Teilnehmer der Gruppe 3 hatten so viel Freude an dem Versuch, dass sie nach Abgabe des *Abschließenden Fragebogens* darauf beharrten, eine weitere Runde zu spielen.

Nach dem Ende des Versuchs wurde jeweils die Datenbank gesichert und die Kommentare der Teilnehmer notiert.

6.4. Ergebnis

6.4.1. Probanden und deren Sehfähigkeit

Die Studie wurden von insgesamt 19 Teilnehmerinnen und Teilnehmern im Alter zwischen 19 und 38 Jahren durchgeführt. Der Altes-Median lag bei 27. Von den Probanden waren 10 männlich, 9 weiblich. 17 bezeichneten sich als technikaffin. 14 Probanden studieren Informatik oder arbeiten in einem informatiknahen Bereich, zwei Probanden studieren Chemie, ein Proband war Zahnarzttechniker, einer Student der Umweltschutztechnik und ein weiterer Diplomingenieur.

Sehhilfen trugen insgesamt 11 Teilnehmer, von denen 3 nach eigener Aussage über kein stereoskopisches Sehen verfügten.

6.4.2. Smartphone-Gebrauch

Auf die Frage, ob sie ein Smartphone mit Touch-Display besäßen, beantworteten 13 Teilnehmer mit „Ja“. 11 von ihnen verfügen darüber hinaus über eine Internet-Flatrate für ihr Mobiltelefon. Bei der Durchführung hatten 3 Probanden, die über kein Smartphone verfügten, Probleme sowohl mit der Menüführung als auch damit das Programm zu bedienen. Diese anfänglichen Probleme legten sich aber meist nach den Proberunden. 12 von den 13 Smartphonebenutzern haben ebenfalls externe „Apps“ installiert, was den Schluss nahelegt, dass diese ihr Smartphone auch sehr intensiv nutzen und sehr gut mit dem Umgang vertraut sind. Als externe Applikationen werden hier Programme bezeichnet, die nicht im Lieferumfang des Geräts enthalten waren. Ein Benutzer, auf den das nicht zutraf, gab an, nicht technikaffin zu sein.

mehrmals pro Woche	0
etwa einmal pro Woche	0
mehrmals im Monat	4
mehrmals im Jahr	12
seltener	3
nie	0

Tabelle 6.2.: Kinobesuch: Häufigkeit (absolute Häufigkeiten vgl. Einstiegsfragebogen Fragen 3.1 im Anhang B)

	Mögen Sie 3D-Filme?	Wie angenehm ist Ihnen das Betrachten von stereoskopischem Inhalt?
Min	1	1
Max	5	5
Durchschnitt	2,53	2,84
Median	2	3

Tabelle 6.3.: Kinobesuch: 3D-Filme, 1 sehr gerne bzw. sehr angenehm, 5 äußerst ungerne bzw. äußerst unangenehm (vgl. Einstiegsfragebogen Fragen 3.3 und 3.4 im Anhang B)

6.4.3. Kinobesuch

Unter den Teilnehmern gab es keine ambitionierten Cineasten, die mindestens einmal in der Woche ins Kino gehen. 4 Teilnehmer gaben an mehrmals im Monat ins Kino zu gehen, nur 3 Teilnehmer gaben an seltener als einmal im Jahr ins Kino zu gehen. Der Großteil (12 Teilnehmer) geht ab und zu ins Kino, also mehrmals im Jahr, aber nicht notwendigerweise jeden Monat (vergleiche Tabelle 6.2).

Alle Teilnehmer haben schon einmal einen stereoskopischen Spielfilm im Kino gesehen. Die Vorlieben für 3D-Filme sind sehr unterschiedlich. 6 Probanden gaben an, 3D-Filme in jedem Fall, der 2D-Fassung vorzuziehen, sofern es eine Wahl gibt. Die 2*D*-Fassung bevorzugten 4 Teilnehmer. Von diesen 4 verfügten zwei über kein stereoskopisches Sehen und ein Teilnehmer hatte massive Probleme mit dem Sehen von stereoskopischem Inhalt. Die meisten Teilnehmer, 9 haben keine besondere Vorliebe für 2D- oder 3D-Filme. Diese machen es vom jeweiligen Film abhängig, ob sie die 2D- oder die 3D-Fassung ansehen. Ein Teilnehmer gab an, in seltenen Fällen Kopfschmerzen zu bekommen und über kein stereoskopisches Sehen zu verfügen. Dennoch ziehe er 3D-Filme den 2D-Fassungen vor.

Auf die Frage „Mögen Sie 3D-Filme“ antworteten die Teilnehmer auf einer Skala von 1 (sehr gerne) bis 5 (sehr ungerne) im Durchschnitt mit 2,53 (Der Median lag bei 2). Die Frage „Wie angenehm ist Ihnen das Betrachten von stereoskopischem Inhalt?“ konnte auf einer Skala von 1 (sehr angenehm) bis 5 (äußerst unangenehm) beantwortet werden. Im Mittel wurde mit einem Wert von 2,84 geantwortet, der Median lag bei 3. 7 Teilnehmer gaben an,

beim Betrachten von dreidimensionalen Inhalten Probleme zu haben. Von diesen gaben 3 an, Kopfschmerzen zu bekommen, 2 Teilnehmern wird schwindelig, und für 2 ist das Betrachten anstrengend für die Augen (vergleiche Tabelle 6.3 auf der vorherigen Seite).

6.4.4. Stereoskopische Anwendungen

Auf die Frage, „Abgesehen von 3D-Filmen, hatten Sie schon einmal mit einer stereoskopischen Anwendung zu tun?“ antwortete über die Hälfte (11 Teilnehmer) mit nein. Von den anderen gaben derzeitige Informatikstudenten oder Alumni an, Präsentationen an einer Powerwall (3 Teilnehmer) gesehen zu haben oder schon selbst an einer Studie zu 3D-Displays teilgenommen zu haben. 2 Teilnehmer haben bereits Computerspiele mit 3D-Brille gespielt, 1 Teilnehmer besitzt Bücher der Reihe „Das Magische Auge“ (vergleiche Abschnitt 2.3.2 auf Seite 21) und 2 Teilnehmer gaben an, Bücher und Videos mit einer Farbanaglyph-Brille betrachtet zu haben.

6.4.5. Befragung zu den jeweiligen Gesten

In den zählenden Durchläufen wurde von den Probanden je Gestenart ein „Fragebogen zur Interaktion“ ausgefüllt. Diese wurden dann ausgeteilt, als die jeweilige Geste zum ersten Mal in einem gültigen Durchlauf verwendet wurden. Dabei wurden für die Fragen 1.1 und 2 (vergleiche Fragebogen B auf Seite 99) standardisierte Fragebögen verwendet. Frage 1.1 war die Adaption des *Questionair For User Interaction Satisfaction* (QUIS) [CDN88] und Frage 2 die Übersetzung der *Simple Usability Scale* (SUS) [Bro96]. Am Ende des gesamten „Fragebogen zur Interaktion“ waren noch zusätzliche Fragen zur verwendeten Geste, die selbst erstellt wurden.

Die aggregierten Ergebnisse der Fragen 1.1 und 2 befinden sich in den Tabellen 6.4 (Frage 1.1 - SUS) und 6.5 (Frage 2 - QUIS). Die Ergebnisse der Fragebögen nach Gruppen aufgeschlüsselt befinden sich im Anhang B. Für die Antworten des QUIS-Teils sei angemerkt, dass die erste Teilfrage im Fragebogen verdreht abgebildet war („wunderbar“ . . . „furchtbar“, anstatt „furchtbar“ . . . „wunderbar). Für die der Auswertung wurde die Frage in die richtige Reihenfolge gebracht und der angegebene Wert vom Wert 9 subtrahiert. Dadurch werden die Ergebnisse vergleichbar mit denen der anderen Fragen.

Es zeigt sich bei den Fragen 1.1 und 2, dass Touch-Gesten generell besser abschneiden als Hand-Gesten, wobei die Ergebnisse von Gruppe zu Gruppe sehr unterschiedlich sind.

6.4.6. Interaktion

Wie schon aus den Fragebögen zur Interaktion, die in Abschnitt 6.4.5 diskutiert wurden, hervor ging, schnitt insgesamt die Touch-Geste leicht besser ab. Das zeigte sich auch in der Frage: „Welche Art der Bedienung ziehen Sie vor?“ Für die Touch-Geste entschieden sich 11 Teilnehmer, 8 für die Hand-Geste. In diesem Fall gab es niemanden ohne Vorliebe.

		Gruppe 1	Gruppe 2	Gruppe 3	Gruppe 4	Gesamt
Hand-Geste	Minimum	22,5	0,0	12,5	20,0	0,0
	Maximum	37,5	30,5	52,0	82,5	82,0
	Durchschnitt	29,0	18,75	28,0	55,0	33,34
	Median	27,5	20,0	27,5	62,5	27,5
Touch-Geste	Minimum	27,5	15,0	12,5	7,5	7,5
	Maximum	45,0	27,5	55,0	62,5	62,5
	Durchschnitt	33,5	21,5	25,5	33,0	28,68
	Median	32,5	20,0	17,5	40,0	27,5

Tabelle 6.4.: SUS: Gruppen verglichen (Wertebereich: 0 bis 100, wobei 0 den besten Wert darstellt), vergleiche Fragebogen in Anhang B, Frage 1.1

		Durchschnitt	σ	Minimum	Maximum
Hand-Geste	furchtbar / wunderbar	4,84	2,43	0	9
	schwierig / leicht	4,58	2,64	0	9
	frustrierend / angemessene Leistung	4,47	2,37	0	9
	langweilig / anregend	6,11	1,83	2	9
	unflexibel / flexibel	4,89	2,45	0	9
Touch-Geste	furchtbar / wunderbar	5,42	2,21	2	9
	schwierig / leicht	4,89	1,92	1	8
	frustrierend / angemessene Leistung	4,95	2,30	1	9
	langweilig / anregend	7,00	1,52	3	9
	unflexibel / flexibel	4,89	1,86	2	8

Tabelle 6.5.: QUIS: Gesamtüberblick (Legende: Werte sind von 0 bis 9, ein Wert von 9 ist am positivsten, σ bezeichnet die Standardabweichung), vergleiche Fragebogen in Anhang B, Frage 2

	natürlich	einfach	kompliziert
Hand-Geste	6	9	4
Touch-Geste	4	12	3

Tabelle 6.6.: User Experience: Wie fanden Sie die Steuerung?

	ja	nein
Hand-Geste	11	8
Touch-Geste	16	3

Tabelle 6.7.: User Experience: Könnten Sie sich vorstellen, ein solches System zu nutzen?

Die Begründungen zur Wahl der Geste fielen sehr unterschiedlich aus. Teilnehmer, die sich für die Touch-Geste entschieden haben, gaben 3 an, dass diese Interaktionsart einfacher zu bedienen sei. 4 Teilnehmer sind der Meinung, dass Touch-Gesten weniger anstrengend sind und 3 gaben an, dass sie mit der Touch-Geste sehr viel besser zielen konnten und diese auch sicherer auslöste. Für die Teilnehmer, die sich für die Hand-Geste entschieden haben, spielte das Erlebnis und die Herausforderung der Interaktion eine große Rolle. Mit der Handgeste sei „eine genauere Interaktion notwendig und möglich“, „macht mehr Spaß, beim Spielen sich zu bewegen“, Handgesten seien „intuitiver“ und „angenehmer und besser zu steuern“. Ein Teilnehmer gab an, bei Touch-Gesten zu verkrampfen, was bei Hand-Gesten nicht der Fall war. Und ein Teilnehmer hob hervor, dass Hand-Gesten auch dann funktionierten, wenn man auf Knöpfe kommt und die Client-Anwendung verlässt.

6.4.7. Indikatoren

Von den Teilnehmern wurde das „Glühen“ am eindeutigsten wahrgenommen. Auf die Frage „Welchen Indikator, dass ein Objekt fangbar ist ziehen Sie vor?“ antworteten 11 Teilnehmer mit „Glühendes Objekt“, 2 Teilnehmer mit „Transparentes Objekt“ und 6 Teilnehmer hatten keine Vorlieben.

Im Diskussionsteil zu dieser Frage wurde von den Teilnehmern angegeben, dass das Glühen eines Objekts besser wahrnehmbar ist. Nur ein Teilnehmer gab an, dass die Transparenz die „eindeutig wahrnehmbarere Veränderung“ sei. Ein anderer wiederum schrieb, dass „man bei transparenten Objekten denkt, es ist schon vorbei“. Die semitransparente Ebene wurde kaum erwähnt. Auf die direkte Frage an die Teilnehmer wurde geantwortet, dass es nicht abschätzbar war, ob ein Objekt sich nun über der Ebene befinde oder nicht. Zudem seien die Ränder der Ebene doppelt wahrnehmbar gewesen, was den Eindruck sehr gestört hat.

Die Teilnehmer hatten weitere Ideen, wie man die Fangbarkeit von Objekten signalisieren könnte. Alleine 7 Probanden schlugen ein schnelleres „Blinken“ oder sonstige Farbänderungen des Objekts vor. Der Übergang beim Glühen sei zu langsam gewesen und der Beginn daher nicht erkennbar. 3 Teilnehmer wünschen sich einen roten Kreis um das Objekt und ein weiterer schlug ein Glühen um das Objekt vor, wie es bei vielen Massively Multiplayer Online Games (MMOG) der Fall ist. Ein Teilnehmer schlug vor, das Mobiltelefon im richtigen Moment vibrieren zu lassen, ein anderer das Objekt selbst optisch zu vibrieren. Ein Proband hatte die Idee, ein Objekt zu rotieren, ein anderer eine Schrift einzubauen, die den Abstand zur Fangzone enthält oder gar einen roten Pfeil über dem Objekt, der als Indikator dient. Die Änderung der Größe wurde von 2 Teilnehmern vorgeschlagen.

6.4.8. Durchgänge und Hintergrund

Die Dauer der Durchgänge war für die meisten Teilnehmer (14) genau richtig. Im Mittel lag die Dauer bei $2,63$ wobei 1 „viel zu“ kurz und 5 „viel zu lange“ bedeutet. Der Hintergrund

	Minimum	Maximum	Durchschnitt	Median
Würde Sie ein solches System vor dem Film stören? „ja, sehr“ (1) / „nein, gar nicht“ (5)	1	5	4,16	5
Ein Kino, das ein solches System verwendet wird würde ich... „meiden“ (1) / „häufiger besuchen“ (5)	1	5	3,39	3

Tabelle 6.8.: Verwendung im Kino:

wurde von vielen nicht besonders wahrgenommen. 1 Teilnehmer fühlte sich von dem Hintergrund gestört, 3 weitere abgelenkt und 6 sogar motiviert. 7 Teilnehmer gaben allerdings eine vierte Antwortmöglichkeit an, die nicht vorgesehen war. Ihnen war der Hintergrund „egal“. Ein Teilnehmer gab den Kommentar, es handele sich eben um einen gewöhnlichen Hintergrund. 2 Teilnehmer machten keine Angabe.

6.4.9. Nutzung und Benutzbarkeit

Auf die Frage, ob sie sie ein solches System in einer realen Kinoumgebung nutzen würden antworteten 8 mit „ja, gerne“, 8 mit „hin und wieder“ und 3 mit „nein, nie“. Für den Fall, dass sie in ein Kino kommen, in dem ein solches System zum Einsatz kommt, würden sich die meisten davon kaum gestört fühlen. Im Durchschnitt stimmten die Teilnehmer mit einem Wert von 4,15 (Median 5), wobei 1 „es würde mich sehr stören“ und 5 „nein, überhaupt nicht“ entspricht. Ob die Teilnehmer ein solches Kino eher meiden (Wert 1) oder häufiger besuchen würden (Wert 5) wurde sehr neutral beantwortet: Der Durchschnitt lag bei 3,39, der Median bei 3. Nur ein Teilnehmer würde tatsächlich ein Kino meiden, in dem ein solches System eingesetzt würde.

6.4.10. Messwerte der Datenbank

Während der Durchläufe wurden alle für die Animation und für das Fangen der Objekte relevanten Daten in einer Datenbank festzuhalten. Neben dem Zeitpunkt, an dem ein fliegendes Objekt in die „Fangzone“ eingetreten ist und dem, an dem es diese wieder verlassen hat, wurden auch sämtliche Gesten registriert. Diese umfassten die Art der Geste, den genauen Zeitpunkt zu dem die Geste in registriert wurde, und auch, ob die Geste noch während der Abklingzeit getätigt wurde und somit ungültig war. In Tabelle 6.9 auf der nächsten Seite befinden sich die aggregierten Daten über alle Gruppen. In den Spalten „Hand“ und „Touch“ sind die empfangenen Gesten aufsummiert, die während der Durchgänge verwendet, in denen eine Hand- oder eine Touch-Geste verwendet wurde. Analog dazu befinden sich in den Spalten „Glühen“ und „Transparenz“ die Summen der Gesten, die während der Durchgänge empfangen wurden, in denen der jeweilige Indikator zum Einsatz kam. Σ bezeichnet die Summe der Gesten aller Durchläufe. Die Zeile „Anzahl gültige Gesten“

	Hand	*Touch*	*Glühen*	*Transp.*	Σ
Fangbare Objekte	217	210	175	252	427
Anz. Gesten	2295	1551	1209	2637	3846
Anz. gültige Gesten	1151	817	666	1302	1968
Anz. Treffer	268	201	158	311	469
Verh. $\frac{gültigeGesten}{Gesten}$	0,502	0,527	0,551	0,494	0,512
Verh. $\frac{Treffer}{gültigeGesten}$	0,233	0,246	0,237	0,239	0,238

Tabelle 6.9.: Aggregierte Datenbankdaten

gibt die Zahl der Gesten, die außerhalb der Abklingzeit ausgeführt wurden und „Anzahl der Treffer“ die Zahl der Gesten, durch die ein fliegendes Objekt gefangen wurde. Die Verhältnisse geben Aufschluss, wie groß der Anteil von gültigen zu allen Gesten, bzw. der Anteil von getroffenen Gesten an allen gültigen Gesten ist. Nach den einzelnen Gruppen aufgeschlüsselte Tabellen befinden sich im Anhang B.3 auf Seite 108.

6.5. Interpretation

Das System Kasia wurde erstellt, um zu prüfen, ob Interaktionen, wie das Fangen von Objekten aus der Luft mit Hilfe eines Smartphones von Benutzern akzeptiert werden oder nicht. Zu diesem Zwecke wurden neben der Hand-Geste, also über eine einfache Handbewegung Objekte fangen kann, eine Touch-Geste angeboten, die durch einfaches Drücken eines Buttons auf dem Touch-Screen, ein Objekt fangen kann. Die Eingangsthese ging davon aus, dass Handgesten, da sie im allgemeinen als natürlicher eingeschätzt werden als eine Touch-Geste und daher die Hand-Gesten bevorzugt würden.

Im Eingangsfragebogen (vergleiche Abschnitte 6.4.1 bis 6.4.4) wurden die Probanden nach ihrer Sehfähigkeit befragt und ebenso, ob sie imstande sind, die Umgebung stereoskopisch wahrzunehmen. Die Auswertung ergab hierbei keine signifikanten Zusammenhänge zwischen der Sehfähigkeit und den Ergebnissen der Messdaten aus der Datenbank. Es zeigte sich durch Beobachtung eine geringe Unbeholfenheit der Teilnehmer, die keine Erfahrung mit Smartphones hatten. In der Auswertung der Fragebögen und der Gestendaten zeigten sich aber keine signifikant schlechteren Werte als bei anderen Teilnehmern.

Bei der Analyse der Datenmessungen (vergleiche Tabelle 6.9) zeigt sich eine signifikant höhere Anzahl an ausgeführten Hand- als Touch-Gesten. Das kann dadurch erklärt werden, dass die Teilnehmer bei Hand-Gesten über die ganze Dauer des Durchgangs versucht haben ein Objekt zu fangen. Selbst dann wenn es noch nicht in Reichweite war und auch dann wenn die Abklingzeit noch nicht abgelaufen war. Bei Touch-Gesten wird meist eher darauf geachtet, dass man erst dann auf den Knopf drückt, wenn man sicher ist das Objekt auch zu fangen. Was die Trefferwahrscheinlichkeit betrifft, so sind die Unterschiede Minimal: In der aggregierten Auflistung über alle Gesten und alle Teilnehmer, beträgt der maximale Unterschied 1,3% von Durchläufen, in denen eine Touch-Geste zu jenen, in denen eine

Hand-Geste verwendet wurde. Das Verhältnis von ausgeführten zu gültigen Gesten ist leicht aussagekräftiger. Zwar ist der Unterschied zwischen den Gestenarten eher gering, doch liegt er zwischen den Indikatoren bei 5,7%. Das deckt sich auch mit der Aussage, dass die Transparenz meist viel schlechter wahrgenommen wurde als das Glühen.

Am deutlichsten sind jedoch die *QUIS*- und *SUS*-Werte und die Fragen, welche Interaktionsmöglichkeit die Teilnehmer bevorzugen würden. In allen beantworteten Fragen wird die Touch-Geste besser akzeptiert als die Hand-Geste. Das hat unterschiedliche Gründe. In Frage 1.3 des Abschlussfragebogens konnten die Teilnehmer ihre Antwort begründen. Zwar meinten einige, dass die Interaktion mit einer Handgeste „interessanter“ und „intuitiver“ sei als bei einer Touch-Geste. Auch wurde angeführt, dass Handgesten „mehr Spaß“ machen würden. Allerdings scheinen die Gründe für die Touch-Geste zu überwiegen: Zwei Teilnehmer berichteten, dass Touch-Gesten „weniger anstrengend“ als Handgesten seien und ebenfalls zwei meinen, dass man mit Touch-Gesten besser zielen konnte.

Die Indikatoren wurden generell von den Teilnehmern schlecht erkannt. Einige Teilnehmer stellten nach der Studie die Frage, was man hätte sehen sollen. Dennoch wurde das Glühen eher akzeptiert als dass das Objekt plötzlich transparent wurde. In Frage 1.4 konnten die Teilnehmer Vorschläge machen, wie man die Indikatoren verbessern könne. Auch dort wurde ein „Blinken“ und eine „Farbänderung“ vorgeschlagen. Nur solle diese Änderung abrupter eintreten, als dass es in der Durchführung der Studie geschah.

Die Transparente Ebene wurde meist als störend empfunden, da sie nicht den gewünschten Bezug zum Objekt herstellen konnte und in der Animation stereoskopisch undeutlich angezeigt wurde.

Obwohl die Hand-Geste nicht die Erwartung erfüllt hat, wird das System als ganzes dennoch sehr positiv aufgenommen. Zumindest 16 Teilnehmer würden das System zumindest hin und wieder nutzen, 8 Teilnehmer von ihnen sogar „gerne“. 3 Teilnehmer würden so ein System nie verwenden und 2 von diesen würden sich auch davon gestört fühlen und Kinos sogar komplett meiden, die ein solches Angebot installiert hätten.

6.6. Fazit

Trotz der Arbeitsthese, dass Hand-Gesten natürlicher sind als Handgesten und daher auch eher akzeptiert werden, stellte sich in der Benutzerstudie heraus, dass die Vorlieben der Benutzer deutlich hin zur Touch-Geste verlagert sind. Obwohl die Hand-Geste nicht die Erwartungen erfüllt hat, ist ein Großteil der Benutzer mit dem System zufrieden. In einer zukünftigen Entwicklung könnte man die Indikatoren, die von den Benutzern vorgeschlagen wurden in das System integrieren und die Schwachstellen bei der Gestenerkennung etwas beseitigen, was in den Fragebogen bereits als Kritik zum Ausdruck kam. Dadurch könnte eine Hand-Geste gezielter eingesetzt werden. Nach diesem Ergebnis sollte die Touch-Geste in jedem Fall enthalten bleiben und eventuell gleichzeitig oder alternativ zur Fanggeste verwendbar sein.

7. Zusammenfassung und Ausblick

Am Ende der Arbeit soll diese noch einmal zusammengefasst und ein Ausblick auf zukünftige Entwicklungen gewagt werden.

7.1. Zusammenfassung

In dieser Arbeit wurde die Möglichkeit vorgestellt, mit Hilfe von Smartphones und Gesten mit einer stereoskopischen Animation zu interagieren. Für dieses Projekt wurde ein Framework aus drei Bestandteilen angefertigt.

- Kasia3D, eine auf dem Betriebssystem Android basierende Smartphone-Applikation, die Gesten mit Hilfe von Beschleunigungssensoren erkennt.
- Kasia3D*Engine*, in der Programmiersprache Java implementiertes Programm, mit dem eine stereoskopische Animation erstellt und deren Aussehen und Verhalten beeinflusst werden kann. Dieses Programm prüft, ob sich Objekte in einer Fangzone befinden und eine ausgeführte Geste das Objekt tatsächlich auch gefangen hat.
- Kasia3D*Server*, ein Programm, das den Datenaustausch zwischen den Client-Programmen regelt.

Im Zentrum der Arbeit stand die These, dass ein System, das natürliche Interaktionsmethoden wie Hand-Gesten nutzt, von Benutzern eher angenommen werden als einfache Touch-Gesten.

In einer Benutzerstudie wurde das System von 19 Teilnehmern in 4 Gruppen evaluiert. Dabei wurde eine Hand-Geste einer Touch-Geste gegenübergestellt. Als Indikator, dass die Objekte in Fangreichweite waren, wurde entweder ein Glühen oder eine Transparenz der Objekte verwendet. Diese Indikatoren wurden Kombination mit einer semitransparenten Ebene evaluiert. Nach der Auswertung der Fragebögen und der aufgezeichneten Messdaten stellte sich heraus, dass Teilnehmer mit Touch-Gesten besser zurecht kommen und sie diese den Hand-Gesten vorziehen. Ein weiteres Ergebnis war, dass als Indikator ein Glühen oder sogar ein offensichtliches Blinken viel besser erkennbar ist als eine Transparenz des Objekts. Die verwendete semitransparente Ebene hatte keinen Einfluss auf das Ergebnis und wirkte eher ablenkend auf die Teilnehmer, als dass sie ihnen half.

7.2. Ausblick

Trotz großem Implementieraufwand hatte das System bis zur Benutzerstudie noch einige Softwarefehler, die erst bei vielen Benutzern offenkundig wurden. So hatte der Server Probleme mit der gleichzeitigen Verarbeitung der Daten. Zudem zeigten sich kleine Mängel, was das Gewinnkriterium betrifft. Dennoch wurde das System von allen Probanden sehr positiv aufgenommen. Eine Gruppe wollte beispielsweise auch nach dem Ende der Studie noch eine zusätzliche Runde spielen. Aus diesem Grunde wird das System auf der Grundlage der Evaluation verbessert, die Fehler beseitigt und in einer Kooperation mit dem Traumpalast Esslingen mit einem breiten Publikum unter realen Bedingungen getestet.

Was die Forschung betrifft gibt es noch viele Möglichkeiten. In einer Weiterentwicklung sollte es Möglichkeiten geben, gezielte Handbewegungen zu verwenden um ein Objekt zu fangen. Durch diese Herausforderung könnte sich der Anreiz vergrößern so ein System auch aktiv zu nutzen. Auf der anderen Seite könnte sich genau das auch frustrierend auf die Benutzer auswirken, denen diese Gesten dann zu kompliziert erscheinen. In jedem Fall muss die Erkennungsgeschwindigkeit erhöht werden, damit keine Latenz zwischen Ausführung der Geste und dem akustischen Feedback wahrgenommen wird. Im nächsten Schritt sollten die Indikatoren eindeutiger gemacht werden, so dass sie von den Benutzern auch richtig erkannt werden. Ein deutliches Blinken oder ein Ring um das Objekt, wie es von Teilnehmern der Benutzerstudie vorgeschlagen wurde, sind mögliche Optionen.

Ob ein solches System tatsächlich in Deutschland zum Einsatz kommt, ist zum heutigen Tag noch fraglich. In einem Gespräch mit den Kinobetreibern in Stuttgart und in Esslingen wurden Bedenken laut, da Mobiltelefone in Kinos aus rechtlichen Gründen nicht erlaubt sind. Eine Kooperation war auch nur dadurch möglich, da von Anfang an gesagt wurde, dass es sich um ein wissenschaftliches Projekt handele, das sich vor dem Spielfilm abspielt. Das Projekt aus Kanada (siehe Abschnitt 3.7) gibt aber Hoffnung, dass solche interaktiven Systeme in Zukunft auch in Deutschland Zuspruch finden werden.

Sofern Kinobetreiber, Werbe- oder Verleihfirmen sich darauf einlassen könnte man sich auch vorstellen, das Smartphone über den ganzen Film über aktiv zu lassen und sich während eines 3D-Films Objekte, wie beispielsweise ein Eis, den „Einen Ring“, etc. aus der Hand des Protagonisten zu „greifen“ und das Gefangene am Ende der Vorstellung an der Kinokasse einzulösen.

Von den Kinobetreibern in Esslingen kam zudem die Idee, dass man dieses System auch einfach dafür einsetzen könnte, Snacks durch fangen einzukaufen und diese dann an der Snack-Theke einfach nur einzulösen.

Die Möglichkeit, mit der Hilfe eines Smartphones Dinge zu steuern wird bei weitem noch nicht ausgereizt. Abgesehen von einem Kino gibt es noch viele Anwendungsgebiete, in denen man eine Interaktionen mit dem Smartphone realisieren könnte. Was sich allerdings durchsetzen wird ist zu diesem Zeitpunkt noch offen.

7.3. Danksagung

An dieser Stelle möchte ich mich bei meinen ganzen Weggefährten bedanken, die mich in den letzten Jahren und vor allem Wochen tatkräftig unterstützt haben. Einen ganz großes Dankeschön geht an meine Lektoren und Motivationsgebern Wiltrud Kessler, Christine Keller, Friederike Rautenberg, Claudia und Hanno Wagner, Svenja Apfelbach, Kaja Spinnler, Veronika Wille, Holger Rode und natürlich auch an Sören Brunk und Stefan Gerzmann. Mein Dank gilt natürlich auch meinen Probanden der Benutzerstudie Johannes Wagner, Nicole Schädel, Frauke Breitgoff, Damian Philipp, Andreas Kull, Michael Behringer, Lukas Schulz, Georg Balatzis, Louis Bergmann, David Schmid, Daniela Schlipf, Sarah Oppold.

Ein besonderer Dank gilt natürlich meinem Betreuer Florian Alt für die tatkräftige Unterstützung, auch was die kurzfristige Anschaffung von technischem Spielzeug angeht und seinen „Stellvertretern“, wenn Florian nicht in der Gegend sein konnte: Thomas Kubitza und Stefan Schneegaß. Maaret Posti möchte ich für die vielen großartigen 3D-Modelle danken, die so das Fliegen gelernt haben. Last but not least danke ich allen Freunden, die ich noch nicht erwähnt habe und die mir über Jahre die Treue gehalten haben.

Zum Schluss danke ich der Fachschaft Informatik und Softwaretechnik für viel Kaffee, einen Schlafplatz in der letzten Phase und für die vielen tollen Jahre. Ihr sitzt jetzt am Steuer und die Verfassten Studierendenschaften können für Euch eine große Chance sein. Dem Akademischen Chor und speziell dem Präsidium möchte ich danken für die vielen schönen Stunden, für den ruhigen Platz und für den Kopierer im Keller.

A. Technische Spezifikationen

A.1. Smartphone: Samsung GT-I9250 „Galaxy Nexus“

Im Galaxy Nexus sind viele unterschiedliche Sensoren und andere leistungsfähige Hardware integriert [Chi11]

Luftdrucksensor Bosch BMP180

Lagesensor (Gyroskop) / Elektrischer Kompass InvenSense MPU – 3050

Touch Screen MELFAS 8PK173

3-Achsen Beschleunigungssensor Bosch BMA220

Global Positioning System (GPS) SiRF GTD4T-9600B

Near Field Communication (NFC) NXP PN544

Kamera S5K4E5YA 5 Mp, 1,4 μm Pixel CMOS Bildsensor

Prozessor Texas Instruments 4460 OMAP Microprozessor (Dual-Core)

Wireless LAN / Bluetooth Broadcom BCM 4330 WiFi/Bluetooth SoC

A.2. Leonvo ThinkPad T520 4242

Hersteller Lenovo

Bezeichnung ThinkPad T520 4242W19

Arbeitsspeicher 4 GB DDR 3 RAM

Prozessor Intel i5-2520M 2,5 GHz, 2 Kerne, 4 logische Prozessoren

Externe Grafikhardware

- **Hersteller** NVIDIA
- **Bezeichnung** NVS 4200M
- **Grafikspeicher** 1GB
- **Ausgänge** DisplayPort, VGA

Verfügt über Stereoskopie Ja

Treiber Version 306.97

Onboard Grafikhardware

Hersteller Intel

Bezeichnung Intel HD Graphics 3000

Grafikspeicher 1,7 GB (Shared Memory)

Ausgänge keine

Verfügt über Stereoskopie Nein

Betriebssystem Microsoft Windows 7 Professional (x64) Service Pack 1

Windows-Leistungsindex 4,4

A.3. Server

Rootserver, gehostet bei der Firma Hetzner[1];

Prozessor Intel Core i7 Quad Core

Arbeitsspeicher 8 GB DDR 3 RAM

Festplatten 2× 750 GB SATA II HDD

Netzwerk- / Internetanbindung 100 MBit/s

Virtuelle Maschine

Von der in Abschnitt A.3 genannten Hardware werden derzeit allerdings nur zwei Prozessorkerne und ein Gigabyte RAM verwendet:

Betriebssystem Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-35-generic x86_64)[2].

Java OpenJDK Runtime Environment (IcedTea7 2.3.3) (7u9-2.3.3-0ubuntu1~12.04.1)[3].

[1] http://wiki.hetzner.de/index.php/Hardware#EQ_4
[2] http://www.ubuntu.com
[3] http://openjdk.java.net

A.4. Institutscomputer

Hersteller Hewlett-Packard

Bezeichnung HP Compay 8200 Elite CMT PC

Arbeitsspeicher 4 GB DDR 3 RAM

Prozessor Intel i5-2400 CPU 3.20 GHz, 4 Kerne, 4 logische Prozessoren

Grafikhardware

- **Hersteller** NVidia
- **Bezeichnung** NVS 300
- **Grafikspeicher** 512 MB DDR 3 RAM
- **Ausgänge** DMS59, per Adapter auch entweder 2 Single Link DVI-I, 2 Displayport, oder 2 VGA
- **Treiber Version** 306.97
- **Verfügt über Stereoskopie** Ja

Betriebssystem Microsoft Windows 7 Professional (x64) Service Pack 1

A.5. Sony VAIO

Hersteller Sony

Bezeichnung VPCF22C5E

Arbeitsspeicher 8 GB DDR 3 RAM

Prozessor Intel i7-2630QM CPU 2.0 GHz, 4 Kerne, 8 logische Prozessoren

Grafikhardware

- **Hersteller** NVidia
- **Bezeichnung** GeForce GT 540M
- **Grafikspeicher** 512 MB DDR 3 RAM
- **Ausgänge** HDMI
- **Verfügt über Stereoskopie** Ja

Betriebssystem Microsoft Windows 7 Home Premium (x64) Service Pack 1

A.6. Philipps 3D-Fernseher

Hersteller Philips

Bezeichnung 55PFL7606H

Bildschirmdiagonale 140 cm (55″)

Bilderzeugung LED

Kontrastverhältnis 500.000 : 1 (dynamisch)

3D-fähig Ja

3D-Bezeichnung Easy 3D

3D-Technologie Zeilenweise Polarisationstechnik

3D-Brillen Passive Polarfilter

3D-Auflösungen 1920×1080 (1080p) 24 Hz und 1280×720 (720p) 60 Hz (HDMI-Standard)

A.7. Epson 3D-Beamer

Hersteller Epson

Bezeichnung EH-TW6000

Projektionstechnik 3LCD-RGB-Projektionssystem mit Flüssigkristallen

Kontrastverhältnis 40.000 : 1

3D-fähig Ja

3D-Technologie Zeilenweise Polarisationstechnik

3D-Brillen Passive Polarfilter

3D-Auflösungen 1920×1080 (1080p) 24 Hz und 1280×720 (720p) 60 Hz (HDMI-Standard)

B. Fragebogen

Nachfolgend befindet sich der Fragebogen, der für die Benutzerstudie (siehe Kapitel 6 auf Seite 79) verwendet wurde.

Einstiegsfragebogen

1. Allgemein

ID:

1.1 Geschlecht:

☐ männlich ☐weiblich

1.2 Alter: ___

1.3 Beruf: _____

1.4 Sind Sie technikaffin?

☐ ja ☐nein

2. Sehen

2.1 Verwenden Sie eine Sehhilfe?

☐ ja ☐nein

2.2 Verfügen Sie über stereoskopisches Sehen?

☐ ja ☐nein

3. Kino

3.1 Wie häufig gehen Sie ins Kino?

☐ mehrmals pro Woche

☐ etwa einmal in der Woche

☐ mehrmals im Monat

☐ mehrmals im Jahr

☐ seltener

☐ nie

3.2 Haben Sie bereits stereoskopische 3D-Filme gesehen?

☐ ja ☐nein

3.3 Mögen Sie 3D-Filme?

sehr gerne ☐ ☐ ☐ ☐ ☐ äußerst ungerne

(bitte wenden)

3.4 Wie angenehm ist Ihnen das Betrachten von stereoskopischem Inhalt?

sehr angenehm ☐ ☐ ☐ ☐ ☐ äußerst unangenehm

3.5 Wenn Sie die Wahl zwischen der (stereoskopischen) 3D-Fassung eines Films und der normalen haben, welche würden Sie vorziehen?

☐ in jedem Fall die 3D-Fassung

☐ in jedem Fall die Standard-Fassung

☐ ich habe keine Vorlieben

3.6 Haben Sie beim Schauen von 3D-Filmen Beschwerden?

☐ nein

☐ ja, und zwar _____

4. Stereoskopische Anwendungen

4.1 Abgesehen von 3D-Filmen, hatten Sie schon einmal mit einer stereoskopischen Anwendung zu tun?

☐ ja ☐ nein

4.2 Wenn ja, bei welcher Gelegenheit?

_____

_____

4.3 Wo könnten Sie sich vorstellen, Stereoskopie einzusetzen?

_____

_____

5. Smartphones

5.1 Besitzen Sie ein Smartphone mit Touchdisplay?

☐ ja ☐ nein

5.2 Haben Sie eine Internetflatrate?

☐ ja ☐ nein

5.3 Haben Sie externe Apps installiert?

☐ ja ☐ nein

Fragebogen zu Interaktionsmethoden

1. ☐ Press-Geste ☐ Hand-Geste **ID:**

1. Fragen zur Benutzerzufriedenheit

1.1 Allgemeine Reaktion auf die Software

	0	1	2	3	4	5	6	7	8	9	
wunderbar	•	•	•	•	•	•	•	•	•	•	Furchtbar
schwierig	•	•	•	•	•	•	•	•	•	•	Leicht
frustrierend	•	•	•	•	•	•	•	•	•	•	angemessene Leistung
langweilig	•	•	•	•	•	•	•	•	•	•	Anregend
unflexibel	•	•	•	•	•	•	•	•	•	•	Flexibel

2 Fragen zur Bedienbarkeit

		stimme ich zu				Stimme ich nicht zu
1	Ich denke, dass ich dieses System gelegentlich nutzen würde	•	•	•	•	•
2	Ich fand das System unnötig komplex	•	•	•	•	•
3	Ich denke, dass das System einfach zu benutzen ist	•	•	•	•	•
4	Ich denke, dass ich die Unterstützen eines technisch versierten Person benötige, um das System zu bedienen	•	•	•	•	•
5	Meiner Meinung nach sind die vielen Funktionen des Systems gut integriert	•	•	•	•	•
6	Ich denke, dass es in dem System zu viele Widersprüche gibt	•	•	•	•	•
7	Ich könnte mir vorstellen, dass die meisten Leute das System schnell erlernen können	•	•	•	•	•
8	Ich finde es mühsam das System zu benutzen	•	•	•	•	•
9	Ich fühlte mich sehr sicher bei der Bedienung des Systems	•	•	•	•	•
10	Ich müsste viele Dinge erlernen, bevor ich das System bedienen könnte	•	•	•	•	•

(bitte wenden)

2.1 Wie fanden sie die Steuerung?

- ☐ natürlich
- ☐ einfach
- ☐ kompliziert

2.2 Können Sie sich vorstellen, ein solches System zu nutzen?

- ☐ ja
- ☐ nein

Abschließender Fragebogen

1. Bedienung **ID:**

1.1 Welchen Indikator, dass ein Objekt fangbar ist ziehen Sie vor?

☐ Glühendes Objekt

☐ Transparentes Objekt

☐ Glühendes Objekt in der transparenten Ebene

☐ Transparentes Objekt in der transparenten Ebene

☐ Keine Vorlieben

1.2 Welche Art der Bedienung ziehen Sie vor?

☐ Touch-Gesten

☐ Hand-Gesten

☐ Keine von beiden

1.3 Aus welchem Grund ziehen Sie diese vor / nicht vor?

_____

_____

_____

_____

_____

1.4 Welche Möglichkeiten, um darzustellen, dass ein Objekt fangbar ist fallen Ihnen noch ein?

_____

_____

_____

_____

_____

1.5 Die Durchläufe waren für mich...

zu kurz ☐ ☐ ☐ ☐ ☐ zu lange

(bitte wenden)

1.6 Der Hintergrund empfand ich als

☐ störend

☐ ablenkend

☐ motivierend

2. System allgemein

2.1 Könnten Sie sich vorstellen, so ein System in einer echten Kinoumgebung vor dem Hauptfilm zu nutzen?

☐ ja, gerne

☐ hin und wieder

☐ nein, nie

2.2 Würde Sie ein solches System vor dem Film stören?

ja, sehr ☐ ☐ ☐ ☐ ☐ nein, gar nicht

2.3 Ein Kino, das ein solches System verwendet würde ich

meiden ☐ ☐ ☐ ☐ ☐ häufiger besuchen

B.1. Questionaire for User Interaction Satisfaction - Gruppenauswertung

		Mittelwert	σ	Minimum	Maximum
	Gruppe 1				
Hand-Geste	furchtbar / wunderbar	4,40	2,06	2	7
	schwierig / leicht	6,00	1,67	3	8
	frustrierend / angemessene Leistung	5,6	1,74	3	8
	langweilig / anregend	6,40	0,8	5	7
	unflexibel / flexibel	4,80	1,17	3	6
Touch-Geste	furchtbar / wunderbar	50	2,28	2	8
	schwierig / leicht	5,40	1,50	3	7
	frustrierend / angemessene Leistung	5,4	2,33	2	8
	langweilig / anregend	6,80	0,75	6	8
	unflexibel / flexibel	5,80	1,72	3	8
	Gruppe 2				
Hand-Geste	furchtbar / wunderbar	5,75	2,38	3	9
	schwierig / leicht	5,75	2,59	2	9
	frustrierend / angemessene Leistung	5,75	2,38	3	9
	langweilig / anregend	6,75	1,48	5	9
	unflexibel / flexibel	6,00	2,74	2	9
Touch-Geste	furchtbar / wunderbar	5,75	2,05	4	9
	schwierig / leicht	6,75	0,43	6	7
	frustrierend / angemessene Leistung	5,25	1,64	4	8
	langweilig / anregend	7,25	1,30	5	8
	unflexibel / flexibel	4,25	1,30	3	6

		Mittelwert	σ	Minimum	Maximum
	Gruppe 3				
Hand-Geste	furchtbar / wunderbar	6,00	0,63	5	7
	schwierig / leicht	5,00	2,19	2	8
	frustrierend / angemessene Leistung	3,80	1,33	2	6
	langweilig / anregend	6,20	1,72	3	8
	unflexibel / flexibel	4,80	2,14	2	8
Touch-Geste	furchtbar / wunderbar	5,60	1,50	3	7
	schwierig / leicht	4,80	1,94	3	8
	frustrierend / angemessene Leistung	4,60	1,36	3	6
	langweilig / anregend	6,40	1,96	3	9
	unflexibel / flexibel	4,60	1,74	2	7
	Gruppe 4				
Hand-Geste	furchtbar / wunderbar	3,4	3,01	0	9
	schwierig / leicht	1,8	1,6	0	4
	frustrierend / angemessene Leistung	3	2,61	0	7
	langweilig / anregend	5,2	2,48	2	9
	unflexibel / flexibel	4,2	3,06	0	9
Touch-Geste	furchtbar / wunderbar	5,4	2,73	2	9
	schwierig / leicht	3	1,10	1	4
	frustrierend / angemessene Leistung	4,6	3,2	1	9
	langweilig / anregend	7,6	1,50	5	9
	unflexibel / flexibel	4,8	2,14	2	8

B.2. Simple Usability Scale - Gruppenauswertung

	Gruppe 1				
	T1	T2	T3	T4	T5
Hand-Geste	22,5	27,5	37,5	27,5	30,0
Touch-Geste	32,5	27,5	35,0	27,5	45,0
	Gruppe 2				
	T6	T7	T8	T9	
Hand-Geste	0	25,0	30,0	20,0	
Touch-Geste	15,0	22,5	20,0	27,5	
	Gruppe 3				
	T10	T11	T12	T13	T14
Hand-Geste	30,0	27,5	52,5	17,5	12,5
Touch-Geste	27,5	17,5	55,0	15,0	12,5
	Gruppe 4				
	T15	T16	T17	T18	T19
Hand-Geste	82,5	27,5	82,5	62,5	20,0
Touch-Geste	42,5	7,5	62,5	40,0	12,5

B.3. Messdaten

Hier befinden sich die Messwerte der Datenbankauswertung. Die Auswertung ist in Gruppen unterteilt zuerst findet sich eine Übersicht mit allen tatsächlich eingestellten Parametern je Gruppe und je Durchlauf. Die Gesamtübersicht befindet sich in Abschnitt 6.4.10 auf Seite 87.

D1 bis D4 bezeichnen die Ergebnisse der Durchgänge 1 bis 4. Die Anzahl der Gesten entspricht der Summe aller Gesten über alle Teilnehmer der Gruppe. Es wird unterschieden zwischen Gesten und gültigen Gesten. Gesten sind alle Gesten, die beim Animationsclient registriert wurden, auch diejenigen, die während der Abklingzeit gesendet wurden. Gültige Gesten sind Gesten, wenn die Gesten außerhalb de Abklingzeit gesendet wurden und nur solche können fliegende Objekte fangen. Treffer sind die Anzahl der Objekte, die tatsächlich durch Gesten gefangen wurden. Der Maximalwert entspricht der Anzahl der Objekte aus Tabelle 6.1 auf Seite 80. „Hand“ bzw. „Touch“ bezeichnet die Summe der Durchgänge, in denen die jeweilige Geste verwendet wurde. „Transp. “ (Transparenz) und „Glühen“ sind analog die Summen der Durchgänge der Indikatoren. Die Durchgänge, über die jeweils summiert wurde, stehen in Klammern in der Spalte unter dem Gruppennamen. Das Verhältnis von gültigen Gesten zur Anzahl der Gesten ist ein Indikator dafür, welcher Anteil aller Gesten tatsächlich gültig war. Das Verhältnis der gültigen Gesten zu Treffern gibt einen Hinweis darauf, wie viele gültige Gesten zum Fangen von Objekten geführt hat.

	Hand	*Touch*	*Glühen*	*Transp.*	*D1*	*D2*	*D3*	*D4*	Σ
				Gruppe 1					
	(D2)	(D1)	(D1)	(D2)					
Anz. Gesten	36	123	123	36	123	36	–	–	159
Anz. gültige	21	95	95	21	95	21	–	–	116
Anz. Treffer	4	13	13	4	13	4	–	–	17
Verh. $\frac{gültigeGesten}{Gesten}$	0,583	0,772	0,772	0,583	0,772	0,583	–	–	0,730
Verh. $\frac{Treffer}{gültigeGesten}$	0,190	0,137	0,137	0,190	0,137	0,190	–	–	0,147
				Gruppe 2					
	(D2+D4)	(D1+D3)	(D1+D4)	(D2+D3)					
Anz. Gesten	361	317	352	326	204	213	113	148	678
Anz. gültige	208	186	215	179	123	116	63	92	394
Anz. Treffer	55	59	55	59	30	30	29	25	114
Verh. $\frac{gültigeGesten}{Gesten}$	0,576	0,587	0,611	0,549	0,603	0,545	0,558	0,622	0,581
Verh. $\frac{Treffer}{gültigeGesten}$	0,264	0,317	0,256	0,330	0,244	0,259	0,460	0,272	0,289
				Gruppe 3					
	(D1+D4)	(D2+D3)	(D2+D4)	(D1+D3)					
Anz. Gesten	476	392	394	474	262	180	212	214	868
Anz. gültige	226	285	266	245	101	141	144	125	511
Anz. Treffer	60	58	60	58	31	31	27	29	118
Verh. $\frac{gültigeGesten}{Gesten}$	0,475	0,727	0,675	0,517	0,385	0,783	0,679	0,584	0,589
Verh. $\frac{Treffer}{gültigeGesten}$	0,265	0,204	0,226	0,237	0,307	0,220	0,188	0,232	0,231
				Gruppe 4					
	(D2+D4)	(D1+D3)	(D3)	(D1+D2+D4)					
Anz. Gesten	1422	719	340	1801	379	251	340	1171	2141
Anz. gültige	696	251	90	857	161	149	90	547	947
Anz. Treffer	149	71	30	190	41	25	30	124	220
Verh. $\frac{gültigeGesten}{Gesten}$	0,489	0,349	0,265	0,476	0,425	0,594	0,265	0,467	0,442
Verh. $\frac{Treffer}{gültigeGesten}$	0,214	0,283	0,333	0,222	0,255	0,168	0,333	0,227	0,232

Literaturverzeichnis

[And] Android Open Source Project. Platform Versions. URL http://developer.android.com/about/dashboards/index.html. (Zitiert auf den Seiten 8, 41 und 42)

[ano] anobjectn. Through Project Native. URL http://www.flickr.com/photos/spacetime/3708458685/. (Zitiert auf den Seiten 8 und 22)

[Bac98] T. Baccei. *Das Magische Auge*. Ars Edition, 1998. (Zitiert auf den Seiten 8, 21 und 22)

[Bro96] J. Brooke. SUS – A quick and dirty usability scale. *Usability evaluation in industry*, 189:194, 1996. (Zitiert auf Seite 84)

[CDN88] J. P. Chin, V. A. Diehl, K. L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, S. 213 – 218. 1988. (Zitiert auf Seite 84)

[Chi11] Chipworks. Inside the Samsung Galaxy Nexus GT-I9250, 2011. URL http://www.chipworks.com/blog/recentteardowns/2011/12/02/inside-the-samsung-galaxy-nexus-gt-i9250/. (Zitiert auf Seite 95)

[FH95] S. Fels, G. Hinton. Glove-TalkII: an adaptive gesture-to-formant interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, S. 456 – 463. 1995. (Zitiert auf Seite 35)

[GG88] R. A. Gardner, B. T. Gardner. The role of cross-fostering in sign language studies of chimpanzees. *Human Evolution*, 3:65 – 79, 1988. (Zitiert auf Seite 28)

[GZ08] Y. Guan, M. Zheng. Real-time 3D pointing gesture recognition for natural HCI. In *Proceedings of the 7th World Congress on Intelligent Control and Automation, 2008*, WCICA 2008, S. 2433 – 2436. 2008. (Zitiert auf Seite 35)

[Hei12] A. M. Heinecke. *Mensch-Computer-Interaktion: Basiswissen für Entwickler und Gestalter*. Springer, 2012. (Zitiert auf den Seiten 18 und 19)

[HOKK10] A. Hyakutake, K. Ozaki, K. M. Kitani, H. Koike. 3-D interaction with a large wall display using transparent markers. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, S. 97 – 100. 2010. (Zitiert auf den Seiten 32 und 33)

[inc13] N. C. inc. Wii Official Site at Nintendo, 2013. URL http://wii.nintendo.com/. (Zitiert auf Seite 29)

[Inn] Innenstadtkinos Stuttgart. Innenstadtkinos Stuttgart. URL http://innenstadtkinos.de/. (Zitiert auf Seite 79)

[Jog] JogAmp. Java Bindings for OpenGL. URL http://jogamp.org/. (Zitiert auf Seite 52)

[Jul60] B. Julesz. Binocular depth perception of computer-generated patterns. *Bell System Technical Journal*, 39:1125 – 1162, 1960. (Zitiert auf Seite 21)

[KKM+06] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, S. Di Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Computing*, 10(5):285 – 299, 2006. (Zitiert auf den Seiten 30, 38 und 39)

[KR10] S. Kratz, M. Rohs. A $3 gesture recognizer: simple gesture recognition for devices equipped with 3D acceleration sensors. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, S. 341 – 344. 2010. (Zitiert auf Seite 38)

[LH10] D.-H. Lee, K.-S. Hong. Game interface using hand gesture recognition. In *Proceedings of the 5th International Conference onComputer Sciences and Convergence Information Technology*, ICCIT '10, S. 1092 – 1097. 2010. (Zitiert auf Seite 35)

[LRS12] R. Llamas, K. Restivo, M. Shirer. Android- and iOS-Powered Smartphones Expand Their Share of the Market in the First Quarter, According to IDC, 2012. URL http://www.idc.com/getdoc.jsp?containerId=prUS23503312#.URkrL2fIt8E. (Zitiert auf Seite 43)

[Mar] Museum of the Marble and Cultural Heritage of Carrara City. URL http://urano.isti.cnr.it:8880/museo/home.en.php. (Zitiert auf Seite 38)

[MFK61] K. Mütze, L. Foitzik, W. Krug, Herausgeber. *ABC der Optik*. Dausien Werner, 1961. (Zitiert auf den Seiten 17, 19 und 21)

[Mic] Microsoft Corperation. Developing games (Windows). URL http://msdn.microsoft.com/en-us/library/windows/apps/hh452744.aspx. (Zitiert auf Seite 53)

[MKKK04] J. Mäntyjärvi, J. Kela, P. Korpipää, S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, MUM '04, S. 25 – 31. 2004. (Zitiert auf den Seiten 9, 30 und 37)

[MPSS06] J. Mäntyjärvi, F. Paternò, Z. Salvador, C. Santoro. Scan and tilt: towards natural interaction for mobile museum guides. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, MobileHCI '06, S. 191 – 194. 2006. (Zitiert auf Seite 37)

[MS99] C. D. Manning, H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999. (Zitiert auf den Seiten 34 und 35)

[MSWRI07] I. Molnar-Szakacs, A. D. Wu, F. J. Robles, M. Iacoboni. Do You See What I Mean? Corticospinal Excitability During Observation of Culture-Specific Gestures. *PLoS ONE*, 2(7):e626, 2007. (Zitiert auf Seite 28)

[Nat12] Nationalgeographic. Gesten und Gefahren. *National Geographic Deutschland*, 4/2012:38, 2012. (Zitiert auf Seite 28)

[Ope] Open Graphics Library (OpenGL). URL http://www.opengl.org/. (Zitiert auf Seite 52)

[PDM+09] A. Pandit, D. Dand, S. Mehta, S. Sabesan, A. Daftery. A Simple Wearable Hand Gesture Recognition Device Using iMEMS. In *Proceedings of the International Conference of Soft Computing and Pattern Recognition*, SOCPAR '09, S. 592 – 597. 2009. (Zitiert auf Seite 36)

[Pla13] Playstation 3 (PS3), 2013. URL http://de.playstation.com/ps3/. (Zitiert auf Seite 34)

[Pup12] M. Puppe. Smartphones sind der Renner im Weihnachtsgeschäft. Internet, 2012. URL https://www.bitkom.org/de/presse/8477_74253.aspx. (Zitiert auf Seite 12)

[Rei96] M. Reim. *Augenheilkunde: 13 Tabellen*. Enke, Stuttgart, 5., durchges. aufl. Auflage, 1996. (Zitiert auf Seite 17)

[RLC99] M. Rider, T. Lacavera, G. Creta. Motion picture theater interactive gaming system. Internet, 1999. (Zitiert auf Seite 40)

[RWBT12] M. Rofouei, A. Wilson, A. Brush, S. Tansley. Your phone or mine?: fusing body, touch and device sensing for multi-user device-display interaction. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, S. 1915 – 1918. 2012. (Zitiert auf den Seiten 31 und 32)

[Sam] Samsung. Samsung Galaxy Nexus - GT-I9250. URL http://www.samsung.com/de/consumer/mobile-device/mobilephones/smartphones/GT-I9250TSADBT-spec. (Zitiert auf Seite 41)

[SHSK08] F. Steinicke, K. H. Hinrichs, J. Schöning, A. Krüger. Multi-Touching 3D Data: Towards Direct Interaction in Stereoscopic Display Environments coupled with Mobile Devices. In *Proceedings of the Advanced Visual Interfaces (AVI) Workshop on Designing Multi-Touch Interaction Techniques for Coupled Public and Private Displays*, S. 46 – 49. 2008. (Zitiert auf Seite 32)

[SP01] S. Starke-Perschke. *Der Brockhaus Psychologie: Fühlen, Denken und Verhalten verstehen*. Brockhaus, 2001. (Zitiert auf Seite 27)

[SP12] M. Shahd, A. Pols. Hochwertige Smartphones werden Standard, 2012. URL http://www.bitkom.org/de/presse/74534_73193.aspx. (Zitiert auf Seite 30)

[TC90] C. W. Tyler, M. B. Clarke. Autostereogram. S. 182–197, 1990. doi:10.1117/12.19904. URL +http://dx.doi.org/10.1117/12.19904. (Zitiert auf Seite 23)

[TGF89] M. Tomasello, D. Gust, G. Frost. A longitudinal investigation of gestural communication in young chimpanzees. *Primates*, 30:35 – 50, 1989. (Zitiert auf Seite 28)

[Tra] Traumpalast. Traumpalast Esslingen. URL http://esslingen.traumpalast.de/. (Zitiert auf den Seiten 25 und 79)

[TY002] E. Tuulari, A. Ylisaukko-oja. SoapBox - A Platform for Ubiquitous Computing Research and Applications. In F. Mattern, M. Naghshineh, Herausgeber, *Pervasive Computing*, Band 2414 von *Lecture Notes in Computer Science*, S. 125 – 138. Springer Berlin Heidelberg, 2002. (Zitiert auf Seite 30)

[VCBE08] T. Vajk, P. Coulton, W. Bamford, R. Edwards. Using a Mobile Phone as a "Wii-like" Controller for Playing Games on a Large Public Display,. *International Journal of Computer Games Technology*, 2008. (Zitiert auf Seite 29)

[VGH12] D. Valkov, A. Giesler, K. Hinrichs. Evaluation of depth perception for touch interaction with stereoscopic rendered objects. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, S. 21 – 30. 2012. (Zitiert auf den Seiten 31 und 32)

[Vle11] E. Vlessing. Canada's Cineplex Entertainment Rolls Out Interactive Branded Entertainment in Theaters, 2011. URL http://www.hollywoodreporter.com/news/canadas-cineplex-entertainment-rolls-interactive-272822. (Zitiert auf Seite 40)

[WWL07] J. O. Wobbrock, A. D. Wilson, Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, S. 159 – 168. 2007. (Zitiert auf den Seiten 36 und 38)

[Xbo13] Xbox. Kinect für Xbox 360, 2013. URL http://www.xbox.com/de-DE/Kinect. (Zitiert auf den Seiten 34 und 53)

Alle URLs wurden zuletzt am 28. 02. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschift