Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Malte Schier

VR Desktop: Konzeption eines virtuellen interaktiven Desktops
mittels einer VR-Brille und Bewegungssteuerung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Malte Schier

VR Desktop: Konzeption eines virtuellen interaktiven Desktops mittels einer VR-Brille und Bewegungssteuerung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Betreuender Prüfer : Prof. Dr. Philipp Jenke
Zweitgutachter : Prof. Dr. Thomas Lehmann

Abgegeben am 11. Juni 2015

Malte Schier

Thema der Arbeit

VR Desktop: Konzeption eines virtuellen interaktiven Desktops mittels einer VR-Brille und Bewegungssteuerung

Stichworte

Virtuelle Realität, alternative Bedienkonzepte, Bewegungssensoren, Immersion, Präsenz, Spezifikation, Entwurf, Betriebssysteme, 3D-Graphik, dreidimensionale graphische Oberflächen

Kurzzusammenfassung

Diese Bachelorarbeit bestimmt die Komponenten, die für eine dreidimensionale graphische Oberfläche auf Basis von Bewegungserkennung und VR-Brille benötigt werden. Hierfür werden ein von VR-Brille und Bewegungserkennung gestütztes Bedienungsschema, die benötigten Systemdienste, mögliche Rahmenbedingungen, sowie mögliche Optimierungen spezifiziert und entworfen. Anschließend wird geprüft, in wie weit sich diese erfassten Sachverhalte mit verfügbarer Technologie realisieren lassen. Hierfür wird ein Prototyp mit einer Teilfunktionalität der zuvor spezifizierten und entworfenen Komponenten erstellt. Zum Ende hin werden die Ergebnisse diskutiert.

Malte Schier

Title of the paper

VR Desktop: Design of a virtual interactive desktop using a VR goggles and motion control

Keywords

Virtual reality, alternative operating concepts, motion sensors, immersion, presence, specification, design, operating systems, 3D graphic, 3-dimensional graphical user interfaces

Abstract

This bachelor thesis determines the components that are needed for a 3-dimensional graphical user interface based on motion detection and VR goggles. For this purpose, an interaction scheme based on VR goggles and motion detection, the required system services, possible conditions and improvements will be specified and designed. It is then checked in how far these specified requirements can be implemented with available technology. For this purpose, a prototype with a part of the functionality of the previously specified requirements is created. At last these results are discussed.

Danksagung
Hiermit möchte ich mich bei allen herzlich bedanken, die mir direkt oder indirekt bei der Erstellung dieser Bachelorarbeit geholfen haben.

Mein fachlicher Dank geht an:
Prof. Dr. Philipp Jenke und Prof. Dr. Thomas Lehmann für die Annahme des Themas und der angenehme Betreuung. Sowie an Fenja Harbke, für die schnelle und kompetente Korrektur.

Mein freundschaftlicher Dank geht an:
Florian und Anna, Dominik und Jenny, Angelo und Marcella, Cenan, Cosimo, Marcel, Olaf, Henning, Sascha, Melanie, Timo, Adam, Huduy, Patrick, Markus, Marko und Hauke, ohne euch wäre ich nicht der Mensch der ich heute bin.

Mein unendlicher Dank geht an:
Meine Eltern, die immer an mich geglaubt haben, selbst wenn ich die Hoffnung selbst schon längst aufgegeben hatte, sowie an meinen Bruder Tom.

Hamburg, den 11.06.2015
Malte Schier

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Auf der Beiliegenden CD befinden sich die folgenden Anlagen:

- Alle vom Autor selbst erstellten Graphiken
- Anforderungen.pdf
- Interaktionsfunktionen.pdf

1 Einleitung

Seit der erfolgreichen Kickstarter Kampagne von Oculus LLC [2.49] im Jahre 2012, erleben wir einen erneuten Boom im Bereich der VR-Technik. Im Gegensatz zu vergangenen Unternehmungen, die es sich zur Aufgabe gemacht hatten, diese Technologie als salonfähig zu etablieren, scheint die Umsetzung diesmal in greifbarer Nähe. Durch die Verfügbarkeit diverser preiswerter Prototypen ist es heute schon möglich, einen Vorgeschmack auf eine mögliche VR-Zukunft zu erhalten. Glaubt man den überwiegend positiven Kritiken, so könnte VR-Technik die Informatik-Landschaft nachhaltig verändern.

Oberflächlich betrachtet ist eine-VR Brille nichts anderes als ein Bildschirm, der direkt vor den Augen eines Anwenders befestigt wird. Bei näherer Betrachtung fällt jedoch auf, dass es trotz der scheinbar trivialen Konstruktion vielfältige Arten von Problemen zu lösen gilt. Diese Probleme sind nicht nur rein technischer Natur, sondern auch momentan angewendete Entwurfsprozesse und heutzutage allgemeingültige Bedienkonzepte könnten in Frage gestellt werden müssen.

Bei genauerer Betrachtung lassen sich mehrere Problemklassen feststellen:

- Welche Hardware-Anforderungen gibt es für VR-Technik?
- Wie wird Software VR-gerecht realisiert?
- Wie wirkt VR auf das Gehirn?

Jedoch wiegt der Kernpunkt, der für eine Verwendung von VR-Technik spricht, wesentlich stärker, als die Problemstellungen, die es für eine breite Marktakzeptanz zu lösen gilt. So lässt sich durch die Wirkung von VR-Technik auf das Gehirn ein Sachverhalt erzeugen, der mit keinem anderen Medium möglich sind.
Dieser einmalige Sachverhalt wird durch das Wort Präsenz beschrieben, was Umgangssprachlich so viel bedeutet wie „Da sein“.

Wenn man sich nun vor Augen führt, dass der Präsenz-Effekt den unterbewussten, menschlichen Entscheidungsprozess manipulieren kann, und es somit keine

abstrahierenden Schichten mehr zwischen dem Gehirn und einem virtuellen Sachverhalt gibt, erkennt man, dass diese Eigenschaften weit über die heutzutage oftmals propangierten Einsatzbereiche der VR-Technik als Gaming-Gimmick hinausgehen.
Dies lässt einen zu der Vermutung gelangen, dass es sich bei VR-Technik vielmehr um ein neues Medium mit unzähligen Anwendungsmöglichkeiten handelt.

1.1 Motivation

Die visuelle Aufbereitung von Daten ist seit jeher ein großes Thema der Informatik. Zu Beginn genügten einfache textuelle Eingaben und Ausgaben, aber mit der Leistungsfähigkeit der Maschinen wuchs auch die Vielfältigkeit der Einsatzgebiete.

Nach einer gewissen Zeit wurden diese textuellen Schemata so vielfältig und komplex, dass es nur noch einer kleinen Gruppe Spezialisten möglich war, diese in ihrer Vollständigkeit zu erfassen.
Als Computer dann immer interessanter für den Massenmarkt wurden, wurde auch der Ruf nach intuitiveren Bedienungsmöglichkeiten lauter. Das Ergebnis sieht man heute bei allen Arten von Computersystemen von Handys bis Supercomputern: Die graphische Oberfläche.
Jedoch könnte auch dieses Konzept langsam an seine Grenzen stoßen. Als graphische Oberflächen in den 70er Jahren erdacht wurden, steckte die Parallelisierung von Programmen noch in den Kinderschuhen oder war aufgrund von fehlender Rechenkraft generell nicht möglich. Heutzutage dagegen ist die Parallelität allgegenwärtig. Der Nutzer bearbeitet ein Textdokument, hat seinen Twitter-Feed im Auge, während er Musik hört und eine DVD brennt, so dass die Übersicht über die parallel ausgeführten Tätigkeiten unter Umständen leiden könnte.
VR-Technik könnte hierbei den nächsten evolutionären Schritt von graphischen Oberflächen darstellen. Anstatt benötigte Informationen auf einem Bildschirm zu betrachten, kann der Benutzer mit Hilfe von VR-Technik in eine Art virtuelle Welt versetzt werden. In dieser könnten dann Informationen an jeder Stelle des Raumes positioniert werden.

Aber egal wie intuitiv die visuelle Aufbereitung von Informationen durch VR-Technik auch ist, der Benutzer benötigt immer ein Gerät zur Interaktion mit den dargestellten Inhalten.
In den letzten Jahren haben Bewegungsteuerungen viel an Popularität gewonnen, so dass auch diese den nächsten evolutionären Schritt darstellen könnten.

Das Interesse an der Kombination von VR-Technik mit Bewegungserfassung, um auf dieser Basis alternative Bedienkonzepte zu untersuchen, bildet die Motivation für die Erstellung dieser Arbeit. Ein weiterer Aspekt ist, dass im Bereich der VR-Technik wie auch der Bewegungssteuerung heute noch Grundlagenforschung betrieben wird und somit der Fokus mehr auf der Suche nach kreativen Problemlösungen, als auf der Optimierung von Sachverhalten liegt.

1.2 Zielsetzung

Im Rahmen der Bachelorarbeit soll ein Konzept erstellt werden das klärt, welche Faktoren erfüllt sein müssen, damit eine dreidimensionalen graphischen Oberfläche auf Basis von polygonalen Netzen, Bewegungserkennung und VR-Technik realisiert werden kann.
Im Detail sollen hierfür ein Bedienungsschema das auf Gestensteuerung basiert, graphische Konzepte, die benötigten Systemfunktionen die zwingend für den Betrieb erforderlich sind und mögliche Optimierungsmöglichkeiten spezifiziert und entworfen werden.

Im Anschluss daran soll geprüft werden, in wieweit sich dieses Konzept mit den momentan verfügbaren technischen Möglichkeiten realisieren lässt. Hierbei soll ein Prototyp implementiert werden, der eine Teilfunktionalität aus dem erstellten Konzept besitzt.

1.3 Aufbau der Arbeit

Diese Arbeit ist in sechs Kapitel gegliedert, die folgendermaßen aufgebaut sind:

Kapitel 1, Einleitung, führt in die Thematik ein und gibt einen Überblick über diese Arbeit.

Kapitel 2, Grundlagen, erläutert zum Einstieg die verschiedenen Aspekte der VR-Technik und geht dabei vertiefend auf die Faktoren Immersion und Präsenz ein. Außerdem wird auf die Wirkung von VR-Technik auf das Gehirn und die daraus resultierenden Probleme eingegangen.

Kapitel 3, die Anforderungsanalyse und Spezifikation, analysiert zunächst die grundlegenden Eigenschaften von graphischen Oberflächen. Daraufhin erfolgt die Spezifikation der benötigten Faktoren.

Kapitel 4, beschreibt den einen Entwurf, der auf den im vorherigen Kapitel spezifizierten Anforderungen basiert. Hierfür werden die verschiedene Sichten der Architektur entwickelt und das Zusammenspiel der dazugehörigen Komponenten und angrenzenden Systemen aufgezeigt.

Kapitel 5, Implementierung, evaluiert die Umsetzbarkeit des Entwurfes und beschäftigt sich mit Implementierungsdetails und möglichen Problemen.

Kapitel 6, Fazit und Ausblick, gibt eine abschließende Zusammenfassung der Arbeit mit Bewertung der Ergebnisse. Der Ausblick erläutert offene Fragestellungen und schließt die Arbeit ab.

2 Grundlagen

In diesem Kapitel sollen die Grundlagen für die folgenden Kapitel vermittelt werden. Es wird vertieft auf die Effekte Präsenz und Immersion eingegangen. Hierfür werden sowohl technische als auch kognitive Aspekte der beiden Effekte genauer betrachtet. Im Anschluss werden die technischen Faktoren von Präsenz erläutert.

2.1 Geschichte der VR Technik

In diesem Abschnitt sollen die wichtigsten Meilensteine der VR-Technik aufgelistet werden. Dabei wird sich auf das absolute Minimum beschränkt. Hierbei geht eher darum, ein zeitliches Gefühl für die VR-Technik zu vermitteln.

"Trying to trace the origins of the idea of virtual reality is like trying to trace the source of a river. It is produced by the accumulated flow of many streams of ideas, fed by many springs of inspiration." - Benjamin Woolley [2.1]

In den 1960 entwickelte Ivan Sutherland [2.2] eine der ersten VR-Brillen überhaupt. Obwohl das Wort Brille hierbei eher im übertragenden Sinne verwendet werden kann. Durch das Gewicht der Apparatur war es nötig, diese fest mit der Decke zu verschrauben.
Sutherland fungierte auch als Doktorvater für Henri Gouraud und Bui Tuong Phong [2.3], die später wichtige Beiträge im Bereich der Computergraphik leisteten.

Ebenfalls in den 1960er Jahren erhielt Mortin Heilig ein Patent für ein Gerät namens Sensorama [2.4]. Dieses rein mechanische VR-Gerät, besitzt die Gestalt eines Passbild-Automaten und sollte alle 5 menschlichen Sinne stimulieren. Ein Zelluloid Film wird hierfür zusammen mit Ton, Force Feedback, verschiedenen Gerüchen und Fahrtwind abgespielt.

1975 stellt Myron Krueger [2.38] Videoplace vor. Hierbei befinden sich zwei Menschen in verschiedenen Räumen, jeder Raum mit einer Kamera und einem Projektor ausgestattet.

Mithilfe dieser zwei Gerätschaften war es möglich, dass die Probanden über die Grenzen der Räume hinweg, über visuelle Gesten, kommunizieren konnten.

1984 präsentiert die NASA [2.39] das Virtual Visual Environment Display kurz VIVED. Diese Apparatur bietet einen stereoskopischen, monochromen Bildschirm und ist den heutzutage erhältlichen VR-Brillen bereits sehr ähnlich.

1988 vermarktet VPL Research das Eyephone [2.40]. Dieses Gerät ist die erste freiverkäufliche VR-Brille.

2.2 Arten von virtueller Realität

Heutzutage taucht der Begriff der virtuellen Realität zumeist in Verbindung mit VR-Brillen auf. Doch gibt es noch eine ganze Reihe anderer Verfahren, die in ihrer Klassifizierung unter den Begriff VR fallen. In diesem Abschnitt soll ein differenziertes Bild des Begriffes Virtual Reality vermittelt werden.
Hierfür wird die Klassifizierung nach Brill [2.43] verwendet. Der Kern dieses Modells ist, dass jedes Verfahren das ein durch den Computer oder durch ein anderes Medium generierte Umgebung anbietet in die der Benutzer eine physikalische, sensorische und oder psychologische Einbindung erlebt, in eine dieser Klassen der VR-Technik fällt [2.44].

Immersive Systeme: Präsentieren eine virtuelle Welt aus der Ich-Perspektive des Benutzers. Hierfür wird zumeist eine VR-Bille verwendet. Diese Brillen beinhalten zumeist eine visuelle und eine akustische Komponente. Sie sind entweder frei beweglich oder können an einer Apparatur befestigt sein.

Argumentierte Realität: Diese sind eine Variante der immersiven Systeme. Hierbei blickt der Anwender durch einen durchsichtigen Computerbildschirm. Auf diesen werden bestimmte Elemente der Realität entweder hervorgehoben oder es werden neue Elemente hinzugefügt.

Desktop Virtualität (WoW): Diese VR-Systeme benutzen gewöhnliche CRT/LCD Monitore zur Darstellung von virtuellen Welten. Dieses Konzept wird bereits 1965 in Ivan Sutherlands Arbeit „The Ultimate Display“ beschrieben und könnte als einer der grundlegenden Sätze der Computergraphik interpretiert werden.
„One must look at a display screen as a window through which one beholds a virtual world. The challenge to computer graphics is to make the picture in the window look real, sound real and the objects act real." - Ivan Sutherland [2.2]

Spiegel-Welt: Eine Variante des WoW Ansatzes. Hierbei wird der Körper des Anwenders mit Hilfe einer Tracking-Komponente der virtuellen Welt hinzugefügt. Zusätzlich dazu kann

er durch die Trackingkomponente mit der virtuellen Welt interagieren. Als Beispiel seien an dieser Stelle die Playstation Eye Kamera [2.5] und die Microsoft Kinect [2.6] genannt.
Eine Verwendung findet dieses Konzept auch in dem eingangs erwähnten Videoplace.

Waldo-Welt: Der Name Waldo entstammt aus der Science Fiction Kurzgeschichte von Robert Heinlein [2.41]. Dieses Verfahren ist dem Motion Capturing sehr ähnlich. Hierbei werden Bewegungsinformationen eines mit diversen Sensoren ausgestatten Probanden direkt auf einen virtuellen Avatar übertragen und dieser dadurch animiert.

Kammerwelt: Diese Variante von immersiven Systemen kann mit der Hilfe von mehreren Projektionsgeräten und Film-Leinwänden erzeugt werden. Die Projektionsgeräte werden so angeordnet, dass sie auf jede Innenwand des Raumes ein Bild werfen. Der Anwender kann dann mithilfe von Bewegungserkennung die projizierten Bilder manipulieren. Eine Implementierung dieser Variante ist CAVE VR System [2.8]

Simulatoren: Praktisch jeder Simulator der die anfangs erwähnten Faktoren nach Brill erfüllt. Hierbei spielt es keine Rolle welchen Zweck die Simulation erfüllt. Es kann sich um realistische Nachbildungen aus der Realität handeln, oder um abstrakte Sachverhalten aus dem Unterhaltungsbereich.

Cyberspace: Der Begriff Cyberspace wird das erste Mal in dem Buch *Neuromancer* von William Gibson [2.42] erwähnt. Ein Cyberspace ist eine globale, virtuelle Realität die simultan von mehreren Menschen durch Zuhilfenahme eines Computernetzwerks besucht werden kann. Ein Beispiel sind im Internet beliebten Multi User Dungeons oder Online Multiplayer Rollenspiele.

Telepräsenz: Bei Telepräsenz entsteht die virtuelle Realität durch Abstraktion der realen Realität. Hierfür werden verschiedene Arten von Sensorik mit den Sinnen eines Anwenders verbunden.
Die Sensorik wird daraufhin mit einem ferngesteuerten Roboter oder ähnlichem verbunden. Anwendung findet dieses Prinzip zum Beispiel im Mars Rover oder chirurgischem Besteck, das mit einer Kamera ausgerüstet ist.

Im weiteren Verlauf wird sich diese Arbeit ausschließlich mit den immersiven Systemen und seinen verwandten Verfahren beschäftigen.

2.3 Immersion

Genau wie der Begriff VR ist auch die Immersion seit dem erneuten VR-Boom in aller Munde. Tatsächlich verbirgt sich dahinter ein weitaus älteres Konzept. Zur genaueren Unterteilung wird die Gliederung nach Ernest Adams [1.10] verwendet, er Identifiziert 4 Arten von Immersion.

Taktische Immersion: Taktische Immersion wird erfahren, durch das Ausführen einer unmittelbaren Tätigkeit die zum Erfolg führt. Ein Beispiel hierfür ist das Fangenspielen auf dem Schulhof.

Strategische Immersion: Im Gegensatz zur taktischen Immersion entsteht die strategischen Immersion bei der Anwendung von mehreren taktischen Entscheidungen, die Teil einer Strategie sind. Ein Schachspiel ist eines der klassischen Beispiele für diese Art der Immersion.

Narrative Immersion: Sie entsteht durch eine emotionale Bindung zu Charakteren und/oder einer Geschichte im Allgemeinen. Kann durch das Lesen von Büchern oder das Ansehen von Filmen erreicht werden.

Spatiale Immersion: Entsteht wenn eine simulierte, virtuelle Welt für die Wahrnehmung des Benutzers glaubwürdig erscheint. Der Benutzer erlebt einen Zustand als wäre er wirklich vor Ort und die Welt sieht und fühlt sich glaubhaft an. Diese Art von Immersion kann nur in Verbindung mit VR-Technik künstlich erzeugen werden.

Anhand der Beispiele lässt sich feststellen, dass für die taktische, strategische und narrative Immersion keinerlei Art von Computertechnik nötig ist. Die für VR entscheidende Immersionsart ist die der spatialen Immersion. Diese lässt sich ausschließliche mit Hilfe von VR-Verfahren erleben.
An dieser Stelle soll auch noch einmal darauf hingewiesen werden, dass spatiale Immersion und Präsenz nicht dasselbe sind. Spatiale Immersion beschreibt den Zustand, von Bildern einer virtuellen Welt umgeben zu sein. Präsenz beschreibt das Gefühl, sich in einer virtuellen Welt zu befinden [2.13].

2.4 Realität, ein Erklärungsversuch

In diesem Abschnitt soll der Begriff Realität näher untersucht werden. Dies liegt darin begründet, dass wenn man verstehen will was eine virtuelle Realität ausmacht, man verstehen muss was die reale Realität ausmacht. Hierfür sollen verschiedene Arbeitsweisen des Gehirns dargelegt werden, sowie beteiligte Faktoren dazu erläutert werden, die dann dazu verwendet werden, den VR-Effekt Präsenz zu erklären.

2.4.1 Die objektive Realität

“Als Realität wird im allgemeinen Sprachgebrauch die Gesamtheit des Realen bezeichnet. Als real wird zum einen etwas bezeichnet, das keine Illusion ist sowie nicht von den Wünschen oder Überzeugungen eines Einzelnen abhängig.“ - Wikipedia [2.15]

Das Wort Realität wird in unserem alltäglichen Wortschatz des Öfteren benutzt und eigentlich sollte man meinen, dass eine Definition des Wortes Realität keine Probleme bereitet.
Aber eine Definition dieses Begriffes ist an so viele Faktoren gekoppelt, dass es schwerfällt diesen universell zu formulieren. Daher soll an dieser Stelle nur der Begriff Realität in Bezug auf VR-Technik definiert werden.

Hierfür betrachten wir als erstes den Ausschnitt des Zitats „…das keine Illusion ist sowie nicht von den Wünschen oder…“. Der Satz könnte implizieren, dass alles was man nicht wahrnehmen kann eine Illusion ist.
Als aufgeklärter Mensch weiß man nun aber, dass Dinge wie Ultraschall oder atomare Strahlung sehr wohl Realität sind. Es drängt sich der Verdacht auf, dass es eine Realität bzw. Faktoren der Realität gibt, die vom Menschen nicht wahrgenommen werden können.

Dies bedeutet, dass schon einmal zwischen einer objektiven, allumfassenden und einer vom Menschen wahrnehmbaren Realität unterschieden werden muss.

2.4.2 Die menschliche Realität

Klassisch bzw. umgangssprachlich wird die menschliche Wahrnehmung nach den Kriterien von Aristoteles definiert und man spricht davon, dass der Mensch die Realität mit 5 Sinnen wahrnimmt. Folgende Angaben wurden aus dem Wikipedia-Artikel Sinn (Wahrnehmung) entnommen [2.16]:

- Das Sehen, die visuelle Wahrnehmung mit den Augen.
- Das Hören, die auditive Wahrnehmung mit den Ohren.
- Das Riechen, die olfaktorische Wahrnehmung mit der Nase.
- Das Schmecken, die gustatorische Wahrnehmung mit der Zunge.
- Das Tasten, die taktile Wahrnehmung mit der Haut.

Jedoch ist dies bezüglich VR noch zu ungenau. Jeder der sich schon einmal in irgendeiner Weise verletzt hat weiß, dass noch andere Faktoren in der Wahrnehmung des Menschen eine Rolle spielen. Die da wären:

- Temperatursinn, Thermorezeption
- Schmerzempfindung, Nozizeption
- Vestibulärer Sinn, Gleichgewichtssinn
- Körperempfindung (oder Tiefensensibilität)

Es lässt sich festhalten, dass gegenüber der objektiven Realität, die menschliche Realität, bestimmt durch ihre Sinnesorgane, nur eine Teilmenge der tatsächlichen verfügbaren Informationen erfassen kann.

2.4.3 Der 6te Sinn, Tiefensensibilität

Ein oftmals vernachlässigter Faktor bei der Verwendung des Wortes VR-Technik ist die Tiefensensibilität.
Zur Erklärung werden hierbei Auszüge aus dem Artikel „Der sechste Sinn“ verwendet [2.17].

Die Tiefensensibilität vermittelt dem Gehirn Informationen über die Lage des Körpers in dreidimensionalen Räumen. Ob wir sitzen, liegen oder laufen, welche Stellung unser Füße haben. Diese Informationen erhält das Gehirn von sogenannten Propriozeptoren. Diese Sensoren sitzen in den Muskeln, Sehnen, Bändern und Gelenken und reagieren auf Druck oder Verformung.
In einen späteren Abschnitt wird beschrieben, welche Rolle diesem Sinn bei Verwendung von VR-Technik zukommt.

2.4.4 Verarbeitung von Sinneseindrücken

Für VR-Technik ist die Frage wichtig, in wie weit unser Bewusstsein in die Verarbeitung von Sinneseindrücken involviert ist.
Prinzipiell lassen sich die Entscheidungen, die wir in unserem täglichen Leben treffen in zwei Kategorien einordnen; bewusst getroffene Entscheidungen und unbewusst getroffene Entscheidungen.
So geht Neurowissenschaftler Prof. Gerhart Roth [2.18] davon aus, dass 90 Prozent der von uns getroffen Entscheidungen unterbewusst getroffen werden und unser Bewusstsein keinen Einfluss auf diese Entscheidungen hat. Hierfür werden scheinbar die von den Sinnesorganen übermittelten Signale vom Gehirn unterbewusst verarbeitet und ein entsprechendes Ergebnis präsentiert. Wichtig ist es hierbei zu verstehen, dass unser bewusstes Denken keinen Einfluss auf diese getroffen Entscheidungen hat [2.50].
Zur besseren Verständlichkeit dieses Sachverhalts soll hierfür eine Graphik [2.19] herangezogen werden. Sie nennt sich “as *Turning the Tables*“ von Roger Shepard [2.20], Abbildung 1. Auf der Graphik sind zwei Tische zu sehen, die scheinbar unterschiedlich groß sind. Jedoch haben diese beiden Tische in Wirklichkeit die absolut identische Größe.

Welche Phänomene hinter dies optischen Täuschung stecken ist nicht relevant. Relevant ist, dass selbst nachdem wir uns von dem Sachverhalt, dass die beiden Tische gleichgroß sind, überzeugt haben, sich unser unterbewusst agierender kognitiver Auswertungsmechanismus weigert, diesen Sachverhalt anzuerkennen und uns weiterhin die optische Illusion präsentiert.
Der Sachverhalt lässt sich beliebig mit anderen optischen Täuschungen wiederholen. Das Ergebnis ist aber immer dasselbe; das bewusste Denken hat keinen Einfluss auf die getroffenen Entscheidungen des unterbewussten Denkens [2.50].

Dies liefert auch die Grundlage für den Präsenz-Effekt. Wenn unsere unterbewusst arbeitenden Prozesse einen dargestellten Sachverhalt als real akzeptieren, dann ist er für uns real, ob wir nun wollen oder nicht.

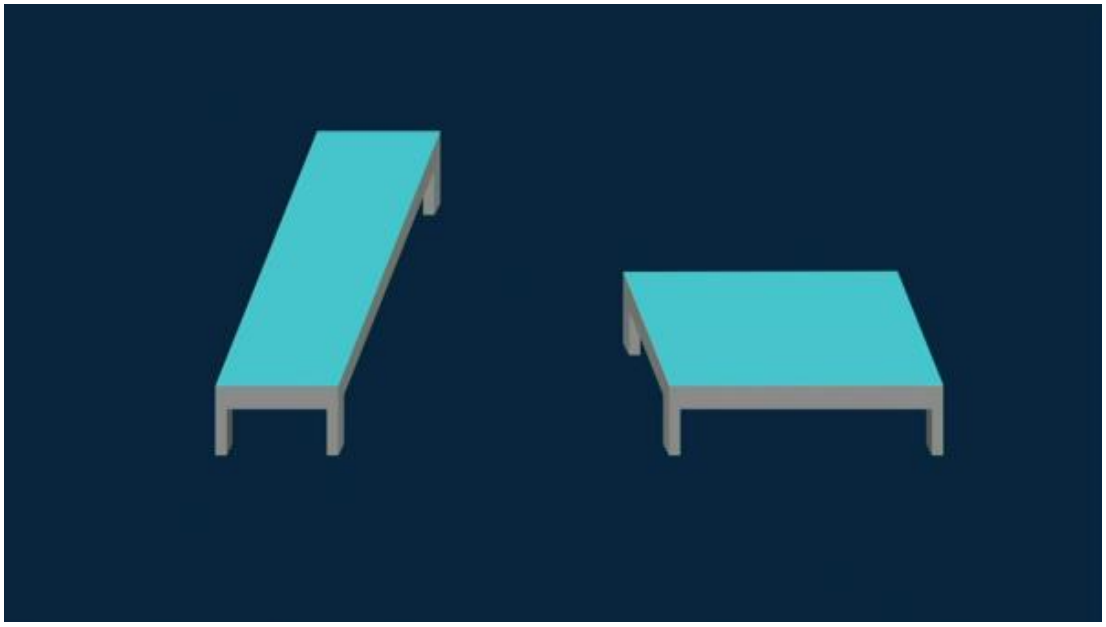

Abbildung 1 - “as Turning the Tables“ von Roger Shepard [2.20]

2.4.5 Interpretation der Realität

Die optische Täuschung von Roger Shepard lässt es schon erahnen, aber es muss ein entscheidender Sachverhalt genauer untersucht werden.
Die Frage die sich nach dieser optischen Täuschung stellt ist; werden die durch die menschlichen Sinnesorgane aufgenommen Informationen über unsere Umwelt lediglich aufgezeichnet und daraus ein Modell der Realität geschaffen oder findet eine unterbewusste Interpretation der erhalten Informationen statt.
Hierfür soll noch einmal ein Beispiel herangezogen werden.

Abbildung 2 - YouTube-Video von Benutzer brusspup [2.21]

Wie man in Abbildung 2 unschwer erkennen kann, handelt es sich hierbei um einen stilisierten Dinosaurier. Aber betrachten wir noch einmal eine andere Perspektive, sichtbar in Abbildung 3.

Abbildung 3 - YouTube- Video von Benutzer brusspup [2.21]

Der Effekt der sich hier beobachten lässt ist der Effekt der Tiefenumkehrung [2.48]. Damit dieser Sachverhalt verständlicher wird, soll nun der Versuchsaufbau diskutiert werden.

Da diese optische Täuschung in Form eines Bildes präsentiert wurde, wurde die durch unsere Sinnesorgane gegebenen Fähigkeiten schon einmal stark beschränkt.
Durch die zweidimensionale Natur von Fotos kann das räumliche Sehen nicht angewendet werden. Aber dies ist kein so entscheidender Faktor, für diese Illusion, wie man meinen sollte, denn dieser Effekt tritt auch bei der Betrachtung von dreidimensionalen Objekten auf.

Bei der Interpretation von Objekten durch unser Gehirn spielen Schattierungen eine große Rolle [2.22]. Je nachdem wie die Schattierungen auf einem Körper anliegen, werden sie von unserem Gehirn interpretiert. Vilayanur S. Ramachandran [2.23] geht davon aus, dass wenn Informationen zur Beleuchtung der Szenerie fehlen oder unzureichend sind, unser Gehirn evolutionsbedingt automatisch eine Lichtquelle von oben annimmt. Wenn mach sich das Video von YouTube-Benutzer brusspup noch einmal betrachtet sieht man, dass in Abbildung 3 der Dinosaurier von unten angestrahlt wird und somit eine falsche Auswertung durch das Gehirn weiter begünstigten wird.

Aber warum wird dieser eindeutig konkave Dinosaurierkopf von unserem Gehirn als konvex wahrgenommen? Auch hierfür sieht Vilayanur S. Ramachandran einen evolutionären Grund. Denn in unsere Umwelt befinden sich mehr konvexe anstatt konkave Objekte, so dass im Zweifelsfall erst einmal ein konvexes Objekt angenommen wird.

Es lässt sich also feststellen, dass das Gehirn nicht einfach Informationen sammelt und dann daraus ein Modell der aktuellen Situation generiert, sondern es werden Annahmen über eingetretene Sachverhalte getroffen.
Dies ist besonders gut sichtbar, wenn wie in dem gezeigten Beispiel Informationen die vom Gehirn benötigt werden, um zu einer richtigen Interpretation der Situation zu gelangen, fehlen. Das unterbewusste Gehirn gibt in diesem Fall eine bestmögliche Schätzung ab und präsentiert sie unserem Bewusstsein.

2.4.6 Zusammenarbeit von Sinnesorganen

Für ein besseres Verständnis, wie das Gehirn Informationen verarbeitet, ist es wichtig zu verstehen, dass zumeist nicht ein Sinnesorgane vom Gehirn zur Bewertung eines Sachverhaltes verwendet wird, sondern vielmehr eine Verbund von Sinnesorganen, die erhalten Informationen interpretieren.

Zur Veranschaulichung soll hierbei der McGurk-Effekt vorgestellt werden [2.24].
Bei diesem Experiment wurden Probanden Videos von Personen gezeigt, die immer wieder dieselbe Silbe wiederholten, wie zum Beispiel ba-ba. Danach wurde dem Probanden dieses

Video erneut gezeigt, jedoch wurde die Tonspur hierbei ausgewechselt, so dass auf der Videospur nach wie vor ein ba-ba zu sehen ist, jedoch auf der Audiospur ein ga-ga ausgesprochen wird. Interessanterweise wurde dies in 98 Prozent der Fälle von den Probanden als ein da-da verstanden. Teilweise ließ sich bei der Verwendung von anderen Buchstabenkombinationen gar ein Überschreiben der akustischen Wahrnehmung durch die visuelle Wahrnehmung feststellen.

Hiermit lässt sich sehr gut verdeutlichen, wie der Sehsinn und der Gehörsinn bei dieser Problemstellung zusammenarbeiten und man sieht; wenn die empfangenen Informationen widersprüchlich sind, können unter Umständen sogar neue Sachverhalte entstehen die so nie passiert sind.
Auf der anderen Seite sieht man jedoch auch, dass es in bestimmten Situationen eine Art hierarchische Ordnung zu geben scheint, mit der das Gehirn die Informationen, die es zur Bewertungen der Situation verwendet, heranzieht.

Einen Link zu einer Webseite die diesen Effekt präsentiert findet sich unter [2.25].

2.4.7 Was ist also Realität in Bezug auf VR-Technik?

Durch das in den vorherigen Abschnitten vermittelte Wissen soll nun ein erneuter Versuch unternommen werden das Wort Realität im Kontext von VR-Technologie zu definieren.
Netterweise kommt uns an dieser Stelle die Popkultur zur Hilfe, in Form eines Zitates aus dem Spielfilm *Matrix*.

"Was ist Realität? Wenn du darunter verstehst, was du fühlst, was du riechen, schmecken oder sehen kannst, ist die Wirklichkeit nichts weiter als elektrische Signale interpretiert von deinem Verstand." – Morpheus [2.31]

Hierfür werden noch einmal die vorherigen Punkte rekapituliert.

Es wurde gezeigt, dass die menschliche Wahrnehmung oder das Modell, aus dem die menschliche Realität erschaffen wird, nur durch die Informationen unserer Sinnesorgane erzeugt wird und so nur einen Teil der tatsächlichen Realität widerspiegeln kann.
Es wurde des Weiteren gezeigt, dass die meisten Prozesse im Gehirn unterbewusst ausgeführt werden und wenn Informationen zur Bewertung fehlen, wird eine Schätzung bzw. Interpretationen durch das Gehirn abgegeben. Anhand der Beispiele wurde auch gezeigt, dass diese Schätzung nicht immer der tatsächlichen Realität entspricht.

Dieses Zitat ist für die Definition des Begriffes Realität, in Bezug auf VR-Technik, mehr als ausreichend. Dies kann im Umkehrschluss bedeuten, dass eine virtuelle Realität zu einer realen Realität werden kann, wenn alle menschlichen Sinnesorgane von der virtuellen Welt

so stimuliert werden, dass die unterbewusst arbeitenden Mechanismen des Hirns annehmen, dass sie aus einer realen Welt stammen.

2.5 Der Präsenz-Effekt

Der Effekt Präsenz entsteht vermutlich dadurch, dass wenn ein menschliches Gehirn durch seine Sinnesorgane mit Signalen aus einer virtuellen Welt so stimuliert wird, die unterbewusst arbeitenden Prozesse des Gehirns, diese Signale für Signale aus der realen Welt halten.

VR-Brillen stimulieren zumeist nur die visuelle Komponente unserer Sinnesorgane. Wie mit Hilfe des McGurk Effektes dargestellt werden konnte, scheint die visuelle Komponente in zumindest einigen Fällen in unserem Gehirn eine höhere Priorität als die akustische Komponente zu besitzen.
Es ist auch noch nicht geklärt, ob der Präsenz-Effekt proportional zur Anzahl der stimulierten Sinnesorgane steigt, oder ob die visuelle Komponente hierfür allein verantwortlich ist.

Bevor die Details von Präsenz besprochen werden, soll die Mächtigkeit dieses Effektes demonstriert werden. Das folgende Bild stammt aus dem Vortag von Michael Abrash auf dem Steam Dev Days 2014 [2.13]

Abbildung 4 - Michael Abrash, Steam Dev Days 2014 [2.13]

Abbildung 4 zeigt einen eher abstrakten Sachverhalt. Geometrisch betrachtet handelt es sich um einen Würfel (Stein-Textur) in einem Würfel (Facebook-Webseite-Textur). Wie sich unschwer erkennen lässt ist dies ein Sachverhalt der in der Realität eher höchst unwahrscheinlich anzutreffen ist. Jedoch wenn die unterbewussten Mechanismen des

Gehirns sich entscheiden, dass dieser Sachverhalt real ist wird auch der Präsenz-Effekt erzeugt. Dann erlebt der Anwender dasselbe mulmige Gefühl, als wenn er von einem hohen Häuserdach herunterblickt.

Im Detail lässt sich das wie folgt erklären: Die Ausgangssituation ist, dass ein Proband eine VR-Brille trägt. Seine Position befindet sich auf dem Würfel mit der Steintextur. Der Proband senkt den Kopf und in Folge dessen erlebt er dasselbe mulmige Gefühl, als wenn er in der Realität von einem hohen Haus blickt.

Alle für diese Handlung benötigten Sinne werden korrekt stimuliert:

- Der Gleichgewichtsinn registriert, dass wir sitzen oder stehen
- Die Tiefensensibilität registriert, dass unser Kopf nach vorne geneigt ist.
- Der Sehsinn registriert durch die Tiefenwahrnehmung eine große Entfernung zum Boden.

Der Präsenz-Effekt kann also auch entstehen, wenn die betrachtete Szenerie eine komplett abstrakte ist. Dies ist möglich, da das Gehirn scheinbar mit allen benötigten Informationen die zur Bewertung der Situation benötigt werden, stimuliert wurde.

2.5.1 Arten von Präsenz

Über die Arten von Präsenz gibt es noch keine einheitliche Meinung. Martijn J. Schuemie [2.32] kommt zu dem Schluss, dass es zwar eine Menge von Modellen gibt die versuchen verschieden Arten von Präsenz zu beschreiben und dass diese sich auch nicht unbedingt gegenseitig ausschließen. Jedoch gehen sie jeweils von unterschiedlichen Implikationen aus. Wenn man Präsenz versucht zu messen hängt dies dann von dem jeweils verwendeten Modell ab.
Kalawsky [2.33] warnt sogar davor, dass Präsenz ein multidimensionaler Parameter sei der vielmehr einen Verbund einer Reihe von psychologischen Faktoren darstelle.

2.5.2 Auswirkung von Präsenz

Wenn im Allgemeinen von Präsenz gesprochen wird, wird der Effekt salopp durch den Satz „sich dort befinden“ beschrieben. Für eine etwas genauere Analyse soll der Effekt jetzt genauer untersucht werden, um die Auswirkungen von Präsenz besser verstehen zu können.

Subjektive Empfindung: Wie bereits erwähnt ist das „sich dort befinden“ die am meisten verwendet Definition von Präsenz, jedoch ist es auch Interessant zu sehen, wie sich Probanden an die von ihnen erlebte Präsenz-Erfahrung erinnern. Slater [2.34] spricht davon, dass das Schlüssel-Resultat von Präsenz sei, dass sich die Probanden, wenn sie sich

an die die virtuelle Welt zurück erinnern, eher an einen Platz denken, als eine Abfolge von Bildern.

Produktivität: Häufig wird angenommen, dass Präsenz sich auf die Produktivität in einer virtuellen Welt auswirkt und diese positive begünstigt. Dies ist aber wissenschaftlich nicht eindeutig bewiesen. Ellis [2.35] kommt sogar zu dem Schluss, dass eine Abstraktion der zu bewältigenden Aufgabe, auf Kosten der Präsenz, zu bevorzugen sei.
Auch das Lernverhalten scheint sich nicht durch die Anwesenheit eines Präsenz-Effektes verbessern zu lassen.
Hierfür wurde von Mania und Chalmers [2.36] ein Experiment durchgeführt, bei dem ein Seminar in einer präsenzerzeugenden Umgebend abgehalten wurde. Danach wurde dieses Seminar erneut ohne präsenzerzeugende Mittel durchgeführt. Dabei konnte keine Verbesserung gegenüber der nicht-präsenzerzeugenden Umgebung festgestellt werden.

Emotionen: Dies ist vermutlich eine der größten Auswirkungen von Präsenz. Mit Präsenz ist es möglich, künstlich dieselben Emotionen zu erzeugen, wie die in der realen Welt. So wurden von Hodges [2.37] erfolgreich mehrere Probanden mit Hilfe des Präsenz-Effektes gegen Höhenangst therapiert.
Diese Ergebnisse werden von anderen Forschungsgruppen mit anderen Phobien bestätigt.

2.6 Simulator Sickness

Simulator Sickness ist ein Phänomen, das durch einen Simulator jeglicher Art ausgelöst werden kann. Von den Symptomen her ähnelt es der Seekrankheit. Hierbei erlebt der Anwender Zustände von Übelkeit, Desorientierung, Schläfrigkeit und/oder generelles Unwohlsein [2.27].
Simulator Sickness entsteht, wenn das Gehirn widersprüchliche Informationen von seinen Sinnesorganen erhält. Im Detail bedeutet das, wenn die vom Auge gesammelten Informationen, die in diesem Fall aus einer virtuellen Welt entstammen, nicht mit den Informationen des vestibulären Sinns und der Tiefensensibilität übereinstimmen [2.28].
Beispielsweise könnte dies durch eine einfache Vorwärtsbewegung in der virtuellen Welt ausgelöst werden.
Simulator Sickness und Präsenz stehen meist in enger Verbindung. Die Abwesenheit des einen bedeutet meist die Anwesenheit des andern. Wenn Präsenz in einer VR-Anwendung nicht vorhanden ist bedeutet das nicht nur, dass sie keine Präsenz erzeugt wird, sondern auch, dass sie in diesen Fall häufig Simulator Sickness auftritt.

2.6.1 Faktoren von Simulator Sickness

Während die genauen Prozesse wie Simulator Sickness entsteht nicht abschließend geklärt sind, lassen sich jedoch drei generelle Faktoren, die zu Simulator Sickness führen feststellen

[2.27]. Zum besseren Verständnis werden diese mit Beispielen aus dem Oculus Rift Best Practices Guide erweitert [2.28].

1.) Simulator Design: Hierbei entsteht die Simulator Sickness durch Fehler in Spezifikation, Entwurf und Konfiguration des Simulators selbst.

Stereoskopisches Rendering: Generelle Unverträglichkeit des Anwenders gegenüber so erzeugte Bildfolgen.
Sichtfeld: Diskrepanz zwischen Sichtfeld der verwendeten Hardware und Sichtfeld der verwendeten Kamera der 3D-Engine.
Latenz: Verzögerung zwischen Eingabe und Ausgabe.
Distortion Korrektion: Falsche Anpassung von Programm-Shadern
Flicker: Stroboskopartige Abfolge von Bildern bei zu langsamer Bildwiederholungsfrequenz.

2.) Simulator Exposure: Hierbei entsteht die Simulator Sickness durch die im Simulator durchgeführten Anwendungsszenarien.

Beschleunigung: Eine starke Beschleunigung, von einem Stillstand bis zur Endgeschwindigkeit.
Kontrolle: Ein Anwender empfindet eher Simulator Sickness, wenn er keine Kontrolle über die virtuelle Situation hat.
Dauer: Unter Umständen kann alleine die Dauer, die in der Simulation verbracht wird, zur Simulator Sickness führen.
Lage: Ungewohnte Lage, die Größe des Avatars in VR 10 cm, Größe es Anwenders in der Realität 180 cm.

3.) Schlussendlich spielt natürlich auch noch die persönliche Veranlagung eine Rolle dabei, in welcher Intensität Simulator Sickness wahrgenommen wird.

2.6.2 Lösungsansätze

Generell kann Simulator Sickness dadurch reduziert werden, dass die Parameter im Simulator Design und im Simulator Exposure richtig gesetzt sind, also dass Hardware-Design und die für die Hardware entwickelten Inhalte VR-verträglichen Konventionen entsprechen.
An dieser Stelle sei noch einmal daraufhin hingewiesen, dass diese Konventionen bei weitem noch nicht ausformuliert sind und so noch regelmäßige Neuentdeckungen in diesem Bereich gemacht werden.
Aber es gibt 2 wesentlich trivialere Möglichkeiten um das Entstehen von Simulator Sickness zu minimieren.
Die erste Möglichkeit ist das Hinzufügen eines statischen Hintergrundes [2.28], der in seiner Orientierung der realen Welt gleicht. Dies ist heutzutage gängige Praxis in vielen 3D-

Computerspielen und kann durch das Hinzufügen einer Skybox oder eines Cockpits erreicht werden.
Eine zweite vielversprechende Möglichkeit ist das Hinzufügen einer virtuellen Nase [2.30]. Anfängliche Test zeigen bereits, dass sich durch das Hinzufügen einer virtuellen Nase die Simulator Sickness um 13.5 Prozent senken lässt. Dabei bemerkten die Probanden diese virtuelle Nase nicht einmal.

2.7 Technische Voraussetzungen für Präsenz

Um den Effekt der Präsenz erzeugen zu können, müssen eine Reihe von technischen Faktoren erfüllt seine. Michael Abrash [2.13] stellt hierfür einen Forderungskatalog auf, der im nachfolgenden detailliert beschrieben und erläutert wird. Nach Abrash müssen dabei alle folgenden Forderungen zu einem gewissen Grad erfüllt sein. Wenn dies nicht der Fall ist, ist das Auftreten von Simulator Sickness sehr wahrscheinlich.

2.7.1 Sichtfeld (Field of View)

Dies bezeichnet die Größe der Fläche des Blickwinkels eines optischen Geräts, innerhalb dessen Ereignisse oder Veränderungen wahrgenommen, aufgezeichnet und verarbeitet werden können.

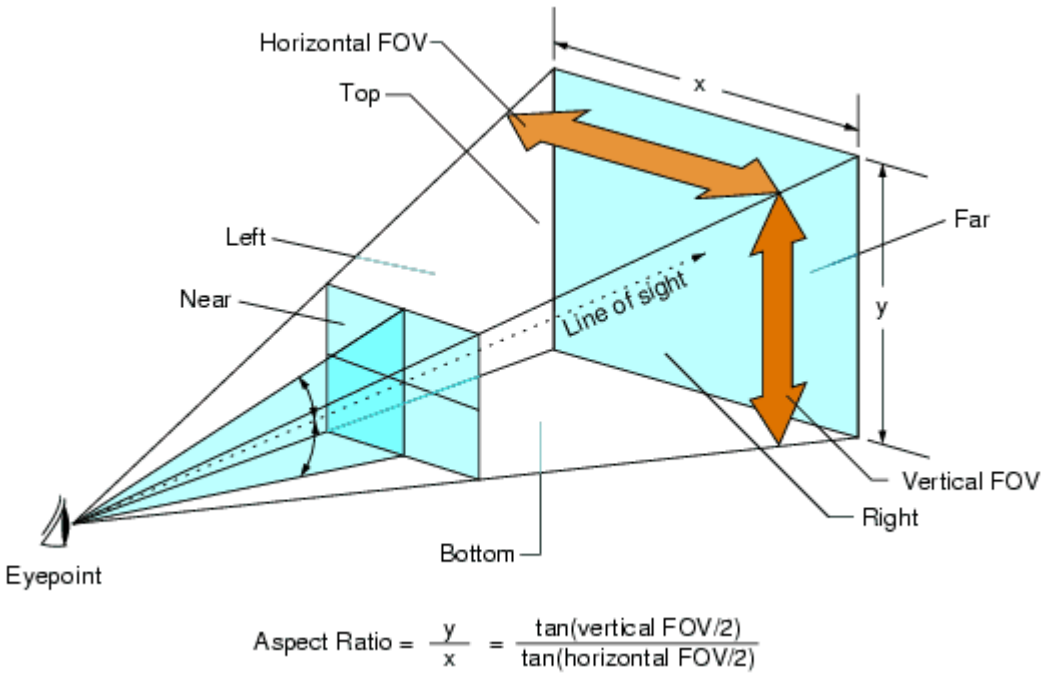

Abbildung 5 - Horizontales und vertikales Sichtfeld [2.14]

Die Größe des Sichtfeldes auf eine virtuelle Welt ist einer der entscheidenden Faktoren die benötigt werden, um den Präsenz-Effekt zu erzeugen. Abrash [2.13] stellt fest, dass der Präsenz-Effekt sich ab einem 80° Sichtfeld einstellt und seine Intensität bis 110° stetig zunimmt.
Im Vergleich dazu verfügt ein handelsüblicher 30 Zoll Computer-Monitor, bei normalem Sitzabstand, mit einer Auflösung von 2560 x 1600 über ein Sichtfeld von 50°.

2.7.2 Adäquate Auflösung

Die Auflösung beschreibt die Anzahl der vertikalen und horizontalen verfügbaren Bildpunkte, die zur Darstellung von Inhalten auf einem Anzeigegerät genutzt werden können. Umgangssprachlich ist des Öfteren von einer HD Ready bzw. Full HD Auflösung die Rede.
Dies beschreibt bei HD Ready eine Anzahl von 1280 × 720 Bildpunkten und bei Full HD 1920 x 1080 Bildpunkten [2.11].
Auflösung wird seit jeher als eines der tragenden Qualitätsmerkale herangezogen um die Qualität eines computergenerierten Bildes zu bewerten. Jedoch muss durch das von VR-Technik benötigte, auf 100° erweiterte Sichtfeld, zusätzlich noch ein weiterer Einflussfaktor addiert werden. Die Pixeldichte. Hierzu soll ein Rechenbeispiel die Problematik verdeutlichen [2.45].

Ein 30 Zoll Monitor mit einer Auflösung von 2560 x 1600 Pixeln füllt das Sichtfeld des Benutzers um 50 Grad. Das ergibt rund 51 Pixel pro Grad im horizontalen Sichtfeld und 32 im vertikalen Schichtfeld.

Betrachtet wird nun eine VR-Brille, die mit einer Full HD Auflösung arbeiten. Dies bedeutet dann 960 x 1080 pro Auge und das bei einem Sichtfeld von 100°. Das ergibt rund 10 Pixel pro Grad im horizontalen Sichtfeld und rund 11 im vertikalen Schichtfeld.

Rechnet man diese wieder auf ein Sichtfeld von 50° um ergibt sich eine Auflösung von 500 x 550 Pixeln auf einem normalen Monitor.

Dieser Effekt kommt zustande, weil das Sichtfeld bei den meisten VR-Brillen durch eine Linse auf die gewünschte Sichtfeld-Größe aufgefächert wird und somit auch die einzelnen Pixel. An Hand es Beispiels wird verständlich, dass VR-Brillen eine vielfach höhere Auflösung besitzen müssen um dieselbe Brillanz eines normalen Monitors bieten zu können. Abrash geht davon aus, dass eine Auflösung von 960 x 1080 pro Auge für den Anfang genügt [2.13].

2.7.3 Geringe Pixel Persistenz

Dies beschreibt die Dauer der Beleuchtung eines Pixels während eines Frames. Die folgenden Angaben sind aus dem Artikel „Why virtual isn't real to your brain: judder" von Michael Abrash entnommen [2.12]. Frames beschreiben die Zeit, die zwischen dem Anzeigen von zwei hintereinander computergenerierten Bildern vergeht. Hierbei wird in der Einheit Hz gemessen.
Viele der heutzutage erhältlichen OLED und LCD Bildschirme benutzen volle Persistenz. Dies bedeutet, dass die Pixel des Bildschirms während der kompletten Zeit eines Frames beleuchtet bleiben. Danach erlöschen sie und das nächste Bild wird aufgebaut.
Bei halber Persistenz Verfahren bleiben sie entsprechend die halbe Dauer eines Frames beleuchtet.
Eine dritte Variante ist die null Persistenz verfahren, hierbei leuchten die Pixel nur kurz in einer hohen Lichtstärke auf und bleiben dann für die restliche Zeit des Frames dunkel.
Das Problem was mit der Forderung nach einem Bildschirm mit geringer Pixel Persistenz gelöst werden soll, ist Judder.

Abbildung 6 - Beispiel Judder Abrash [2.13]

Man kann in Abbildung 6 erkennen, dass Judder eine ungewollte Form von Bewegungsunschärfe ist. Judder entsteht durch eine Augenbewegung relative zu einem Bildschirm. Ist das Sichtfeld so groß wie bei VR-Brillen, äußert sich dieser Effekt durch ein Verwischen des Bildes.
Dies entsteht dadurch, dass das Licht der einzelnen Pixel, durch die Augenbewegung verschiedene Teile der Retina trifft. Die Stärke des Effektes resultiert dabei aus der absoluten Zeit, die die Pixel beleuchtet bleiben. Ein Bildschirm mit voller Persistenz benötigt nach Abrash schätzungsweise eine Bildwiederholungsfrequenz von 1000 Hz um diesen Effekt auszumerzen. Dies ist jedoch alles andere als praktikable. Daher sind Bildschirme mit einem null Persistenz Verfahren zu bevorzugen. Die Pixel werden nur kurz beleuchtet und eine mögliche Augenbewegung findet somit eher bei einem abgedunkelten Zustand des Bildschirms, statt.

2.7.4 Globales Display

Die folgenden Angaben wurden aus dem Artikel, *Raster-Scan Displays: More Than Meets The Eye* [2.46] von Michael Abrash entnommen.
Prinzipiell gibt es zwei Möglichkeiten, wie ein Bildschirm pro Bild seine Pixel aufbauen kann. Die erste ist Raster Scanning, hierbei werden die Pixel nacheinander aktualisiert.

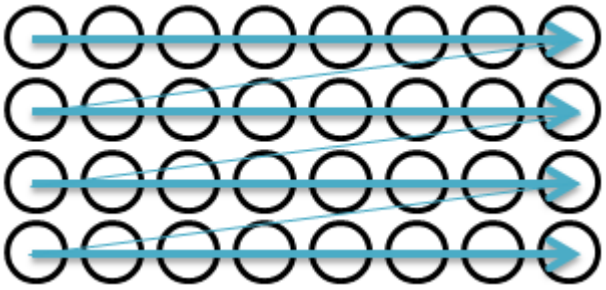

Abbildung 7 - Beispiel für die Aktualisierung von einem Rasterdisplay [2.46]

Dies hat zur Folge, dass der Pixel in der rechten unteren Ecke ein paar Millisekunden später aktualisiert wird als der links oben. Dieses Verfahren stammt noch aus der Zeit der Kathoden-Strahl-Röhren und repräsentiert den Weg des Kathoden-Strahls bei der Erstellung eines Bildes. Tatsächlich wird dieses Verfahren auch noch heute in Zeiten von LCD und LED häufig verwendet.

Globale Displays [2.46] aktualisieren alle Pixel auf einmal. Hierzu wird gewartet, bis die kompletten Pixel-Daten des Frames transferiert wurden und dann werden alle Pixel des Bildschirms kurz abgedunkelt und aktualisiert neu beleuchtet.

Globale Displays sind aus dem Grund zu bevorzugen, weil die Raster-Scanning Variante in vielerlei Hinsicht eine stärkere Artefakt Bildung im jeweiligen Bild verursachen kann. Dieser Effekt wird bei VR-Brillen noch verstärkt durch mögliche Augen- und Kopfbewegung.

2.7.5 Optik

Obwohl bei den heute verfügbaren VR-Brillen in dem meisten Fällen Plastiklinsen beiliegen die ihre auf Aufgabe scheinbar zuverlässig erledigen, gibt es an dieser Stelle viel Platz für Verbesserungen. Jedoch sind momentan die Linsen-Modelle die dieses Problem optimal lösen können viel zu schwer, um sie in ein Headset zu verbauen [2.13]. Die Linse die das Problem der Lichtbrechung optimal löst, benötigt 9 in sich geschachtelte Linsen [2.13].

2.7.6 Optische Kalibrierung

Es hat sich heraus gestellt, dass optische Kalibrierung einen großen Anteil bei der Erzeugung von Präsenz hat [2.13]. Der menschliche Sehsinn ist sehr sensibel gegenüber kleinsten Fehlern, besonders wenn gerade Linien und Bewegung beteiligt sind [2.13]. Eine VR-Szene kann, wenn statisch, also ohne Bewegung betrachtet, ansprechend erscheinen jedoch bei Kopfbewegung, sofort den Präsenz-Effekt zerstören. Abrash [2.13] schlussfolgert, dass die Abstimmung der Linsen und des computergenerierten Bildes genauestens aufeinander abgestimmt werden müssen.

2.7.7 Tracking

Durch die Tracking-Komponente der VR-Brille werden die Kopfbewegungen des Anwenders ermittelt. Diese muss mindestens Bewegungen auf der X, Y und Z Achse erfassen, fordert Abrash [2.13]. Iribe [2.47] fordert zusätzlich ein Tracking auf sechs Achsen, so dass auch die Position im Raum bestimmt werden kann und dies bei einem Radius von 360°. Beide veranschlagen dieselbe Genauigkeit, bei Positionsbestimmung im mm Bereich, bei Drehung 0.25° genau und dies in einem Radius von 1.5 Metern auf jeder Seite des Kopfes.

2.7.8 Geringe Latenz

Dies bedeutet, dass die Zeit in der eine Bewegung, wie zum Beispiel eine Kopfdrehung, ausgeführt wird, über die Zeit die das System braucht um daraus ein Bild zu generieren, bis zu dem Moment in dem die Photonen die Retina treffen möglichst gering sein sollte.
Um den Effekt der Präsenz aufrecht zu erhalten muss das korrespondierende Bild zur Bewegung zeitnah erfolgen. Abrash [2.13] veranschlagt hierfür 20 Millisekunden.

2.7.9 Hohe Bildwiederholungsraten

Die Forderung nach hohen Bildwiederholungsraten resultiert aus der Forderung nach niedriger Pixel-Persistenz. Dadurch, dass die Pixel nur einen kurzen Moment während des Frames beleuchtet sind, beginnt diese Art von Bildschirm stärker zu flackern, als ein Voll-Persistenz-Bildschirm bei gleicher Bildwiederholungsfrequenz.
Abrash [2.13] bestätigt, dass eine Bildwiederholungsfrequenz von 90Hz ausreicht, um jegliches Flackern zu eliminieren.

3 Anforderungsanalyse und Spezifikation

In diesem Kapitel werden die Anforderungen erfasst, die für die Spezifikation und Entwicklung einer dreidimensionalen graphischen Oberfläche benötigt werden. Hierfür wird zunächst eine Analyse durchgeführt, die die grundlegenden Eigenschaften der Interaktion mit graphischen Oberflächen verdeutlichen. Danach werden die Teilaspekte der zu erstellenden dreidimensionalen graphischen Oberfläche nacheinander spezifiziert.

3.1 Ausgangssituation

In der Informatik ist Moors Gesetz [3.14] allgegenwärtig. Es besagt, dass sich alle 18 Monate die Anzahl der Transistoren verdoppelt. Dies lässt auch indirekt Rückschlüsse auf die gesamte Leistungssteigerung des Computers zu. Natürlich spielen bei der Bewertung der Gesamtleistung eines Computersystems auch noch andere Faktoren eine Rolle. Aber insgesamt lässt sich feststellen, dass die Rechenleistung steigt und somit im Umkehrschluss auch immer billiger wird.

Als Anfang der 1990er graphische Oberflächen immer beliebter wurden, brachten sie die damalige gängige PC Hardware an ihre Leistungsgrenzen. Heute, 20 Jahre später, hat sich rein äußerlich nicht viel an diesen Konzepten verändert. Zwar hat sich die Auslösung vervielfältigt und der Einsatz von Vektorgraphiken hat das graphische Niveau angenehm erhöht.
Wenn man sich jedoch einmal die Systemauslastung, mit Hilfe der auf jedem Betriebssystem verfügbaren Tools ansieht stellt man schnell fest, dass der Computer während der täglichen Arbeit meist einen geringen Prozentsatz seiner möglichen Leistung nutzt. Natürlich gibt es Programme die sehr viel Rechenleistung benötigen, wie Videobearbeitung, CAD Anwendungen oder Computerspiele, aber es sind Anwendungen

wie Browser, Textverarbeitung, Chatclients und Mailprogramme die heutzutage das definieren, was man unter „tägliche Arbeit mit dem Computer“ versteht.
Im Großen und Ganzen bedeutet es, dass in vielen Computersystemen mögliche Leistung ungenutzt brach liegt. Aber was ist, wenn man diese ungenutzten Ressourcen nutz, um sie für ein technisch aufwendigeres, jedoch für den Benutzer intuitiveres Bedienkonzept zu nutzen? Angelehnt an dem Konzept, wie der Benutzer mit seiner Umwelt interagiert.
Für die Realisierung eines solchen intuitiveren Bedienkonzepts gibt es drei interessante Forschungsfelder, die solche Anforderungen erfüllen könnten.
Das erste Gebiet wäre die 3D-Graphik auf Basis von polygonalen Netzen. Das zweite Gebiet ist die Forschung an Bewegungssensoren und das drittes die Forschung an VR-Technik.

Während vor 25 Jahren die 3D-Graphik noch in den Kinderschuhen steckte und die durch sie produzierten Objekt eher abstrakter Natur waren, ist es heutzutage möglich, komplexe Objekte zu erschaffen, die von ihrem realen Ebenbild nicht mehr zu unterscheiden sind.
Bewegungssensoren auf der anderen Seite sind heute so genau, dass sie einzelne Fingerglieder messen können.
Der Reifegrad der VR-Forschung hat natürlich noch nicht die Fortschritte erzielt, wie die anderen beiden Forschungsbereiche, aber die Möglichkeit mit VR Technik den Effekt Präsenz erzeugen zu können ist etwas, das mit keinem anderen uns bekannten Medium möglich ist und wenn sich die bereits angesprochen Kinderkrankheiten eliminieren lassen, kann die VR Technik, die Art wie wir mit dem Computer interagieren, nachhaltig verändern.
Wenn man jetzt diese 3 Forschungsfelder miteinander verknüpft lässt sich daraus eine graphische Oberfläche erzeugen, die den Benutzer in eine 3D-Welt versetzt, wo er einen 360° rundum Blick besitzt und durch seine eigenen Hände mit dem System interagiert.
Dies ist für die Informatik eine interessante Möglichkeit die Interaktion mit dem Computer noch natürlicher zu gestalten als es bisher der Fall ist. So lassen graphische Elemente auch in der Tiefe des Raumes verschieben und um 360 Grad um den Benutzer herum anordnen und orientiert sich damit, weitaus näher als eine Maus oder Tastatur, an den Interaktionsvorgängen der realen Welt.

Sicherlich ist es fraglich, ob ein heutiger Office PC dazu imstande ist, die Leistungsfaktoren von Bewegungstracking, VR und 3D-Graphik zufriedenstellend zu bedienen. Aber das bringt uns zurück zu Moors Gesetz. Wenn wie heute noch nicht genügend Rechenleistung haben, haben wir sie vielleicht schon in 18 Monaten.

3.2 Analyse

Zuerst soll eine Analyse durchgeführt werden die grob erklärt, was die grundsätzliche Interaktion mit einer graphischen Oberfläche ausmacht. Sie fungiert als eine Art Bestandsaufnahme und definiert die Hauptziele, die mit der Entwicklung des VR-Desktops erreicht werden sollen.

Hierfür wird als erstes eine Ist-Analyse durchgeführt, die sich mit gegenwärtig verfügbaren Lösungen beschäftigt. Danach folgt die Soll-Analyse, die mögliche Verbesserungen zum Ist-Zustand aufzeigt. Für die Analyse wird die Sicht des Anwenders eingenommen, technische Aspekte nehmen hierbei eine untergeordnete Rolle ein.

3.2.1 Ist-Analyse

Visualisierung: Heutzutage benutzen alle Arten von Computern in irgendeiner Form, egal ob nun Desktop PC, Laptop oder Smartphone, einen Monitor zur Ausgabe von Inhalten.
Der Benutzer blickt auf einen Monitor, kann den Neigungswinkel und die Entfernung ändern. Dieser füllt hierbei, je nach Abstand und Neigungswinkel zum Benutzer, einen Teil des Sichtfeldes aus.
Die Ausgabe der verarbeiteten Inhalte erfolgt mit einer zweidimensionalen Repräsentation auf Basis von Pixel- und/oder Vektor-Grafik. Diese Elemente lassen sich auf einem zweidimensionalen kartesischen Koordinatensystem beliebig anordnen.

Organisation von graphischen Oberflächen: Zentrale Elemente von graphischen Oberflächen sind Fenster, die Menüleiste und die Desktopoberfläche.
In Fenstern werden Anwendungen eingebettet. Diese Anwendungen operieren dann auf den ihnen zugeordneten Daten.
Die Menüleiste gibt Übersicht über alle verfügbaren Programme. Sie kann kleine Anwenderprogramme, wie zum Beispiel eine Uhr, Datum oder aktuelles Wetter, aber auch Betriebssystem-organisatorische Inhalte enthalten.
Die Desktop-Oberfläche ist ein virtuelles Äquivalent zum echten Schreibtisch. Auf ihm können Daten, Programme und Verknüpfungen zu Programmen, die zum aktuellen Arbeiten benötigt werden abgelegt werden.

Bedienung: Die Interaktion mit dem Computer findet auf zweidimensionaler Basis statt. Egal ob nun klassisch mit Desktop und Laptop oder mit Touch Devices.
Der Benutzer agiert mit einer zweidimensionalen Oberfläche. Diese kann durch andere zweidimensional arbeitende Eingabegeräte unterstützt werden. Dieses System arbeitet mit einer X und Y Achse. Eingabegerät oder Finger bewegen sich entlang dieser beiden Achsen. Löst der Benutzer ein Klick- oder Touch-Ereignis aus überprüft das Betriebssystem, ob es Teil einer systemrelevanten Interaktion ist und reagiert dann entsprechend.

Leistungsmerkmale: Kompakte und effiziente zweidimensionale graphische Oberflächen, wie die neuste Windows Version, stehen bereits für Kleinstcomputern wie den Raspberry Pie zur Verfügung [3.3].
Graphische Oberflächen sind entweder mit dem darunterliegenden Betriebssystemkern verzahnt, wie im Falle von Windows, oder sie sind entkoppelt und können frei ausgetauscht werden, wie im Falle von Linux.

Sie bieten Programmierschnittstellen an, damit die Funktionalität der graphischen Oberfläche durch Programme erweitert werden kann.
Im Lieferumfang befinden sich Anwendungsprogramme die eine Grundfunktionalität für das tägliche Arbeiten bereitstellen.

3.2.2 Soll-Analyse

Visualisierung: Zur Darstellung wird eine VR-Brille benutzt. Hierfür ist es erst einmal egal, ob es sich um eine VR-Brille oder AR-Brille handelt. Das Sichtfeld des Benutzers wird somit durch die dreidimensionale graphische Oberfläche annähernd komplett ausgefüllt.
Korrespondierend zur Kopfbewegung wird die Sicht auf die graphische Oberfläche verändert. Das Headset bietet somit einen 360° rundum Blick.
Graphik auf Basis von polygonalen Netzen oder Trixeln. So lassen sich grundlegende Operation wie Translation, Skalierung und Rotation einfach anhand von mathematischen Funktionen durchführen.

Organisation der Graphischen Oberfläche: Die graphische Repräsentation wird um die dritte Dimension erweitert.
Zentrale Elemente und Kernkonzepte wie Fenster, Menüleiste und Desktop bleiben erhalten und übernehmen dieselbe Funktionalität.

Bedienung: Um für eine dreidimensionale graphische Oberfläche eine angemessen Bedienung zu bieten muss das bisher gültige Bedienungsschema für graphische Oberflächen um die Z Achse erweitert werden.
Die Z Achse sorgt dafür, dass der Benutzer mit der Tiefe des Raumes interagieren kann. Das neue Bedienkonzept soll sich dabei daran orientieren, wie ein Mensch in der realen Welt interagiert.

Leistungsmerkmale: Die Erzeugung von dreidimensionalen Graphiken stellen weitaus größere Anforderungen an die benötigten Hardware-Ressourcen als äquivalente zweidimensionale graphische Oberflächen.

3.2.3 Fazit

Damit eine dreidimensionale graphische Oberfläche, im folgenden VR-Desktop genannt, die auf Basis von VR-Brille, 3D Graphik und Bewegungstracking arbeitet, einen wirklichen Mehrwert bietet, muss die Bedienung mindestens genauso intuitiv und fehlerfrei funktionieren wie die mit der Maus.
Auch bei Touch-Interfaces fallen bei näherer Betrachtung schnell die Parallelen zur Maus auf. Problematisch ist, dass beide Verfahren lediglich auf zweidimensionaler Basis arbeiten.

Daher können nur Teile des Bedienschemas übernommen werden. Der Rest muss von Grund auf neu entwickelt werden.
Bewegungssensoren und VR sind Forschungsgebiete in denen aktiv Grundlagenforschung betrieben wird. Es gibt somit keine etablierten Standards und es ist wichtig, dass Überlegungen schnell vom Papier in ein lauffähiges System überführt werden können, da damit zu rechnen ist, dass sich theoretische Vorüberlegungen in der Praxis als problematisch erweisen. Um diese Situation zu erleichtern ist es nötig die erarbeiteten Ergebnisse möglichst schnell Quelloffen zu veröffentlichen, damit eine vielseitigere Diskussion in Gang gesetzt werden kann.
In Bezug auf die Funktionalität muss ein Mindestmaß an Anwendungen geboten werden, damit der VR-Desktop produktiv eingesetzt werden kann. Für die Erweiterung darüber hinaus muss eine API definiert werden, damit der VR-Desktop an die individuellen Bedürfnisse des Anwenders angepasst werden kann.
Schlussendlich ist es wichtig den VR-Desktop einem möglichst breiten Publikum zur Verfügung zu stellen.

Die 6 wichtigsten Aspekte der VR-Desktop-Entwicklung sind daher:

- Eine immersive und Präsenz erzeugende Benutzer-Erfahrung erschaffen.
- Entwicklung eines Bedienungsschemas mit VR-Brille und Bewegungssensoren.
- Wenn die Entwicklung weit genug fortgeschritten ist, eine Veröffentlichung als Open Source Projekt.
- Entwickelung einer VR-Desktop-API
- Hohe Portierbarkeit durch Abstraktion
- Effizienz durch Implementieren eines VR-Desktop Applikationslebenszyklus.
- Bereitstellen von grundlegenden Betriebssystemprogrammen

3.3 Zielbestimmung

3.3.1 Abgrenzungskriterien

Heutzutage ein Betriebssystem von Grund auf neu zu entwickeln ist ein ambitioniertes Unterfangen, daher ist es unabdingbar Abgrenzungskriterien zu untersuchen die bestimmen, was eigentlich tatsächlich selbst entwickelt werden muss und wann auf bereits bestehende Lösungen zurückgegriffen werden kann.
Die Verwendung von bestehenden Komponenten bietet einen Vorteil der häufig übersehen wird; bestehende Komponenten, die sich bereits im produktiven Einsatz befinden, haben bereits eine gewisse Reife. Sie sind bereits ausgiebig auf Funktionalität und Effizienz getestet und können so in das System übernommen werden ohne das befürchtet werden muss, dass sich die Systemstabilität gravierend verschlechtert.

Aus diesem Grund soll die folgende Funktionalität aus einem Betriebssystemkern entnommen und nicht selbst Implementiert werden:

Geräteverwaltung: Der Kernel stellt ein Treibermodell zur Verfügung, um von der in dem jeweiligen Rechner verbauten Hardwarekomponente zu abstrahieren und so das Ansprechen der Komponente zu vereinheitlichen.

I/0 Verwaltung: Regelt jegliche Operationen die mit Daten-Elementen in Verbindung stehen. Dies reicht über hardwarenahe Funktionen, wie Daten in dem persistenten Speicher organisiert sind, über sicherheitsrelevante Aspekte bis zu Zugriffsrechten des jeweiligen Benutzers.

Prozessverwaltung: Stellt jegliche Funktionalität bereit, die ein Programm bzw. ein Prozess in der Ausführung benötigt. Unter anderem wird ein Scheduler bereitgestellt der Multitasking überhaupt erst ermöglicht.

Speicherverwaltung: Stellt jede benötigte Funktion bereit, die zur Verwaltung des RAMs benötigt wird. Zum Beispiel das Swapping zur Erweiterung des RAMs auf einen virtuellen Speichers.

Somit lässt sich die erste zentrale, nicht funktionale Anforderung des VR-Desktops formulieren. Die Schnittstellen des Betriebssystemkerns, auf dem der VR-Desktop aufsetzt, müssen offen liegen, damit die benötigte Funktionalität entnommen werden kann.

3.3.2 Muss-Kriterien

Zunächst muss die Frage beantwortet werden, welche grundlegenden Gegebenheiten überhaupt erfüllt sein müssen, damit mit dem VR-Desktop gearbeitet werden kann. Hierfür werden die Ergebnisse der Soll-Analyse übernommen und erweitert.
Hierbei beschreiben die Muss-Kriterien lediglich den absoluten minimalen Funktionsumfang der erfüllt sein muss, damit der VR-Desktop überhaupt funktioniert. Für die Spezifikation dieser funktionalen Anforderungen wird die Sicht des Benutzers eingenommen.

Eine detailliertere, formale Auflistung dieser funktionalen Eigenschaften findet sich im separaten Anhang: Anforderungen. Zusätzlich befindet sich in diesem Anhang auch noch eine Reihe von Wunschkriterien. Diese sind zwar nicht für einen möglichen Betrieb elementar, jedoch sollte je nach Möglichkeit versucht werden, diese zu erfüllen.

Visualisierung: Diese erfolgt auf Basis einer Art VR- oder AR-Brille

Audio: Die Verwendung von akustischen Signalen bietet neben visuellen Hinweisen die Möglichkeit, den Benutzer über den Zustand einer Anwendung oder Funktion zu informieren.

Bewegungserkennung: Das Erkennen von Benutzerinteraktion erfolgt mithilfe von Kameras oder anderer Sensorik. Dabei ersetzen Hände und der Kopf die allgemein verwendeten zweidimensionalen Eingabegeräte.

Bedienkonzepte zur Interaktion mit dem Desktop: Damit das System auf Benutzerinteraktion reagieren kann muss ein Schema festgelegt werden, das Benutzergesten auf Betriebssystem-Funktionen abbildet.

Graphische Repräsentation: Die visuellen Bausteine und Fenster sind die VR-Desktop-Variante der graphischen Kernkonzepte, die jeder graphischen Oberfläche innewohnen.

Logische Repräsentation: Damit die graphischen Repräsentationen, eine durch den Computer verarbeitet bare, Programm technische Funktionalität erfüllen kann.

Fehler-Handhabung: Im Fehlerfall ist ein Anwender gezwungen Fehler auszuwerten, um diese zu beheben. Damit dies möglich ist müssen ihm verschiedene Funktionalitäten geboten werden.

System-Programme: Damit ein Betriebssystem für das tägliche Arbeiten eingesetzt werden kann, muss ein Mindestmaß an Funktionalität geboten werden. Diese Funktionalität wird durch die System-Programme bereitgestellt.

3.4 Produkteinsatz

Im folgenden Abschnitt soll darauf eingegangen werden, wie und von wem der VR-Desktop eingesetzt werden kann.

3.4.1 Anwendungsbereiche

Der VR-Desktop soll sowohl im akademischen, professionellen als auch im privaten Umfeld eingesetzt werden können. Die 360° Visualisierung der VR-Brille und die verwendete Bewegungserkennung sollen eine evolutionäre Weiterentwicklung zu den heute allgemeingültigen Bedienstandards von zweidimensionalen graphischen Oberflächen darstellen.

Der Desktop soll hierfür die grundlegenden Bedienelemente bzw. Funktionen, die für das Arbeiten und die Interaktion mit dem Computer benötigt werden bereitstellen, damit der als vollwertiger Ersatz für bestehende Desktop-Lösungen eingesetzt werden kann.

3.4.2 Zielgruppen

Akademischer Anwender: Allen voran momentan die wichtigste Benutzergruppe. Bei dreidimensionalen Oberflächen gibt es momentan keine etablierten Standards an die man sich halten könnte. Der VR-Desktop soll hierfür eine Diskussionsplattform bilden. So können an ihm mögliche Bedienschemata und graphische Konfigurationen ausprobiert und diskutiert werden.

Professionelle Anwender: Bei professioneller Anwendung profitiert der Anwender besonders von der Möglichkeit, die durch die graphischen Visualisierung eines 3D-Raumes gegeben werden und die dadurch übersichtlicheren Anordnungsmöglichkeiten.
Ein anderer großer Vorteil liegt bei CAD-Anwendungen, so lassen sich 3D Modelle noch während sie entstehen perspektivisch korrekt durchwandern.

Privater Anwender: Der private Anwender profitiert vorrangig von den durch VR-Technik erweiterten Multimedia-Fähigkeiten. So lassen sich 3D-Inhalte wie Spiele oder Filme nativ in 3D bzw. VR konsumieren. Beispielweise können Fotos, die mit einer stereoskopischen Kamera aufgenommen wurden, in 3D betrachtet werden.

3.4.3 Betriebsbedingungen

Der VR-Desktop kann in zwei Betriebsmodi eingesetzt werden:

Minimaler Modus: Dieser Modus benötigt Kopftracking und Handtracking. Der VR-Desktop wird in diesem Fall im Sitzen bedient. Dabei ist es unerheblich ob das System Zuhause, im Büro oder unterwegs genutzt wird.
Für diesen Modus benötigt der Benutzer eine VR- oder AR-Brille, die mit einer Tracking-Fähigkeit für Hand und Kopfbewegungen ausgestattet ist. Für diesen Modus müssen alle in 3.2.2 (Anhang Anforderungen) beschrieben funktionalen Anforderungen erfüllt sein.

Erweiterter Modus: Dieser Modus erweitert den minimalen Modus um eine Ganzkörper-Bewegungstracking-Komponente. Diese erfasst die Bewegung des Körpers auf der X, Y und Z Achse. Hierfür werden Sensoren entweder im Raum verteilt oder der Raum von der verwendeten VR-Brille gescannt. In diesem Modus ist es möglich, sich durch den virtuellen Raum zu bewegen. Hierbei ist zu beachten, dass der virtuelle Raum dem realen Raum von der Größe und Beschaffenheit her im Großen und Ganzen gleichen muss, um etwaigen

Verletzungen vorzubeugen. Dieser Modus muss zusätzlich zu den Anforderungen aus 3.2.2 auch 3.2.3.3 (Anhang Anforderungen) implementieren.

3.4.4 Anwendungsdauer

Da der VR-Desktop eine Alternative zu einer herkömmlichen graphischen Oberfläche darstellen soll, kann die Betriebszeit täglich im Bereich zwischen 8 und 12 Stunden liegen

3.5 Übersicht

Bisher wurden die einzelnen Anforderungen an den VR-Desktop und dessen Komponenten für sich isoliert betrachtet. In diesen Abschnitt soll daher eine Übersicht darüber geboten werden, wie der Anwender mit dem VR-Desktop interagiert und welche Komponenten daran beteiligt sind. Hierfür werden die Kernkonzepte textuell umrissen.

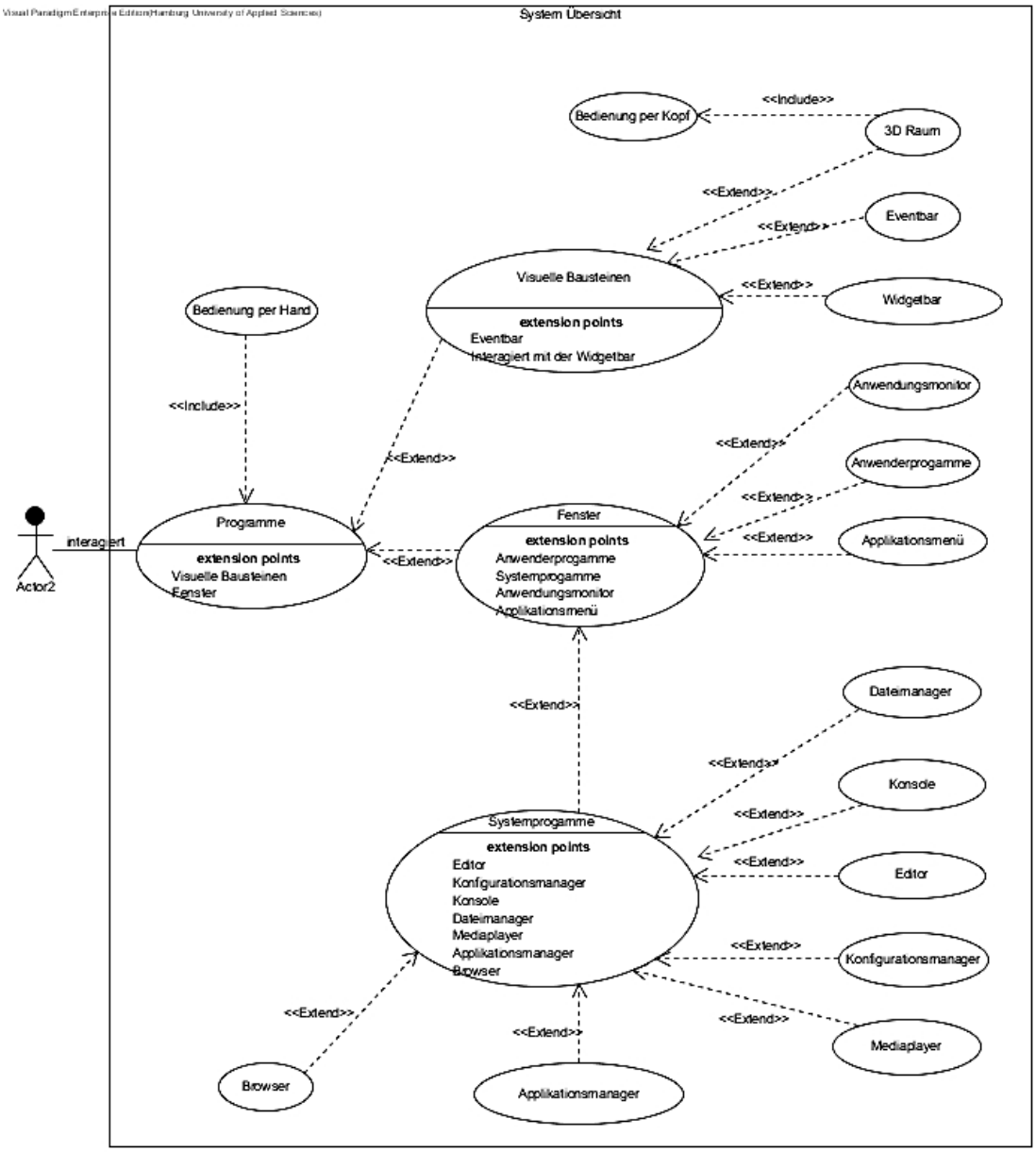

Abbildung 8 - Übersicht VR-Desktop

3.5.1 Programme

In der VR-Desktop-Spezifikation werden generell zwei Arten von Programmen unterschieden. Diese unterscheiden sich in der Form ihrer graphischen Repräsentation. Die erste Art sind die Fenster. Sie sind die graphische Standard-Repräsentation von Programmen. Die zweite Art sind die visuellen Bausteine. Diese besitzen keine fest vorgegebene graphische Struktur und repräsentieren VR-Desktop-spezifische graphische Konzepte.

3.5.2 Tracking Komponenten

Bedienung per Kopf: Bei Kopfbewegungen wird der 3D-Raum der Bewegung korrespondierend angepasst.

Bedienung per Hand: Um differenzierte Gesten an das System zu übergeben muss das Tracking auf Granularität der Finger erfolgen. Zur besseren Orientierung wird im 3D-Raum eine graphische Repräsentation der Hände angezeigt.

3.5.3 Fenster

Wie bei 2D-Oberflächen werden mit ihrer Hilfe Programme in die graphische Oberfläche eingebettet. Sie bieten somit ein Mittel zur graphischen Repräsentation von Programmen, die auf dem VR-Desktop ausgeführt werden sollen. Den folgenden auf Fenster basierenden Programmen kommt aus VR Desktopsicht eine dedizierte Aufgabe zu und sollen deshalb hier genannt werden.

Anwendungsprogramm: Die zur Verfügung stehenden Möglichkeiten der Visualisierung beschreibt ein vom VR-Desktop angebotenes Framework. Es wird durch Elemente wie Buttons, Listen und Tabellen beschrieben, die im Quellcode deklariert und dann mithilfe des VR-Desktop-Rendering-System visualisiert werden.

Anwendungsmonitor: Durch die Natur des 3D-Raums ist es möglich, Fenster in einen für den Benutzer toten Winkel zu verschieben. Um dieses Problem abzuschwächen bietet der Anwendungsmonitor eine Übersicht über alle momentan im 3D-Raum vorhanden Fenster an. Der Nutzer kann so das gewünschte Fenster zentriert zu seinem momentanen Blickwinkel anzeigen lassen. Zusätzlich lässt sich über ihn die Prozess-Priorität festlegen.

Applikationsmenü: Dieses Menü bietet eine Übersicht über alle installierten Programme. Dies beinhaltet Anwendungs- sowie Systemprogramme.

3.5.4 Systemprogramme

Jegliche Interaktion mit Dateien findet über Programme statt. Ist für einen bestimmten Dateityp kein passendes Programm verfügbar, kann nicht mit der Datei interagiert werden. Um zumindest eine grundlegende Funktionalität bereitzustellen, müssen mindestens die folgenden Systemprogramme verfügbar sein.

Dateimanager: Wird verwendet um organisatorische Operationen auf Dateien auszuführen.

Konsole: Bildet Brücken zum unterliegenden Betriebssystemkern um über textuelle Kommandos mit ihm zu interagieren

Editor: Wird verwendet, um Klartext-Dateien zu bearbeiten.

Konfigurationsmanager: Dieser Manager soll sich um alle Konfigurationsbelange des VR-Desktops kümmern. Dies reicht von Hardware-Einstellungen bis zu ästhetischen Belangen.

Mediaplayer: Zum Abspielen von Audio- und Video-Dateien.

Applikationsmanager: Installiert und deinstalliert Applikationen. Zusätzlich verwaltet der Applikationsmanager die Liste, welche Dateitypen welchen Programm zugeordnet sind. Bei Interaktionen mit Dateien wählt er das der Datei zugeordnete Programm aus und startet dieses.

Browser: Zur Bereitstellung von Internetkonnektivität.

3.5.5 Visuelle Bausteine

Eventbar: Über die Eventbar kann der Benutzer über eingetroffene System-, Applikation- und andere nicht vom Benutzer initiierte Ereignisse informiert werden. Genauer gesagt versteht man unter einem solchen Ereignis-Event das Ergebnis einer Funktion die im Hintergrund ohne direkte Interaktion des Benutzers oder von außerhalb des VR-Desktops eingeleitet worden ist. Dies sind zum Beispiel Verfügbarkeit von Updates oder das eintreffen einer E-Mail.

Widgetbar: Hierunter versteht man einen dedizierten Bereich im 3D-Raum der dafür reserviert ist Icons bzw. Verknüpfungen zu Programmen und Miniprogrammen (Widgets) aufzunehmen.

3D-Raum: Äquivalent zum Desktop wie man ihn von Windows, Gnome und KDE her kennt. In ihm können Fenster, Applikationsmenü und Anwendungsmonitor frei bewegt werden oder sie sind fest positioniert wie die Widgetbar und Eventbar.

Der Benutzer hat keine Einschränkungen und kann zum Beispiel ein Fenster theoretisch an jeden Punkt des 3D-Raums verschieben, die er mit der Hand bzw. durch Handtracking erreichen kann. Praktisch wird aber mithilfe der Eventbar und der Widgetbar versucht, die tiefst-mögliche Position für ein Fenster auf Brusthöhe zu beschränken, um so mögliche Kollisionen mit dem Möbel und anderen Interieur zu vermeiden.

3.6 Funktionen zur Benutzer-Interaktion

Im folgenden Abschnitt soll das Bedienkonzept des VR-Desktops näher beschrieben werden. Hierfür werden die einzelnen Bedienarten beschrieben und wie diese in Verbindung zueinander stehen.
Eine detaillierte Auflistung der Anforderungen, die an das Bedienkonzept gestellt werden, befindet sich im separaten Anhang: Anforderungen und Interaktionsfunktionen.

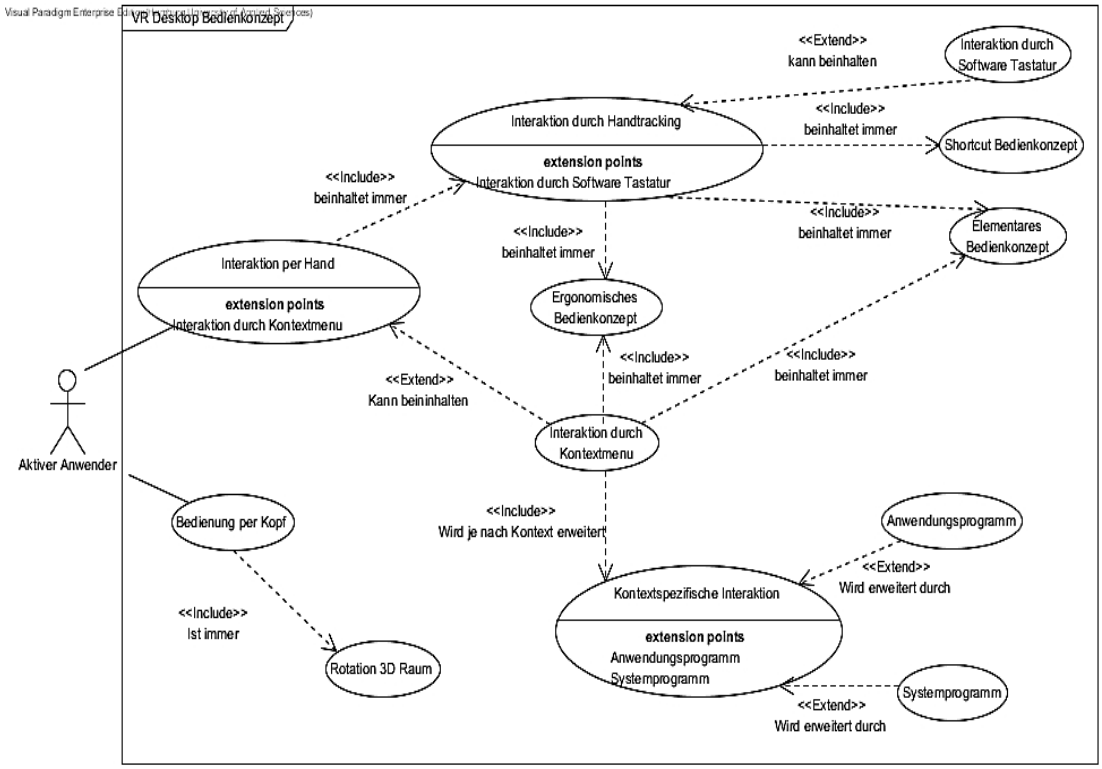

Abbildung 9 - Interaktionsmöglichkeiten VR-Desktop

3.6.1 Arten von Bedienung

Prinzipiell gibt es 5 verschiedene Arten von Bedien-Gesten, über die mit dem VR-Desktop interagiert werden kann.

Elementares Bedienkonzept: Diese Gruppe beinhaltet die grundlegenden Kommandos die zur Interaktion benötigt werden. Darunter fallen das Öffnen, Schließen, Verschieben, Markieren und Kontextmenü Aktivieren von graphischen Elementen.

Ergonomisches Bedienkonzept: Hier drin enthalten sind alle Kommandos die verwendet werden können, um verschiedene ergonomische Aspekte zu steuern. Darunter fallen das Vergrößern und Gruppieren von Elementen

Shortcut Bedienkonzept: Bietet eine Anzahl von Gesten, die verwendet werden können um bestimmte Gestenabfolgen abzukürzen, wie zum Beispiel das Schließen von allen Fenstern gleichzeitig.

Interaktion durch Software-Tastatur: Für textuelle Eingaben soll eine Software-Tastatur verwendet werden. Diese ist nur nötig wenn eine VR-Brille ohne Außenkamera verwendet wird. Ansonsten kann eine reale Tastatur verwendet werden.

Interaktion durch Kontextmenü: Das Kontextmenü ist eine alternative Möglichkeit das elementare und das ergonomische Bedienkonzept auszuführen. Des Weiteren lassen sich die Eigenschaften des gewählten Elements betrachten.

3.6.2 Kontextspezifische Interaktion

Je nach Kontext kann das Kontextmenü durch anwendungsspezifische Eigenschaften erweitert werden. Diese Eigenschaften werden von der Anwendung definiert, die dem jeweils ausgewählten Dateityp zugeordnet wurde.

3.6.3 Interaktion per Hand

Die Gesten des elementaren, ergonomischen und Shortcut-Bedienkonzepts besitzen systemweite Gültigkeit. Über diese Gesten wird das komplette Interaktionsspektrum des VR-Desktops abgewickelt. Alternativ kann ein Kontextmenü aufgerufen werden. Anwendungen können optional eine Interaktionsmöglichkeit per Software-Tastatur einbinden.

3.7 Produktdaten

Der VR-Desktop bietet, wie jede graphischen Oberflächen auch, die grundlegende Funktionalität, die für die Interaktion auf graphischer Basis mit einen Computersystem benötigt wird. Es ist wichtig zu verstehen, welche Daten verarbeitet werden und welche Bedeutung diese im systemweiten Kontext besitzen.
Hierfür werden in diesem Abschnitt die relevanten Daten umgangssprachlich als Teil einer fachlichen Domäne erhoben und ihr technischer Bezug erklärt.

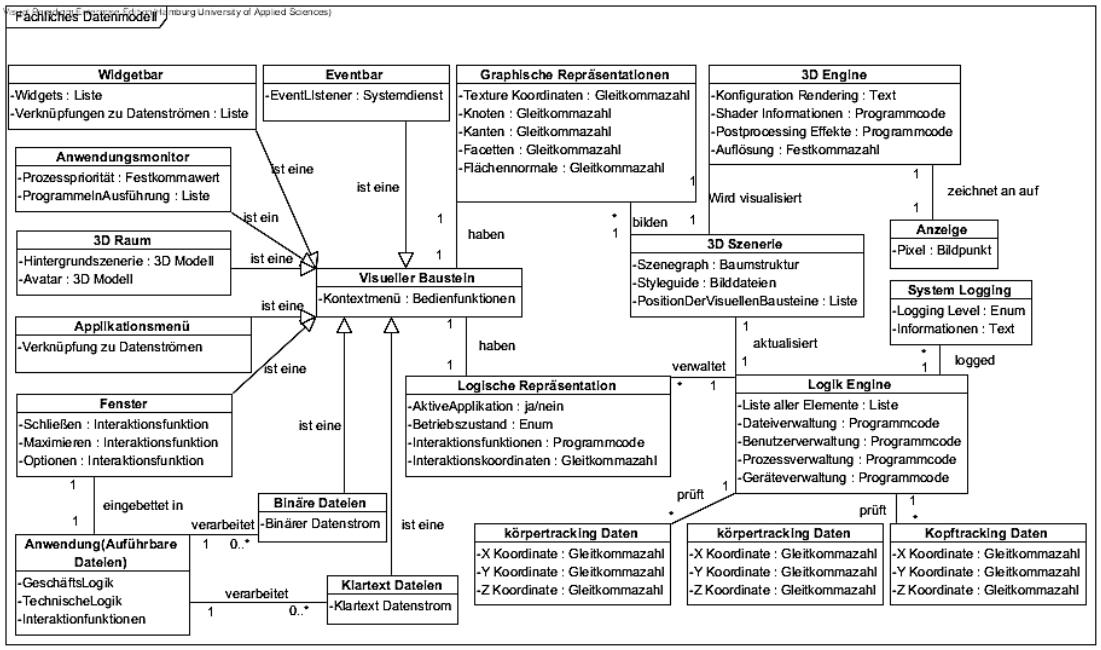

Abbildung 10 - Fachliches Datenmodell VR-Desktop

3.7.1 Grundlegende Datenarten

Rein fachlich wird beim VR-Desktop zwischen vier Datenarten unterschieden.

Klartext-Datenströme: Liegen leserlich für den Anwender vor. Hierbei handelt es sich zum Beispiel um txt-, html- oder XML-Dateien.

Binäre Datenströme: Liegen in Maschinencode oder Bytecode vor. Hierbei handelt es sich zum Beispiel um mp3-, avi-, oder jpg-Dateien.

Ausführbare Datenströme: Sind eine Spezialform von binären Datenströmen. Sie können entweder direkt vom VR-Desktop in der VR-Desktop Laufzeitumgebung (Anhang

Anforderungen FNWA3) ausgeführt werden oder werden von dem VR-Desktop unterliegenden Betriebssystemkern verarbeitet bzw. ausgeführt. Sie liegen zum Beispiel als jar oder exe vor.

Verknüpfungen auf Datenströmen: Sind ein Stellvertreter für einen Datenstrom jeglicher Art. Sie können auf beliebigen Orten Innerhalb des Systems verteilt werden. Sie verweisen entweder auf einen Datenstrom im Persistenz-Speicher oder auf einen laufenden Prozess.

3.7.2 Komponenten Datenarten

Fenster und visuelle Bausteine: Damit der VR-Desktop diese Elemente verarbeiten kann benötigt jedes Element eine graphische und eine logische Repräsentation im System.

Graphische Repräsentation: Durch sie wird beschrieben, welche Dimensionen das entsprechende Element in der 3D-Szenerie hat. Einschränkungen sind im Anhang Anforderungen unter FNBB9 beschrieben.
Diese geschieht durch: Knoten, Kanten und Facetten. Damit das Element später mit Texturen überzogen werden kann enthält es zusätzlich Texturkoordinaten für mögliche Texturen. Für die Schattierung einzelner Elemente werden vorher offline die Flächennormale berechnet.

Logische Repräsentation: Sie beschreibt den logischen Betriebszustand des jeweiligen Elements. Interaktionskoordinaten legen fest an welchen Punkten des Elements der Anwender Interagieren kann.
Der Betriebszustand (Anhang Anforderungen FNLE1) legt fest wie der zu der logischen Repräsentation gehörige Prozess vom System verarbeitet wird bzw. in welchem Teil des Applikationslebenszyklus der Prozess gerade befindet. Das Feld aktive Applikation signalisiert dem System, ob dieses Element momentan von einem Anwender bearbeitet wird.

3D-Szenerie: Alle Elemente die sich momentan im 3D-Raum befinden sind Teil des Szenegraphs. Der Styleguide legt fest, welche Texturen auf die einzelnen Elemente gelegt werden. Zusätzlich besitzt die 3D-Szenerie eine Liste mit Koordinaten zu allen graphischen Repräsentationen im Raum.

Rendering Engine: Wird verwendet zur Bilderzeugung von graphischen Repräsentationen. Hierfür wird der Szenegraph durchwandert und alle Elemente der Rendering-Pipeline hinzugefügt. Des Weiteren besitzt sie verschiedene Shader, wie zum Beispiel den zur Bildkrümmung von der VR-Brille benötigte Distortion-Shader, sowie verschiedene Postprocessing-Effekte, die durch Pixelshader über das Bild gelegt werden können.

Anzeige: Liest aus dem Framebuffer der Graphikkarte mit welcher Farbe die einzelnen Pixel der VR-Brille beleuchtet werden sollen.

Logik-Engine: Die Logik-Engine ist ein Verbund von VR-Desktop-Systemprogrammen, die verschiedene Schnittstellen zum verwendeten Betriebssystemkern nutzen, um deren Funktionalität Anwendungen des VR-Desktops entweder direkt oder indirekt verfügbar zu machen. Indirekt bedeutet in diesem Fall, dass die vom Betriebssystemkern angebotenen Schnittstellen dafür benutzt werden, um eigene VR-Desktop spezifische Funktionalität zu implementieren. Beispielsweise wird die Schnittstelle zum Gerätetreiber dafür genutzt, um Tracking Daten aufzufangen und auf deren Basis das Bedienkonzept zu erstellen.

Hand-, Körper- und Kopftracking-Daten: Sind Gleitkommazahlen, die durch die Logik-Engine mit Hilfe der Geräteverwaltung am Betriebssystemkerns abgefragt werden. Die Logik-Engine verändert auf Basis dieser Daten die 3D-Szenerie bzw. prüft im Falle der Handtracking-Daten, ob durch den Benutzer eine Interaktionsfunktion mit einem der graphischen Repräsentationen ausgelöst wurde.

System Logging: Bestehen aus einem Logging-Level, dass die Gewichtung des eingetreten Ereignisses beschreibt und einer dazugehörigen textuelle Beschreibung. Spezifiziert in Anhang Anforderungen unter 3.3.2.6 Fehler Handhabung.

Eventbar:. Bietet einen Hintergrunddienst, dessen Schnittstelle von anderen Hintergrundprogrammen genutzt werden kann, um den Benutzer über eingetroffene Ereignisse zu informieren.

Widgetbar:. Verwaltet eine Liste von Widgets, die aktuell ausgeführt werden. Widgets sind kleine Programme, die über eine Teilmenge der VR-Desktop-API definiert werden können. Zusätzlich hierzu verwaltet die Widgetbar eine Liste von Verknüpfungen zu Datenströmen.

Anwendungsmonitor:. Besitzt eine Liste mit Verknüpfungen von allen momentan ausgeführten Programmen. Kann mithilfe der Logik-Engine die Prozesspriorität des jeweiligen Programms verändern.

3D-Raum: Bei Verwendung einer VR-Brille eine beliebige 3D-Szenerie. Zusätzlich kann ein Avatar verwendet werden. Wenn der Anwender eine AR-Brille benutzt wird stattdessen die Umgebung in der sich der Anwender befindet angezeigt und seine leibliche Erscheinung verwendet.

Applikationsmenü: Präsentiert dem Anwender eine Liste von Programmverknüpfungen mit der interagiert werden kann.

3.8 Technische Leistungen

Bisher wurde in der Spezifikation des VR-Desktop vorrangig auf die funktionalen Anforderungen eingegangen, die direkt durch den Benutzer und seine Interaktion formuliert werden. Dieser Abschnitt beschäftigt sich nun mit den grundlegenden Leistungsmerkmalen die erfüllt sein müssen, damit die technische Umsetzung, die von ihr erwarteten Ergebnisse liefern kann.

Diese Anforderungen werden im Anhang Anforderungen genauer beschrieben.

Die folgenden Leistungsmerkmale stehen zumeist in Verbindung mit der VR-Technik. Dies hat den Grund, dass ganz bestimmte technische Voraussetzungen erfüllt sein müssen, damit die durch VR gebotenen Effekte Präsenz und Immersion entstehen können.

Die folgenden technischen Leistungsmerkmale wurden mit Hilfe von Michael Abrashs Steam Dev Days 2014 Präsentation [2.13] und dem Oculus Rift Best Practice Guide [2.28] erstellt.

3.8.1 Leistungsmerkmale der VR-Brille

- Das erzeugte Sichtfeld der VR-Brille muss mindestens 80° betragen.
- Der Bildschirm der VR-Brille sollte minimal 1080p für beide Augen haben. Somit 960 x 1080 pro Auge auflösen.
- Die Beleuchtung der Pixel auf dem Bildschirm darf nicht länger als 3 ms betragen um das Verschmieren des Bildes zu vermeiden.
- Um ein Flackern des Bildschirms zu vermeiden muss die Bildwiederholungsfrequenz bei ungefähr 90 Hz liegen.
- Die VR-Brille muss einen Bildschirm verwenden, der alle Pixel gleichzeitig aktualisiert anstatt zeilenweise.
- Zwischen dem Registrieren einer Kopfbewegung, der anschließenden Verarbeitung bis zu dem Moment wo die Photonen die Netzhaut treffen, dürfen höchsten 25 ms vergehen.
- Bewegungserkennung der Kopfbewegung bestimmbar auf die X, Y, Z Achse
- Genauigkeit bei Bewegung im Millimeterbereich, bei Drehung um ¼°, bei einer Weite von 1.5 Meter pro Kopfseite.

3.8.2 Leistungsmerkmale von Rendering/Logik.

- Die Anzahl der Bilder pro Sekunde muss minimal bei 60 Bildern pro Sekunde liegen. Optimal genauso hoch wie Bildwiederholungsfrequenz der VR-Brille

- Normalen-Mapping wirkt unnatürlich wenn in 3D dargestellt. Es sollte daher stattdessen Parallax-Mapping eingesetzt werden.
- Durch ungünstiges Verschieben eines Fensters kann es passieren, dass dieses mit Bewegungserkennung des Kopfes durchstoßen wird. Deshalb muss Back-Face-Rendering unterstützt werden

3.8.3 Leistungsmerkmale von Handtracking

- Bei 10 ausgeführten Gesten sollten 9 korrekt erkannt werden und zur gewünschten Aktion führen. Die maximale Fehlertoleranz von 10% darf nie überschritten werden.

3.9 Qualitätsanforderungen

In diesem Abschnitt soll über die Qualitätsmerkmale des VR-Desktops gesprochen werden.
In vorherigen Abschnitten wurden bereits einige dieser Qualitätsmerkmale beiläufig grob definiert. Hierfür wird der Abschnitt in drei Teile geteilt. Als erstens werden die Qualitätsstufen definiert und welche Maßnahmen ergriffen werden müssen, um die Qualität auf der jeweiligen Stufe zu gewährleisten.
Als zweites werden die verwendeten Testarten vorgestellt. Zum Schluss werden die Qualitätsmerkmale im Detail besprochen.
Die hier aufgestellten Qualitätsmerkmale werden nach der ISO/IEC 9126 Norm [3.10] erstellt. Es ist bekannt, dass diese Norm bereits veraltet ist und gegen die ISO/IEC 25000 Norm [3.9] ersetzt wurde. Aufgrund der Kompaktheit der ISO/IEC 9126 Norm, wird diese aber präferiert.

3.9.1 Qualitätsstufen

Nicht relevant: Es existiert kein dedizierter Testfälle für Qualitätsmerkmale auf dieser Stufe. Entweder werden die Merkmale implizit durch andere Testfälle indirekt mit getestet oder sind vernachlässigbar.

Normal: Es existieren ein oder mehrere Testfälle, die die Funktionalität des Qualitätsmerkmals sicherstellen. Diese Testfälle werden als Komponententest und Integrationstest bzw. Regressionstest ausgeführt, wenn Änderungen an dem Qualitätsmerkmal, wie zum Beispiel Implementierungsarbeiten vorgenommen wurden. Können zur Zeitersparnis im Systemtest ausgelassen werden.

Wichtig: Beinhaltet alle Maßnahmen der Stufe Normal. Des Weiteren existieren Performance-Tests und es werden statische Codeanalysen durchgeführt, um einen Teil der

nicht funktionalen Anforderungen sicher zu stellen. Die Testfälle werden bei jedem Systemtest ausgeführt.

Kritisch: Beinhaltet alle Maßnahmen der Stufe Normal und Wichtig. Des Weiteren existieren Performance-, Stress- und wenn möglich Usability-Tests, die die nicht funktionalen Anforderungen sicherstellen.
Die Testfälle werden bei jedem Systemtest ausgeführt. Zusätzlich sind diese Merkmale Teil des Akzeptanztest.

3.9.2 Testverfahren

Komponententest ist ein Test, der auf Ebene der einzelnen Komponente oder Teilprogrammen des VR-Desktops durchgeführt wird.

Regressionstest ist das wiederholte Ausführen von Testfällen, um sicherzustellen, dass Modifikationen an bereits getesteten Komponenten keine Fehler verursachen.

Integrationstest testet die Zusammenarbeit zweier oder mehrerer voneinander abhängiger Komponenten. Der Testschwerpunkt liegt auf den Schnittstellen der beteiligten Komponenten.

Systemtest testet das gesamte System gegen alle in dieser Spezifikation definierten Anforderungen, sowohl funktionale als auch nicht-funktionale.

Akzeptanztest beschreibt das Testen der Funktionalität des VR-Desktop durch einen nicht direkt an der Entwicklung des VR-Desktop beteiligten Benutzers. Im Falle des VR-Desktops werden Akzeptanztest bei jedem Major Release ausgeführt.

Usabilitytest testen den Bedienkomfort durch einen nicht in den Entwicklungsprozess eingebunden Probanden.

Performanztest prüft die Effizienz des Systems. Dies kann auf Komponenten-Ebene erfolgen oder das komplette System beinhalten.

Stresstest. Eine Art von Performanztest, der versucht das System an seine Belastungsgrenzen zu bringen und so entweder zum Teil oder komplett versagen zu lassen.

3.9.3 Funktionalität

Die festgelegten Eigenschaften von einzelnen Funktionen des VR-Desktops.

Tabelle 1 - Qualitätsmerkmale zur Funktionalität

Funktionalität	Kritisch	Wichtig	Normal	Nicht relevant
Richtigkeit	**X**			
Interoperabilität		**X**		
Sicherheit		**X**		
Angemessenheit		**X**		

Angemessenheit: Um ein möglichst breites Leistungsspektrum an Hardware zu unterstützen ist es nötig, den Selbstverwaltungsaufwand zur Laufzeit des VR-Desktops so gering wie möglich zu halten. Hierfür sollte so viel Programmcode wie möglich in systemweit verfügbare Bibliotheken und Hintergrunddienste ausgelagert werden. Diese können dann von anderen Programmen wiederverwendet werden, wenn diese eine bestimmte Funktionalität benötigen. Beschrieben in Anhang Anforderung unter FNWA2.

Richtigkeit: Während an die Richtigkeit der Anwender und Systemprogramme des VR-Desktops keine besonderen Anforderungen gestellt werden bzw. diese weitestgehend der Sorgfalt des jeweiligen Entwickler überlassen werden können, ist die Richtigkeit der Tracking-Informationen kritisch. Es können bei zu ungenauer Auswertung der Trackingkomponenten, leicht Frust über die Bedienung und im schlimmsten Fall Simulator Sickness entstehen.

Interoperabilität: Es ist nötig eine Abstraktionsschicht zwischen den Systemdiensten des verwendeten Betriebssystemkerns und dem Aufruf im VR-Desktop zu implementieren, damit im Falle einer Portierung des VR-Desktops nur die verwendeten Systemaufrufe geändert werden müssen. Diese Anforderung ist unter FNPU1 beschrieben.

Sicherheit: Sicherheitskonzepte des verwendeten Betriebssystemkern müssen vollständig durch den VR-Desktop implementiert werden. Darunter zählen Datenverwaltung, Benutzerverwaltung und Verschlüsselungsalgorithmen. Beschrieben im Anhang Anforderungen unter 3.11.1 Sicherheit.

3.9.4 Zuverlässigkeit

Fähigkeit des VR-Desktops das Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.

Tabelle 2 - Qualitätsmerkmale zur Zuverlässigkeit

Zuverlässigkeit	Kritisch	Wichtig	Normal	Nicht relevant
Fehlertoleranz	**X**			
Wiederherstellbarkeit			**X**	

Fehlertoleranz: Der VR-Desktop muss das vom unterliegenden Betriebssystemkern angebotene Fehler-Handling komplett implementieren und um seine eigenen Funktionen erweitern. Im Anhang Anforderungen unter 3.3.2.6 beschrieben. Besonderes Augenmerk ist hierbei auf das Bewegungstracking zu richten. Es gilt zu klären, wann der Benutzer tatsächlich mit dem VR-Desktop interagiert und wann lediglich die Hände bewegt, zum Beispiel zur Entspannung.

Wiederherstellbarkeit Die Stabilität des Systems ist kritisch. Bei einem Ausfall kann dies bedeuten, dass die Interaktion mit dem System vorübergehend komplett unmöglich wird und ein möglicher Verlust von Daten droht.
Um diesen schwerwiegenden Faktor etwas abschwächen zu können ist es nötig, jegliche bereitgestellte Systemfunktion, die eine Wiederherstellung erleichtert, aus dem Betriebssystemkerns auf dem der VR-Desktop aufgesetzt zu implementieren bzw. zu nutzen.
Zusätzlich müssen für VR-Desktop spezifische Funktionen eigene Fehler-Behandlung und Fall-Back-Mechanismen entwickelt werden, wie im Anhang Anforderungen unter 3.3.2.6 beschrieben.

3.9.5 Benutzbarkeit

Welche Anforderungen werden für den Einsatz des VR-Desktops an den Benutzer gestellt?

Tabelle 3 - Qualitätsmerkmale zur Benutzbarkeit

Benutzbarkeit	Kritisch	Wichtig	Normal	Nicht relevant
Verständlichkeit	**X**			
Erlernbarkeit	**X**			
Bedienbarkeit	**X**			
Attraktivität			**X**	
Lange Benutzung	**X**			

Verständlichkeit: Die Steuerung des VR-Desktops mit Hilfe von Bewegungssteuerung bietet eine natürliche Art mit dem Computer zu interagieren. Der entscheidende Faktor ist, dass diese Bedienung durch das ganze System konsequent durchgehalten wird.

Erlernbarkeit: Um den VR-Desktop für den Benutzer so verständlich wie möglich zu gestalten sollten, wenn möglich, etablierte, organisatorische Konzepte von bestehenden graphischen Oberflächen übernommen werden. So können die bei der Nutzung von herkömmlichen, graphischen Oberflächen erlernten Problemlösungsstrategien weiterhin verwendet werden und dem Benutzer den Umstieg so erleichtern.

Bedienbarkeit: Die Bedienbarkeit ist das wichtigste Qualitätsmerkmal des VR-Desktops. Es muss besonders darauf geachtet werden, dass die Gesten mit denen der VR-Desktop gesteuert wird mit möglichst wenig körperlichem Aufwand erzeugt werden können.
Genauso wichtig ist das korrekte Erkennen der verwendeten Gesten.

Attraktivität: Die Verwendung von Bewegungstracking und VR-Brille bietet einen „Unique Sellingpoint“ für den VR-Desktop, daher es ist eine gewisse Attraktivität von vornherein gegeben.

3.9.6 Effizienz

Wie ist das Verhältnis zwischen Leistungsfähigkeit des VR-Desktop und eingesetzten Hardware Ressourcen?

Tabelle 4 - Qualitätsmerkmale zur Effizienz

Effizienz	Kritisch	Wichtig	Normal	Nicht relevant
Zeitverhalten	**X**			
Verbraucherverhalten		**X**		

Zeitverhalten: Hierbei muss grundlegend zwischen Applikationen die auf dem VR-Desktop ausgeführt werden und dem vom VR-Desktop angebotenen bzw. ausgeführten Bedienungsschemata unterschieden werden.
Während das Zeitverhalten der auf dem VR-Desktop laufenden Applikation maßgeblich zur Benutzererfahrung beiträgt, spielt es eine untergeordnete Rolle ob die Applikation nun 0.5 oder 2 Sekunden zur Ausführung benötigt. Wichtig ist nur, dass von der Applikation ein angemessener zeitlicher Rahmen eingehalten wird.
Die Zeitspanne von der Eingabe des Benutzers, über die Verarbeitung, bis zu dem Moment wo die Photonen des erzeugten Bildes Netzhaut treffen ist kritisch. Das Verarbeiten von Tracking-Daten muss daher immer Priorität besitzen.

Verbrauchsverhalten: Da die visuellen Elemente des VR-Desktop ausschließlich auf der Basis von Polygon-Netzen entstehen ist anzunehmen, dass sein Verbrauchsverhalten in Bezug auf System-Ressourcen wesentlich höher ist, als die einer äquivalenten graphische Oberfläche auf 2D-Basis.

Außerdem muss auch sichergestellt sein, dass der Detailgrad der darzustellenden Polygon-Netze möglichst vielseitig konfigurierbar ist, um die Ressourcen auf der GPU freizuhalten.

3.9.7 Wartbarkeit/Änderbarkeit

Wie sollte sich der VR-Desktop in Bezug auf seine Wartbarkeit und Änderungen verhalten? Änderungen können Korrekturen, Verbesserungen, Anpassungen oder Änderungen der Umgebung beinhalten, aber auch funktionale Anforderungen an das System.

Tabelle 5 - Qualitätsmerkmale der Wartbarkeit und Änderbarkeit

Wartbarkeit	Kritisch	Wichtig	Normal	Nicht relevant
Analysierbarkeit	**X**			
Modifizierbarkeit	**X**			
Stabilität	**X**			
Testbarkeit	**X**			

Analysierbarkeit: Da VR eine Technik ist, die noch in den Kinderschuhen steckt, ist die Analysierbarkeit kritisch. Daher ist es nötig das System auf mögliche Fehlerquellen, schnell und ausgiebig auszuwerten. Dies soll mit Hilfe von Systemlogging (Anhang Anforderungen FH2) erreicht werden.

Modifizierbarkeit: Der VR-Desktop bietet nur die absolut notwendigen Systemfunktionen, beschrieben im Anhang Anforderungen Abschnitt 3.2.2.7. Bei allen darüber hinausgehenden Benutzungsszenarien verlässt man sich auf die Expertise der jeweiligen Fachdomäne. Hierfür soll eine wohldefinierte API erstellt werden, die das Erstellen von Applikationen und deren graphischen Repräsentation ermöglicht.
Bei der unterstützen Hardware muss vor allen Dingen auf die Kompatibilität mit OpenGL geachtet werden.

Stabilität: Die Stabilität soll durch einen modularen Aufbau und eine hohe Wiederverwendung von Programmfunktionen erreicht werden. Dies ermöglicht im Optimalfall, dass wenn es zu einem Fehler kommt, nur ein Single „Point of Failure“ untersucht werden muss.

Testbarkeit: Um die Richtigkeit der Funktionalität des VR-Desktops auf einem konstanten Qualitätsniveau zu gewährleisten ist es nötig, dass das VR-Desktop-Projekt eine hohe Testabdeckung besitzt. Zur Effizienzsteigerung soll diese Testbasis soweit wie möglich automatisiert werden, damit im Falle der Projekterstellung, aber auch in definierten

zeitlichen Intervallen, routinemäßig Unit Tests und durch die Qualitätssicherung definierte Test-Szenarien automatisch ausgeführt werden können.
Zum anderen beschreibt das Wort Testbarkeit im Bereich der Forschung auch immer indirekt, und dies ist aufgrund der Reife von VR-Technik momentan ein viel wichtigerer Faktor, die Fähigkeit zum Erstellen von Prototypen. Also wie schnell sich ein Erkenntnisstand in ein lauffähiges System übertragen lässt. Im Anhang Anforderungen unter SD3 erhoben.

3.9.8 Übertragbarkeit

Eignung des VR-Desktops von einer Umgebung in eine andere übertragen werden zu können.

Tabelle 6 - Qualitätsmerkmale der Übertragbarkeit

Übertragbarkeit	Kritisch	Wichtig	Normal	Nicht relevant
Anpassbarkeit			**X**	
Installierbarkeit		**X**		
Koexistenz				**X**
Austauschbarkeit				**X**

Anpassbarkeit: Hierbei geht es um die Fähigkeit des VR-Desktops mit verschiedenen Betriebssystemkernen zusammen zu arbeiten. Betriebssysteme müssen nur ihre Schnittstellen zum Betriebssystemkern offen legen. Daraufhin kann die Abstraktionsschicht des VR-Desktop auf den Betriebssystemkern angepasst werden.

Installierbarkeit: Einfachheit der Installation zu gewährleisten. Es muss durch Testen auf verschiedenen Hardwarekonfigurationen sichergestellt werden, dass alle benötigten Betriebsmittel vorhanden sind, bzw. erkannt und angesteuert werden können oder welche Vorgänge eingeleitet werden wenn sie fehlen.

Koexistenz und Austauschbarkeit: Da es sich bei dem VR-Desktop um eine graphische Oberfläche handelt die elementare Funktionalitäten, die für das Arbeiten mit Computer benötigt werden, bereitstellt, kann er Isoliert betrachtet werden und die Koexistenz ist nicht relevant. Durch die Nutzung von Schnittstellen bzw. Entkoppelung des Betriebssystemkerns ist die Austauschbarkeit automatisch gegeben. Durch die Einhaltung des Posix-Standards soll dies zusätzlich vereinfacht werden. Anhang Anforderung unter FNPU2 aufgenommen.

3.10 Graphische Oberflächen

In diesem Abschnitt werden die Anforderungen an die graphischen Elemente formuliert. Eine formale Beschreibung zur Funktionalität der graphischen Elemente befindet sich in Anhang Interaktionsfunktionen in Form von Anwendungsfällen

Fenster: Jedes Fenster des VR-Desktops muss über die Interaktionsfunktionen Maximieren, Minimieren und Schließen verfügen. Ansonsten gleicht es seinem zweidimensionalen Äquivalent.

Widgetbar: Die Widgetbar befindet sich auf Höhe des Bauches des Anwenders. Sie besitzt die geometrische Form eines Ringes, in dessen Mitte sich der Anwender befindet. Auf ihr befindet sich ein Mülleimer in den nicht mehr benötigte Verknüpfungen verschoben werden können. Des Weitern befinden sich auf ihr standardmäßig Verknüpfungen zu:

- Applikationsmenü
- Anwendungsmonitor
- Konsole
- Dateimanager
- Konfigurationsmanager

Diese können nicht vom Anwender gelöscht werden.

Eventbar: Befindet sich am oberen Ende der Widgetbar. Sie besitzt ebenfalls eine Ringform in dessen Mitte sich der Anwender befindet. Bei Eintreffen eines neuen Events blinkt sie kurz auf und zeigt textuelle Informationen zum eingetroffen Event in Form von Laufschrift.

Kontextmenü: Zur besseren Anpassung an die Bewegungssteuerung wird das Kontextmenü des VR-Desktops als Tortendiagramm dargestellt. Neben den bisher spezifizierten Funktionen bietet es zusätzlich die Möglichkeit Verknüpfungen zu erstellen.

Anwendungsmonitor: Wird in einem VR-Desktop-Fenster angezeigt. Alle momentan laufenden Anwendungen werden in einer graphischen Kachel-Liste präsentiert.

Applikationsmenü: Wird in einem VR-Desktop-Fenster angezeigt. Des Weiteren werden auf ihm folgende Verknüpfungen standardgemäß zur Verfügung gestellt:

- Mülleimer
- Konfigurationsmanager
- Konsole
- Dateimanager

Diese können nicht vom Anwender gelöscht werden.

Dateien: Alle Datenarten werden im VR-Desktop mit Hilfe von Icons auf Vektorgraphik-Basis visualisiert.

3.11 Gliederung in Teilprodukte

Im folgenden Abschnitt wird die fachliche Architektur des VR-Desktops präsentiert und erläutert. Hierfür werden die Anforderungen einer jeweiligen Komponente zugeordnet.

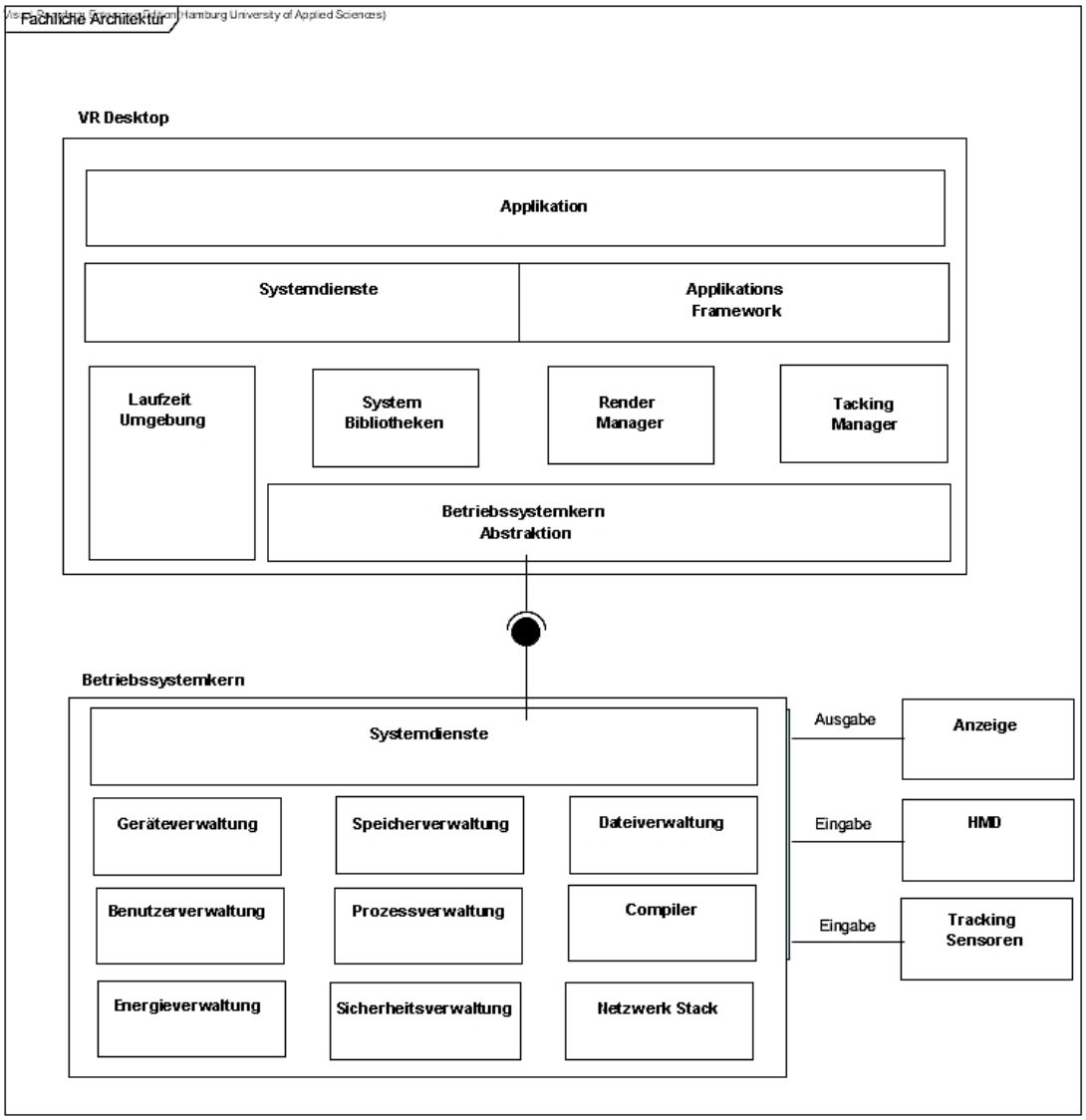

Abbildung 11 - Fachliche Architektur des VR-Desktops.

3.11.1 Betriebssystemkern Abstraktion

Um ein hohes Maß an Portabilität gewährleisteten zu können ist es nötig, die verwendeten Schnittstellen des Betriebssystemkerns zu entkoppeln. Prinzipiell ist die Betriebssystemkern-Abstraktion nichts anders als ein Adapter-Entwurfsmuster.
Ein Vorteil der durch die Anwendung dieser Technik entsteht ist, dass die darüber liegenden Schichten des VR-Desktop gegen vereinheitlichte Schnittstellen entwickelt werden können. Der VR-Desktop kann so komplett aus sich selbst heraus definiert werden und ist in sich geschlossen. Bei Portierungen müssen nur die Systemaufrufe im jeweiligen Adapter angepasst werden.

3.11.2 Laufzeit-Umgebung

Mit Hilfe der Laufzeit-Umgebung werden Programme auf dem VR-Desktop ausgeführt. Um zeitaufwendiges Kompilieren von Anwendungen zu minimieren und die Portabilität des VR-Desktop zu stützen ist es nötig, eine Programmiersprache zu verwenden, die mit einem virtuellen-Maschinen-Konzept arbeitet.
Daher wird auf Applikationsebene des VR-Desktops Java eingesetzt. Es gibt eine Menge Implementierungen verschiedener Java-Virtuelle-Maschine, deren Implementierungen jeweils verschiedene Ziele verfolgen, wie zum Beispiel Hotspot (Portierbarkeit) [3.12], Dalvik (Effizienz)[3.11] oder Kaffe (nur Freie Software) [3.13]. Java ist eine der am weitesten verbreiteten Programmiersprachen und kann so vielen Programmierern die Möglichkeit geben, Anwendungen für den VR-Desktop zu entwickeln.

3.11.3 System-Bibliotheken

System-Bibliotheken werden aus Effizienzgründen in C oder C++ geschrieben und bei der Installation des VR-Desktops in den binären Dialekt des Host Betriebssystemkerns kompiliert. Sie können danach als native Bibliotheken in die System-Dienste des VR-Desktop eingebunden werden.

Sie enthalten unter anderen Funktionen für die grundlegenden Systemeigenschaften:

- Graphik- Bibliothek basierend auf OpenGL.
- Webkit als Grundlage für den Browser.
- Multimedia Codecs zum Abspielen von Medien-Dateien.

3.11.4 Render Manager

Der Render Manager übernimmt die Visualisierung des VR-Desktops. Er verwaltet den Szene-Graphen der dem Nutzer während der Benutzung präsentiert wird. Systemdienste aus der darüber liegenden Schicht können Änderungen am Szene-Graphen vornehmen.

3.11.5 Tracking Manager

Dieser Manager verwaltet die eingehenden Kopf- und Hand-Tracking Daten und stellt sie den darüber liegenden Schichten zur Verfügung. Des Weiteren benachrichtigt der Tracking Manager die laufenden Anwendungsprozesse, wenn eine Interaktion mit ihrem Fenster stattgefunden hat.

3.11.6 Systemdienste

Systemdienste verwalten Hardware- und Systemressourcen und stellen dem Applikation-Framework Dienstleistungen zur Verfügung. Dies können System-organisatorische oder anwendungsspezifische Funktionen sein.

3.11.7 Applikation Framework

Stellt alle benötigten Funktionen bereit, die zum Entwickeln von Anwendungen für den VR-Desktop benötigt werden. Darunter fallen graphische Elemente, wie Buttons und Listen, aber auch Schnittstellen zu den darunterliegenden Systemdiensten.

4 Entwurf

In diesem Kapitel sollen die Konzepte die im Kapitel 3 spezifiziert wurden entworfen werden. Im Laufe des Kapitels werden hierfür im Detail, das VR-Desktop-Bedienschema, VR-Desktop-Applikationslebenszyklus, das VR-Desktop-Applikationsframework, sowie die technische Architektur und ein Ausschnitt des Verhaltens zur Laufzeit entwickelt.

4.1 Bedienung

Zur Interaktion mit dem VR-Desktop müssen Systemweit gültige Basis-Kommandos bzw. Gesten festgelegt werden. Als Inspiration hierfür werden die allgemein anerkannten Smartphone Gesten benutzt und erweitert. Hierfür wird sich im Detail an den von Android [4.4] und dem IPhone [4.3] definierten Bedienschemata orientiert.
Die in Kapitel 3 gestellte Anforderung einer Handtracking Genauigkeit auf Fingerebene ermöglicht es, dass differenzierte Kommandos zur Interaktion mit dem System erstellt werden können. Aus technischer Sicht ist dies möglich, da an jeder Fingerspitze ein Kollisionspunkt angelegt werden kann und somit das System immer weiß, welcher Finger mit einem Element interagiert hat.
Die hier präsentierten Bediengesten bilden den systemweiten Standard, jedoch können Applikationen, die auf dem VR-Desktop-Applikationsframework basieren, selbstständig entscheiden, wie auf bestimmte Gesten die auf ihre Interaktionsfunktionen ausgelöst werden, zu reagieren ist.

Eine formale Beschreibung der Anforderungen die an das VR-Desktop Bedienschemata gestellt werden, findet sich im Anhang Anforderungen.

4.1.1 Elementares Bedienkonzept

Dieser Teil des Bedienschemas wird verwendet um die Elementare Interaktion mit dem VR-Desktop bereitzustellen. Dies beinhaltet Das Öffnen von Dateien, Starten von Programmen,

die Interaktion mit Verknüpfungen und das Auslösen von Interaktionsfunktionen der einzelnen Programme. Des Weiteren werden Gesten zum Verschieben von Fenstern und Scrollen von Textelementen definiert.

1.) **Antippen** Der Benutzer streckt den Zeigfinger aus und führt dann schnell eine Antipp-Geste auf ein Element aus (Abbildung 12). Danach kehrt der Finger in eine entspannte Haltung zurück und hat keinen Kontakt mehr mit dem Element.

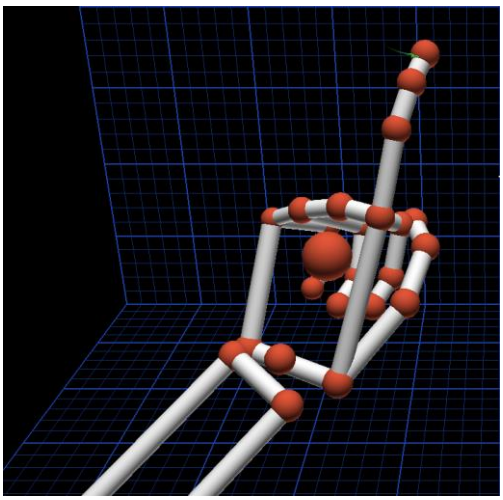

Abbildung 12 - Ein-Finger-Geste

- **Fenster und Visuelle Bausteine:** Fokus wird auf das Element gelegt und als aktiv markiert.

- **Fenster und Visuelle Baustein Interaktionsfunktion:** Der Programmcode der Interaktionsfunktion, die sich hinter dem graphischen Element verbirgt wird aufgerufen.

- **Dateien und Verknüpfungen:** Öffnet die Datei oder Verknüpfung, mit der Anwendung dessen Dateiendung mit der Datei verknüpft ist oder öffnet die Anwendung selbst.

- **Auf 3D Raum:** Wenn ausgeführt wird die Markierung oder Fokussierung von allen Elementen entfernt.

2.) Antippen und halten Der Benutzer streckt den Zeigfinger aus und führt schnell eine Antipp-Geste auf ein Element aus, danach verweilt der Finger in einer ausgestreckten Lage auf dem Element.

- **Fenster und Visuelle Bausteine:** Bei Verwendung ruft die Geste das Kontextmenü mit seinen Standardoptionen kombiniert mit dem kontextspezifischen Kontextmenü auf. Zum Beispiel Fensterrahmen, dann Fenster-Optionen. Wenn Datei, dann Datei-Operationen.

3.) Ziehen: Der Benutzer zieht den Finger über eine graphisches Element. Dabei hat der Finger ständig Kontakt mit diesem Element.

- **Textuelle Inhalte:** Spezifische Geste wird auf Textelemente angewendet. Durch Verwendung dieser Geste kann schnell durch textuelle Inhalte gescrollt werden.

4.) Zwei-Finger-Halten und -Ziehen: Der Benutzer streckt den Zeigfinger und Mittelfinger aus und führt schnell eine Tipp Geste auf ein Element aus (Abbildung 13), danach verweilt der Finger in einer ausgestreckten Lage auf dem Element.

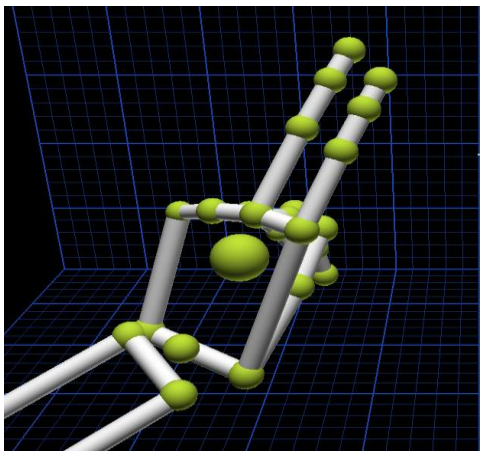

Abbildung 13 - Zwei-Finger-Geste

- **Fensterrahmen:** Fenster wird an der virtuellen Hand befestigt und der Benutzer kann im Anschluss das Fenster frei durch den 3D-Raum bewegen. Ein Doppeltes antippen löst das Fenster wieder von der Hand.

- **Textuelle Inhalte:** Langsames und genaueres Scrollen durch textuelle Inhalte.

5.) Zwei-Finger doppeltes Antippen: Der Benutzer streckt den Zeigfinger und Mittelfinger aus und führt schnell eine doppel-Antipp Geste auf eine Element aus, danach verweilt der Finger in einer ausgestreckten Lage.

- **Fensterrahmen:** Das betreffende Fenster wird so nah an den Benutzer gezoomt das das es einen Großteil seines Sichtfelds ausmacht.

4.1.2 Ergonomisches Bedienkonzept

Das Ergonomische Bedienkonzept stellt eine Anzahl von Operationen zur Verfügung, um das Arbeiten mit dem VR-Desktop an die Vorlieben des Anwenders anzupassen. Da in diesem Abschnitt die Funktionen anhand der Gesten erklärt werden, wurden bereits unter 1.1.1 Gesten besprochen, die auch in diese Kategorie fallen, jedoch im Kontext dort besser präsentiert werden konnten.

1.) Anpassung der Größe von Elementen: Der Benutzer verwendet dieselbe, vom Smartphone bekannte Pinch Geste (Abbildung 14), um den Vorgang auszuführen. Hierfür berührt der Benutzer das gewünschte Element mit dem Zeigefinger und dem Daumen. Durch die Verringerung oder Vergrößerung des Abstands der beiden Finger wird das Element entweder vergrößert oder verkleinert.

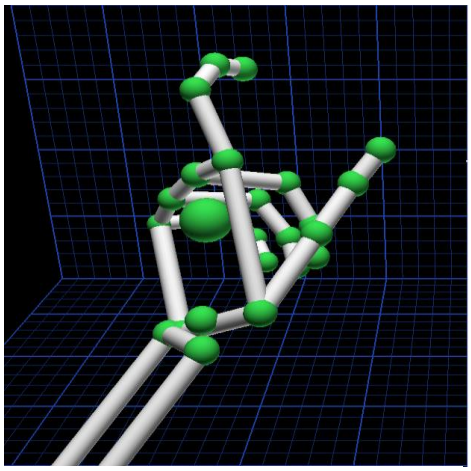

Abbildung 14 - Zoom-Geste

- **Fenster und visuelle Bausteine:** Fenster werden vergrößert oder verkleinert.
- **Textuelle Inhalte:** Vergrößert und verkleinert textuellen Inhalte.

2.) Markieren und Gruppieren: Der Benutzer formt mit einer Hand eine Faust und mit der anderen Hand tippt er auf die gewünschten Elemente (Abbildung 15).

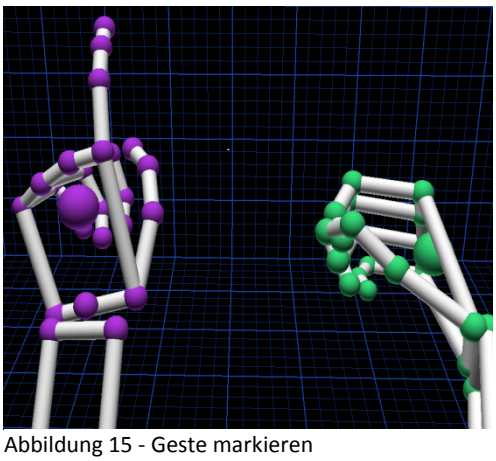

Abbildung 15 - Geste markieren

- **Fensterrahmen:** Alle Fensterrahmen, die während dieser Geste berührt werden, werden als markiert dargestellt. Nachdem der Benutzer die gewünschte Menge an Fenstern markiert hat können auf alle Elemente simultan verschiedene Operationen ausgeführt werden. Zum Beispiel Aufruf des Kontextmenüs oder das Verschieben im Raum.

- **Dateien :** Funktionalität und Ausführung ist dieselbe wie bei Fensterrahmen

4.1.3 Shortcut Bedienkonzept

Zur effizienteren Bedienung von Computersystemen werden in der Regel Shortcuts von sich häufig wiederholenden, mehrstufigen Ausführungsschritten erstellt. Exemplarisch soll hier nun veranschaulicht werden, wie mögliche Shortcuts auf dem VR-Desktop realisiert werden können.

Öffnen der Eventbar: Der Benutzer formt mit beiden Händen eine Faust (Abbildung 16). Nach Ausführung dieser Geste wird die Eventbar geöffnet und dem Benutzer passend zu seinem Blickwinkel präsentiert.

Schließen aller Fenster: Der Benutzer legt die Fingerspitze des betreffenden Finger an die korrespondierende Fingerspitze der anderen Hand (Abbildung 18).

Öffnen des Anwendungsmonitors: Der Benutzer öffnet die Hände und richtet die Handinnenflächen in Richtung seines Körpers (Abbildung 19).

Öffnen des Applikationsmenüs: Der Benutzer legt den Zeigefinger und Mittelfinger der einen Hand auf die korrespondierenden Finger der anderen Hand (Abbildung 17).

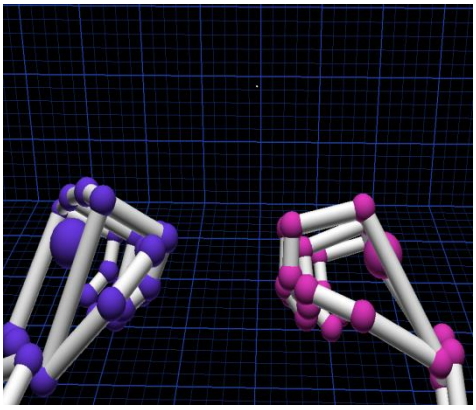

Abbildung 16 - Öffnen Eventbar

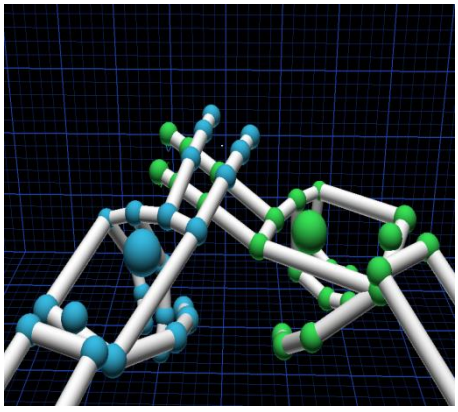

Abbildung 17 - Öffnen Applikationsmenüs

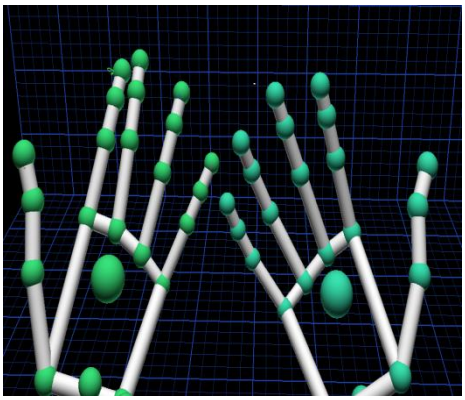

Abbildung 18 - Öffnen Anwendungsmonitor

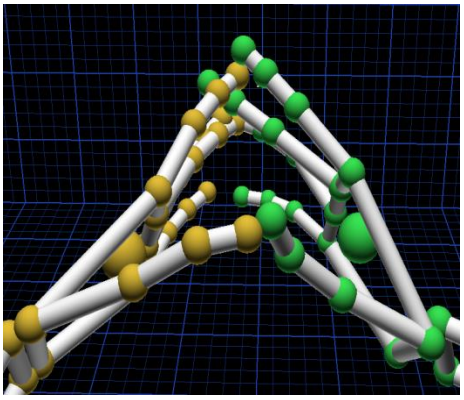

Abbildung 19 - Schließen aller Fenster

4.2 Applikation Lebenszyklus

Da alle graphischen Elemente des VR-Desktops auf Basis von polygonaler Graphik entstehen ist davon auszugehen, dass er mehr Speicherplatz im Hauptspeicher und mehr Prozesszeit zur Eigenverwaltung benötigt als für eine zweidimensionale Oberfläche mit gleichem Funktionsumfang. Der VR-Desktop Applikation Lebenszyklus stellt dabei einen Versuch dar die benötigten System-Ressourcen zu minimieren.
Hierbei werden Konzepte des Android [4.1] und LibGDX [4.2] Applikation Lebenszyklus angewendet und erweitert. Nachfolgend eine Erklärung der Methoden in Reihenfolge in der sie auf gerufen werden.

create(): Die Create-Methode ist die erste Methode die VR-Desktop Applikation Lebenszyklus vom System aufgerufen wird. Hier sollte die komplette Initialisierung der Applikation erfolgen, wie das Laden von Assets und Initialisierung der verwendeten Systemdienste.

render(): In dieser Methode kann die graphische Repräsentation der Anwendung angepasst werden. Dies kann zum Beispiel nötig sein, wenn in der Anwendung flächendeckend ein Untermenü aufgerufen werden soll.

slowDown(): Diese Methode ist die Besonderheit des VR-Desktops und soll im Folgenden genauer erklärt werden. Zur Vereinfachung wird dieses Beispiel lediglich am horizontalen Sichtfeld erklärt. In der Implementierung wird dieser Ansatz aber auch für das vertikale Sichtfeld verwendet. Für die folgende Behauptung, wird davon ausgegangen das der VR-Desktop im Sitzen, also ohne Verwendung von Körpertracking-Daten, bedient wird.

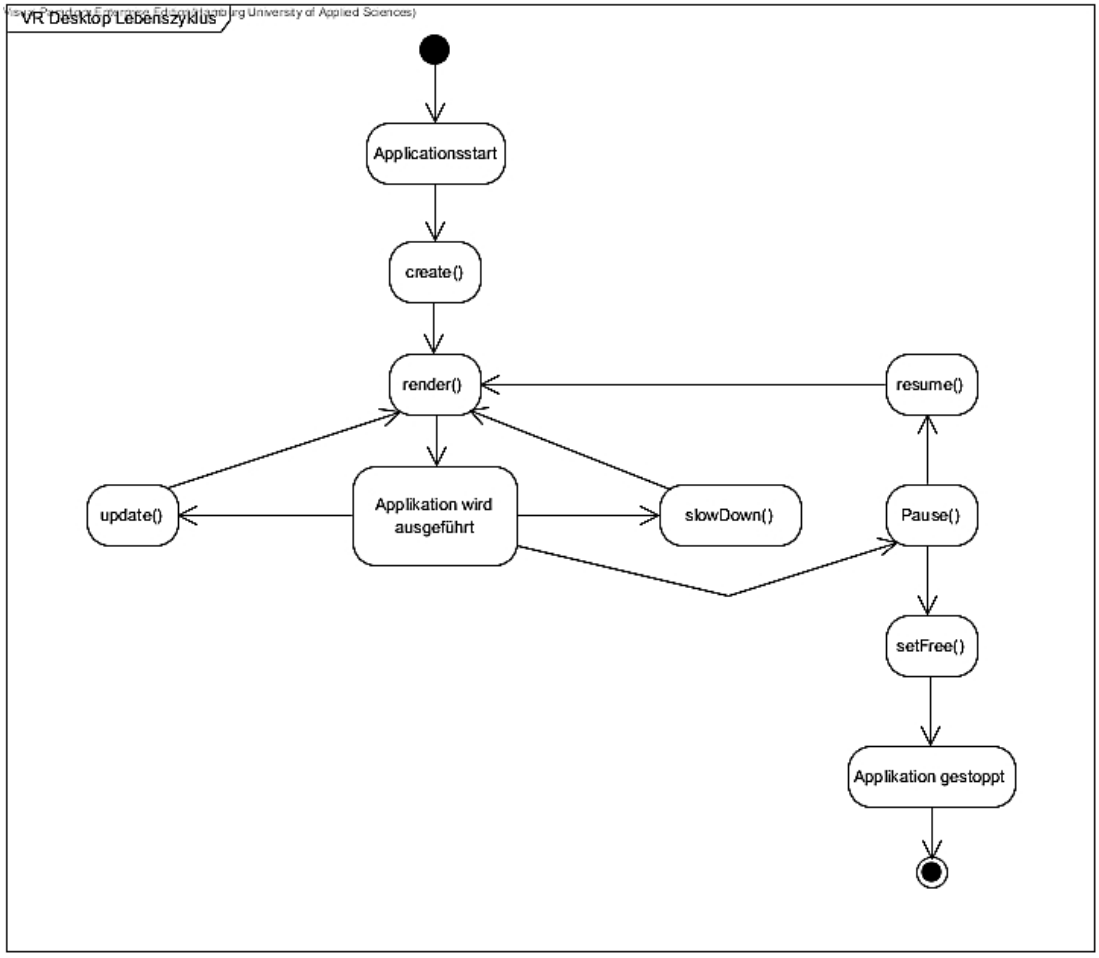

Abbildung 20 - VR-Desktop Applikationslebenszyklus

Durch die 360° Freiheit, die durch das Kopftracking in Kombination mit dem Handtracking ermöglicht wird, kann der Anwender theoretisch an jeder Stelle, die er mit seinen Fingern erreichen kann, graphische Elemente positionieren.
Die Annahme die beim Entwurf des VR-Desktop getroffen wird ist, dass wenn man konzentriert an einer Aufgabe arbeitet, man diese so im 3D-Raum positioniert, dass sie bequem zu erreichen sind. Die zu bearbeitenden Elemente werden dann so angeordnet, dass der Körper eine entspannte Haltung einnehmen kann. Man bedenke an dieser Stelle, dass bei der Benutzung des VR-Desktops mehr körperliche Arbeit verrichtet werden muss, als bei einer äquivalente Bedienung mit Maus und Tastatur.

Die Anordnung der zu bearbeitenden Elemente wird dann vermutlich irgendwo vor dem Benutzer auf Augenhöhe liegen.
Dies definiert den primären Sichtbereich des Benutzers. In diesem erledigt er die meisten seiner Arbeiten.

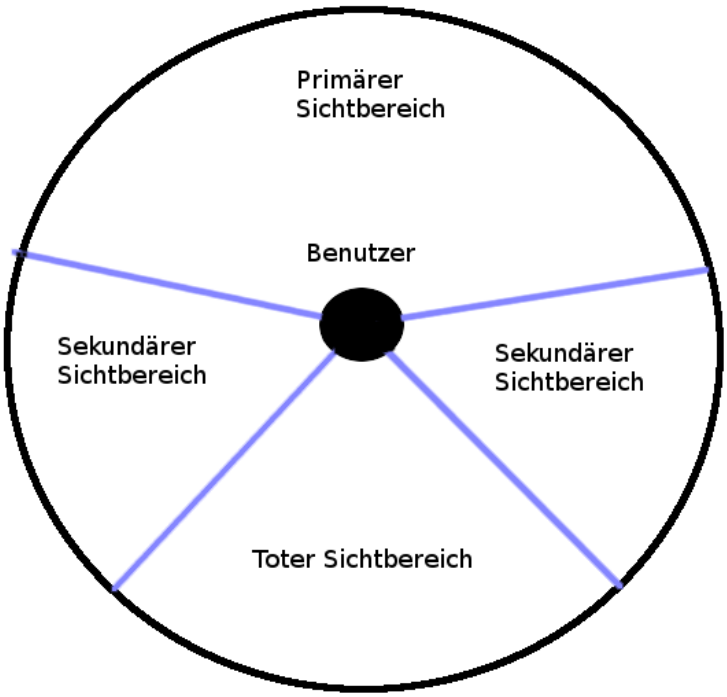

Abbildung 21 - VR Desktop Sichtbereich

Der sekundäre Sichtbereich beschreibt den Bereich in dem der Benutzer hin und wieder blickt.
Der tote Sichtbereich beschreibt den Bereich an den der Benutzer höchst selten blickt.
Zur Ermittlung dieser Bereiche werden die Kopftracking-Daten des Benutzers permanent analysiert.
Je nachdem wie der Benutzer mit dem VR-Desktop arbeitet werden diese Bereiche in ihrer Größe und Orientierung während des laufenden Betriebes angepasst.
Ziel ist es hierbei die Leistungsaufnahme der VR-Desktop-Anwendungen im sekundären Sichtfeld und im toten Sichtfeld zu minimieren. Wenn die Systemauslastung einen gewissen Wert übersteigt werden die betroffenen Progamme in ihrer Prozesspriorität nach unten geregelt. Hierfür wird die Prozessverwaltung des Betriebssystemkerns konsultiert.
Es muss auch angemerkt werden, dass dieser Sachverhalt wissenschaftlich nicht belegt ist und daher als eine reine Entwurfsentscheidung angesehen werden kann.

pause(): Methode ist der Zeitpunkt um System und Benutzer relevante Information, wie zum Beispiel Benutzerdaten, zu speichern. Sie wird aufgerufen wenn sich eine VR-Desktop-Anwendung im toten Sichtbereich befindet und mehr freie System-Ressourcen benötigt werden.

Resume(): Wenn ein möglicher Systeme-Leistungs-Engpass überstanden wurde, können pausierte Anwendungen wieder gestartet werden.

setFree(): Die setFree-Methode ist die letzte Methode die aufgerufen wird bevor der VR-Desktop komplett beendet ist. Hier sollten allokierte Ressourcen wieder freigeben werden.

4.3 Architektur Bausteinansicht

Im Folgenden soll die Komponenten-Sicht auf die Architektur vermittelt werden, nach einer graphischen Veranschaulichung folgt eine textuelle Beschreibung der einzelnen Komponenten. Der Fokus liegt hierbei auf den vom VR-Desktop benötigten Systemprogrammen.

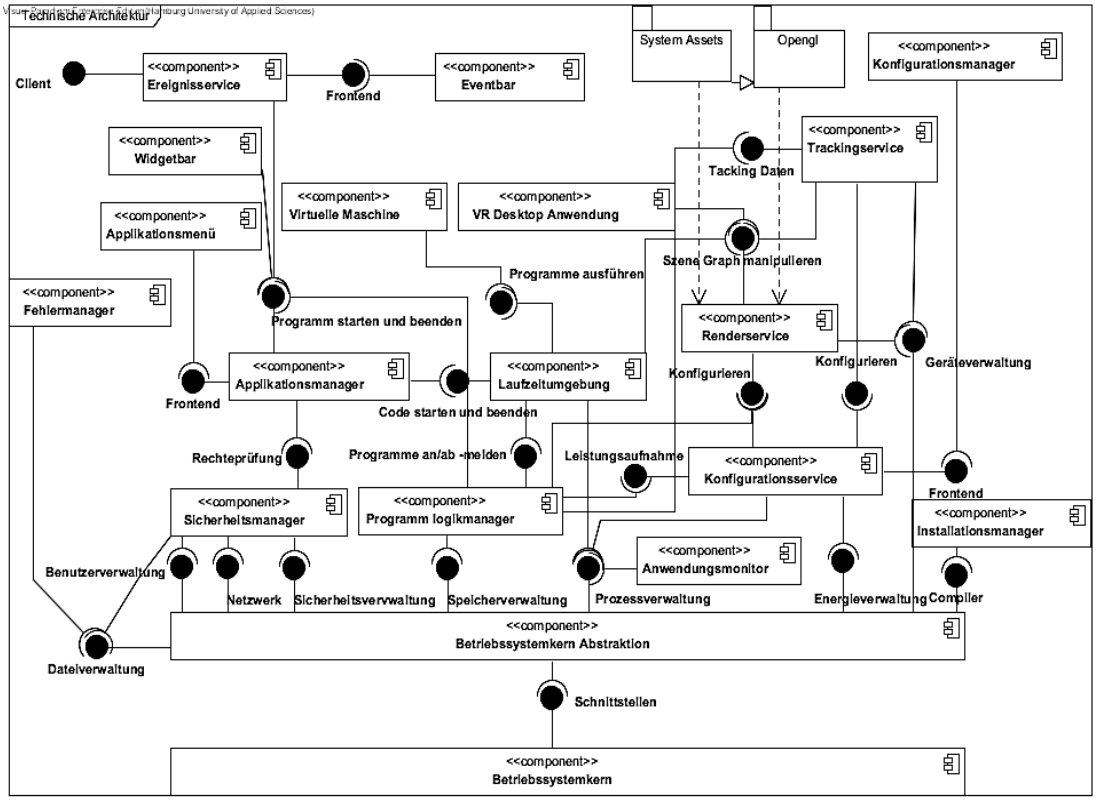

Abbildung 22 - Technische Architektur VR-Desktop

4.3.1 Systemdienste

Fehlermanager: Loggt jegliche Art von System-Ereignissen. Die Möglichkeiten die diese Komponente bietet, werden im Anhang Anforderungen detailliert beschrieben.

Applikationsmanager: Findet Verwendung in der Organisation von Anwendungen. Mit seiner Hilfe werden Anwendungen gestartet, beendet, installiert und deinstalliert. Als Frontend wird das Applikationsmenü verwendet.

Sicherheitsmanager: Prüft alle sicherheitsrelevanten Aspekte des Systems. Um die Wahrscheinlichkeit von Schadcode zum Minimieren werden alle VR-Desktop Applikationsframework Komponenten, die Operationen auf dem Betriebssystemkern ausführen durch den Sicherheitsmanager geprüft. Dies gilt auch wenn VR-Desktop Anwendungen gestartet werden.

Ereignisservice: Backend-Komponente der Eventbar. Hintergrunddienst der von VR-Desktop Anwendungen der verwendet werden kann, um Resultate von Hintergrundprozessen die keine graphische Repräsentation besitzen, anzeigen lassen zu können.

Laufzeitumgebung: Erledigt organisatorischen Aufgaben, die vor der Ausführung einer Benutzeranwendung erfüllt sein müssen, bevor sie von der virtuellen Maschine ausgeführt werden kann. Dies beinhaltet sowohl grundlegende Aufgaben, wie das Laden von Programmcode in den Hauptspeicher, also auch VR-Desktop-spezifische Aufgaben, wie das beim Start einer Applikation benötigte Registrieren am Programmlogikmanager und Renderservices.

Programmlogikmanager: Interpretiert die Werte des Trackingmanagers und leitet sie an die betroffene Anwendung weiter. Hierfür wird aus dem Trackingdatenstrom des Trackingservices, die durch den Benutzer geformte Bediengeste ermittelt, und der Ort bestimmt wo diese im 3D-Raum stattgefunden haben. Anhand einer Liste mit Positionen zu allen ausgeführten Anwendungen, entscheidet der Programmlogik Manager dann, ob eine Anwendung betroffen ist. Falls eine Anwendung betroffen ist wird diese benachrichtig und die ermittelten Informationen dann entsprechend an diese weitergeleitet. Somit stellt Programmlogikmanager die logische Repräsentation für jedes Programm dar. Zusätzlich steuert er indirekt die Benutzeranwendungen durch den VR-Desktop Applikationslebenszyklus.

Konfigurationsservice: Wird verwendet um leistungsrelevante Aspekte des Gesamtsystems anzupassen. Dies reicht von visuellen Faktoren bis über die Konfigurationsmöglichkeiten des verwendeten Betriebssystemkerns. Zur Anpassung der VR-Desktop-Systemleistung prüft der Konfigurationsservice die Systemauslastung des Betriebssystemkerns. Wenn ein

kritischer Wert überschritten wurde, wird eine Nachricht an den Programmlogikmanager übermittelt, daraufhin regelt dieser mit Hilfe des VR-Desktop Applikationslebenszyklus die Gesamtleistung nach unten.

Renderservice: Visualisiert die graphischen Repräsentationen von VR-Desktop-Anwendungen. Hierfür besitzt er einen 3D-Szenegraphen, über Schnittstellen kann dann mit diesem interagiert werden. Den graphischen Aufbau des einzelnen Elements entnimmt er hierbei aus den Styleguide der System Assets. Des Weiteren enthält er jegliche programmtechnischen Elemente, die zur Generierung von Computer-generierten Bildern nötig sind und ruft die render() Methode des Desktop Applikationslebenszyklus auf.

Trackingservice: Stellt einen ungefilterten Stream von Trackingdaten zur Verfügung. Da für Kopftracking-Daten lediglich Positionsangaben dargestellt werden, werden diese direkt weiter an den Renderservice geschickt und dort zur Anpassung des Szenegraphs verwendet.

Installationsmanager: Wird zur Kompilierung von Systembibliotheken verwendet. Dies kann entweder zur Laufzeit oder bei der Installation von Anwendungen, sowie bei der Installation des VR-Desktop selbst geschehen.

4.4 Architektur Laufzeitansicht.

In diesem Abschnitt sollen nun anhand von ein paar exemplarischen Beispielen gezeigt werden, wie sich der VR-Desktop zur Laufzeit verhält. Diese wurden so gewählt, dass sie gewisse Eigenheiten der Architektur hervorheben.

4.4.1 Starten von VR Desktop Anwendungen.

An dem Start von einer VR Desktop Applikation sind viele Komponenten des VR-Desktops beteiligt. Abbildung 23 zeigt das im Folgenden referenzierte Sequenzdiagramm.

1.) Applikationsmanager prüft mit Hilfe des Sicherheitsmanagers, ob die Entität die ein Programmstart angefordert hat die nötigen Rechte besitzt.
2.) Übergibt die Persistenz-Speicher-Adresse des Programmes an den Laufzeitmanager, damit dieser das Programm in den Hauptspeicher laden kann.
3.) Der Laufzeitmanager weißt die virtuelle Maschine an, das Programm auszuführen.
    Die virtuelle Maschine registriert das Programm bei Programmlogikmanager und Renderservice.
4.) Renderservice und Programmlogikmanager beginnen damit den VR-Desktop Applikationslebenszyklus auszuführen.

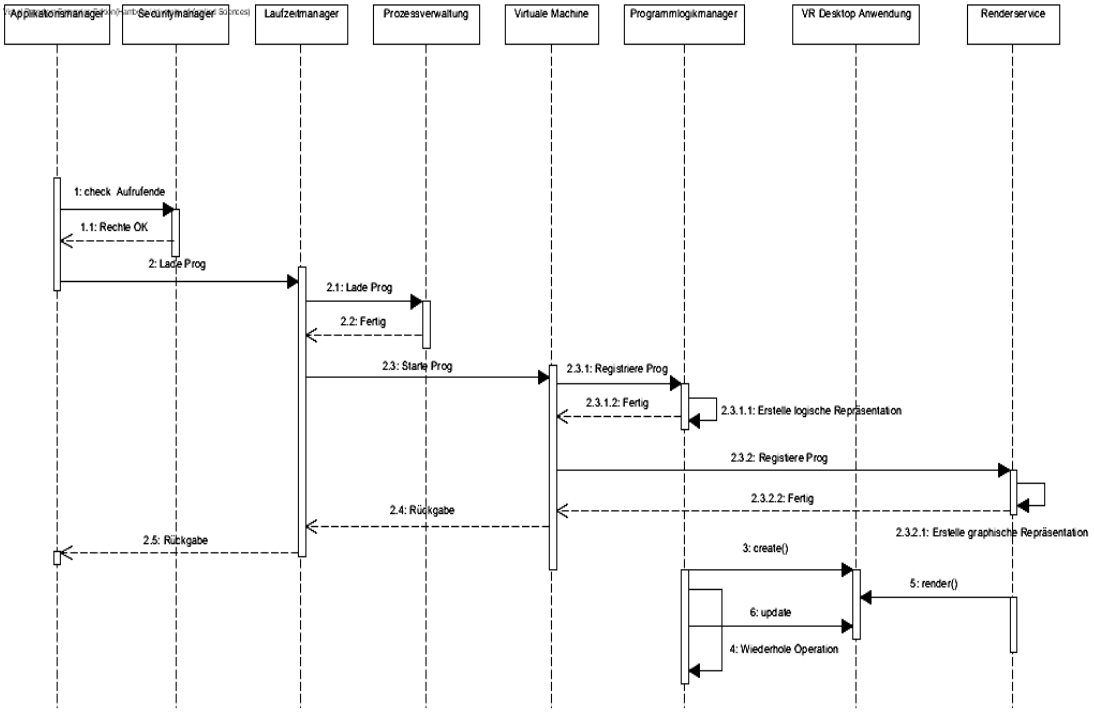

Abbildung 23 - Laufzeitansicht Start VR-Desktop Anwendung

4.4.2 Interaktion mit VR-Desktop-Anwendungen

Das Fenster ist im Kontext des VR-Desktops eines der tragenden visuellen Konzepte. An dieser Stelle soll gezeigt werden, wie programmtechnisch die Interaktion mit dem Fenster abläuft. Dafür wird einmal das Laufzeitverhalten gezeigt, wenn mit dem Fensterrahmen interagiert wird und einmal, wenn mit einer Interaktionsfunktion eines Fensters kommuniziert wird.

Interaktion mit Interaktionsfunktion.

1.) Programmlogikmanager findet Trackingdaten, die die Interaktionsfunktion der VR-Desktop-Anwendung X betreffen. Daraufhin ermittelt er aus dem Trackingstrom die dazugehörige Geste die der Benutzer ausgeführt hat.
2.) Programmlogikmanager informiert Anwendung X, indem er die ermittelten Daten an sie weiterleitet.
3.) Anwendung X reagiert im nächsten Aufruf der Update- bzw. Rendering-Methode auf die eingetretene Benutzer-Interaktion.

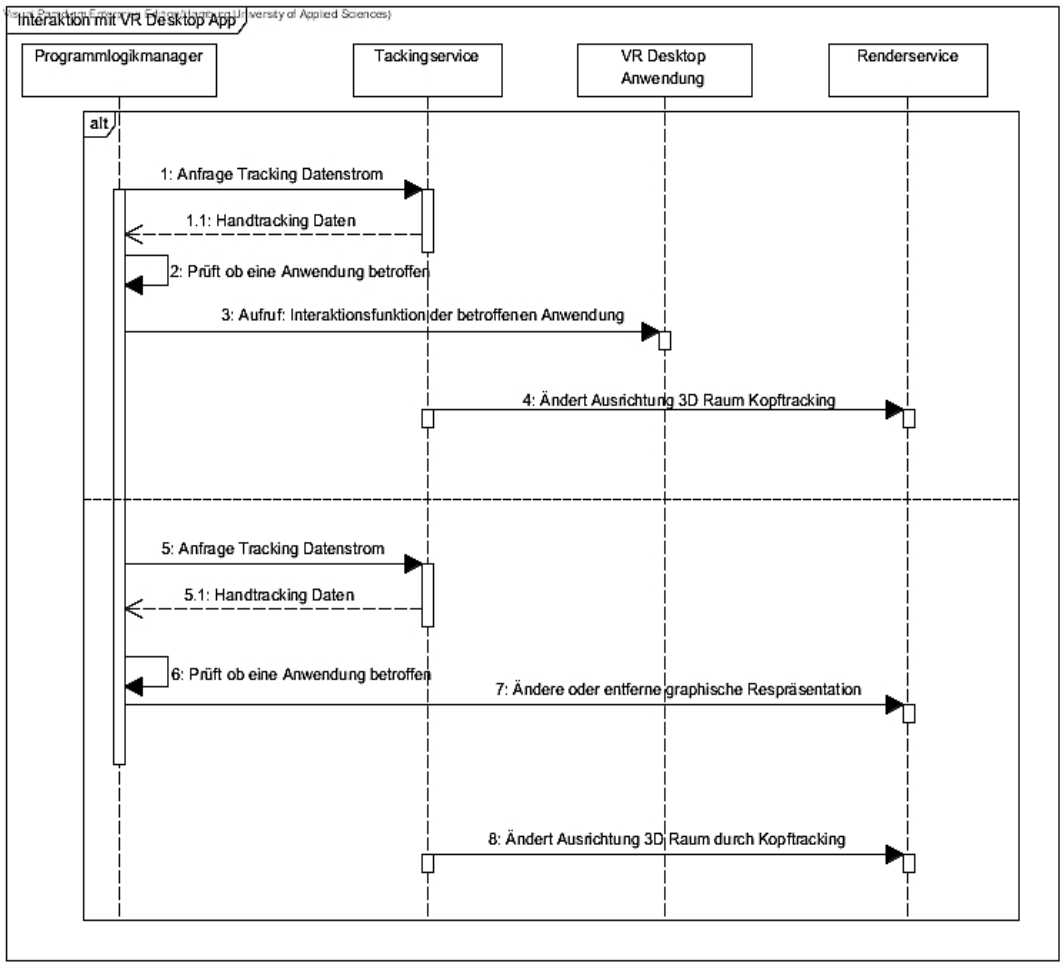

Abbildung 24 - Laufzeitansicht Interaktion mit VR-Desktop-Anwendung

Interaktion mit Fensterrahmen.

1.) Programmlogikmanager findet Trackingdaten, die den Fensterrahmen von VR-Desktop-Anwendung X betreffen. Daraufhin ermittelt er aus dem Trackingstrom die dazugehörige Geste, die der Benutzer ausgeführt hat und gegebenenfalls die Interaktionsfunktion, falls zu Beispiel ein Fenster geschlossen werden soll.
2.) Programmlogikmanager ändert oder entfernt die graphische Repräsentation aus dem Szenegraph des Renderservice.

4.4.3 Auszug Verhalten VR-Desktop Applikationslebenszyklus

Da die slowDown() Methode des Applikationslebenszyklus eine Besonderheit darstellt, soll diese hier noch einmal sein Verhalten zur Laufzeit präsentiert werden.

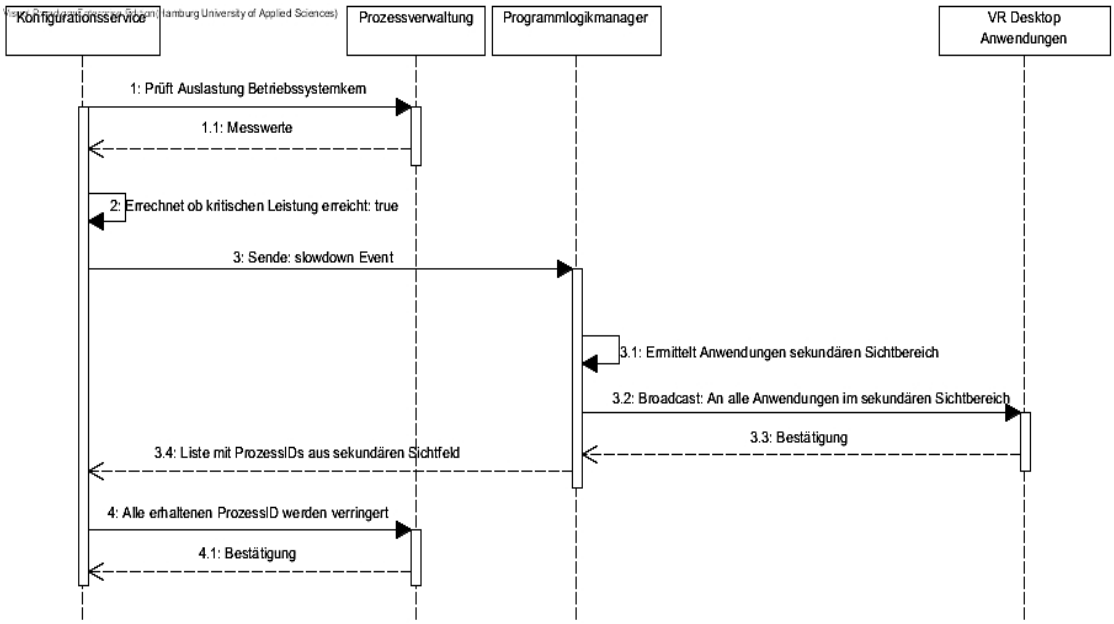

Abbildung 25 - Laufzeitansicht Steuerung durch VR-Desktop Applikationslebenszyklus

1.) Konfigurationsmanager prüft periodisch die Systemauslastung mit Hilfe des Betriebssystemkerns. Wenn ein gewisser Prozentwert überschritten worden ist kontaktiert er den Programmlogikmanager und teilt ihm mit, dass er die Leistungsaufnahme der sich in Ausführung befindlichen VR-Desktop-Anwendungen reduzieren soll.
2.) Der Programmlogik Manager ermittelt daraufhin welche Anwendungen sich momentan im sekundären Sichtbereich befinden und ruft die slowDown-Methode auf.
3.) Anwendungen können nun darauf reagieren, indem sie ihre Rechenzeit minimieren oder ihre graphische Repräsentation, wenn von Anwendung selbst definiert, vereinfachen.
4.) Egal ob die Anwendung Mittel ergreift oder nicht, ändert der Konfigurationsservice die Prozesspriorität der Anwendungen, mit Hilfe der vom Programmlogikmanager erhalten IDs, im Betriebssystemkern herunter.

4.5 Applikationsframework.

Damit der VR-Desktop in seiner Funktionalität erweitert werden kann ist es nötig potenziellen Programmieren eine Sammlung von Bibliotheken zur Verfügung zu stellen, damit diese Anwendungen für den VR Desktop implementieren können. Hierfür soll an dieser Stelle ein einfaches, exemplarisches Framework erstellt werden. Eine Übersicht der Klassen ist in Abbildung 26 dargestellt.

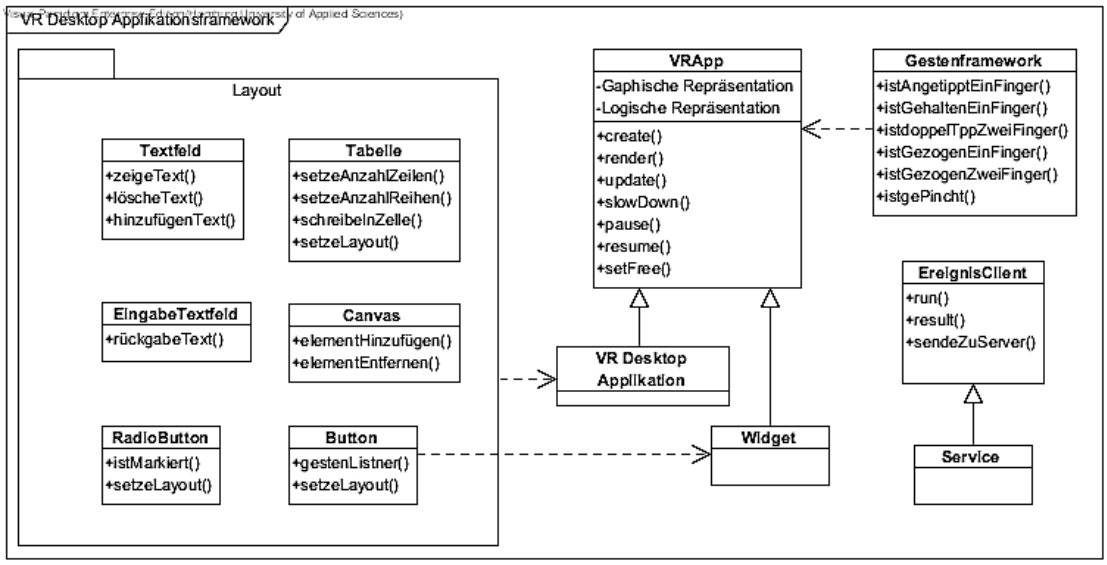

Abbildung 26 - VR-Desktop Applikationsframework

Der VR Desktop Unterscheidet drei verschiedene Arten von Programmen. Die Klasen VR Desktop Applikation und Widget leiten beide von VRApp ab und Implementieren so den VR Desktop Applikationslebenszyklus.

- **VR Desktop Applikation:** Der Standard Anwendungstyp des VR-Desktops. Dieser hat vollen Zugriff auf die Funktonalität des VR-Desktop Applikationsframework.

- **Widget:** Sind eine leichtgewichtige Variante der VR-Desktop-Programme. Als graphische Elemente können sie nur die Buttons des Layout Packages implementieren. Dieser kann durch Animation etc. optisch aufgewertet werden. Diese Animation kann dann einfache Sachverhalte präsentieren, wie die Uhrzeit oder aktuelles Wetter. Die GestenListener() Methode, der Klasse Button, kann dazu benutzt werden um weiterführende Schritte einzuleiten. So kann ein Buttondruck die Mutterapplikation oder eine externe Webseite aufgerufen. Von den Framework Klassen dürfen nur das Netzwerkframework und Dateiframework verwendet werden.

- **Service:** Diese Art von Programm besitzt keine graphische Oberfläche. Alle anderen Framework Klassen können aber verwendet werden. Es kann genutzt werden um Aktionen und Funktionen im Hintergrund auszuführen. Sollte es doch einmal nötig sein, dass eine Ausgabe benötigt wird, dann kann dafür der Ereignis-Service verwendet werden.

Exemplarisch wird in dem vorherigen Diagramm auch Funktionalität gezeigt, die von einer VR Desktop Applikation eingebunden werden kann. Wichtig ist es an dieser Stelle noch einmal zu verdeutlichen, das mit Hilfe der Layout Klasse die Grundlage gelegt wird, dass Entwickler eigene Interaktionsfunktionen erstellen können.

- **Layout:** Stellt die Menge der graphischen Elemente gekoppelt mit deren Interaktionsfunktion.

- **Gestenframework**: Wird verwendet um die verschieden Gesten des VR Desktop Bedienschemata auswerten zu können.

Für jegliche weitere benötigte Funktonalität kann auf die Java Standard Bibliothek zurückgegriffen werden.

5 Implementierung

In diesem Kapitel soll die Möglichkeiten untersucht und beschrieben werden, mit welchen Mitteln man einen Prototypen zu dem in Kapitel 3 und 4 beschriebenen System verwirklichen kann. Der Prototyp soll einen vertikalen Schnitt durch die aufgenommen Anforderungen bieten. Hierfür wird zunächst der zu realisierende Funktionsumfang beschrieben. Danach werden die drei Hauptkomponenten evaluiert.

5.1 Prototype Funktionsumfang

Um die Arbeitsweise des VR-Desktops zu veranschaulichen ist es nötig, einen minimalen Umfang aller Systemebenen zu realisieren. Zu allererst müssen die Hardwarekomponenten mit Softwarekomponenten vereinigt werden. Die Softwarekomponenten werden dafür entsprechend vereinfacht und implementieren nur einen Teil der in Kapitel 3 und 4 spezifizieren Funktionen.
Dies bedeutet im Detail, dass softwareseitig Teile der Logik-Engine, zur Verarbeitung der Tracking Informationen, und Teile der Render-Engine, zur Darstellung der graphischen Repräsentationen, erstellt werden und im Anschluss mit der Hardware von VR-Brille und Bewegungserkennung verbunden werden.

Damit der Prototyp zumindest optisch einem vollentwickelten VR-Desktop gleicht, werden hierfür die spezifizierten, visuellen Bausteine realisiert. Jedoch wird nicht die für sie spezifizierte Funktionalität entwickelt. Sie dienen lediglich dazu das „Look and Feel“ des VR-Desktops zu vermitteln. Die folgenden Anforderungen wurden aus dem Anhang Anforderungen entnommen.

- **SN1** 3D Raum
- **SN3** Eventbar um den User über System und Anwendungsevents zu informieren
- **SN6** Fenster zur Einbettung von Programmen:
- **SN7** Widgetbar zum Anlegen von Applikationsverknüpfungen.

Damit auch ein Stück weit mit dem Prototypen interagiert werden kann, werden Teile des Bedienkonzepts implementiert. Hierfür sollen die folgenden Teile der Bewegungsschemata des VR-Desktops implementiert werden.
Damit das Bewegungsschema dem Anwender ein Feedback liefert sollen sich zumindest Fenster verschieben lassen. Die folgenden Anforderungen wurden aus dem Anhang: Anforderungen entnommen.

- **BD1** Finger besitzen graphische Repräsentation mit Freiheitsgraden auf der X, Y, Z Achse.
- **BD2** Geste zur Ausführung von Programmen und Funktionen.
- **BD3** Geste zum „Drag und Drop“ von Elementen im virtuellen Raum.
- **BD6** Geste Aktion abbrechen.

Schlussendlich soll etwas Systemfunktionalität Implementiert werden, um damit einen Einblick in die alltäglichen Arbeitsabläufe und deren Abarbeitung zu vermitteln. Hierfür wird der unter **SD2** spezifizierte Medienplayer implementiert.

5.2 Evaluation

An dieser Stelle soll bestimmt werden, mit welchen Verfahren der Prototyp am besten zu implementieren ist. Hierfür werden verschiedene, mögliche Vorgehensweisen vorgestellt.

5.2.1 VR-Brille

Zum Zeitpunkt der Erstellung dieser Arbeit beschränkt sich die Auswahl der tatsächlich verfügbaren bzw. erwerbbaren VR-Brillen auf eine überschaubare Anzahl.
Diese sind entweder die Oculus Rift und dessen Derivat Samsung Gear VR [5.5], sowie diversen Low-End-Varianten wie die Durovis Dive 5 [5.18], die aus einer Kombination von Halterung und einen Smartphone bestehen.

5.2.2 Logik und Rendering Komponente

Die Rendering Komponente visualisiert Benutzer-Eingaben und mögliche Systemevents und stellt diese mit Hilfe von Polygonnetzen dar. Zur Realisierung gibt es mehrere Herangehensweisen. Da diese Komponente das zentrale Element der Entwicklung darstellt wird sie umfangreicher als die anderen Komponenten evaluiert.

1.2.2.1 Eigenbau: Rendering Subsysteme werden im Laufe der Bachelorarbeit selbst erstellt. Gearbeitet wird vorrangig mit OpenGL Elementen. Diese werden zu neuen Elementen zusammengefasst umso neue Abstraktionsebenen zu schaffen.

- **Vorteile:** 100 prozentige Kontrolle über die Quellcode. Einzelne Funktionen lassen sich genau auf die von ihnen zu lösenden Aufgaben anpassen.

- **Nachteile:** Sehr aufwendig. Bevor mit der eigentlichen Ausgabenstellung begonnen werden kann, müssen elementare Grundsysteme für Graphik, Animation und Sound erstellt werden. Durch die große Anzahl von zu implementierenden Subsystem kann dies zu einer erhöhten Fehleranfälligkeit führen.

1.2.2.2 Middleware: OpenGL-Elemente sind bereits zu neuen Elementen abstrahiert. Je nach verwendeter Middleware werden verschiedene grundlegende Operationen und Manipulationen auf diese angeboten, wie zum Beispiel Import und Export von bestehenden Polygonnetzen, Funktionen für die Animation von Polygonnetzen, aber auch Speichermanagement. Produkte die in dieser Kategorie sind zum einen das kommerzielle Autodesk Gameware [5.2] und das freie LibGDX [5.3].

- **Vorteile:** Grundlegende Subsysteme sind bereits entworfen, es kann zügig mit der Implementierung der eigentlichen Aufgabe begonnen werden.

- **Nachteile:** Oftmals kostenpflichtig, wenn frei dann unter Umständen schlechte Dokumentation oder Pflege.

1.2.2.3 3D Engine: Bietet neben den Funktionen einer Middleware auch Editoren zur Bearbeitung von verschieden Teilaufgaben. Funktionserweiterungen erfolgen durch Scripting-Sprachen wie Lua [5.1], oder eine speziell für die entsprechende 3D-Engine entwickelte Scripting Sprache.

- **Vorteile:** Alle benötigten Funktionen sind bereits vorhanden, es kann sofort mit der Implementierung der Problemstellung begonnen werden.

- **Nachteile:** Oftmals kostenpflichtig. Die Architektur und Quellcode der Engine ist oftmals proprietärer Quellcode. Anpassung der angeboten Funktionalität der Engine oftmals nur indirekt über Scripting-Sprachen möglich.

5.2.3 Bewegungs-Sensorik

Bei der Auswahl von Bewegungssensoren kann zwischen zwei grundlegenden Verfahren gewählt werden. Einmal wird die Sensorik auf den Betrachter gerichtet und einmal wird die Sensorik am Anwender selbst befestigt und in den Raum gerichtet. In erste Kategorie fallen Sensoren wie die Microsoft Kinect [5.6]. In zweite fallen Geräte wie der Motion Leap Sensor [5.7].

5.2.4 Fazit

Am besten scheint die Verwendung einer Middleware geeignet. Sie bietet das Beste aus beiden Welten, der Code kann gut an die benötigte Funktionalität angepasst werden und die grundlegenden benötigten Funktionen sind bereits vorhanden. Deshalb soll der VR Desktop auf Basis des LibGDX-Frameworks entstehen. Zusätzlich bietet das Framework ein hohes Maß an Portabilität. Die Funktionalität muss nur einmalig implementiert werden und kann dann von allen LibGDX unterstützten Plattformen ausgeführt werden.
Für die Tracking Aufgaben soll der Leap Motion Sensor verwendet werden. Er bietet den Vorteil, dass er sich direkt an einer VR-Brille befestigen lässt. So kann sichergestellt werden, dass wenn sich ein Anwender um 180 Grad dreht, immer noch Handtracking Daten erfasst werden können. Dieses Szenario ist mit einer Kinect und der vorher festgelegten Position zum Anwender nicht möglich. Durch eine 180 Grad Drehung würden unter Umständen die Finger des Anwenders durch seinen Rücken verdeckt werden. Dies würde den Aktionskreis, mit dem der Anwender graphische Elemente auf dem VR-Desktop platzieren kann unweigerlich einschränken und somit eines der offensichtlisten Vorteile gegenüber herkömmlichen 2D-graphischen Oberflächen stark einschränken. Ein andere wichtiger Aspekt ist, dass der Leap Motion Sensor im Gegensatz zur Kinect mehrere Verschiedene Betriebssystemplattformen unterstützt wie zum Beispiel Android [5.11]. Damit kann eine höhere Portabilität des VR-Desktops erreicht werden.

Zur Darstellung der VR-Desktop Inhalte soll das Oculus Rift DK1 [5.8] verwendet werden. Diese Entscheidung basiert alleine auf der Verfügbarkeit der Brille. Aber insgesamt ist eine AR-Brille oder eine VR-Brille mit Außenkameras gegenüber der Oculus Rift zu bevorzugen. Mit ihrer Hilfe können die indirekt durch Oculus Rift vorgegebene Anwender-Anforderung des Blindschreibens auf der Tastatur bzw. die Anforderung einer Software Tastatur eliminiert werden.

5.2.5 Exkurs LibGDX

Das LibGDX-Projekt ist eine Middleware für Cross-Plattform Spiele-Entwicklung. Das Framework ist in Java geschrieben und nutzt für Performance-kritische Abschnitte das Java Natives Interface zur schnelleren Verarbeitung.
Das Framework abstrahiert die Unterschiede zwischen den einzelnen Zielplattformen auf Basis von OpenGL.
Wie bei Cross-Development üblich, benötigt man bei der Entwicklung mit LibGDX [5.3] lediglich eine Codebasis. Der übliche Entwicklungszyklus besteht aus Programmieren und anschließendem Testen auf einem Desktop PC.
In regelmäßigen Abständen (z.B. mit Milestone-Versionen) wird dann überprüft, ob das Projekt noch auf anderen Zielplattformen, wie zum Beispiel Android funktioniert.

Eines der Hauptziele von LibGDX ist, eine vollständige Abstraktion über Desktop und mobile Geräte, so dass der einzige Unterschied die Geschwindigkeit bzw. Rechenleistung der einzelnen Zielplattformen ist.

5.2.6 Resümee Prototyp 1

Das Erstellen einer graphischen Oberfläche ist ein ambitioniertes Unterfangen. Als die Evaluation durchgeführt wurde war sich der Autor über die Komplexität der einzelnen Aufgaben entweder nicht bewusst oder hat den zeitlichen Aufwand unterschätzt.
Ein Problem ist das von VR-Technik benötigte stereoskopische Rendering. Dieses wird nicht von LibGDX native nicht unterstützt und sollte selbst entwickelt werden.
Dies sollte mit Hilfe des OpenGL Befehls glScissor [5.9] geschehen. Leider stellte sich dabei heraus, dass die Verwendung dieser Funktion gewisse Seiten-Effekte mit sich bringt. Was eine umfangreichere Einarbeitung in OpenGL erforderlich gemacht hätte. Daraufhin entschied sich der Autor die Problemlösung zurückzustellen und das zu Implementierende System auf weitere, mögliche, unvorhersehbare Komplikationen zu untersuchen. Nach längeren Experimentieren mit dem von Oculus Rift und der zum Betrieb benötigten Teil-Funktionalität, wie Headtracking, Distortion-Shader und stereoskopischen Rendering, beschloss der Autor, sich erst einmal mit der LibGDX-3D-Api zu beschäftigen und stieß dabei auf ein weiteres Problem. Obwohl die 3D-Api von LibGDX durchaus potent erscheint, bietet sie lediglich eine unzureichende Dokumentation.
Nachdem dieser weitere Störfaktor ausgemacht wurde begann der Autor an dem Ergebnis seiner Evaluation zu zweifeln. Es handelte sich lediglich um kleine Teilaspekte der gesamten Aufgabenstellung und diese wurden durch die in der Evaluation festgelegte Struktur zu umfangreichen Problemstellungen und benötigten daher eine wesentlich größere Einarbeitungszeit als vorher veranschlagt.
Daraufhin entschied sich der Autor noch einmal die Evaluation zu überdenken. Dies wurde zusätzlich durch den Umstand beschleunigt, dass mit anderen Teilaspekten die zur Lösung der Aufgabenstellung benötigt wurden noch nicht einmal begonnen worden war. Unter diese Teilaspekte fallen die Erstellung von 3D-Modellen und das komplette Ausformulieren der Programmlogik.

Schlussendlich ließ sich feststellen, dass die Abstraktionsmechanismen die das LibGDX-Framework bereitstellt nicht ausreichen, um im Rahmen einer Bachelorarbeit im entsprechenden Zeitrahmen zu einer zufriedenstellenden Lösung zu gelangen. Das bedauernswerte an diesen Umstand ist, dass nach dem Wechsel der Implementierungstechnik der Autor auf ein paar andere Frameworks aufmerksam geworden ist, die diese Aufgaben vielleicht besser bewältigten hätten können, dies aber im Rahmen der Bachelorarbeit nicht mehr abschließend geklärt werden kann. Exemplarisch sei hierbei das jovr [5.10] Projekt genannt.

5.3 Evaluation zweiter Versuch

Nach dem Scheitern der Implementierung des ersten Prototyps auf Basis einer Middleware Lösung und den daraus gewonnenen Erkenntnissen, kam für die Implementierung nur noch der Einsatz einer 3D-Engine in Frage. Das Hauptauswahlkriterium in dieser zweiten Evaluation war der Wunsch zur Reduzierung der Komplexität der gegebenen Aufgabenstellung.
Im Gegensatz zur ersten Evaluation die als Hauptauswahlkriterien die möglichen Portabilität des VR Desktops vorsah.
Die andere Auswahlmöglichkeit, alles im Eigenbau zu entwickeln, hätte den benötigten Aufwand nur noch weiter erhöht und wird daher nicht weiter untersucht.
Die in der ersten Evaluation ausgewählte VR-Brille und Tracking-Sensorik werden in den zweiten Prototypen ohne Änderung übernommen.

Aufgrund der Kombination von Leap Motion Sensor und Oculus Rift, ist die Auswahl potentiellen 3D Engines auf 2 beschränkt. Momentan bieten nur die Unreal 4 Engine [5.13]. und die Unity3D Engine [5.16] ein, von offizieller Seite unterstütztes, Leap Motion Plug-In an.
Technisch gesehen sind zwischen den beiden Engines keine gravierenden Unterschiede festzustellen [5.14]. Deshalb war die Tatsache, dass der Autor bereits Erfahrungen mit der Unreal Engine 3 hat einer der Hauptgründe für die Verwendung dieser. Diese wurde noch zusätzlich dadurch begünstigt, dass kurz bevor diese Entscheidung getroffen wurde, der Quellcode der Engine freigeben [5.15] und Revenue Modell so geändert wurden, dass sie nun ohne monatliche Kosten verwendet werden kann.

5.3.1 Unreal Engine Exkurs

Die Unreal Engine ist eine Spieleentwicklungsumgebung der Firma Epic Games [5.13]. Diese liegt momentan in ihrer 4ten Iteration vor. Sie bietet neben einem Graphik-Kern, einen Drag und Drop Level-Editor und andere Editoren die für Zwischensequenzen, Material und Scripting.
Die Erweiterung der Funktionalität der Engine erfolgt entweder in Kombination von C++ und einer von Epic Games selbst entwickelten visuellen Scripting Sprache namens Blueprint. In diesen Fall werden die erarbeiteten Resultate in Form einer direct link library, der Engine hinzugefügt und sind dann im Blueprint Editor verfügbar. Oder die Erweiterung der Funktionalität kann auch komplett in Blueprint Editor erstellt werden. Hierfür liefert die Engine vorgefertigte Klassen, die dann verwendet werden können.
Zum besseren Verdeutlichung folgend ein Beispiel.

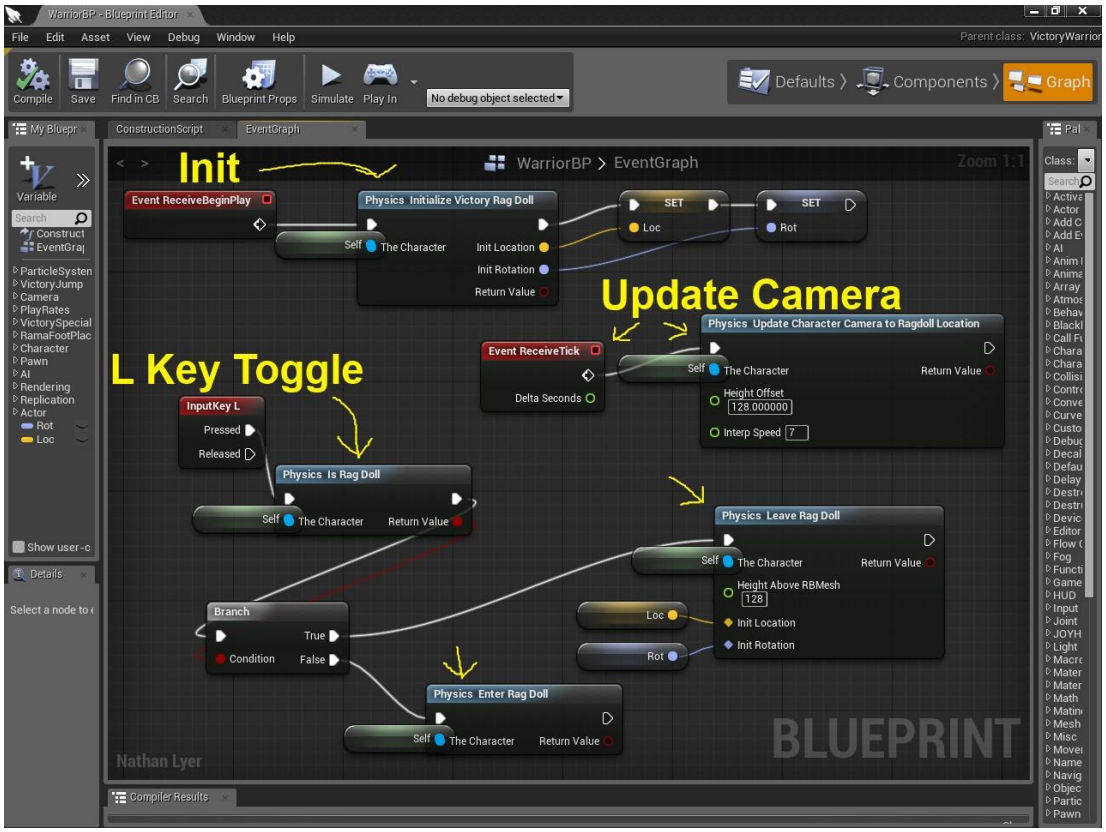

Abbildung 27 - Beispielskript in Unreal Blueprint [5.12].

Die Elemente mit einer roten Titelleiste sind Werte, die von der Engine abfragt werden können. Diese Werte können entweder periodisch oder durch ein bestimmtes Ereignis auftreten. In der Graphik wird zum Beispiel das Event Receive Tick verwendet. Dieses Element wird einmal pro Frame aufgerufen.

Elemente mit einer blauen Titelleiste sind Funktionen, die auf eingetreten Events reagieren und ausgeführt werden können. Mit Hilfe von C++ lassen sich eigene solcher Funktionen erstellen.

5.4 Resümee Prototyp 2

In der Theorie soll die Oculus Rift, sobald sie an einen Computer angeschlossen wird, durch die Unreal Engine erkannt werden. Wenn dies der Fall ist, wird der Level-Editor daraufhin um VR-spezifische Funktionalität erweitert. In der Praxis erwies sich dieser Sachverhalt als nicht ganz so reibungslos wie in der Theorie beschrieben. Eine genaue Fehlerquelle konnte dabei nicht ausgemacht werden. So wurde die Brille mal erkannt und mal nicht.

Für die Verwendung des Leap Motion Sensors wurde es erforderlich die Engine selbst zu kompilieren, da das Plug-In momentan nicht in der aktuellen vorkompilierten Version der Engine vorhanden ist.
Das Kompilieren der Quellcode-Version selbst verlief reibungslos. Jedoch musste der Autor beim ersten Buildprozess eines Projekts feststellen, dass ein Fehler in der Engine verhinderte, dass das Projekte außerhalb des Editors erstellt bzw. ausgeführt werden könnte. Daraufhin wurde der Code nochmals kompiliert, doch das Problem bestand weiterhin. Daraufhin wurde das Betriebssystem nach möglichen Fehlerquellen untersucht. Jedoch ohne Erfolg.
Um diese Variante des Prototyps trotzdem voran zu treiben, wurde dieses Problem nach hinten angestellt und der Autor begann mit dem Leap Motion Sensor zu experimentieren. Zu seiner Überraschung musste er feststellen, dass das Leap Motion Plug-In momentan nur einen Teil der gesamt Funktionalität bereitstellt [5.17], was das Implementieren einer Drag und Drop Funktionalität unmöglich machen würde. Dieser Umstand wurde trotz gründlicher Recherche vorher nirgends entdeckt.
Eine weitere Herausforderung ist die Unreal Engine selbst. Während sich, solange man den Standard Workflows hält, schnell Resultate produzieren lassen, stellt ein Prototyp wie der virtuelle Desktop eine dezente Zweckentfremdung der Engine dar. Dies führt dazu, dass man ein tieferes Verständnis für die Materie benötigt und so die Lernkurve der Engine stark ansteigt.
Schlussendlich wurde das Unterfangen zugunsten der textuellen Ausarbeitung der Bachelorarbeit aufgegeben.

5.5 Fazit Prototypen Entwicklung

Ursprünglich waren für diese Arbeit 5 Wochen Implementierungsaufwand vorgesehen.
Tatsächlich wurden 15-17 Wochen dafür aufgewendet. Dies ist zum Teil der aufwendigen Einarbeitungszeiten in jeweilige Technologie oder Frameworks geschuldet. Zum anderen dem Anspruch des Autors, eine funktionierende Version zu produzieren.
Momentan verhält es sich mit den Software Lösungen, die als Basis für die Entwicklung von VR-Inhalten in Frage kommen, so, wie mit der VR-Technik selbst. Alle Grundlagen sind vorhanden, jedoch fehlt es an Feinschliff. Die für die Arbeit betrachtete Software, auf deren Basis der VR-Desktop realisiert werden sollte, wurde oftmals nicht mit einer Funktionalität für VR im Hinterkopf entwickelt.
Der Wechsel der Technologie hat die Probleme nur verlagert. Während der LibGDX-Prototyp Probleme mit den Hardware Komponenten hatte, bestand die Schwierigkeit des Unreal-Prototypen bei der Implementierung der benötigten Funktionalität.
Das bedeutet vermutlich, dass es momentan immer irgendwo einen Flaschenhals gibt, der massive Probleme verursacht. Die VR-Technik und der während dieser Arbeit verwendete Leap Motion Sensor sind einfach noch zu junge Konzepte, um auf deren Basis zügig einen Prototypen zu realisieren. Sie befinden sich eher noch in einem experimentellen Stadium. Im Endeffekt ist der Autor genau auf die Fragestellung getroffen, die er in der Einleitung

aufwirft. Wie wird Software VR gerecht realisiert? Die Prototypen verdeutlichen diesen Sachverhalt. Momentan gibt es einfach noch keine Antwort auf diese Frage. Vielmehr findet momentan ein kollektiver Lernprozess statt, der versucht diese Frage zu klären und ein häufiges Scheitern gehört zu so einem Lernprozess dazu.

6 Fazit

Die vorliegende Arbeit hat gezeigt, dass es konzeptionell möglich ist mit den heutzutage erhältlichen Technologien eine dreidimensionale, graphische Oberfläche zu erstellen, die als Basis VR-Technik, Bewegungserkennung und 3D-Graphik verwendet.

6.1 Technische Aspekte

Aber die technischen Gegebenheiten, die für die Realisierung eines solchen VR-Desktops nötig sind, sind zum Zeitpunkt der Erstellung dieser Arbeit noch in einem sehr frühen Stadium. Viele der zur Realisierung benötigten Hilfsmittel, wie Treiber und Frameworks, befinden sich entweder noch in einer Betaphase oder die benötigte Funktionalität werden nur von Heimentwicklern bereitgestellt, weil sie von offizieller Seite nicht unterstützt werden. Als Beispiel sei hier der Beta Unreal Engine Treiber von Leap Motion und die Java Unterstützung der Oculus Rift genannt.
Leider kann im Rahmen dieser Arbeit nicht abschließend geklärt werden, ob die Verwendung einer anderen Komponenten-Konstellation, wie zum Beispiel die Verwendung der Unity-3D-Engine in Kombination mit einer Kinect Kamera zu einem besseren Ergebnis geführt hätten.

6.2 Organisatorische Aspekte

Insgesamt lässt sich feststellen, dass der benötigte Aufwand der betrieben werden muss um eine graphische Oberfläche zu konzeptionieren und Implementieren sehr hoch ist. Bei der Entwicklung einer solchen Oberfläche bedarf es der Lösung einer Unmenge an verschiedenen Problemstellungen. Diese erstrecken sich über viele Teilbereiche der Informatik und reichen von hardwarenahen bis zu Oberflächen-gestalterischen Aspekten.
Selbst nachdem eine große Menge der verschieden Problemstellungen erfasst wurde, stellt sich dennoch die Frage, mit welcher Granularität diese bearbeitet werden müssen, damit sie im Umfang einer Bachelorarbeit angemessen präsentiert werden können.

6.3 Konzeptionelle Aspekte

Natürlich müssen auch Fragen, die durch die Eigenschaften der im VR-Desktop verwendeten Technik entstehen noch abschließend geklärt werden. Denn zwischen einem Prototypen und einem produktiv einsetzbaren System liegen Welten. Selbst wenn alle in dieser Arbeit definierten Anforderungen erfüllt sind muss geklärt werden, ob es sich bei dem VR-Desktop eher um eine technische Spielerei handelt, oder ob die in ihm verwendeten Konzepte einen wirklichen Mehrwert für die tägliche Arbeit mit dem Computer liefern. Die für diese Arbeit verwendeten Hardwarekomponenten lassen einige Problemstellungen entstehen, die für die tägliche Arbeit hinderlich sind.
Ein großes Problem ist momentan die Kombination von VR-Brille und Hand-Bewegungserkennung. Textuelle Eingaben an ein Computersystem sind aus dem täglichen Alltag nicht weg zu denken. Die reicht vom Suchen von Daten mit einem Dateimanager, Eingeben von URLs, bis zum Schreiben von E-Mails und Texten in Textverarbeitungen.
Der für diese Arbeit verwendetet Motion Leap Sensor [5.7] benötigte viel zu viel Zeit um Handbewegungen korrekt zu erfassen, um als eine ernsthafte Alternative zu einer realen Tastatur verwendet werden zu können. Sicherlich kann man die reale Tastatur als einen Fal-Back-Mechanismus verwenden, doch hier kommt ein Problem der momentan angekündigten VR-Brillen hinzu. Von allen durch den Autor evaluierten VR-Brillen war nur eine, die mit einer Außenkamera bestückt war. Dies bedeutet, dass die VR Brille entweder abgenommen werden muss, wenn Text eingegeben wird, oder der Anwender ist im Stande blind zu tippen (10-Finger-Schreiben).
Dieses Problem lässt sich mit einer AR-Brille leicht lösen. Hierbei kann der Anwender ohne Einschränkungen oder andere Hilfsmittel auf eine normale Tastatur zurückgreifen. Dies lässt den Schluss zu, dass momentan AR-Brillen gegenüber VR-Brillen in Kombination mit dreidimensionalen graphischen Oberflächen bevorzugt werden sollten.

Aber die Bewegungserkennung allein besitzt schon eine gravierende Tücke, die die Verwendung für das tägliche Arbeiten mit dem Computer problematisch machen könnte. Das Problem der Informationssicherheit bzw. Vertraulichkeit. Bewegungsgesten lassen sich wesentlich leichter Abhören, als eine vergleichbare Interaktion mit traditionellen Eingabegeräten. Ein solches Szenario ist denkbar einfach zu entwerfen.
Im einfachsten Fall müssen sich nur zwei Anwender gegenüber sitzen. Bob führt seine tägliche Arbeit aus und Eve beobachtet ihn dabei. Eve spiegelt die Tracking-Daten von Bob und kann so in Echtzeit seine Interaktionen verfolgen.
Die Frage ist, ob sich dieses Problem überhaupt zufriedenstellend lösen lässt. Ganz zu schweigen davon, wie ein rechtlicher Rahmen dafür ausgelegt werden müsste.

Ein weiteres Problem könnten die benötigte Hardware Anforderungen, die für den Betrieb eines solches System benötigt werden, darstellen. Es lässt sich schwer abschätzen welche Hardware-Anforderungen ein voll entwickeltes System tatsächlich benötigt. Während sich die graphische Darstellung meisten über geometrische Primitive realisieren lässt und somit

entsprechend wenig Aufwand rendern lässt, liegt die wahre Schwierigkeit in der Häufigkeit, in der dies getan werden muss. Der „Oculus Rift best pratice Guide“ [2.28] empfiehlt, dass die Bildrate genauso hoch ist wie die des in der Oculus Rift verwendeten Displays. Dies würde bedeuten, dass die Inhalte 90-mal pro Sekunde gerendert werden müssten. Darüber hinaus soll dies bei aktivieren VSync und ohne die Verwendung eines Framebuffers geschehen, was zu einer zusätzliche Belastung der verwendeten Hardware führt.

6.4 End-Fazit

Trotz alle dessen wurde gezeigt, dass die Konzeption eines solchen Systems sich zum größten Teil mit heutzutage verwendeten Konzepten Umsätzen lässt. Die Interaktionsweise mit dem Computer muss nicht komplett neu erfunden werden, viele Konzepte von graphischen Oberflächen lassen sich in eine dreidimensionale Oberfläche übernehmen und können angepasst werden. Auch die bestehenden logischen Konzepte der Betriebssystem-Forschung, die außerhalb der graphischen Oberfläche wirken, lassen sich verwenden, so dass die hier angesprochen technischen Probleme mehr als „Kinderkrankheiten“ verstanden werden können, die sich zum größten Teil mit der Verbesserung der Software und Hardware von VR-, AR- und Tracking-Technik von selbst lösen lassen.

Eine große Frage ist sicherlich, ob die Informationssicherheit durch eine auf Bewegungsgesten basierenden Steuerung, jemals befriedigend geklärt werden kann. Aber dieser Sachverhalt muss Individuell von jeder Anwendungsdomäne oder Person selbst entschieden werden.

Schlussendlich lässt sich feststellen, dass jetzt ein sehr guter Zeitpunkt ist, die Grundlagen Forschung an solchen alternativen Bedienkonzepten voran zu treiben.

6.5 Ausblick

Die Jahre 2015 und 2016 sind spannende Jahre für die VR-Technik. Bis zum Ende des ersten Halbjahr 2016 wollen drei der führenden Entwickler von VR-Technik Ihr Produkt auf dem Markt platzieren.
Interessant ist dies vor allen Dingen deshalb, weil sich die Herangehensweisen und die Ausgangsituationen der drei total verschieden sind. Da wäre zu aller erst Oculus [6.2], die ein paar der führenden Köpfe der VR-Technik hinter sich vereinigen können. Ihr Fokus liegt eindeutig auf dem PC-Gaming-Markt.
Dann wäre da Sony [6.3] mit ihrem Projekt Morpheus, die im Gegensatz zu Oculus mit der PlayStation 4 an eine feste Hardware gebunden ist. Sieht man sich, die weitaus höheren, von Oculus präsentieren Hardware Anforderungen an [6.4], so wirft dies schnell die Frage

auf, ob das benötigte Minimum an Hardware Ressourcen für VR-Technik vielleicht doch niedriger ist als gedacht.
Als dritter sei noch die HTC Vive genannt, die im Gegensatz zu den anderen beiden, den kompletten Raum virtualisieren möchte, in dem sich der Benutzer befindet.

Natürlich sollte in diesem Zusammenhang auch Microsofts Hololens erwähnt werden, die einen AR-Ansatz verfolgt. Eine Koppelung zwischen der Hololens und Windows 10 bietet eine Vielzahl von Möglichkeiten. Dies könnte möglicherweise bedeuten, dass eine Art von VR-Desktop bereits in naher Zukunft verfügbar ist.

Aber auch auf dem Bewegungssensor Markt gibt es interessante Entwicklungen, so wurde zum Beispiel Nimble VR [6.6] von Oculus gekauft. Nimble-VR bietet eine ähnliche Technologie wie der Motion Leap Sensor [6.7]. Dies ist exemplarisch für ein Zusammenwachsen der Bewegungs-Sensorik und VR-Technik.

Schlussendlich werden die komplette VR-Branche, sowie der Anwender von diesen unterschiedlichen Lösungsansätzen profitieren und die Entwicklung von VR-Technik um viele Aspekte erweitern.

Glossar

A
AR – Argumentierte Realität,
Applikationsmenü – VR Desktop graphisches Frontend zur Verwaltung von Applikationen
Anwendungsmonitor -
Animiation – Graphisches 3D-Model, das sich bewegt.

B
Backend – Software-Komponente, die überwiegend logische Aufgaben übernimmt

C
CAD- Computer Aided Design

D
Distortion Shader – Shader der eine Bildkrümmung verursacht Bsp. Fischaugen Effekt.

E
Eventbar – VR-Desktop-Konzept. Bietet graphische Oberfläche für Hintergrund-Prozesse.

F
Framebuffer – Teil der Graphikkarte, der ein Computer generiertes Bild solange zwischenspeichert, bis es vollständig aufgebaut. ist.
Frontend – Zumeist graphische Repräsentation eines Software-Systems.

G
Graphische Oberfläche – Visuelle Aufbereitung von Daten zur Interaktion mit dem Benutzer.

H
Handtracking- Ermittlung von Handbewegungen mittels Sensorik.

I

Immersion – Psychologisches Konzept, wie sehr sich jemand in einen Sachverhalt einbezogen fühlt.

Interaktionsfunktion – Beschreibt eine Fläche in Raum, zumeist auf einem graphischen Element angebracht, mit dem der Benutzer ein Stück Programmkode ausführen kann.

J

K

Kopftracking – Ermitteln von Kopfbewegung mittels Sensorik.

Konvex - Adj. Bsp. Ein nach außen gewölbter Gegenstand.

Konkave - Adj. Bsp. Ein nach innen gewölbter Gegenstand.

L

Logging – Schreiben von Status-Information zwecks späterer Auswertung.

M

Motion Capturing – Verfahren, um menschliche Bewegungen auf den Computer zu übertragen

N

O

Overhead – Verwaltungsaufwand. Leistung, die zusätzlich zur Leistung der Problemlösung benötigt wird.

P

Polygonale Netze – Grundlage von 3D Graphik

Postprocessing – Nachbearbeitung des Computer-generierten Bildes.

Pixel Shader – Wird pro Render-Vorgang einmal aufgerufen, operiert auf der Pixelebene des Bildes.

primärer Sichtbereich – VR-Desktop-Konzept, beschreibt die Richtung in die der Anwender am längsten Blickt.

Q

R

RAM – Random Access Memory, Hauptspeicher des Computers.

S

Spezifikation – Sammlung von Anforderungen die an ein Software-Produkt gestellt werden.

Swapping – Beschreibt den Vorgang einen RAM Speicher auf einen Festplattenspeicher zu erweitern.
Schnittstelle – Beschreibt die Menge der Funktionen, die eine Komponente einer anderen zur Verfügung stellt.
Styleguide – Beschreibt das Aussehen des VR-Desktops, besteht aus Texturen 3D Modellen.
Sekundärer Sichtbereich – VR-Desktop-Konzept. Beschreibt eine Richtung in die der Benutzer ab und zu blickt
Skybox – Hohle Kugel auf deren Innenseite eine Textur gelegt wird. Dient zur Erschaffung eins künstlichen Horizonts.

T
Trixel- dreidimensionaler Pixel
Toter Sichtbereich – VR-Desktop-Konzept, beschreibt die Richtung in der Benutzer an wenigsten blickt

U

V
VR – Virtuelle Realität

W
Widgetbar – VR Desktop Konzept, beschreibt Fläche im 3D Raum

X

Y

Z

1...9

3D-Graphik – Besteht aus Knoten, Kanten und Fassetten, die sich beliebig in einem dreidimensionalen Koordinatensystem anordnen lassen.
3D-Raum – Im Zusammenhang dieser Arbeit beschreibt das Wort ein graphisches Konzept des VR-Desktops. Dieses bildet ein Äquivalent zu einem Hintergrundbild bei zwei dimensionalen Oberflächen.

Literaturverzeichnis

[2.1] WOOLLEY, Benjamin. *Virtual worlds: A journey in hype and hyperreality*, 1993.
[Online]https://books.google.de/books?id=a2aw_PAa2UAC&lpg=PP1&pg=PP1&hl=de#v=onepage&q=like%20river&f=false
(Zugriff am 08.06.2015)

[2.2] SUTHERLAND, Ivan E. The ultimate display. *Multimedia: From Wagner to virtual reality*,1965
[Online] http://worrydream.com/refs/Sutherland%20-%20The%20Ultimate%20Display.pdf
(Zugriff am 08.06.2015)

[2.3] Wikipedia Eintrag Ivan Sutherland
[Online] http://en.wikipedia.org/wiki/Ivan_Sutherland
(Zugriff am 08.06.2015)

[2.4] *Sensorama simulator*. U.S. Patent Nr. 3,050,870, 1962.
[Online] http://www.mortonheilig.com/SensoramaPatent.pdf
(Zugriff am 08.06.2015)

[2.5] Playstation Eye Infoseite
[Online]http://de.playstation.com/ps3/accessories/detail/item78898/PlayStation%C2%AEEye/
(Zugriff am 08.06.2015)

[2.6] Microsoft Kinect Infoseite
[Online]https://www.microsoft.com/en-us/kinectforwindows/
(Zugriff am 08.06.2015)

[2.7] NESAMALAR, E. Kiruba; GANESAN, G. *An Introduction to Virtual Reality Techniques and its Applications*.
[Online]http://www.academia.edu/7762611/AN_INTRODUCTION_TO_VIRTUAL_REALITY_TECHNIQUES_AND_ITS_APPLICATIONS
(Zugriff am 09.06.2015)

[2.8] the CAVE Virtual Reality System
[Online]https://www.evl.uic.edu/pape/CAVE/
(Zugriff am 09.06.2015)

[2.9] Mars Rover Infoseite
[Online]http://mars.nasa.gov/mer/home/
(Zugriff am 09.06.2015)

[2.10] ADAMS, Ernest. *Fundamentals of game design*. Pearson Education, 2014.
[Online]https://books.google.de/books?id=Lm1jAgAAQBAJ&pg=PA20&lpg=PA20&dq=Ernest+Adams+immersion&source=bl&ots=7eHAQrp5V7&sig=E2O0eETFsfc3KvI0cpnFcXyVN_0&hl=de&sa=X&ei=3AppVeHSMIHaUrXPgLAN&ved=0CDMQ6AEwAg#v=onepage&q=Ernest%20Adams%20immersion&f=false
(Zugriff am 09.06.2015)

[2.11] High Definition (HD) Image Formats for Television Production
[Online]https://tech.ebu.ch/docs/tech/tech3299.pdf
(Zugriff am 09.06.2015)

[2.12] ABRASH, Michael. *Down the VR rabbit hole: Fixing judder*, 2013.
[Online] http://blogs.valvesoftware.com/abrash/down-the-vr-rabbit-hole-fixing-judder/
(Zugriff am 09.06.2015)

[2.13] ABRASH, Michael. Steam Dev Days presentation, 2014.
[Online]http://media.steampowered.com/apps/abrashblog/Abrash%20Dev%20Days%202014.pdf
(Zugriff am 09.06.2015)

[2.14] Abbildung 5 – Horizontales und vertikales Sichtfeld [2.14].
[Online]http://www.incgamers.com/wpcontent/uploads/2013/05/6a0120a85dcdae970b0120a86d9495970b.png
(21.05.2015)

[2.15] Wikipedia Eintrag Realität
[Online] http://de.wikipedia.org/wiki/Realit%C3%A4t
(Zugriff am 09.06.2015)

[2.16] Wikipedia Eintrag Sinn(Wahrnehmung)
[Online]http://de.wikipedia.org/wiki/Sinn_%28Wahrnehmung%29
(Zugriff am 09.06.2015)

[2.17] KRÄMER Tanja, *Der Sechste Sinn*. 2011
[Online]https://www.dasgehirn.info/wahrnehmen/fuehlen-koerper/der-sechste-sinn
(Zugriff am 09.06.2015)

[2.18] ROTH, Gerhard. *90 Prozent sind unbewusst*. Psychologie heute, 2002, 2. Jg., S. 44ff.
[Online]http://sowiport.gesis.org/search/id/dzi-solit-0132557
(Zugriff am 09.06.2015)

[2.19]Abbildung Shepards Table
[Online]http://www.breakingnews.ie/remote/snappa.static.pressassociation.io/assets/2015/03/26191019/1427397017-7afbb1602613ec52b265d7a54ad27330-600x337.png?width=600
(Zugriff am 09.06.2015)

[2.20] Shepard's Tables – what's up?
[Online]http://www.opticalillusion.net/optical-illusions/shepards-tables-whats-up/
(Zugriff am 09.06.2015)

[2.21] Amazing T-Rex Illusion!
[Online]https://www.youtube.com/watch?v=A4QcyW-qTUg#t=25
(Zugriff am 09.06.2015)

[2.22] MAMASSIAN, Pascal; GOUTCHER, Ross. *Prior knowledge on the illumination position*. Cognition, 2001, 81. Jg., Nr. 1, S. B1-B9.
[Online]http://mamassian.free.fr/papers/mamassian_goutcher01.pdf
(Zugriff am 09.06.2015)

[2.23] RAMACHANDRAN, Vilayanur S. *Perceiving shape from shading*. Scientific American, 1988, 259. Jg., Nr. 2, S. 76-83.
[Online]http://cbc.ucsd.edu/pdf/Shape%20From%20Shading%20-%20Scientific%20American.pdf
(Zugriff am 09.06.2015)

[2.24] MCGURK, Harry; MACDONALD, John. *Hearing lips and seeing voices*. 1976.
[Online]http://wexler.free.fr/library/files/mcgurk%20%281976%29%20hearing%20lips%20and%20seeing%20voices.pdf
(Zugriff am 09.06.2015)

[2.25] Try The McGurk Effect! - Horizon: Is Seeing Believing? - BBC Two
[Online] https://www.youtube.com/watch?v=G-lN8vWm3m0
(Zugriff am 09.06.2015)

[2.26] OUNI, Slim, et al. *Visual contribution to speech perception: measuring the intelligibility of animated talking heads*. EURASIP Journal on Audio, Speech, and Music Processing, 2007, 2007. Jg., Nr. 1, S. 3-3.
[Online] http://dl.acm.org/citation.cfm?id=1287491
(Zugriff am 09.06.2015)

[2.27] ROE, Cheryl; BROWN, Timothy; WATSON, Ginger. *Factors Associated with Simulator Sickness in a High-Fidelity Simulator*. Education, 2007, 251. Jg., S. 5A.
[Online]http://www.nads-sc.uiowa.edu/bio_pub_pdf/Roe_et_al_-_Factors_associated_with_simulator_sickness.pdf
(Zugriff am 09.06.2015)

[2.28] Oculus Best Practices Guide
[Online]http://static.oculus.com/sdkdownloads/documents/Oculus_Best_Practices_Guide.pdf
(Zugriff am 09.06.2015)

[2.29] PROTHERO, Jerrold D., et al. *The use of an independent visual background to reduce simulator side-effects*. Aviation, space, and environmental medicine, 1999, 70. Jg., Nr. 3 Pt 1, S. 277-283.
[Online]http://www.ncbi.nlm.nih.gov/pubmed/10102741
(Zugriff am 09.06.2015)

[2.30] How to Reduce VR Sickness? Just Add a Virtual Nose
[Online]http://www.wired.com/2015/04/reduce-vr-sickness-just-add-virtual-nose/
(Zugriff am 09.06.2015)

[2.31] The Matrix Film Zitat
[Online]http://www.filmzitate.info/indexlink.php?link=http://www.filmzitate.info/suche/film-zitate.php?film_id=299
(Zugriff am 09.06.2015)

[2.32] SCHUEMIE, Martijn J., et al. *Research on presence in virtual reality: A survey*. CyberPsychology & Behavior, 2001, 4. Jg., Nr. 2, S. 183-201.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.3775&rep=rep1&type=pdf
(Zugriff am 09.06.2015)

[2.33] KALAWSKY, Roy S. *The validity of presence as a reliable human performance metric in immersive environments*. In: proceedings of the Presence Workshop'00. 2000.
[Online] http://astro.temple.edu/~lombard/ISPR/Proceedings/2000/Kalawsky.pdf
(Zugriff am 09.06.2015)

[2.34] PERTAUB, D. P.; SLATER, M.; BARKER, C. *An experiment on fear of public speaking in virtual reality*. Studies in health technology and informatics, 2001, S. 372-378.
[Online]http://www.researchgate.net/profile/Mel_Slater/publication/12017692_An_experiment_on_fear_of_public_speaking_in_virtual_reality/links/00b495219d7573d4fa000000.pdf
(Zugriff am 09.06.2015)

[2.35] ELLIS, Stephen R.. *Presence of mind*., 1996, 5. Jg., Nr. 2, S. 247-259.
[Online]http://telerobotics.gsfc.nasa.gov/papers/Ellis1996.pdf
(Zugriff am 09.06.2015)

[2.36] MANIA, Katerina; CHALMERS, Alan. *A user-centered methodology for investigating presence and task performance*. 2000.
[Online]http://astro.temple.edu/~lombard/ISPR/Proceedings/2000/Mania%20and%20Chalmers.pdf
(Zugriff am 09.06.2015)

[2.37] HODGES, Larry F., et al. *Presence as the defining factor in a VR application: Virtual reality graded exposure in the treatment of acrophobia*. 1994.
[Online]https://smartech.gatech.edu/bitstream/handle/1853/3584/94-06.pdf
(Zugriff am 09.06.2015)

[2.38] Media Art Net Videoplace
[Online] http://www.medienkunstnetz.de/works/videoplace/
(Zugriff am 09.06.2015)

[2.39] Time Machines: NASA goes virtual at CES
[Online] http://www.engadget.com/2013/12/15/time-machines/
(Zugriff am 09.06.2015)

[2.40] VPL EyePhone Model 1
[Online]https://vrwiki.wikispaces.com/VPL+EyePhone
(Zugriff am 09.06.2015)

[2.41] Waldo (short story)
[Online] http://en.wikipedia.org/wiki/Waldo_%28short_story%29
(Zugriff am 09.06.2015)

[2.42] Neuromancer
[Online] http://de.wikipedia.org/wiki/Neuromancer
(Zugriff am 09.06.2015)

[2.43] BRILL, L.. *Museum Virtual Reality: Part II*. Virtual Reality World, vol. 3

[2.44] DIFFERENT KINDS OF VIRTUAL REALITY
[Online]http://www.aect.org/edtech/ed1/15/15-03.html
(Zugriff am 09.06.2015)

[2.45] ABRASH, Micheal, *When it comes to resolution, it's all relative*. 2012.
[Online]http://blogs.valvesoftware.com/abrash/when-it-comes-to-resolution-its-all-relative/
(Zugriff am 09.06.2015)

[2.46] ABRASH, Michael. *Raster-Scan Displays: More Than Meets The Eye*. 2012
[Online] http://blogs.valvesoftware.com/abrash/raster-scan-displays-more-than-meets-the-eye/
(Zugriff am 09.06.2015)

[2.47] Oculus Shares 5 Key Ingredients for Presence in Virtual Reality
[Online] http://www.roadtovr.com/oculus-shares-5-key-ingredients-for-presence-in-virtual-reality/
(Zugriff am 09.06.2015)

[2.48] Wikipedia Tiefenumkehr
[Online]http://de.wikipedia.org/wiki/Tiefenumkehr
(Zugriff am 09.06.2015)

[2.49] Oculus Rift: Step Into the Game
[Online]https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game
(Zugriff am 09.06.2015)

[2.50] Das automatische Gehirn - 1v2 - Die Magie des Unbewussten
[Online]http://www.dailymotion.com/video/x22yii0_das-automatische-gehirn-1v2-die-magie-des-unbweussten-2011-by-artblood_lifestyle
(Zugriff am 09.06.2015)

[3.3] Windows 10 läuft kostenlos auf dem Raspberry Pi 2
[Online]http://www.golem.de/news/internet-der-dinge-windows-10-laeuft-kostenlos-auf-dem-raspberry-pi-2-1502-112100.html
(Zugriff am 09.06.2015)

[3.5] What would you like to see most in minix?
[Online]https://groups.google.com/forum/#!msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ
(Zugriff am 09.06.2015)

[3.9] ISO/IEC 9126-1:2001
[Online] http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749
(Zugriff am 09.06.2015)

[3.10] Wikipedia Artikel ISO/IEC 9126
[Online]http://de.wikipedia.org/wiki/ISO/IEC_9126
(Zugriff am 09.06.2015)

[3.11] Index of /android/mydroid/dalvik/docs
[Online] http://www.netmite.com/android/mydroid/dalvik/docs/
(Zugriff am 09.06.2015)

[3.12] Java SE HotSpot at a Glance
[Online]http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html
(Zugriff am 09.06.2015)

[3.13] The Kaffe Virtual Machine
[Online] http://www.kaffe.org/
(Zugriff am 09.06.2015)

[3.14] Wikipedia Artikel Moorische Gesetz
[Online]http://de.wikipedia.org/wiki/Mooresches_Gesetz
(Zugriff am 09.06.2015)

[4.1] Managing the Activity Lifecycle
[Online]http://developer.android.com/training/basics/activity-lifecycle/index.html
(Zugriff am 09.06.2015)

[4.2] The life cycle
[Online]https://github.com/libgdx/libgdx/wiki/The-life-cycle
(Zugriff am 09.06.2015)

[4.3] Interactivity and Feedback
[Online]https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/InteractivityInput.html
(Zugriff am 09.06.2015)

[4.4] Touch mechanics
[Online]http://www.google.com/design/spec/patterns/gestures.html#gestures-touch-mechanics
(Zugriff am 09.06.2015)

[5.1] Getting started with Lua
[Online] http://www.lua.org/start.html
(Zugriff am 09.06.2015)

[5.2] Produktseite Gameware
[Online] http://gameware.autodesk.com/
(Zugriff am 09.06.2015)

[5.3] LibGDX Webseite
[Online] http://libgdx.badlogicgames.com/
(Zugriff am 09.06.2015)

[5.4] Oculus Rift Webseite
[Online]https://www.oculus.com/
(Zugriff am 09.06.2015)

[5.5] Produktseite Samsung Gear VR
[Online] http://www.samsung.com/de/promotions/galaxynote4/feature/gearvr/
(Zugriff am 09.06.2015)

[5.6] Produktseite Windows Kinect
[Online]https://www.microsoft.com/en-us/kinectforwindows/
(Zugriff am 09.06.2015)

[5.7] Produktseite Leap Motion
[Online] https://www.leapmotion.com/
(Zugriff am 09.06.2015)

[5.8] Oculus Rift Hardware Specs
[Online]http://www.heise.de/preisvergleich/oculus-rift-development-kit-1-dk1-a937438.html
(Zugriff am 09.06.2015)

[5.9] Dokumentation Opengl ,Befehl glScissor
[Online]https://www.opengl.org/sdk/docs/man/html/glScissor.xhtml
(Zugriff am 09.06.2015)

[5.10] Java access to the Oculus Rift sensor device
[Online]https://github.com/jherico/jovr
(Zugriff am 09.06.2015)

[5.11] Android SDK Progress Update
[Online]https://community.leapmotion.com/t/android-sdk-progress-update/2115
(Zugriff am 09.06.2015)

[5.12] Abbildung 26 – Beispielskript in Unreal Blueprint .
[Online]https://forums.unrealengine.com/attachment.php?attachmentid=1388
(Zugriff am 09.06.2015)

[5.13] Produktseite Unreal 4 Engine
[Online]https://www.unrealengine.com/unreal-engine-4
(Zugriff am 09.06.2015)

[5.14] Unreal Engine 4 vs. Unity: Which Game Engine Is Best for You?
[Online]http://blog.digitaltutors.com/unreal-engine-4-vs-unity-game-engine-best/
(Zugriff am 09.06.2015)

[5.15] If You Love Something, Set It Free
[Online]https://www.unrealengine.com/blog/ue4-is-free
(Zugriff am 09.06.2015)

[5.16] Produktseite Unity
[Online] https://unity3d.com/unity
(Zugriff am 09.06.2015)

[5.17] Unreal Engine 4 Twitch Broadcast - Leap Motion Plugin, News, and More - Live from Epic HQ
[Online]https://youtu.be/r2fdpr_iMCc?t=33m25s
(Zugriff am 09.06.2015)

[5.18] Produktseite Durovis
[Online]https://www.durovis.com/index.html
(Zugriff am 09.06.2015)

[6.1] VR-Brille Valve Vive im Detail: Intensive Virtual Reality dank Positionstracking
[Online]http://www.heise.de/newsticker/meldung/VR-Brille-Valve-Vive-im-Detail-Intensive-Virtual-Reality-dank-Positionstracking-2567623.html
(Zugriff am 09.06.2015)

[6.2] Oculus Rift kommt Anfang 2016 in den Handel
[Online]http://www.heise.de/newsticker/meldung/Oculus-Rift-kommt-Anfang-2016-in-den-Handel-2635597.html
(Zugriff am 09.06.2015)

[6.3] Project Morpheus: Release der PS4-VR-Brille erfolgt 2016
[Online]http://www.netzwelt.de/project-morpheus/151494-release.html
(Zugriff am 09.06.2015)

[6.4] Oculus Rift System Requirements Announced
[Online]http://www.cinemablend.com/games/Oculus-Rift-System-Requirements-Announced-71938.html
(Zugriff am 09.06.2015)

[6.5] Help shape the future of holographic computing.
[Online] https://www.microsoft.com/microsoft-hololens/en-us/developers
(Zugriff am 09.06.2015)

[6.6] Nimble VR is joining Oculus
[Online] http://nimblevr.com/
(Zugriff am 09.06.2015)

Versicherung über Selbstständigkeit

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

Hamburg, den _____ _____