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

Bachelorarbeit Nr. 70

Blickbasierte Interaktion mit 3D Displays

Entwicklung eines Prototyps zur Evaluierung blickbasierter Interaktion

Jonas Auda

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Albrecht Schmidt
Betreuer/in:	Prof. Dr. Florian Alt
	M.Sc. Stefan Schneegaß
Beginn am:	2. Mai 2013
Beendet am:	1. November 2013
CR-Nummer:	H.5.2

Kurzfassung

Maus und Tastatur sind die Eingabegeräte am Computer schlechthin. Doch es gibt noch eine Reihe weiterer Methoden, wie Menschen mit Computern interagieren können. In dieser Arbeit sollen blickbasierte Interaktionsmethoden auf 3D-Bildschirmen untersucht werden. Mithilfe eines Eyetrackers und eines 3D-Bildschirms sollen von einer Beispielapplikation bereitgestellte Aufgaben gelöst und die Performanz der unterschiedlichen Lösungsansätze gemessen und diskutiert werden. Das Konzept, die Entwicklung der Applikation, die durchgeführte Studie und die dazugehörige Evaluierung werden in dieser Arbeit besprochen. Zuletzt werden Ansätze vorgestellt, welche in Zukunft die Leistung solcher Systeme möglicherweise noch erhöhen könnten.

Abstract

The mouse and the keyboard are traditionally used as input devices. However, there are more methods how people can interact with computers. In this thesis, methods for gaze based interaction with 3D displays are investigated. Utilizing an eye tracker and a stereoscopic display, different tasks should be solved and the performance of the interaction methods within a prototypical application is measured. The concept, the development of the prototyp, and the evaluation are part of this thesis. Finally, some approaches are discussed that could improve the performace of such systems in the future.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Neben standardmäßigen Eingabemöglichkeiten - wie Maus und Tastatur - gibt es eine ganze Reihe weiterer Methoden der Interaktion mit Computern, die auf verschiedene Körpermerkmale des Menschen abzielen. Beispielsweise ist man durch immer bessere technische Möglichkeiten mittlerweile in der Lage, den Blick eines Menschen genau zu erfassen und die Koordinaten des Blickfokusses auf einer zweidimensionalen Fläche - wie einem Bildschirm - können exakt bestimmt werden. Dies eröffnet neue Wege der Steuerung und Eingabe in Applikationen. Der Blick eines Nutzers kann vieles aussagen. Durch das Auswerten der Augeneigenschaften können verschiedenste Aussagen über die Intention des Nutzers gemacht werden. Das menschliche Auge bietet viele messbare Größen, die in unterschiedlichsten Kontexten als Eingabe für Programme dienen können.

Eyetracking, also die Verfolgung des Blickes und die damit einhergehende Analyse des menschlichen Auges, kann als Bedienelement von Software genutzt werden und bietet somit neue Interaktionsmöglichkeiten in den unterschiedlichsten Bereichen. Eyetracking kann am Computer, an öffentlich zugänglichen Plätzen [DBE+10], im Auto oder mobil mit speziellen Helmen und Kameras [HKB+12] eingesetzt werden. Durch diese Vielfalt an unterschiedlichen Einsatzmöglichkeiten und die steigende Zahl von Rechnern in unserer modernen Welt, stellt Eyetracking eine sehr innovative Bereicherung auf dem Gebiet der Mensch-Computer-Interaktion dar.

In dieser Arbeit liegt der Fokus auf stationärem Eyetracking und dies nicht auf gängigen 2D-Bildschirmen, sondern auf 3D-Bildschirmen. Die Technik des Eyetrackings soll auf den 3D-Fall erweitert und evaluiert werden. Verschiedene Methoden zur Auswertung des Eyetrackings sollen auf 3D-Displays angewendet und auf Performanz überprüft werden. Hierzu wurde eine Beispielapplikation entwickelt, welche eine 3D-Spielumgebung bereitstellt, in der der Nutzer verschiedene Aufgaben erfüllen muss. Es soll evaluiert werden, ob sich die Technik des Eyetrackings problemlos auf 3D-Bildschirme erweitern lässt und ob das menschliche Auge zusätzliche Attribute besitzt, die explitzit in einer 3D-Umgebung genutzt werden können, um anwendungsspezifische Aufgaben zu bewältigen. Die Leistung dieser Ansätze soll in einer Studie gemessen werden, in der die Probanden mithilfe eines Eyetrackers bestimmte Aufgaben erfüllen müssen. Teil dieser Aufgaben wird unter anderem das Anvisieren von Objekten sein, die sich hinter anderen, transparenten Objekten befinden und somit nicht durch "direktes Anschauen" fokussiert werden können.

Die Methoden werden in einer Beispielapplikation mit Weltraumkontext getestet. Es wurden zwei verschiedene Level mit unterschiedlichen Aufgaben entwickelt. In Level 1 soll der Nutzer mithilfe seines Blickes einen Planeten in der Mitte des Bildschirms vor einschlagenden Meteoriten schützen. Der Blick des Nutzers dient dem Zielen. Somit ist der Eyetracker das

maßgebliche Eingabegerät. Auch Jönsson beschäftigte sich schon mit augengesteuerten Computerspielen [Jön05] - mit dem Fokus auf dem Zielvorgang durch den Blick des Spielers. Die Meteoriten können durch den Blickfokus zerstört werden. Ein Mausklick oder ein Tastendruck ist hierzu nicht erforderlich. Zusätzlich gibt es besondere Meteoriten, die nicht auf dem Planeten einschlagen werden. Diese sollten nach Möglichkeit vorbeifliegen und nicht vom Nutzer zerstört werden. Im Falle einer Zerstörung werden zwei neue Meteoriten generiert, die dann auf den Planeten zufliegen und zusätzlich vom Benutzer abgeschossen werden sollten. Um diese nicht zu zerstörenden Meteoriten zu erkennen, sind sie grün eingefärbt und somit deutlich von den anderen zu unterscheiden.

In einem zweiten Level soll untersucht werden, ob mithilfe von Eyetracking die Möglichkeit besteht, Objekte mit einer bestimmten Tiefe im Raum anzuvisieren. Hierzu fliegen Meteoriten über die Bildfläche von links nach rechts. Hinter jedem dritten Meteorit versteckt sich ein feindliches Raumschiff, welches vom Nutzer anvisiert und dann per Mausklick (einfacher Linksklick, kein explizites Klicken auf das Objekt) abgeschossen werden soll. Meteoriten mit einem Raumschiff dahinter werden transparent dargestellt, um dem Spieler überhaupt die Möglichkeit zu geben, diese sehen zu können. Wird das Raumschiff abgeschossen, fliegt der Meteorit auf seiner Flugbahn mit unveränderter Geschwindigkeit weiter. Wird der Meteorit vor einem Raumschiff zerstört, beschleunigt das Schiff und ist somit schwerer zu treffen. Ziel dieses Levels sollte sein, so wenig Raumschiffe wie möglich passieren zu lassen. Um den Tiefenfokus des Nutzers zu erkennen, wurden zwei verschiedene Techniken implementiert. Diese Techniken nutzen verschiedene Augenmerkmale um das Objekt, welches im Fokus steht, zu bestimmen.

Die gesamte Applikation bietet die Möglichkeit zwischen 2D- und 3D-Darstellung zu wechseln, um zu untersuchen welche Eigenschaften und Techniken zum Tragen kommen, wenn der Nutzer mit diesen unterschiedlichen Darstellungsarten konfrontiert wird. Es werden die technischen Möglichkeiten, die mathematischen Hintergründe und die Herausforderungen besprochen, die mit dem Eyetracking auf 3D-Displays in Zusammenhang stehen.

Die blickbasierte Interaktion könnte in der Zukunft neuartige und hilfreiche Methoden zu Verfügung stellen, welche in unterschiedlichsten Bereichen Anwendung finden. Körperlich beeinträchtigte Menschen profitieren schon heute von diesen Technologien, indem eine erleichterte Bedienung in Anwendungen durch den Blick realisiert wird [DSI09]. Immer mehr autostereoskopische Displays kommen auf den Markt und somit stellt sich auch die Frage, wie die Interaktion mit derartigen Geräten aussehen könnte. Beispiele hierfür wären CAD-Anwendungen [PPK00]. Gerade in diesem Bereich wäre eine Tiefenerkennung im 3D-Raum vorteilhaft, um mit dem Blick die unterschiedlichen Konstruktionsteile zu erfassen, ohne dass eine zusätzliche Eingabe erforderlich ist. Durch immer günstigere Hardware könnte auch in Zukunft Eyetracking - wie die 3D-Technik in den letzten Jahren - fester Bestandteil von Systemen werden, welche heute weit verbreitet sind. Mobile Endgeräte, Spielekonsolen und der herkömmliche Computer könnten mit dieser Technik ausgestattet werden, um blickbasierte Eingaben zu ermöglichen. Das Sehen, welches uns ermöglicht die Umwelt visuell wahrzunehmen kann durch eine solche Technologie zu einem Instrument werden, um unsere Umwelt manipulieren zu können.

Die Entwicklung der Beispielapplikationen und die Evaluierung der Interaktionstechniken geschahen in Zusammenarbeit mit Rufat Rzayev. Einige Abbildungen in dieser Arbeit sind auch Gegenstand der Arbeit von Rufat Rzayev und wurden mit dem $\star$ - Symbol markiert.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 1 – Einleitung: Einleitend werden die Grundlagen erläutert, welche Motivation hinter dieser Arbeit steht und die grundsätzlichen Ideen erklärt. Die grundsätzliche Idee, der Aufgabenkontext und die Ziele der Arbeit werden definiert. Der Leser erhält eine Übersicht über die technischen Grundlagen, die Problemstellung und den Aufbau dieser Arbeit.

Kapitel 2 – Verwandte Arbeiten: Hier werden Arbeiten aus dem Themengebiet Eyetracking, sowie 3D Displays vorgestellt. Prototypen, Interaktionstechniken und deren Evaluierung sowie 3D-Technologien werden hier besprochen. Auch Arbeiten, welche Eyetracking und 3D-Technologie vereinen werden hier vorgestellt.

Kapitel 3 – Konzept: Hier wird das grundlegende Konzept der Arbeit und der Beispielapplikation erläutert. Das Auge - als zentrales Organ - beim Eyetracking und die damit einhergehenden technischen Grundlagen sind Gegenstand dieses Kapitels. Techniken zur Erstellung von 3D-Inhalten werden ebenfalls behandelt und wie diese vereint mit Eyetracking in der Beispielapplikation eingesetzt werden können.

Kapitel 4 – Implementierung: In diesem Kapitel wird die Implementierung der Beispielapplikationen besprochen. Es wird Auskunft über die verwendeten Programme, die Funktionsweise der Algorithmen und die Entwicklung der einzelnen Komponenten gegeben.

Kapitel 5 – Evaluierung: Die Ergebnisse einer Benutzerstudie werden in diesem Kapitel diskutiert. Durch das Programm erfasste Daten, sowie die von den Benutzern ausgefüllten Fragebögen werden hier analysiert, ausgewertet und diskutiert.

Kapitel 6 – Zusammenfassung und Ausblick: Gewonnene Erkenntnisse und zukünftige Ideen werden hier besprochen. Es werden Technologien und Ideen vorgeschlagen, welche in Zukunft die blickbasierte Interaktion mit 3D-Displays weiter verbessern könnten.

2 Verwandte Arbeiten

In diesem Kapitel sollen Arbeiten aus dem Bereich des Eyetrackings und der 3D-Technologie vorgestellt werden. In einigen dieser Arbeiten, werden Ansätze vorgestellt welche die beiden Techniken kombinieren.

2.1 Eyetracking

Eyetracking kann in vielen Bereichen eingesetzt werden. Zur Steuerung von Applikationen oder zur Datenerfassung in verschiedenesten Bereichen. So wurde beispielsweise Eyetracking als neue Technik herangezogen, um die Usability von Computerspielen zu untersuchen und zu evaluieren [JNR08]. Eyetracking kann im Hintergrund geschehen und lenkt den Spieler nicht ab, sodass er sich natürlich in der Spielwelt bewegen kann. Durch Eyetracking erhalten Entwickler von Spielen Informationen über das Sehverhalten der Spieler. Trotz der Kosten, welche noch mit dieser Technik verbunden sind, kann es sich lohnen Eyetracking einzusetzen, da nicht alle Evaluierungsmöglichkeiten, welche für beispielsweise Office Anwendungen genutzt werden können, in der Spieleentwicklung einsetzbar sind. Durch Eyetracking sind Entwickler in der Lage zu sehen, was die Spieler sehen. Diese Erkenntnisse können von Spieledesignern genutzt werden um beispielsweise bessere Userinterfaces zu entwickeln, welche vom Spieler besser wahrgenommen werden können.

In der Arbeit von Isokoski und Martin [IM06] wurde Eyetracking als Eingabe für einen "First Person Shooter" (FPS) verwendet. Hier wurden die Blickdaten ausschließlich für das Zielen verwendet. Geschossen wurde mit den Maustasten. Das Fortbewegen wurde nicht durch Eyetracking realisiert, sondern durch standardmäßige Tastatureingaben. Die Sicht in den Spielszenen wurde nicht durch Eyetracking realisiert, da kein Weg gefunden wurde, die Blickdaten hierfür ordnungsgemäß heranzuziehen.

Das Nutzerverhalten kann ebenfalls durch Eyetrackingsysteme analysiert werden. Ein Beispiel hierfür ist die Websuche [GJG04]. Durch Eyetracking kann herausgefunden werden, wo die Benutzer einer "Websearchengine" nach einer Suchanfrage hinsehen, wie lange sie den Fokus auf ein bestimmtes Objekt halten und welche Auswahl sie schließlich treffen. Das erste Ergebnis einer Suchanfrage war in dieser Studie immer das am längsten fokussierte und letztendlich auch das am häufigsten ausgewählte Objekt. Durch das Blickverhalten der Nutzer kann festgestellt werden, wie eine Webseite aufgebaut werden kann, um das Interesse von Nutzern zu wecken. Die Ergebnisse solcher Studien sind auch im Bereich Design und Aufbau von Lernwebseiten hilfreich, um einen möglichst großen Lernerfolg zu erzielen.

Eyetracking auf dem Gebiet der Spielentwicklung kann in zwei grundlegende Bereiche eingeteilt werden - einerseits als Eingabetechnik [HO11] und andererseits als Analyseverfahren [AVRM11]. Durch die Verwendung der Augen als Eingabeorgan können traditionelle Verfahren wie Maus, Tastatur oder Joystick ergänzt oder abgelöst werden. Besonders für Menschen mit eingeschränkten motorischen Fähigkeiten bietet Eyetracking eine Lösung für die Interaktion mit Anwendungen wie beispielsweise Computerspiele.

Beim Spielen ist man gezwungen, das Objekt anzusehen, welches man beispielsweise abschießen möchte. Dieser Zielvorgang könnte via Eyetracking realisiert werden [LM04]. Interessant ist hier das Messen der Genauigkeit und Geschwindigkeit, mit der das Zielen vonstatten geht. Hierzu wurde eine Beispielapplikation entwickelt, welche das Zielen mit der Maus und mit dem Tobii ET-1750 Eyetracker realisiert. In der Beispielapplikation flogen Bälle mit einer bestimmten Geschwindigkeit im Sichtfeld des Benutzers und konnten mit einem Linksklick abgeschossen werden. Gezielt wurde entweder mit der Maus oder dem Eyetracker. Gemessen wurden die Genauigkeit des Zielens, die Zeit die ein Benutzer für den Zielvorgang benötigte sowie die "Completion Rate", also die Rate, mit der die Bälle von einm Benutzer in einem bestimmten Zeitfenster abgeschossen werden konnten. Insgesamt schnitt hier das Zielen mit der Maus besser ab, als das mit dem Eyetracker. Die Genauigkeit des Eyetrackings spielt hier eine besondere Rolle. Nur wenn der Eyetracker sehr präzise Ergenisse liefert, funktioniert ein solches System ordnungsgemäß. Wenn der Eyetracker nicht zuverlässig für das Zielen verwendet werden kann, schwindet das Vertrauen in diese Technologie und die Nutzer greifen auf herkömmliche Technologien - wie die Maus - zurück.

Abbildung 2.1: Modifiziertes Breakout. Unten rechts ist das Brett zu erkennen, welches der Spieler kontrolliert, um mit dem Ball (rotes Objekt rechts über dem Brett) die oben befindlichen Quader zu zerstören, ohne diesen in den Abgrund fallen zu lassen.

Es ist also wichtig, dass Eyetrackingsysteme zuverlässige Daten liefern, bevor die als Eingabegeräte in Anwendungen aller Arten eingesetzt werden können. So wurde in weiteren Arbeiten ebenfalls untersucht, ob die Blickverfolgung gegen die Maus bestehen könnte [DBMB07]. Auch hier wurde ein Spiel entwickelt, welches sich mit der Maus oder dem Blick steuern lässt. Hier wurden von einem Eyetrackingsystem erfasste Daten per Netzwerkprotokoll an das Spiel gesendet und somit das Spielen durch Eyetracking realisiert. Das Eyetrackingsystem musste vor Beginn kalibriert werden, was mit einem externen Programm außerhalb des Spiels vonstatten ging. Das Spielkonzept (siehe Abbildung 2.1) orientierte sich an "Breakout" aus dem Jahre 1976. Das Prinzip des Spiels ist, einen Ball mit einem "Brett" vor dem Absturz in die Tiefe zu schützen und gleichzeitig mit ihm Objekte zu zerstören, welche sich oberhalb des "Brettes" in der Luft befinden. Dies geschieht durch Berührung (Ball mit Objekt). Um dieses Spiel mit den Augen zu kontrollieren, soll sich der Spieler grundsätzlich auf die Position konzentrieren, wo der Ball auf das "Brett" trifft. Das Ergebnis dieser Arbeit war eindeutig. Das blickbasierte Spielen schlug das Spielen mit der Maus bei der Anzahl der gewonnenen Spiele. Fast zwei Drittel der Spiele wurden von den Spielern gewonnen, welche das Spiel mit ihrem Blick steuerten.

Die Kalibrierung eines Eyetrackingsystems stellt eine besondere Schwierigkeit dar. Vor allem, wenn ein Benutzer sich oder seinen Kopf frei bewegen kann. Hierzu können Trackingsysteme wie die Microsoft Kinect [LMW+11] herangezogen werden [FMO12]. Die Kinect bietet Modelle an, welche eine 3D-Umgebung abstrahieren kann und dadurch wichtige Aussaugen bezüglich ihrer Natur und der darin befindlichen Personen gemacht werden können. Die Bestimmung der Kopfposition reduziert den Aufwand für die Blickrichtungsbestimmung drastisch und kann somit unterstützend zum Eyetracking herangezogen werden.

Eyetracking ist genauso wie die gestenbasierte Interaktion eine natürliche Form der Eingabe in Anwendungen, da in der echten Welt Augen als Sensoren und Hände als Aktuatoren genutzt werden. Um in 3D-Umgebungen zu interagieren, können diese auch in Kombination genutzt werden [SBG12]. Hier wird die Kombination aus Geste und Blickrichtung als Signal genutzt, um bestimmte Aktionen in der virtuellen Welt auszulösen. Hierzu wurde die Microsoft Kinect und ein Eyetracker von Mirametrix[1] verwendet. Als 3D-Umgebung diente das Onlinespiel "Second Life"[2]. Durch die Kombination von Kinect, Eyetracker und Second Life wurde hier ein Prototyp erstellt, der blick- und gestenbasierte Interaktion mit 3D-Inhalten ermöglicht.

Eine andere Weise Eyetracking zu verwenden, um Interaktion mit 3D-Inhalten zu ermöglichen, wurde von Tanriverdi und Jacob [TJ00] untersucht und mit traditionellen 3D-Interaktionmethoden verglichen - wie dem "Zeigen" (Handgesten) auf 3D-Objekte. Die Interaktion mit 3D-Objekten stellt einige Probleme dar. Unter anderem das fehlende haptische Feedback, beim "Berühren" solcher Objekte. Des weiteren können Objekte außerhalb der Reichweite des Betrachters liegen, sodass es nicht mehr möglich ist, diese Objekte beispielsweise mit dem Arm zu erreichen. Tanriverdi und Jacob entwickelten eine Interaktiontechnik, welche auf Augenbewegungen basierte, um diese mit herkömmlichen "Zeige"-Methoden zu

[1] http://mirametrix.com/, letzter Zugriff am 20.10.2013
[2] http://secondlife.com/, letzter Zugriff am 20.10.2013

vergleichen. Hier beobachtet der Computer den Benutzer und interpretiert dessen Aktionen, um zu erkennen, welche Befehle ausgeführt werden sollen. Hierfür sollte keine explizite Eingabe vom Benutzer selbst erforderlich sein. Um einen Vergleich zwischen dem "Zeigen" mit der Hand und den blickbasierten Ansatz zu ziehen, wurden zwei Beispielanwendungen entwickelt, welche jeweils eine Interaktionstechnik implementierten. Basis der Anwendung war ein 3D-Raum mit Objekten, welche als Texturen einen Brief enthielten. Diese konnten durch die zwei verschiedenen Interaktionsmethoden (Eyetracker/Handgeste) ausgewählt und somit vergrößert werden. Wurde ein Objekt ausgewählt änderte sich seine Farbe so, dass der Brief lesbar wurde. In einer Studie, welcher ein Trainigstag vorrausging, um sich mit der Technik (Head-mounted Display, Eyetracker und Interaktionstechniken) vertraut zu machen, wurden die Techniken evaluiert. Die Aufgabe bestand darin, zwei Objekte (Zylinder) zu finden, welche einen bestimmten Brief enthielten. Es wurde die Zeit erfasst, die die Probanden benötigten um die Aufgabe zu erfüllen. Anschließend wurden die Probanden gebeten Fragebögen auszufüllen und somit ein Feedback über die verwendeten Methoden zu geben. Die Messung der Zeit, die die Probanden benötigten, um die Aufgabe zu erfüllen ergab, dass die Probanden unter der Verwendung der auf Eyetracking basierten Lösung signifikant schneller waren, als bei der Handgestenmethode. Des weiteren konnten mit distanzierte Objekte per Eyetracking besser interagiert werden, als mit Handgesten.

Ein weiterer Ansatz, blickbasierte Interaktion auf Displays zu ermöglichen, ist das Einbeziehen eines in der Hand haltbaren Geräts. Somit kann beispielsweise der Blick für das Fixieren eines Inhaltes genutzt werden und das Gerät schlussendlich für das Selektieren. Stellmach und Dachselt [SD12] haben diese Kombinationsmöglichkeit untersucht. Eine solche Interaktionsmethode ist insbesondere für große Displays, wie Powerwalls oder große Fernsehsysteme geeignet. Ein "Auswählen" ist in Applikationen ein zentraler Bestandteil, welcher nahezu überall zu finden ist. In dieser Arbeit wurden unterschiedliche Methoden vorgestellt, welche nach dem Prinzip arbeiten, dass der Blick auswählt und das mobile Handgerät bestätigt. In anderen Worten: man schaut etwas an und bestätigt dies per Knopfdruck.

2.2 3D-Technologie

Dreidimensionale Darstellungstechniken sind nicht neu, aber die Verfügbarkeit hat sich in den letzten Jahren drastisch geändert. Mittlerweile sind viele Unterhaltungssysteme - insbesondere Bildschirme - 3D-fähig und somit haben die dafür gebräuchlichen Technologien auch den Weg in die Privathaushalte gefunden und sind nicht mehr nur in Kinos oder anderen öffentlich zugänglichen Einrichtungen gegenwärtig [GHM99].

Es stellt sich nun die Frage, welche Vorteile - neben den visuellen - haben dreidimensionale Darstellungsarten bezüglich der herkömmlichen 2D-Darstellung und welche Probleme oder Herausforderungen gibt es im Bereich der dreidimensionalen Darstellung?

Um dreidimensionale Bilder zu generieren, gibt es unterschiedliche Verfahren, welche jeweils Vor- und Nachteile haben. Volbracht et al. [VDSF97] haben verschiedene 3D-Modi im Bezug auf ihre Effektivität untersucht. Als Kontext der hier entwickelten Beispielapplikation wurde Chemie gewählt und für die Visualisierung ein Molekülmodell herangezogen. Als

Darstellungstechniken dienten das Anaglyphenverfahren, perspektivisches Sehen und ein Shutterbrillen-System. Die Aufgaben bestanden aus Indentifikation, Vergleiche und Positionierung von Molekülen. Insgesamt wurden fünf Aufgaben entwickelt. Aufgaben 1 und 2 bestanden darin, sowohl simple als auch komplexe Moleküle zu identifizieren. Die dritte Aufgabe verlangte den Vergleich zweier simplen Moleküle. In Aufgabe 4 sollten komplexe Moleküle verglichen werden und in der fünften und letzten Aufgabe war das Ziel, einen Benzol Ring[3] parallel zur Bildfläche zu positionieren. Gemessen wurde die Zeit und die Genauigkeit, mit der die Aufgaben bewältigt wurden. Die Aufgabe wurde interaktiv für jeden der 3D-Modi durchgeführt. Die Eingabe beschränkte sich auf das Rotieren der Moleküle per Maus. Ausgewertet wurde die Zeit für die Antwort, sowie deren Korrektheit. Aufgabe 2 (komplexe Moleküle) konnte gegenüber Aufgabe 1 (simple Moleküle) besser unter der Verwendung des Anaglyphverfahrens und Shutterbrillenverfahren bewältigt werden, als beim perspektivischen Sehen. Die Genauigkeit war in diesen Fällen höher und die Probanden waren schneller. Auch für die dritte und vierte Aufgabe, für die räumliche Wahrnehmung relevant war, war das perspektivische Sehen nicht geeignet. Lediglich bei Aufgabe 4 unterschieden sich Anaglyph- und Shutterbrillenverfahren signifikant. Die Auswertung der Daten für die letzte Aufgabe ergab, dass Positionierungsfehler kleiner waren, wenn das Shutterbrillen- oder Anaglyphenverfahren eingesetzt wurde. Zeittechnisch war das Anaglyphenverfahren hierbei signifikant besser geeignet. Zwischen Shutterbrillenverfahren und perspektivischem Sehen gab es keine nennenswerte Unterschiede bezüglich der Zeit. Im direkten Vergleich aller hier evaluierten Darstellungsarten schnitt das Anaglyphenverfahren unter den Gesichtspunkten "Kosten vs. Performanz" am besten ab. Dieses Ergebnis muss aber unter dem Aspekt von Farbeinbußen bei diesem Verfahren betrachtet werden.

Dreidimensionale Inhalte sind nicht an stationäre Ausgabegeräte gebunden. Im Bereich mobiler Geräte ist ein Leistungsstandard erreicht worden, welcher für komplexe grafische Anwendung mehr als ausreichend ist. Zudem sind mittlerweile stereoskopische Bildschirme in Smartphones oder anderen mobilen Geräten, wie Spielekonsolen, integriert. Auch die Sensortechnologie in diesem Bereich eröffnet neue Interaktionsmöglichkeiten, welche weit über die Möglichkeiten von Maus, Tastatur oder Joysticks gehen [DLK12]. Daiber et al. verwendeten diese Sensortechnologien, um ein Spiel auf mobilen 3D-fähigen Endgeräten zu entwickeln. Die zentralen Aufgaben in diesem Spiel sind das Navigieren, um Bewegung zu realisieren und das Manipulieren, um mit Objekten zu interagieren. Bewegungen (Rotieren oder Kippen des Gerätes) in der echten Welt ließen die Kamera rotieren und somit konnte das Gerät als greifbare Kamera angesehen werden. Dadurch war es möglich die Kamera zu bewegen und somit die Perspektive und den Fokus zu ändern. Der zweite Teil - das Manipulieren von Objekten - geschah durch Berührung des Touchdisplays. Das Kippen und Rotieren des Gerätes trug zusätzlich zur Interaktion durch Berührung bei. Objekte konnten durch einfaches "Antippen" ausgewählt werden. Durch die Navigationstechniken konnte hier ein einfaches und intuitiv verständliches "Auswählen" von Objekten realisiert werden, auch wenn diese Objekte von anderen Objekten verdeckt waren. Hierfür war ein einfaches Rotieren ausreichend, um das dahinter befindliche Objekt auszuwählen. Objekte konnten durch die

[3] http://de.wikipedia.org/wiki/Benzol, letzter Zugriff am 20.10.2013

Rotation des Gerätes gedreht werden, wenn sie durch dauerhafte Berührung ausgewählt wurden. Translationen von Objekten konnten per "Drag" durchgeführt werden.

Mit diesen Interaktionstechniken wurde ein Spiel entwickelt, welches an "Flight Control"[4] angelehnt ist. Ziel des Spiels ist das Landen von heranfliegenden Flugzeugen ohne Kollisionen zu verursachen. Die Flugzeuge und Helikopter kommen periodisch angeflogen und müssen ordnungsgemäß und ohne Unfall gelandet werden, indem sie ausgewählt werden und eine Route zu einem Flughafen gezeichnet wird. Man verliert das Spiel, wenn Flugobjekte kollidieren, wodurch dem Spieler eine präzise Planung der Flugrouten abverlangt wird. Das Spiel, welches im Original zweidimensional gestaltet ist, wurde auf 3D erweitert. Durch drei verschiedene Ebenen - Boden, mittlere Luftebene und hohe Luftebene - wurde eine 3D-Welt erschaffen. Um Kollisionen zu vermeiden, konnte der Spieler die Flugzeuge auf verschiedenen Ebenen fliegen lassen. Die Bodenebene diente der Landung der Flugzeuge. Flogen die Flugobjekte auf unterschiedlichen Ebenen, war eine Kollision ausgeschlossen. Folgerichtig konnte der Spieler die Flugobjekte auf den verschiedenen Ebenen verteilen, um die Aufgabe zu bewältigen. Durch Kippen des Geräts konnte kontinuierlich zwischen Drauf- und Seitenansicht gewechselt werden. Abhängig von der Ansicht kann der Benutzer dann eine Route für das Flugzeug definieren. Um die Flugzeuge ordnungsgemäß zu landen musste der Benutzer eine solche Route definieren und einen Sinkflug einleiten. Um eine Route zu definieren konnte der Benutzer ein Flugobjekt auswählen und mit dem Finger über den Bildschirm fahren, um die Richtung anzugeben. Der Prototyp dieser Anwendung wurde für das mobile Betriebssystem Android[5] und OpenGl ES 2.0[6] für embedded Systems entwickelt. Diese Anwendung bestand aus zwei Grundkomponenten. Ein 3D-Renderer und ein Sensor-Controller. Die erste Komponente diente zur Erstellung von 3D-Inhalten unter der Verwendung des Anaglyphenverfahrens. Der Sensor-Controller übernahm die Auswertung von Beschleunigungs-, Touch- und Orientierungssensor, um somit eine Eingabe anhand der so ermittelten Daten zu realisieren. Der Prototyp wurde von Benutzern getestet, welche ein positives Feedback gaben. Sowohl das Spiel, als auch die Interaktionsmethoden wurden positiv bewertet. Aufgrund dieser ersten Rückmeldungen sollte in Zukunft der Prototyp Gegenstand einer Studie sein, um die Interaktionskonzepte zu evaluieren. Hier soll auch die 3D-Wahrnehmung und das Navigieren in dreidimensionalen Umgebungen tiefgehender analysiert werden. Auch die Verwendung unterschiedlicher 3D-Technologien soll Teil der zukünftigen Forschung sein.

3D-Inhalte müssen nicht zwangsläufig von einem 3D-fähigen Display generiert werden. Eine besondere Displayart sind die "Volumetric Displays". Diese Displays können Inhalte dreidimensional darstellen und beispielsweise eine Ansicht von jeder Seite ermöglichen. Das heißt, der Betrachter kann auf jeder Seite des Geräts stehen und den Inhalt betrachten. Grossman et al. [GWB04] stellten ein System vor, welches die Manipulation von 3D-Objekten - dargestellt auf einem solchen Display - ermöglichte. Durch eine Bewegungserkennung der Finger soll der Benutzer dieses Systems mit den Objekten interagieren. Dazu wurden grund-

[4] http://de.wikipedia.org/wiki/Flight_Control, letzter Zugriff am 20.10.2013
[5] http://www.android.com/, letzter Zugriff am 20.10.2013
[6] http://www.khronos.org/opengles/, letzter Zugriff am 20.10.2013

legend notwendige Operationen definiert, welche für eine Manipulation von 3D-Objekten notwendig sind. Es wurden Handgesten entwickelt, die zur Navigation im System dienten. Oft genutzte Befehle, wie Löschen oder Auswählen, wurden mithilfe von Buttons realisiert, welche durch ein Antippen der Oberfläche des Displays gedrückt werden konnten. Somit wurde ein Prototyp für gestenbasierte Interaktion mit 3D-Volumen-Displays geschaffen, welcher als Grundlage für weitere Projekte in diesem Bereich dienen könnte.

2.3 Eyetracking auf 3D-Displays

Die Eyetrackingtechnologie zusammen mit der 3D-Darstellung eröffnet weitere Forschungsbereiche und Möglichkeiten der Interaktion.

Kwon et al. haben gezeigt, dass es möglich ist, auf stereoskopischen Displays mit 3D-Inhalten zu interagieren [KJK+06]. Hier wurde ein Eyetracker verwendet und der Inhalt auf einem stereoskopischen Display ausgegeben. Um die Blicktiefe zu messen, wurden beide Augen analysiert. Das zu Grunde liegende Konzept war hier die Messung der Distanz zwischen den beiden Pupillenmitten (engl. Pupil Center Distance - PCD). Das Konzept der Tiefenerkennung ist in Abbildung 2.2 verdeutlicht.

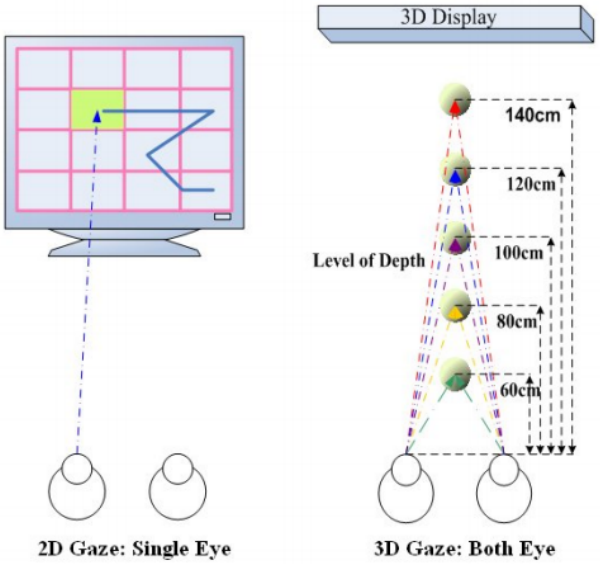


Abbildung 2.2: Die unterschiedlichen Konzepte der Blickerfassung für den 2D- und 3D-Fall. Rechts im Bild das Prinzip hinter der Tiefenerkennung. Hier werden Daten beider Augen verwendet, um die Tiefe des Blickfokusses zu ermitteln. (Quelle: [KJK+06])

Um eine 3D-Blickerfassung zu ermöglichen, ist neben der Blickrichtung auch die Blicktiefe erforderlich. Bisher lagen die meisten Forschungsarbeiten auf dem Gebiet des Eyetrackings

auf 2D-Displays. Hier ist es ausreichend nur die Daten eines Auges zu analysieren, um die Blickrichtung und somit die Position auf dem Bildschirm festzustellen. In dieser Arbeit wurde ein Algorithmus vorgestellt, welcher Blickerfassung und Tiefenerkennung auf einem Parallaxenbarriere-Display [Dod05] realisiert. Um die Tiefe zu ermitteln wurde hier die PCD verwendet - also die Distanz zwischen linker und rechter Pupillenmitte. Nun konnte festgestellt werden, wohin und mit welcher Tiefe ein Benutzer ein Objekt betrachtete. In einer Evaluierung konnte die Methode mit Erfolg, unter gewissen Rahmenbedingungen, ausgewertet werden. Ein anderes Konzept, um eine Tiefenerkennung zu ermöglichen, ist das Einbeziehen des Pupillendurchmessers. Dieser variiert bezüglich der Blicktiefe [RHFL10]. Dies kann die Grundlage für eine Evaluierung der Korrelation zwischen Blicktiefe und Pupillendruchmesser bilden.

Das Messen des Blicks auf stereoskopischen Displays war auch Gegenstand der Arbeit um Duchowski [DPHW11]. Auch hier wurde mit einem stereoskopischen Display gearbeitet, um eine Tiefenerkennung zu realisieren. Es wurde außerdem in einer Studie gezeigt, dass die Vergenz (gegensinnige Augenbewegung) als Maß für die Blicktiefe genutzt werden kann. Die Studie sollte den Zusammenhang zwischen Blicktiefe und 3D-Repräsentationstiefe mittels der Vergenz evaluieren. Ein Eyetracker wertete zu diesem Zweck beide Augen des Benutzers aus, wenn dieser auf dem 3D-Display bestimmte Stimuli zu sehen bekam. Die primäre Aufgabe in der Studie bestand darin, dass ein Proband einen Würfel visuell fixieren sollte, wenn dieser wackelte. Doch zunächst musste das System kalibriert werden. Dazu wurde zuerst eine zweidimensionale Kalibrierung durchgeführt, indem der Proband einem Punkt auf dem Bildschirm folgte, der zwischen neun verschiedenen Stellen hin und her wechselte. Nach diesen neun Zielen wurde der Kalibrierungsvorgang vom Studienleiter beendet. Nun wurde dem Probanden eine von vier Variationen einer "Cube-Grid" (einem Gitter aus Würfeln, die auf dem Display erschienen) gezeigt. War ein stereoskopischer Effekt für den Probanden sichtbar, teilte er dies dem Studienleiter mit. Wie gut der Tiefeneindruck war, sollte der Proband auf einer sieben Punkte Likert Skala angeben. Vor der eigentlichen Studie wurde die Korrelation zwischen der Vergenz und dem Sehziel untersucht. Zusätzlich wurden aus den hier gewonnen Informationen zusätzliche Verbesserungen für die eigentliche Studie vorgenommen. Insgesamt nahmen 20 Probanden an der Studie teil. Es konnte gezeigt werden, dass die Vergenz genutzt werden kann, um eine Tiefenerkennung zu realisieren. Dies konnte durch den Eyetracker und die Stimuli auf dem stereoskopischen Display nachgewiesen werden.

3 Konzept

Die Idee der blickbasierten Interaktion mit 3D-Displays soll durch eine Beispielapplikation evaluiert werden. Diese Applikation soll mithilfe eines Eyetrackers gesteuert und die Spielwelt auf einem 3D-fähigen Full-HD-Fernseher ausgegeben werden. Die Eingabe soll im Grunde nur aus den Augendaten bestehen und alles, was der Nutzer sieht, soll auch in 3D sichtbar sein. Das menschliche Auge dient als Eingabeorgan und sollte deshalb auf seine Merkmale, welche für die Eyetrackingtechnik relevant sind, untersucht werden [Daho6]. Um eine 3D-Spielwelt zu erzeugen, muss eine 3D-Technik gewählt und diese darstellungstechnisch in der Beispielapplikation umgesetzt werden. In diesem Kapitel liegt der Fokus auf den unterschiedlichen Bereichen, welche zusammenspielen müssen, um eine solche Beispielapplikation zu ermöglichen.

3.1 Eyetracking

Eyetrackingsysteme nutzen spezielle Augenmerkmale, um die Blickrichtung zu berechnen. Um die Funktionsweise zu verstehen, ist es nötig die Anatomie (siehe Abb. 3.1 und 3.2) des menschlichen Auges etwas genauer zu betrachten. In Abbildung 3.2 ist die "Hornhautreflexion" dargestellt. Diese Reflexion stellt einen wichtigen Bezugspunkt für den Eyetracker dar [OMY02].

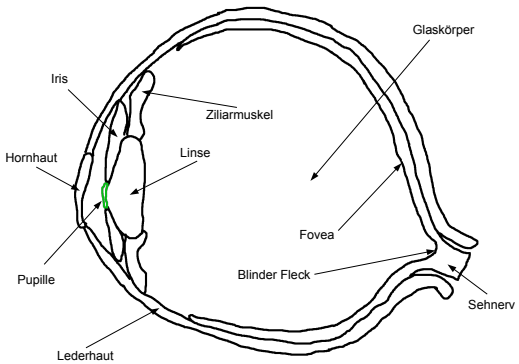


Abbildung 3.1: Das menschliche Auge und seine Bestandteile. Grün dargestellt die Pupille, welche eine tragende Rolle bei der Blickerkennung spielt.

Durch das Bewegen des Auges verschiebt sich die Position der Pupille und der "Hornhautreflexion" (engl. corneal reflection) [PB06]. Über die Positionen der Pupille und der punktförmigen Reflexion kann man die Blickrichtung ausrechnen und somit relativ zum Eyetracker bestimmen, wo der Benutzer hinschaut. Dabei kommt infrarotes Licht zum Einsatz. Kann bestimmt werden, wohin der Blick eines Benutzers fällt, kann dies direkt als Eingabe für eine Applikation genutzt werden. Auf Grundlage verschiedener Werte, wie beispielsweise der Blickposition oder des Pupillendurchmessers, soll die Anwendung gesteuert werden.

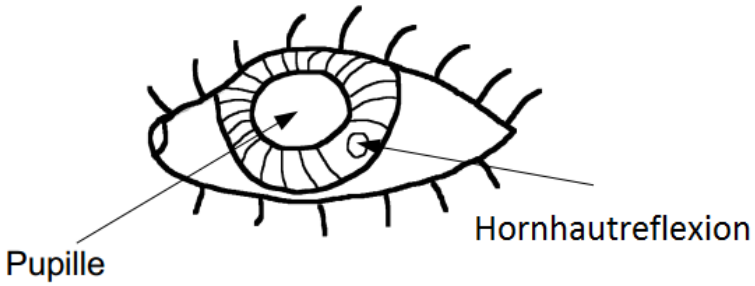


Abbildung 3.2: Frontale Sicht auf das Auge. Die punktförmige Reflexion auf der Hornhaut kann vom Eyetrackingsystem zur Blickbestimmung genutzt werden.

3.2 3D-Technik

Um überhaupt dreidimensional sehen zu können, benötigt man zwei Augen. Jedes Auge sieht von seiner Position aus ein anderes Bild. Im Gehirn werden diese beiden Bilder zusammengeführt, erst dadurch kann man seine Umgebung dreidimensional wahrnehmen. Um diesen Effekt künstlich zu erzeugen, müssen also beide Augen mit unterschiedlichen Bildern versorgt werden. Dies kann durch verschiedene Techniken geschehen, von denen einige im Folgenden kurz erläutert werden.

3.2.1 Polarisationsbrillen

Um mit Polarisationsbrillen eine stereoskopische Darstellung zu erzielen, wird mit polarisiertem Licht gearbeitet [RW]. Die Bilder für das linke und rechte Auge werden jeweils in entgegengesetzt polarisiertem Licht ausgestrahlt. In der Brille befinden sich Polarisationsfilter, welche das Licht, welches für das jeweilige Auge bestimmt ist, passieren lassen und das für das andere herausfiltern. Diese Technik benötigt keine Kommunikation zwischen Brille und Anzeigegerät. Für die Beispielapplikation, die in dieser Arbeit evaluiert wird, wurde diese 3D-Technik ausgewählt. Es gibt aber noch weitere Techniken zur Erstellung von dreidimensionalen Bildern, beispielsweise Shutterbrillensysteme.

3.2.2 Shutterbrillensysteme

Shutterbrillen [Wit10] [RW] nutzen Flüssigkristalle, um die Brillengläser abwechselnd verdunkeln zu können, um den 3D-Effekt zu erzeugen. Dadurch kann einem Auge die Sicht auf den Bildschirm verwehrt und dem anderen gestattet werden. Auf dem Bildschirm wird passend dazu, welches Auge gerade die Sicht auf den Bildschirm erhält, ein Bild speziell für dieses Auge dargestellt. Durch synchrones Umschalten des Bildes auf dem Display und des dazugehörigen Auges, kann stereoskopisches Sehen ermöglicht werden. Es ist also eine Kommunikation zwischen Brille und Anzeigegerät notwendig, um den 3D-Effekt zu erzeugen.

3.3 Konzept des zu implementierenden Systems

Um die gewünschte Beispielanwendung zu entwickeln, sind die bereits genannten Techniken erforderlich. Man benötigt also ein Eyetrackingsystem und einen 3D-fähigen Bildschirm, um eine solche Anwendung zu entwickeln. Damit die Augendaten als Eingabe genutzt werden können, muss überlegt werden, welche Berechnungen in der Beispielapplikation gemacht werden müssen, um die gewünschten Funktionalitäten zu erhalten. Was kann man anhand der Blickposition in einer 3D-Welt bestimmen? Verhält sich das menschliche Auge in der künstlich erzeugten Welt ähnlich wie beim Betrachten der natürlichen, echten Welt? Was sagen Pupillendurchmesser und andere Augenattribute aus?

Die vom Eyetrackingsystem gelieferten Daten müssen in eine 3D-Welt transformiert werden, um die in ihr enthaltenen Objekte zu manipulieren. Die 3D-Welt (oder besser die Anwendung, welche diese generiert) muss sich problemlos auf einem 3D-Display darstellen lassen - und das alles möglichst in Echtzeit.

Um eine 3D-Anwendung zu erstellen, bietet es sich an, eine 3D-Engine zu benutzen. Es gibt es eine ganze Reihe an nutzbaren Engines, welche viele verschiedene Programmiersprachen unterstützen. Einige Beispiele sind JAVA 3D, Ogre oder Unity 3D. Die in dieser Arbeit entwickelte Beispielapplikation wurde mit der Unity 3D Engine erstellt. Diese Engine wurde gewählt, da sie ein sehr mächtiges und einfach zu benutzendes Werkzeug ist, welches in verschiedensten Anwendungsbereichen genutzt werden kann [MCP+12]. Mit einem 3D-Modellierungsprogramm (beispielsweise Blender) lassen sich komplexe Geometrien erstellen und direkt in ein Format, welches von Unity 3D akzeptiert wird, exportieren. Zudem kann in Unity 3D JavaScript, C# oder Boo verwendet werden, was dem Benutzer relativ große Freiheit bei der Wahl der Programmiersprache einräumt. Unity 3D hat eine hoch optimierte Grafikpipeline für DirectX und OpenGL [WMZ+10]. Somit eignet diese Engine sich hervorragend für die Entwicklung von Echtzeitgrafikanwendungen.

Um die 3D-Welt ordnungsgemäß auszugeben, benötigt man einen 3D-fähigen Bildschirm. Da in dieser Arbeit auch Unterschiede zwischen 2D- und 3D-Darstellung untersucht werden sollen, sollte der Bildschirm ein Wechseln zwischen zweidimensionaler und dreidimensionaler Anzeige ermöglichen.

Damit sind die Voraussetzungen für eine 3D-Anwendung geschaffen. Um nun noch die Augen als Eingabeorgan einzubinden, sollte das Eyetrackingsystem gewisse Voraussetzungen erfüllen. Die Frequenz, also die Rate, mit der die Daten der Augen ermittelt werden, darf nicht zu niedrig sein, um ein Spielen in Echtzeit zu ermöglichen. Das in dieser Arbeit verwendete Eyetrackingsystem ist ausreichend schnell und generiert automatisch sehr gut verwendbare Werte, welche für das Spielen in einer 3D-Welt geeignet sind. Um die Augendaten in die Beispielapplikation zu übertragen, bietet sich eine Übertragung über das Netzwerk an. Das Eyetrackingsystem ist so konzipiert, Augendaten per Netzwerkprotokoll zu verschicken.

Durch die Kombination aller drei Komponenten kann eine Applikation entwickelt werden, mit welcher blickbasierte Interaktion mit 3D-Displays möglich sein sollte. Anhand solch einer Applikation, deren Entwicklung in Kapitel 4 beschrieben ist, soll auch eine Evaluierung der implementierten Techniken möglich sein, aber auch die Ansprüche, die an den Benutzer gestellt werden, sollen untersucht werden.

4 Implementierung

Das Gesamtsystem ist aus drei Komponenten aufgebaut. Aus Gründen der Performanz und da bestimmte genutzte Software an spezielle Betriebssysteme gebunden ist, wurde diese Aufteilung gewählt. Für die Funktionsweise der Interaktionsmethoden kann auch eine abweichende Gesamtarchitektur gewählt werden. Die Kommunikation zwischen den unterschiedlichen Teilen des Systems geschah über das lokale Netzwerk via UDP. Alle Applikationen, die im Rahmen dieser Arbeit entwickelt wurden, entstanden in Zusammenarbeit mit Rufat Rzayev.

1. Eyetracker mit dazugehörigem Rechner + Trackingsoftware (Windows XP)
2. Serversoftware für die Kommunikation zwischen Eyetracker und Applikationsrechner (Plattformunabhängige JAVA-Implementierung)
3. Beispielapplikation mit Ausgabe für einen 3D-Bildschirm (Unity 3D Engine Pro Version, Windows 7)

4.1 Systemarchitektur

Wie zuvor erwähnt, besteht das Gesamtsystem aus drei Teilen. Die Beispielapplikation und der Kommunikationsserver wurden im Rahmen dieser Arbeit entwickelt. Da das Eyetrackingsystem erfasste Augendaten über das Netzwerk verbreitet, erschien die Entwicklung eines Servers zwischen der eigentlichen Applikation und dem System sinnvoll. Über diesen Server lässt sich das Eyetrackingsystem steuern. Zudem ist es durch einen solchen Server möglich, eine API[1] für das Steuern des Eyetrackingsystems zu implementieren. Der Server kann folglich für jede mögliche Applikation genutzt werden. Hierzu muss nur ein Protokoll zwischen Server und Applikation realisiert werden, welches dann auf dem Server die vorhandenen Methoden zur Steuerung und Konfiguration des Eyetrackers nutzen kann.

Die eigentliche Beispielapplikation, welche auf einem eigenen System läuft, dem Applikationsrechner (siehe Abbildung 4.1, rechts), ist mithilfe der 3D-Engine "Unity 3D" entwickelt worden. Wie Unity 3D funktioniert, wird weiter hinten beschrieben.

[1] Application Programming Interface

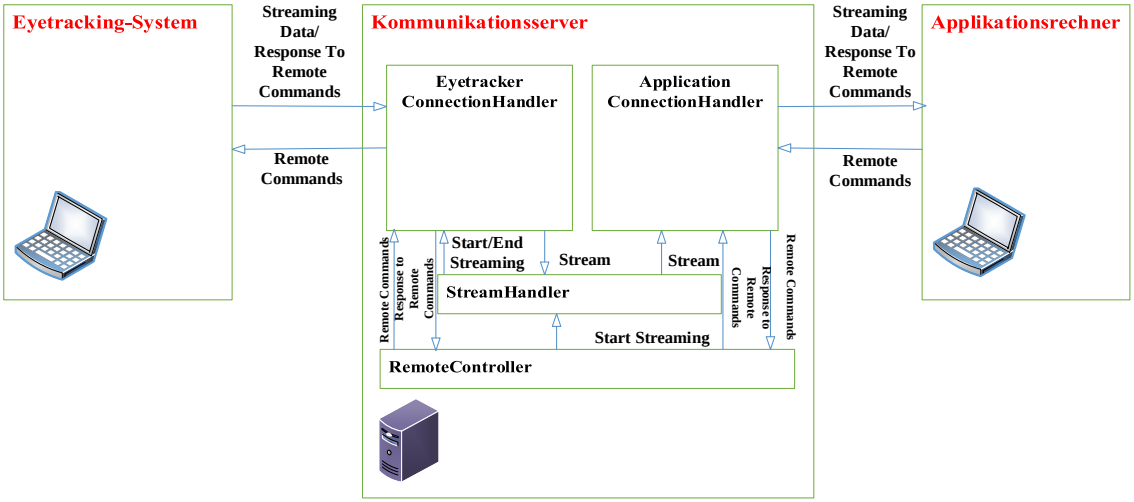


Abbildung 4.1: Systemarchitektur. Links: das Eyetrackingsystem. Mitte: der Kommunikationsserver zur Datenvermittlung. Rechts: der Rechner mit Beispielapplikation und Anbindung an den 3D-Bildschirm.⋆

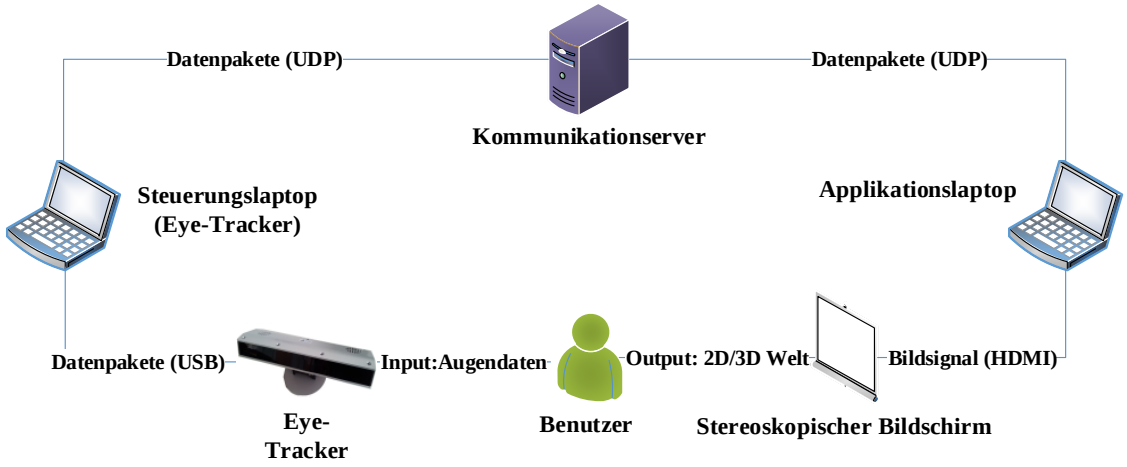


Abbildung 4.2: Schematische Darstellung der verschiedenen Abschnitte des Gesamtsystems und des Benutzers.⋆

4.2 Unity 3D

Um eine realistische und optisch ansprechende 3D-Umgebung zu erstellen, wurde die Engine Unity 3D[2] verwendet. Diese bietet eine breite Palette an vorgefertigten Objekten und Funktionen, die die Erstellung von 3D-Umgebungen stark vereinfachen. Solche Objekte können verschiedene Geometrien (wie Kugeln, Würfel oder komplexere Modelle) sowie leere Objekte sein, welche zwar keine Geometrie haben, jedoch Funktionalitäten realisieren können. In Unity 3D lassen sich Objekte per Drag & Drop in der 3D-Welt platzieren, rotieren und skalieren. Texturen lassen sich ebenfalls direkt auf die Objekte ziehen und passen sich direkt der gewählten Geometrie an. Um die Objekte zu animieren, unterstützt Unity 3D die Skriptsprachen C#, Javascript und Boo. Die Beispielapplikation dieser Arbeit wurde ausschließlich mit C# geschrieben. Jedem Objekt können ein oder mehrere Skripte zugewiesen werden. Diese Skripte bestehen grundsätzlich aus zwei Methoden: "Start" und "Update". Eine Applikation besteht aus einer oder mehreren Szenen. Beim Start der Applikation wird eine Szene initialisiert. Dies hat zur Folge, dass die Startmethoden aller Objekte dieser Szene ausgeführt werden. In dieser Methode hat man also die Möglichkeit, Variablen zu initialisieren und Parameter zu setzten. Anschließend wird in jedem Frame die Methode "Update" aller in der Szene befindlicher Objekte ausgeführt. In dieser Methode wird also das Verhalten der Objekte in einer Szene beschrieben.

Ein Beipsiel: Um ein Objekt um seine eigene Achse rotieren zu lassen, kann man den in Listing 4.1 dargestellten Befehl aufrufen.

Listing 4.1 Updatemethode in Unity 3D. Dieser Programmauszug lässt ein Objekt um eine bestimmte Achse rotieren. Um welche Achse rotiert werden soll, wird mit dem ersten Parameter bestimmt. Die Eins als zweiter Operand des "Vectors3" spezifiziert die y-Achse als Rotationsachse, der zweite Parameter den Winkel pro Frame.

```
void Update() {
        this.transform.Rotate(new Vector3(0,1,0), 0.25f);
       }
```

Hierbei bezieht sich das "this" auf das Objekt - in Unity "GameObject" genannt - in der 3D-Umgebung. Über "transform" lässt sich die gesamte Bewegung von Objekten im Raum beschreiben.

Um Interaktion zwischen Objekten zu ermöglichen, können Objekten physikalische Größen (beispielsweise die Masse) und Eigenschaften zugewiesen werden. Dies geschieht über "Rigidbodys". Rigidbodys ermöglichen zudem die Wirkung von Kräften auf Objekte und können diese beispielsweise beschleunigen oder abbremsen. Um Kollisionen mit anderen Objekten zu erkennen, können den Objekten "Collider" hinzugefügt werden. Durch diese werden Kollisionen automatisch berechnet und bestimmte Methoden ausgeführt, wenn es zu einer Berührung kommt. Die Methoden, welche bei einem solchen Ereignis aufgerufen

[2] www.unity3D.com, letzter Zugriff am 20.10.2013

werden, können zum Beispiel die Zerstörung eines Objekts realisieren. Dieses Konzept wird auch genutzt, um Objekte zu erfassen, auf die gezielt wird. Somit wird dem Spieler ermöglicht, mit Objekten in einer Szene zu interagieren.

4.2.1 Szenen

Die Beispielapplikation dieser Arbeit besteht aus sieben Szenen. In Unity 3D werden auch die Menüs in Szenen modelliert. Somit besteht die Applikation aus einem Hauptmenü, zwei Kalibrierungszenen, zwei Spielszenen und zwei Szenen, in denen nach dem Spielen der erreichte Punktestand, die Spielzeit und andere Informationen über den Spielverlauf dargestellt werden.

Hauptmenü

Im Hauptmenü können Spielername und die IP-Adresse zu dem Kommunikationsserver eingegeben werden, welcher mit dem Eyetrackingsystem kommuniziert. Außerdem bietet es die Möglichkeit, aus verschiedenen Berechnungsarten für das Zielen mit den Augen zu wählen. Von dort aus kann der Benutzer auch das System kalibrieren, um ein möglichst genaues Eyetracking zu ermöglichen. Schließlich kann der Benutzer noch das Level wählen, welches er spielen möchte.

Level

Die Beispielapplikation bietet zwei verschiedene spielbare Szenen (Level 1 und Level 2). In Level 1 ist die Aufgabe des Spielers einen Planeten (in diesem Fall die Erde) vor heranfliegenden Meteoriten zu schützen (siehe Abbildung 4.3). Um die Erde kreist der Mond, welcher ebenfalls mit heranfliegenden Objekten interagieren kann. Das Heranfliegen der Meteoriten ist in Wellen organisiert. Pro Welle, die generiert wird, steigt die Anzahl der Meteoriten um eins. Es gibt zwei verschiedene Arten von Meteoriten (siehe Abbildung 4.5). Eine Art fliegt auf Flugbahnen, welche die Position der Erde schneiden und somit auf ihr einschlagen würden, wenn der Benutzer sie nicht abschießt. Die andere Art fliegt in einem gewissen Abstand vorbei. Um die verschiedenartigen Meteoriten zu unterscheiden, sind die vorbeifliegenden grün eingefärbt. Ziel des Levels ist es, einschlagende Meteoriten abzuschießen und vorbeifliegende passieren zu lassen. Wird ein grüner Meteorit abgeschossen, werden an der Abschussposition zwei neue Meteoriten generiert, welche wiederum auf die Erde zufliegen.

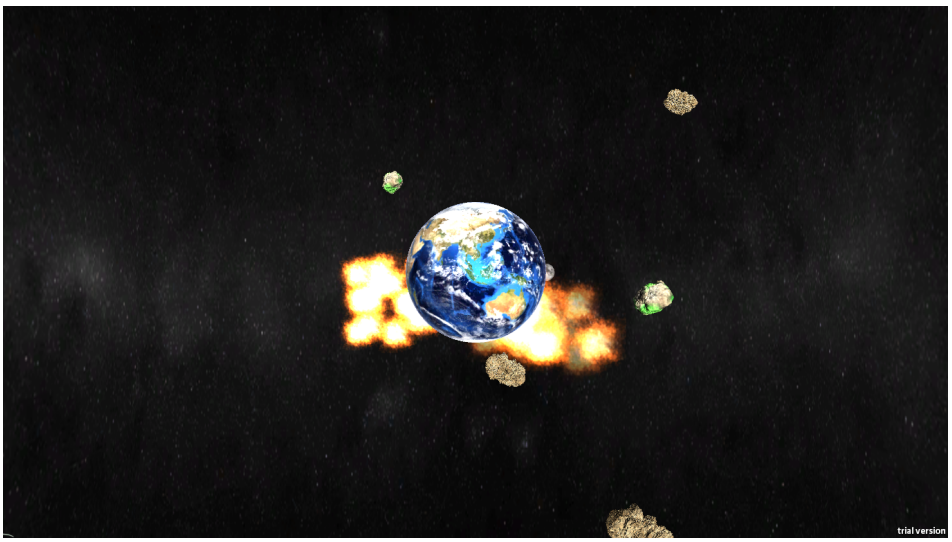


Abbildung 4.3: Die Spielansicht in Level 1. Die Erde in der Mitte wird von Meteoriten getroffen, welche auf der Oberfläche explodieren. Rechts und links oben fliegen grüne Meteoriten, welche die Erde lediglich passieren, aber nicht auf ihr einschlagen.

Das Zielen und Abschießen der Meteoriten wird komplett mit den Augen gesteuert. Hierzu wird an der vom Benutzer fokussierten Stelle ein Fadenkreuz eingeblendet. Ist ein Meteorit im Fokus des Benutzers, wird er augenblicklich zerstört. Ein Klicken oder Tastendruck ist nicht notwendig. Die Erde und der Mond können nicht abgeschossen werden. Gespielt wird drei Minuten. Danach wird der Punktestand und die Anzahl der abgeschossenen Meteoriten in einer extra Szene dargestellt, von dieser aus der Spieler wieder ins Hauptmenü gelangen kann.

Abbildung 4.5: Meteoritentypen. Ganz links der grüne Meteorit, welcher nach Möglichkeit nicht abgeschossen werden sollte.

Level 2 bietet eine andere Aufgabe. In dieser Szene fliegen Meteoriten von links nach rechts über den Bildschirm. Hinter jedem dritten Meteorit versteckt sich ein feindliches Raumschiff, welches nach Möglichkeit nicht den rechten Bildschirmrand erreichen sollte. Um die Raumschiffe für den Benutzer sichtbar zu machen, wird der Meteorit, hinter dem sich ein Raumschiff befindet transparent dargestellt und grün umrandet. Ziel dieses Levels ist die

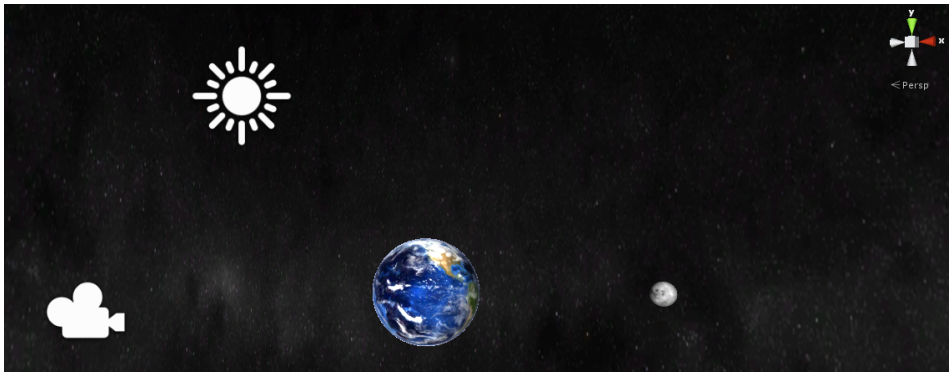


Abbildung 4.4: Entwicklungsansicht in Unity 3D. Das Kamerasymbol links zeigt die Position der Kamera, durch die der Spieler die Szene sieht. Oben zu sehen ist ein Symbol für eine Lichtquelle, welches die Sonne simuliert. In der Mitte unten sind die Erde und der Mond zu erkennen. Als Geometrie dienten lediglich Kugeln, auf die unterschiedliche Texturen gemappt wurden.

Raumschiffe, welche hinter den Meteoriten fliegen, zu zerstören, ohne den Meteorit davor abzuschießen. Der Spieler soll sich also möglichst auf das dahinter fliegende Raumschiff konzentrieren und nicht auf den Meteoriten, welcher auf einer Ebene deutlich näher am Betrachter fliegt. Das Fadenkreuz wird wieder an der Position gerendert, die vom Benutzer fokussiert wird. Um zu schießen, ist ein Linksklick erforderlich (hier ist es egal, worauf der Benutzer klickt, die Bewegung der Maus hat keinen Einfluss auf das Spiel). Dies gibt dem Spieler die Möglichkeit, sich vorerst auf das abzuschießende Objekt zu konzentrieren. Auch hier wird das Spiel nach drei Minuten beendet und der Spieler bekommt den Highscore und die Anzahl der zerstörten Schiffe sowie Meteoriten angezeigt. Anschließend gelangt der Spieler von hier aus wieder ins Hauptmenü.

Kalibrierungsszene 1

Das Eyetrackingsystem muss vor dem Gebrauch kalibriert werden. Dies erfolgt in einer extra Szene, in welcher der Nutzer bestimmte Punkte auf der Bildfläche fokussieren und diese dann mit einem Mausklick bestätigen muss. Zu Beginn ist in der Mitte ein blaues Kreuz mit einem kleineren roten Kreuz in dessen Mitte dargestellt (siehe Abbildung 5.3). Der Benutzer sollte genau auf das kleine Kreuz schauen und dann mit einem Linksklick das Eyetrackingsystem auffordern, seinen aktuellen Blick für diesen Punkt zu akzeptieren. Nimmt das System den Blick des Benutzers an, werden weitere acht Punkte an verschiedenen Positionen auf der Bildfläche dargestellt, welche auf die gleiche Weise vom Benutzer abgearbeitet werden müssen. Nach neun Kalibrierungspunkten ist die Kalibrierung abgeschlossen. Nun kann der Benutzer an Meteoriten, welche an den neun Kalibrierungspunkten nacheinander angezeigt werden, seine Kalibrierung testen. Kann der Benutzer alle neun Meteoriten abschießen, gelangt er in die nächste Kalibrierungsszene.

Kalibrierungsszene 2

Für den zweiten Level des Spiels ist eine Tiefenkalibrierung erforderlich, da erfasst werden soll, welches Objekt aus einer Menge von hintereinander angeordneten Objekten unterschiedlicher Tiefe im Blickfokus des Benutzers liegt. Dazu werden an bestimmten Kalibrierungspositionen Meteoriten angezeigt, welche vom Benutzer fokussiert werden sollen. Hat der Benutzer das Objekt genau im Blick, muss dies mit einem Linksklick bestätigt werden. Die Applikation zeichnet nun verschiedene Werte zweier Augenattribute auf - zum einen den Pupillendurchmesser und zum anderen den Abstand beider Pupillen zueinander. Diese Werte werden für alle neun Kalibrierungspositionen gespeichert und dies zwei mal, denn der Benutzer muss neun mal auf einer ihm näheren Ebene kalibrieren und neun mal auf einer Ebene weiter von ihm entfernt. Diese Ebenen entsprechen von der Entfernung her den Ebenen des zweiten Spiellevels. Auf der vorderen Ebene fliegen die Meteoriten und auf der hinteren die Schiffe.

4.2.2 Zielen und Schießen

Während des Spiels werden die Augendaten vom Eyetracker über den dazugehörigen Laptop zum Kommunikationsserver geschickt. Dieser verarbeitet die Datenpakete und berechnet zusätzliche Werte, wie zum Beispiel die Distanz zwischen linker und rechter Pupille. Danach werden die Daten weiter an den Rechner mit der Beispielapplikation geschickt.

Folgende Augendaten $\vec{e}$ werden schließlich vom Kommunikationsserver an die Beispielapplikation geschickt:

$\vec{e} = (\vec{l}, \vec{r}, \vec{d_l}, \vec{d_r}, d_p)$

$\vec{l} = (l_x, l_y)$: Blickposition in Pixel $((\pm 2^{32} Pixel) * 32)$ auf der Bildfläche des linken Auges.

$\vec{r} = (r_x, r_y)$: Blickposition in Pixel $((\pm 2^{32} Pixel) * 32)$ auf der Bildfläche des rechten Auges.

$\vec{d_l} = (dl_x, dl_y)$: Durchmesser in Pixel $((0...2^{32} Pixel) * 32)$ der Pupille in x und y Richtung des linken Auges.

$\vec{d_r} = (dr_x, dr_y)$: Durchmesser in Pixel $((0...2^{32} Pixel) * 32)$ der Pupille in x und y Richtung des rechten Auges.

d_p : Abstand zwischen der linken und rechten Pupille in Pixel $((0...2^{32} Pixel) * 32)$.

Diese Daten bieten die Grundlage für das Rendern des Fadenkreuzes und die Schussberechnung.

Um das Fadenkreuz an der Stelle zu platzieren, die im Blickfokus des Benutzers liegt, reicht es aus, entweder die Blickposition des rechten oder des linken Auges anzugeben. Da $\vec{l}$ und $\vec{r}$ in Pixeln angegeben sind, müssen die Werte nicht mehr umgerechnet werden, sondern können direkt in Unity 3D verwendet werden. In jedem Frame, also in der Updatemethode des Skripts für das Zielen, werden diese Werte genutzt, um das Fadenkreuz über den Bildschirm zu bewegen. Ändert der Nutzer seinen Blickfokus, reagiert das Fadenkreuz direkt

darauf und folgt somit dem Blick des Spielers. Zusätzlich wird in jedem Frame geprüft, ob der Blick des Nutzers auf ein Objekt im 3D-Raum fällt. Dies geschieht mithilfe von Strahlen (in Unity 3D Rays genannt). Unity 3D bietet hier die Möglichkeit von der Kamera (den Kameras) aus über Bildschirmkoordinaten (in Pixeln anzugeben, also die Werte der Vektoren $\vec{l}$ und $\vec{r}$) Strahlen in die Spielwelt zu schießen. Dies kann genutzt werden, um festzustellen, welches Objekt hinter dem Fadenkreuz liegt, also vom Benutzer angeschaut wird.

Da im ersten Level keine zusätzliche Eingabe für das Schießen notwendig ist, wird jedes Objekt, welches von einem solchen Strahl getroffen wird, zerstört. Ausnahmen sind im ersten Level die Erde in der Mitte und der darum kreisende Mond. Der Nutzer ist so nur in der Lage heranfliegende Meteoriten zu zerstören. Wird ein Meteorit getroffen, wird das Objekt, welches ihn repräsentiert zerstört und eine Explosionsanimation instanziiert. Über ein zusätzliches Objekt, das "GameObserverObject", werden solche Abschüsse von der Applikation protokolliert und im CSV-Format gespeichert.

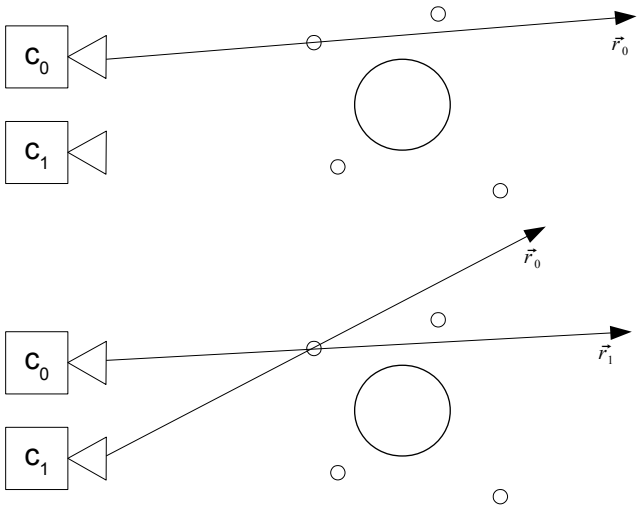


Abbildung 4.6: Schnitttests im Raum. Oben mit einem Strahl r_0 die direkte Methode. Unten mit zwei Strahlen r_0 und r_1: nur wenn beide Strahlen das selbe Objekt treffen, wird es zerstört.

Insgesamt wurden zwei unterschiedliche Auswertungsmethoden implementiert. Die erste Methode benötigt nur die Daten des linken oder rechten Auges, also entweder $\vec{l}$ oder $\vec{r}$. In unserem Fall wurde $\vec{l}$ gewählt (siehe Abbildung 4.6, oberer Teil). Nun wird ein Strahl vom Betrachter über die Koordinaten von $\vec{l}$ direkt in die Szene geschossen. Wie oben beschrieben, werden dann die getroffenen Objekte ermittelt. In Unity 3D wird die Szene durch Kameras betrachtet. Diese sind der Ursprung der Strahlen. Um eine dreidimensionale Darstellung zu ermöglichen, werden zwei Kameras benötigt.

Die zweite Auswertungsmethode nutzt diese beiden Kameras als Ursprung für ihre Strahlen (siehe Abbildung 4.6, unterer Teil). Es werden von zwei Kameras aus Strahlen in die Szene

geschossen und nur wenn beide dasselbe Objekt treffen, wird dieses zerstört. Für die Richtung der beiden Strahlen werden diesmal beide Blickpositionen ($\vec{l}$ und $\vec{r}$) genutzt. Mathematisch gesehen fehlt diesen Vektoren noch eine zusätzliche Dimension. Hier reicht es aus, den dritten Wert für die Richtung eines Strahls auf eins zu setzen und nur die beiden ersten Einträge auf die in Spielkoordinaten transformierten Werte der beiden Augen zu setzen. Diese Transformation kann mithilfe der Kamera in Unity 3D automatisch geschehen, da die hierfür notwendigen Operationen durch das Kameraobjekt bereitgestellt werden.

Die Anforderungen an den Spieler in Level 2 verlangen einige Erweiterungen der Zielmethoden und die Einbindung weiterer Augenattribute. Im zweiten Level wurden zwei verschiedene Ansätze implementiert. Hierzu wurde zum einen der Pupillendurchmesser und zum anderen die Distanz zwischen der Pupille des rechten und der Pupille des linken Auges einbezogen.

Die Methode, die den Pupillendurchmesser nutzt, erhält Kalibrierungswerte aus der zweiten Kalibrierungsszene. Es wird der Durchschnitt aller neun Werte (Pupillendurchmesser/Pupillendistanz) vom jeweils linken und rechten Auge ermittelt. Beim Schießen werden die Durchschnittswerte genutzt, um zu erkennen, ob der Betrachter ein Objekt auf der vorderen oder der hinteren Ebene fokussiert. Da beide Objekte vom Betrachter aus hintereinander auf einer Geraden liegen, führt dies zwangsläufig dazu, dass Unity 3D immer das vordere Objekt beim Zieltest zurückliefert. Um jedoch ein hinteres Objekt zu treffen, wird beim Auswerten zusätzlich auf den Pupillendurchmesser geachtet. Dabei wird beim Schießen die Distanz des aktuellen Pupillendurchmessers zu den Durchschnittswerten der Pupillendurchmesser der vorderen und hinteren Ebene berechnet. Ist die Differenz zum Durchschnittswert der vorderen Ebene kleiner als die Differenz zum Durchschnittswert der hinteren, wird das vordere Objekt - der Meteorit - zerstört, andernfalls das hintere, also das Raumschiff.

Abbildung 4.7: Transparente Meteoriten und Raumschiffe, welche sich hinter den Meteoriten verstecken. Durch die transparenten Meteoriten soll der Benutzer die Raumschiffe erfassen können. Dazu wird entweder der Pupillendurchmesser oder die Pupillendistanz (zwischen rechter und linker Pupille) genutzt.

Ähnlich funktioniert die zweite Methode. Als Vergleichswert dient hier allerdings der Abstand zwischen der linken und rechten Pupille. Ansonsten wird nach dem gleichen Prinzip ausgewertet wie beim Pupillendurchmesser. Wie zuvor beschrieben werden die Durchschnittswerte (Abstände der Pupillen) wieder mit dem aktuellen Abstand beider Pupillen verglichen, um so festzustellen, ob Objekte der vorderen oder hinteren Ebene fokussiert wurden.

4.2.3 Datenerfassung

Um eine spätere Benutzerstudie durchführen zu können, erfasst die Beispielapplikation Daten während des Spielvorgangs. Hierfür wurden zwei Skripte implementiert, jeweils eins für den ersten und für den zweiten Level. Diese Skripte stellen Methoden zur Verfügung, welche überall in der Beispielapplikation aufgerufen werden können. Diese Methoden wurden in allen relevanten Abschnitten integriert und zeichnen bei bestimmten Ereignissen Werte auf. Wird ein Meteorit abgeschossen, werden Daten in Form eines Log-Eintrags in eine Liste eingefügt. Diese Daten enthalten unter anderem den Zeitpunkt des Ereignisses, die aktuelle Position des Meteorits, alle vom Eyetrackingsystem erhaltene Blickdaten $\vec{e}$ und eine kurze Information über die Art des Ereignisses (beispielsweise der Einschlag eines Meteoriten

auf der Erde oder der Abschuss eines Meteoriten durch den Benutzer). Zusätzlich werden Start- und Endzeitpunkt des Spielvorgangs und die vom Spieler erreichte Punktzahl, sowie die genutzte Zielmethode und der Spielername erfasst. Die gesammelten Daten werden im CSV-Format gespeichert und können dadurch leicht in Tabellenkalkulationsprogrammen genutzt werden.

4.3 Eyetrackingsystem

Zur Erfassung der Augen diente ein Eyetracker der Firma SMI (SensorMotoric Intruments) des Typs "RED"[3]. Es handelt sich hierbei um einen stationären Eyetracker auf Infrarotbasis. Zwei Lampen am linken und rechten Rand strahlen den Benutzer an. Weiter in der Mitte liegen Kameras, welche die Augen aufnehmen. Der Eyetracker ist mit einem extra System verbunden, auf welchem die mitgelieferte Software läuft. Diese Software ist für die Erfassung der Augen zuständig und liefert, via UDP [4], die gewünschten Daten an im Netzwerk befindliche Systeme. Dazu gibt man die IP-Adresse an, an welche die Datenpakete geschickt werden sollen. Beim Empfänger können diese dann leicht aus dem UDP-Datenpaket ausgelesen werden. Da es sich bei UDP um ein verbindungsloses Protokoll handelt, ist weder garantiert, dass die Pakete in der richtigen Reihenfolge ankommen, noch dass sie nicht unterwegs verloren gehen. Während des Spielens stellt dies jedoch kein Problem dar, da ein verlorenes Paket in der Menge der Pakete untergeht. Der Eyetracker sendet Daten mit einer Frequenz von 30 Hz. Verlorene Daten werden deshalb nicht einmal bemerkt, da das nächste Paket genutzt werden kann, um die Augendaten weiter auszuwerten. Verlorene Pakete während des Kalibrierens sind problematischer. Das Protokoll, welches den Datenaustausch während der Kalibrierung realisiert, ist deshalb so konzipiert, dass derartige Verluste erkannt werden. In diesem Fall besteht die Möglichkeit, den verlorenen Vorgang noch einmal zu wiederholen. Hierzu zählt zum Beispiel das Akzeptieren eines Kalibrierungspunktes.

4.4 Kommunikationsserver

Der Kommunikationsserver stellt das Bindeglied zwischen Beispielapplikation und Eyetrackingsystem dar. Der in JAVA implementierte Server vermittelt nicht nur die Datenpakete zwischen den beiden Komponenten, sondern bietet noch zusätzliche Funktionen, wie diverse Filter für die Augendaten (Glättung der Blickposition, also ein Entgegenwirken gegen das Zittern des Blicks) und ermöglicht das Ausführen weiterer "Remote Commands" des Eyetrackers. Außerdem wird beispielsweise der Abstand der beiden Pupillen ad hoc auf dem Server berechnet und an die Applikation weitergegeben. Der Server ist aus verschiedenen Modulen aufgebaut (siehe Abbildung 4.1). Die Kommunikation geschieht über das lokale

[3] http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/red-red250-red-500.html, letzter Zugriff am 20.10.2013
[4] User Datagram Protocol

Netzwerk via UDP. In diesem Abschnitt sind die wichtigsten Komponenten des Kommunikationsservers beschrieben. Eine genauere Beschreibung ist in der gleichnamigen Arbeit von Rufat Rzayev zu finden.

4.4.1 ConnectionHandler

Diese Komponente organisiert die Kommunikation zweier Geräte im Netzwerk. Der Server verfügt über zwei ConnectionHandler-Instanzen - eine für die Vermittlung von Paketen vom Eyetrackingsystem zum Kommunikationsserver und eine für die Weiterleitung der Pakete zur Beispielapplikation. Die ConnectionHandler-Instanz zwischen Eyetrackingsystem und Kommunikationsserver implementiert ein spezielles Protokoll. Über UDP können sogenannte "Remote Commands" gesendet werden. Diese werden als Strings in UDP-Pakete integriert und dann an das Eyetrackingsystem geschickt. Für jeden relevanten "Remote Command" wurde eine eigene JAVA-Methode auf dem Server implementiert, welche die übergebenen Parameter in ein UDP-Paket integriert und dieses an den Eyetracker schickt. Über den ConnectionHandler kann das Eyetrackingsystem aus der Ferne konfiguriert und gesteuert werden. Wie die "Remote Commands" funktionieren, ist dem Handbuch [Sen09] des Eyetrackers zu entnehmen.

Eine zweite ConnectionHandler-Instanz implementiert ein weiteres Protokoll zur Kommunikation zwischen der Beispielapplikation und dem Kommunikationsserver. Dieses Protokoll nutzt ebenfalls in UDP-Pakete integrierte Strings, welche Anweisungen sowie Augendaten enthalten.

4.4.2 StreamHandler

Diese Komponente leitet den Stream (die Augendatenpakete), welche vom Eyetrackingsystem geliefert werden, direkt an die Applikation weiter und berechnet, wenn gewünscht, einen Filter zur Glättung des Blickfokusses. Diese Komponente wird gestartet, wenn der Benutzer den Eyetracker erfolgreich kalibriert hat. Nun werden bis zum Abbruch des Streams Augendatenpakete vom Eyetrackingsystem geliefert und über den Server an die Beispielapplikation geschickt. Befindet sich die Applikation im Spielmodus, können diese Pakete als Benutzerinput verwendet werden, um das Spielen per Eyetracking möglich zu machen.

4.4.3 RemoteController

Der RemoteController kümmert sich um das Ausführen zusätzlicher Befehle an das Eyetrackingsystem (Remote Commands). Beispiele hierfür sind das Stoppen des Datenstreams oder der Abbruch der Kalibrierung.

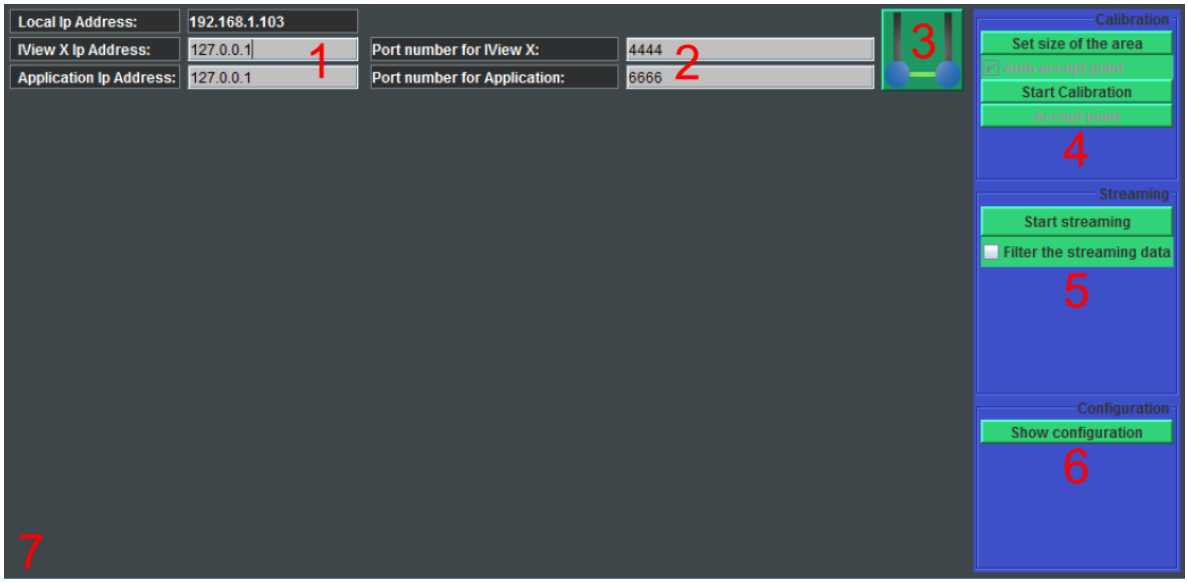


Abbildung 4.8: Grafische Oberfläche des Kommunikationsservers. 1. Eingabefelder für die IP-Adressen zum Eyetrackingsystem und zum Beispielapplikation. 2. Portnummern zu Eyetrackingsoftware (oben) und Beispielapplikation (unten). 3. Startknopf um den Server in Bereitschaft zu versetzen. 4. Kalibrierungskontrolle. 5. Streamkontrolle und Filteroption. 6. Konfigurationsansicht. 7. Informationsfenster

4.4.4 Grafische Benutzerschnittstelle

Zur Erleichterung der Bedienung wurde der Server mit einer grafischen Benutzeroberfläche ausgestattet (siehe Abbildung 4.8). Diese ermöglicht neben der Überwachung der Kommunikation auch das Ausführen von "Remote Commands", um beispielsweise den Datenstream zu starten/stoppen oder die Kalibrierung zu konfigurieren. Zusätzlich können die bereits erwähnten Filter aktiviert werden, welche dem Zittern des Blickfokusses entgegenwirken. Im Normalfall benötigt man nur die Startfunktion des Servers (Nr. 3 in Abbildung 4.8). Die grafische Benutzerschnittstelle ermöglicht im Fehlerfall beispielsweise den Stream manuell stoppen, um ein reibungsloses Neustarten der Applikation zu ermöglichen. Ein nicht gestoppter Stream kann zu einem Paketverlust zwischen Beispielapplikation und Kommunikationsserver führen.

4.5 Generierung von 3D-Inhalten

Zur dreidimensionalen Darstellung der Beispielapplikation wurde ein 3D-Fernseher mit 55"-Bildschirmdiagonale der Marke Phillips genutzt. Dieser nutzt Polarisationsbrillen, um den 3D-Effekt zu erzeugen. Diese Brillen haben eine leichte Tönung und beeinträchtigen

somit das Eyetracking. Reflexionen auf der Brille erschweren zusätzlich die korrekte Arbeit mit dem Eyetrackingsystem. Nur wenn die Lichtverhältnisse stimmen und der Benutzer sich nicht bewegt, funktioniert das Gesamtsystem ordnungsgemäß. Das Display stellt Modi für 2D- und 3D-Darstellung zur Verfügung. Um den gewünschten 3D-Effekt zu erzielen, wurde die "Side-by-Side" Technik gewählt (siehe Abbildung 4.9). Die Szene wird von zwei Kameras aufgenommen, welche so die zwei unterschiedlichen Bilder für die beiden Augen generieren. Mithilfe dieser beiden Bilder kann nun das dreidimensionale Bild erzeugt werden. Um die zwei verschiedenen Bilder zu generieren, wurde in Unity 3D per Skript eine zweite Kamera erzeugt. Diese ist lediglich eine Kopie der ersten Kamera mit einer anderen Position. Dazu wurde ein Skript verwendet (Unity Stereoskopix 3D v027[5]), welches den dazu benötigten Programmcode zur Verfügung stellt. Das Skript stellt unterschiedliche Methoden zur Verfügung, um 3D-Bilder zu generieren. Dazu zählt unter anderem auch die Side-By-Side-Technik (neben- und untereinander). Um das Skript nutzen zu können, wird es der Kamera in Unity 3D hinzugefügt und die entsprechende 3D-Technik ausgewählt, welche das 3D-Display unterstützt.

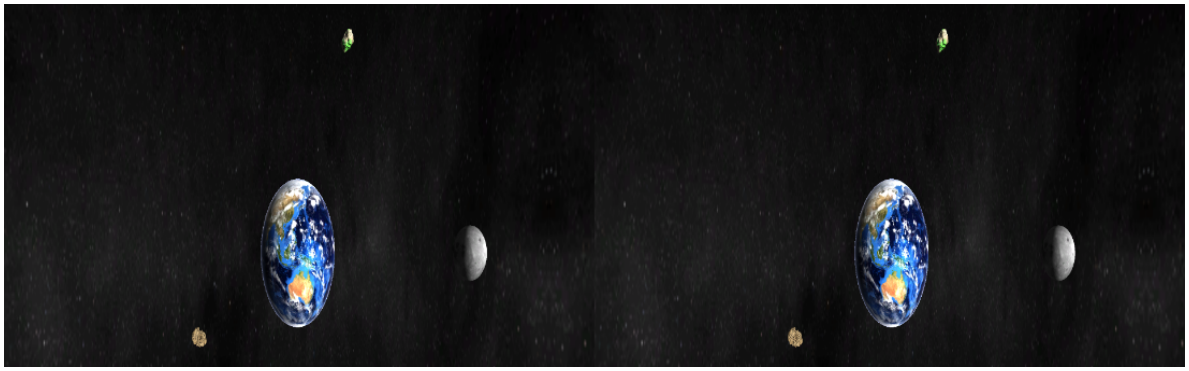


Abbildung 4.9: Side-By-Side-Technik: Links und rechts die Bilder, generiert von linker und rechter Kamera. Aus diesen Bildern kann der 3D-Bildschirm ein Bild erzeugen, welches durch eine Polarisationsbrille betrachtet den gewünschten 3D-Effekt im Gehirn hervorruft.

[5] http://forum.unity3d.com/threads/63874, letzter Zugriff am 20.10.2013

5 Evaluierung

Für die Evaluierung wurden eine Versuchsumgebung aufgebaut und Probanden eingeladen. Die Studienzeit erstreckte sich über zwei Tage. Für jeden Probanden wurde ca. eine dreiviertel Stunde benötigt. Insgesamt teilgenommen haben 15 Personen, davon 14 männliche und eine weibliche. Tatsächlich haben 17 Personen an der Studie teilgenommen, aber zwei Datensätze mussten verworfen werden, da Kalibrierungsprobleme und Umwelteinflüsse die Interaktion unmögliche machten. Die Evaluierung wurde zusammen mit Rufat Rzayev im Labor des Simulation Technologie - Gebäudes an der Universität Stuttgart durchgeführt.

5.0.1 Technische Daten

Die technischen Daten sind in Tabelle 5.1 dargestellt. Das Eyetrackingsystem als Gesamtpaket mit zugehörigem Laptop wurde bezüglich der Hardware nicht verändert.

5.0.2 Versuchsumgebung

Als Versuchsumgebung wurde ein Raum gewählt, welcher komplett von Tageslichteinflüssen abgeschottet werden konnte, um durch künstliches Licht immer gleiche Lichtverhältnisse für jeden einzelnen Probanden zu erhalten. Die drei Komponenten des Gesamtsystems wurden verbunden und der Eyetracker in einem Abstand von 171.5 cm zur Bildfläche des 3D-Displays aufgestellt (siehe Abbildung 5.1). Vor dem Eyetracker wurde eine Kopfstütze (siehe Abbildung 5.2) befestigt, die die Probanden daran hindern sollte, ihren Kopf zu bewegen. Die Bildschirme von Eyetrackinglaptop, Applikationsrechner und Kommunikationsserver

	Eyetracking-Laptop	Kommunikations-server	Applikations-rechner
Hersteller	Lenovo	ASUS	Sony
Modell	ThinkPad X200	X83V	VPC-F22C5E
Prozessor	2.4Ghz	2x2.0Ghz	8x2.0GHz
Arbeitsspeicher	2GB	4GB	8GB
Grafikkarte	Mobile Intel 4 Series Express Chipset Family	NVIDIA GeForce 9300M GS	NVIDIA GeForce GT 540M
Betriebssystem	Windows XP	Windows 7	Windows 7

Tabelle 5.1: Technische Daten der verwendeten Systeme.

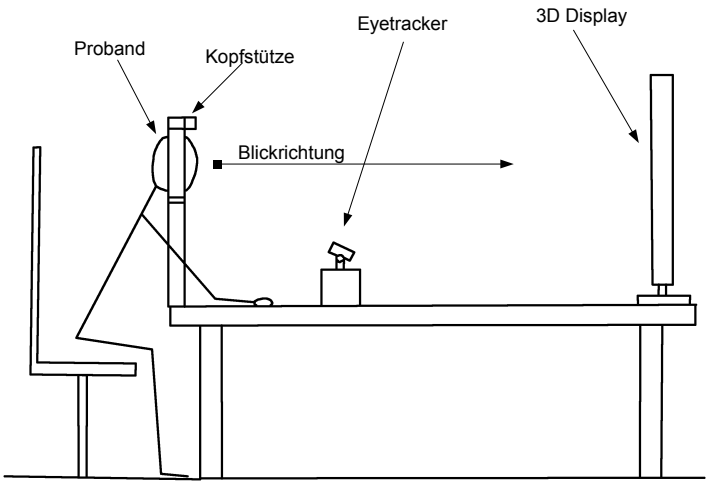


Abbildung 5.1: Versuchsaufbau. Der Proband (links) blickt durch die Kopfstütze auf den 3D-Display. In der Mitte auf dem Tisch befindet sich der Eyertracker, welcher den fixierten Proband genau im Fokus hat.

waren für die Probanden nicht einsehbar. Die Probanden hatten somit nur den 55"-Display im Blick. Für die Probanden wurde eine Polarisationsbrille und ein stationärer Stuhl (ohne Rollen) bereitgestellt.

5.1 Prozedur

Die Studie war mit einer dreiviertel Stunde pro Teilnehmer angesetzt und wurde im Labor 2 des Simtech-Gebäudes an der Universität Stuttgart durchgeführt. Getestet wurden 15 Personen. Zu Beginn wurden die Probanden belehrt. Die Probanden wurden darüber informiert, dass sie die Studie jederzeit abbrechen könnten. Zusätzlich wurden Informationen über etwaige Sehschwächen erhoben, insofern die Probanden einverstanden waren. Alter und Geschlecht, sowie Berufsfeld wurden ebenfalls erfasst.

5.1.1 Kalibrierung

Bevor die Probanden mit dem Spielvorgang beginnen konnten, musste das System kalibriert werden. Dies geschah vor jedem Level, um gleichbleibende Bedingungen zu gewährleisten, da die Teilnehmer nach jedem Level Fragebögen ausfüllen mussten und somit ihre Position veränderten. Dem Proband wurde kurz erklärt, was für die Kalibrierung erforderlich war.

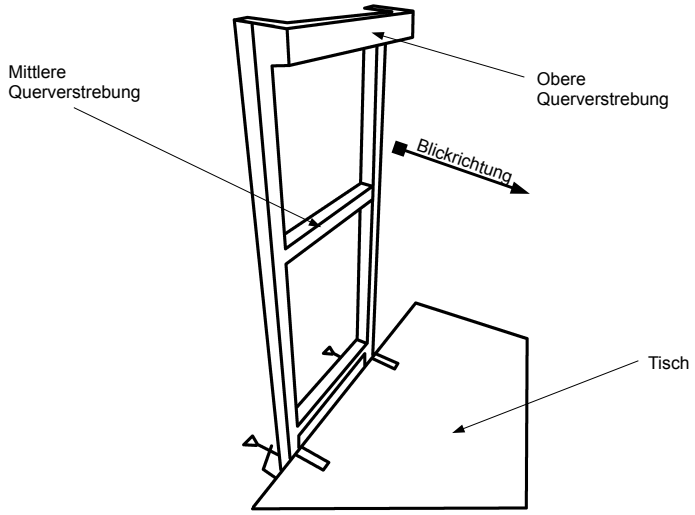


Abbildung 5.2: Schematische Darstellung der Kopfstütze. Der Proband legt seinen Kopf mit dem Kinn auf der in der Mitte befindlichen Querverstrebung ab und berührt mit der Stirn die obere, etwas weiter vorne liegende Querverstrebung.

Der Teilnehmer sollte nacheinander die neun Kalibrierungspositionen abarbeiten. Dazu sollte er genau auf die Mitte des Kalibrierungssymbols schauen. Dort war ein kleines rotes Kreuz zu sehen, welches die genaue vom Eyetrackingsystem geforderte Position darstellte (siehe Abbildung5.3). Mit Linksklick wurde der momentane Kalibrierungspunkt bestätigt. Konnte das System den Blick des Probanden nicht genau erfassen, wurde der Punkt nicht akzeptiert und musste erneut bestätigt werden. Nach einer erfolgreichen Kalibrierung konnte die Testperson noch in der Kalibrierungsszene ein paar Testschüsse machen. Dazu wurden nacheinander Meteoriten an den neun Positionen, welche zuvor fixiert werden mussten, generiert und konnten durch Anschauen zerstört werden. Nach neun Meteoriten gelangte der Proband wieder ins Hauptmenü. War ein Nutzer nicht zufrieden mit der Kalibrierung, konnte diese auf Wunsch wiederholt werden. Nun konnte ein Level ausgewählt und schließlich gespielt werden. Eine Besonderheit gab es beim Kalibrieren für den zweiten Level. Wie im Implementierungsteil beschrieben, musste hier noch zusätzlich eine Tiefenkalibrierung durchgeführt werden.

Abbildung 5.3: Kalibrierungsobjekt. Man sieht das kleine rote Kreuz in der Mitte, welches die genaue Position des zu fixierenden Punktes darstellt.

5.1.2 Spielen

Nach einer erfolgreichen Kalibrierung konnte gespielt werden. Pro Proband wurde die Reihenfolge der Level geändert. Der erste Proband begann mit dem ersten Level, der zweite mit dem zweiten und der dritte wieder mit dem ersten. Auch die jeweiligen Darstellungsarten wurden abwechselnd genutzt. Manche Probanden begannen mit der zweidimensionalen Darstellung und manche mit der 3D-Darstellung. Im zweiten Level wurden nach dem selben Prinzip die Zielmethoden abgewechselt. Jeder Proband spielte jeden Level drei Minuten. Nach der Spielzeit wurde der Highscore angezeigt und der Proband sollte eine Reihe von Fragebögen ausfüllen.

5.1.3 Fragebögen

Nach dem Spielen füllten die Probanden je nach Level unterschiedliche Fragebögen aus. Für den ersten Level wurden NASA TLX (Task Load Index) [nas] , SUS (System Usability Scale) [Bro96] und PRC (Product Reaction Cards) [BM02] verwendet. Bei Level 2 mussten die Probanden lediglich den PRC-Fragebogen ausfüllen. Zusätzlich konnten die Teilnehmer nach jedem Spielvorgang Feedback in Form eines Freitextes, einer Zeichnung oder mündlich geben.

5.1.4 Probanden

Teilgenommen haben an der Studie wie zuvor angegeben 15 Personen (14 männlich und eine weiblich) im Alter von 22 bis 28 Jahren (Durchschnittsalter 24,666), deren Daten schlussendlich verwendet werden konnten. Die Teilnehmergruppe bestand ausschließlich aus Studenten, vorwiegend aus dem Bereich Informatik und Softwaretechnik. Des weiteren nahmen Maschinenbaustudenten und eine Person aus dem Bereich Sozialwissenschaften teil. Fünf der insgesamt 15 Probanden trugen wären der Studie eine Brille unter der 3D-Brille und ein Teilnehmer benutzte Kontaktlinsen.

5.1.5 Ablauf

Nach der Belehrung wurde der Proband gebeten vor der Kopfstütze Platz zu nehmen. Er sollte seinen Kopf auf der ersten Querverstrebung der Kopfstütze ablegen. Anschließend wurde die 3D-Brille aufgesetzt und geprüft, ob der Eyetracker den Probanden gut erfassen konnte. War dies der Fall, konnte kalibriert werden. Nach der Kalibrierung wurde ein Level gespielt. Die Probanden begannen abwechselnd mit Level 1 oder Level 2 und auch die Reihenfolge der Darstellungsmethoden (2D/3D in Level 1) oder der Zielmethoden (Abstand zwischen rechter und linker Pupille oder Pupillendurchmesser im zweiten Level) wurden pro Durchlauf getauscht. Nach dreiminütigem Spielen wurde der Highscore angezeigt und die Probanden gebeten die jeweiligen Fragebögen auszufüllen. Dazu durften die Teilnehmer sich wieder normal sitzen und den Kopf aus der Kopfstütze nehmen. Insgesamt wurde dieser

Vorgang vier mal durchgeführt - einmal für Level 1 in 2D, einmal für Level 1 in 3D und zwei mal für Level 2 mit unterschiedlichen Zielmethoden. Nach jedem Spielvorgang sollten die Probanden die entsprechenden Fragebögen ausfüllen. Zum Schluss konnten die Probanden weiteres Feedback geben, welches notiert wurde und zu den Fragebögen hinzukam.

5.2 Datenanalyse

Die Evaluierung der Ergebnisse bestand aus zwei Teilen. Zum einen die Analyse der aufgezeichneten Daten während des Spielvorgangs und zum anderen die Auswertung der Fragebögen und des Freitextfeedbacks der Probanden.

5.2.1 Auswertung Level 1

Die Auswertung des ersten Levels legte den Fokus auf Usability und Taskload. Diese wurden per Fragebögen ermittelt. Des weiteren wurde die Erfolgsrate untersucht und ob diese variiert, je nachdem ob im 2D- oder 3D-Raum gespielt wird. Dazu wurden die während des Spielens gespeicherten Daten ausgewertet.

Auswertung der Fragebögen zu Usability und Taskload

Zunächst wurde der System Usabilty Scale Fragebogen (SUS) ausgewertet. Über alle Teilnehmer der Studie hinweg ergaben sich folgende Werte: durchschnittlicher SUS-Score 2D: 74,53 und durchschnittlicher SUS-Score 3D: 74,33 (siehe Abbildung. 5.5). Die Werte unterscheiden sich nur im Nachkommabereich. In Abbildung 5.4 ist abzulesen, in welchem Wertungsbereich die Beispielapplikation liegt. Die Beispielapplikation oder mehr das Gesamtsystem liegt also im zweidimensionalen sowie im dreidimensionalen ziemlich genau auf (knapp darüber) der Grenze zum "guten" Bereich.

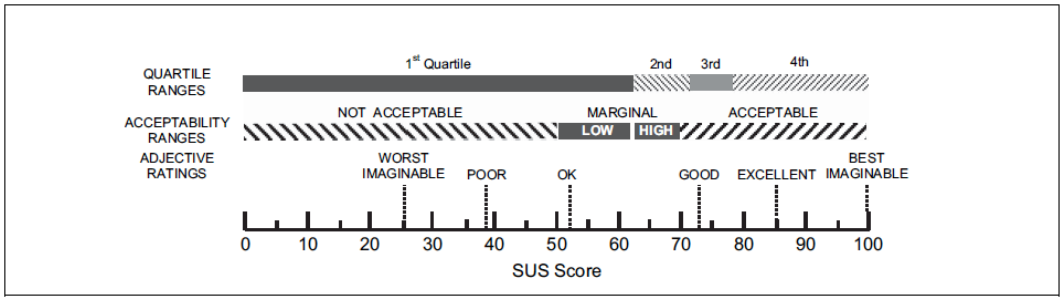


Abbildung 5.4: System Usability Score Umrechung. Je nach erreichter Punktzahl liegt die Anwendung in einem bestimmten Bereich. Je höher die erreichte Punktzahl, desto besser fällt die Bewertung der Usability aus. Quelle: http://blog.seibert-media.net/2011/04/11/usablility-analysen-system-usability-scale-sus/, letzter Zugriff am 20.10.2013

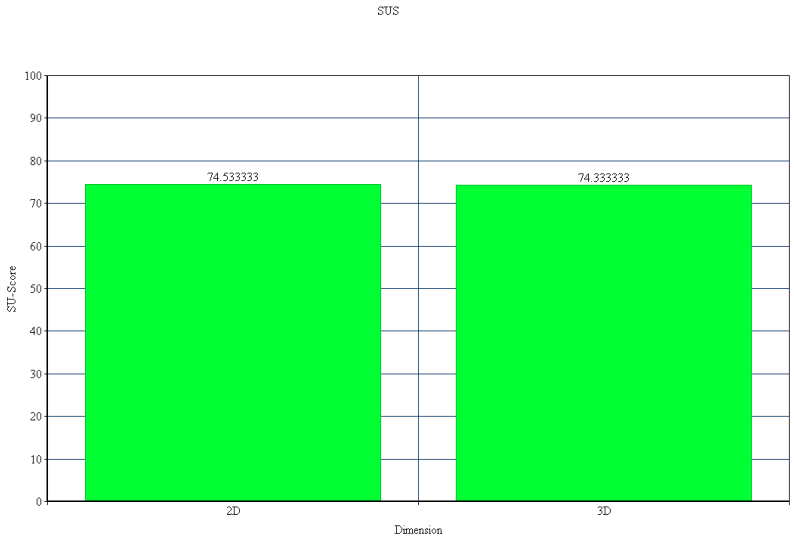


Abbildung 5.5: System Usability Score Ergebnisse (Durchschnittswerte über alle Teilnehmer). Es ist kein signifikanter Unterschied feststellbar. Beide Darstellungstechniken haben einen guten SU-Score erzielt.

Als nächstes wurde der NASA TLX ausgewertet, um die Taskload zu ermitteln. Auch hier konnte kein signifikanter Unterschied zwischen 2D- und 3D-Spielumgebung festgestellt werden. Die Durchschnittswerte über alle Studienteilnehmer liegen für 2D bei 53,4 und für 3D bei 53,26 (siehe Abbildung 5.6). Demnach ist kein deutlicher Unterschied bezüglich der

Taskload zwischen zwei- und dreidimensionaler Darstellung erkennbar. Folglich scheint die 3D-Darstellung keinen besonderen Einfluss darauf zu haben, wie belastend die Aufgabe für die Probanden ist.

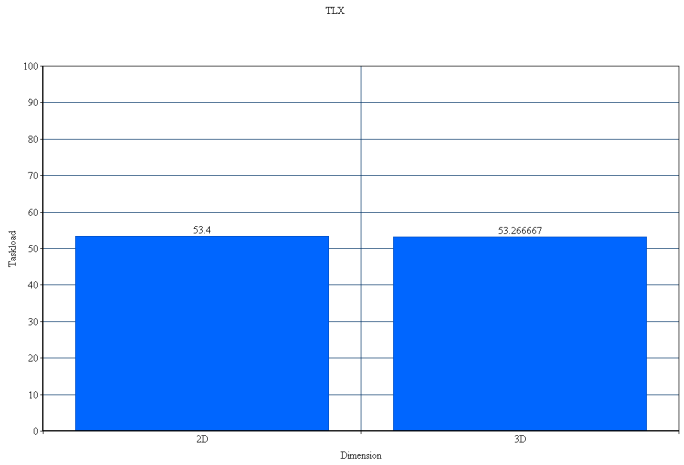


Abbildung 5.6: NASA TLX Ergebnisse (Durchschnittswerte über alle Teilnehmer). Auch hier ist kein signifikanter Unterschied zwischen 2D- und 3D-Darstellung feststellbar. Beide Werte sind relativ mittig angesiedelt.

Als drittes und letztes wurde der PRC-Fragebogen ausgewertet. Jedes markierte Wort wurde gezählt (bei Wörter mit positiv/negativ Option wurden die Markierungen miteinander verrechnet, beispielsweise: 3 + und 2- ergaben 1+). Das Ergebnis ist in Abbildung 5.7 zu sehen. Einige Wörter fallen hier auf, da ihr Vorkommen bezüglich der Darstellungsart (2D/3D) stark variiert.

Das Attribut "Fast" (dt. schnell) wurde sehr stark negativ angegeben, wenn Level 1 im 2D-Modus gespielt wurde. Im 3D-Modus allerdings wurde es nur noch ein mal negativ angegeben. Dies lässt auf Unterschiede bei der Geschwindigkeitsauffassung bezüglich der Darstellungsart schließen. Möglicherweise nahmen die Studienteilnehmer die Geschwindigkeit anders wahr, wenn sie im 3D-Modus spielten. Die hohe negative Wertung im 2D-Modus und die geringe negative Wertung im 3D-Modus lässt darauf schließen, dass die Probanden im 3D-Raum die Geschwindigkeit als weniger problematisch ansahen.

Das Attribut "Visually pleasant" (dt. optisch angenehm) wurde für den dreidimensionalen Fall vier mal angegeben und kommt im 2D-Fall gar nicht vor. In Anbetracht des Spielkontexts, welcher selbst eine 3D-Umgebung ist, ist eine dreidimensionale Darstellung offenbar ansprechender als eine 3D-Welt, dargestellt auf einem 2D-Display. Mit realistischen Texturen und Geometrien kann also eine 3D-Umgebung gestaltet werden, welche dann auch auf einem stereoskopischen Display für den Benutzer ansprechende Bilder erzeugt.

"Entertaining" (dt. unterhaltend) erzielte im 3D-Fall einen doppelt so hohen Wert, wie bei der 2D-Darstellung. Wie bei "Viusally pleasant" kommt die 3D-Welt besser zur Geltung, was scheinbar zu einem höheren Unterhaltungswert führte. Die dreidimensionale Darstellung kann demnach positiv zum Unterhaltungswert einer solchen Anwendung beitragen.

"Playful" (dt. spielerisch) erreichte im zweidimensionalen doppelt so viele Markierungen, wie in 3D. Möglicherweise ist die 2D-Welt Benutzern als Spielwelt geläufiger, da die meisten Computerspiele, welche der Beispielapplikation ähneln, zwar eine 3D-Welt simulieren, aber auf 2D-Displays dargestellt werden.

Viele Wörter (beispielsweise Fun, Exciting oder Innovative) haben für beide Darstellungsarten ähnlich viele Markierungen erhalten. Es gibt also Eigenschaften, welche offenbar nicht von einer räumlichen Darstellung abhängen.

Viele andere Begriffe erhielten keine erwähnenswert hohe Anzahl an Markierungen (meist unter zwei).

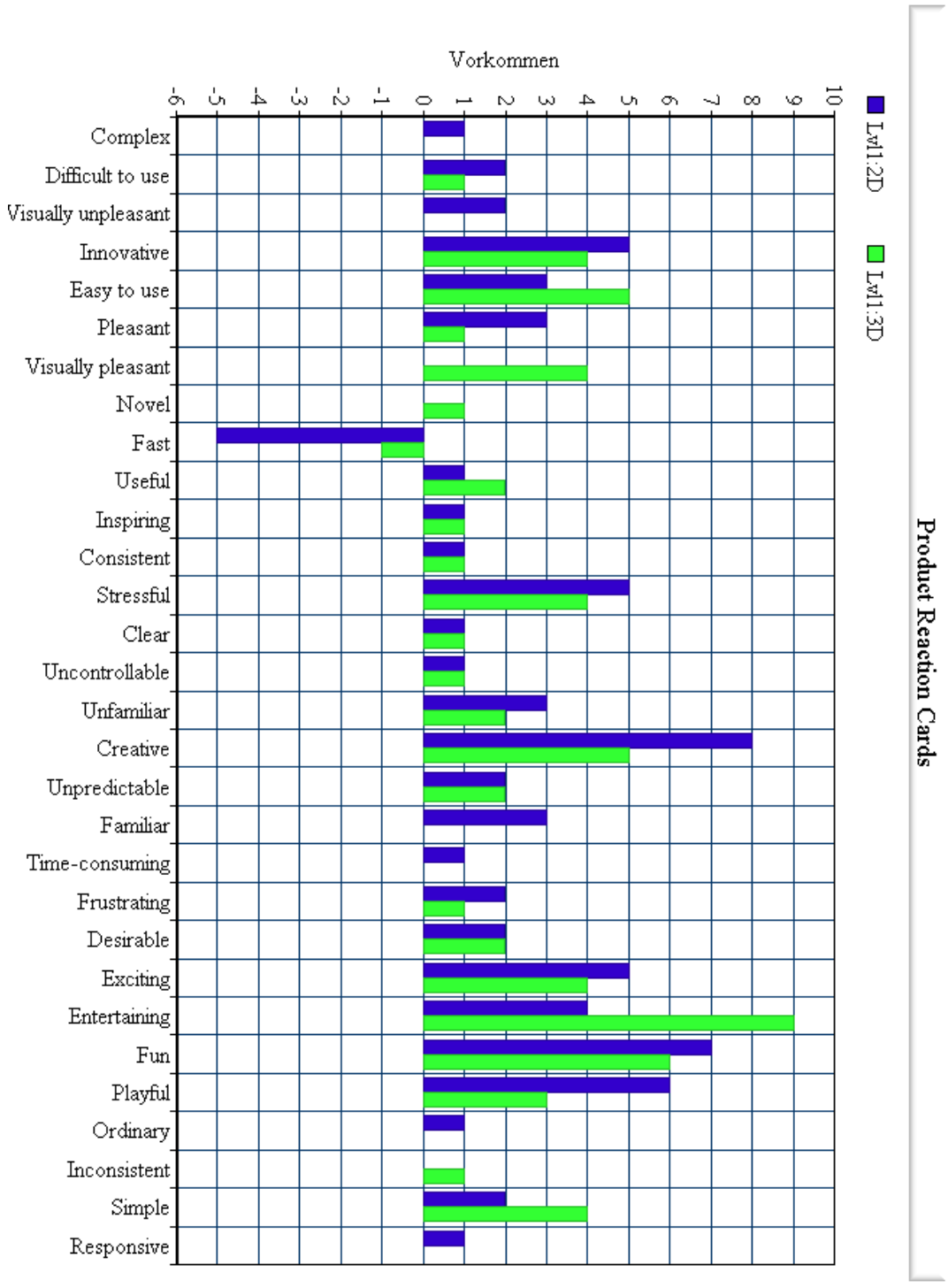


Abbildung 5.7: PRC Ergebnisse für den ersten Level. Blau dargestellt die Ergebnisse für die zweidimensionale Darstellung und grün die für die dreidimensionale.

Auswertung der aufgezeichneten Daten - Vergleich 2D/3D

Um die Erfolgsraten der Probanden bezüglich der Darstellungsart zu ermitteln, wurden die aufgezeichneten Daten analysiert. Diese wurden in den Abbildungen 5.8 für 2D und 5.9 für 3D zusammengefasst. Zunächst erkennt man, dass mit der Zahl der Meteoriten auch die Zahl der Abschüsse und Einschläge steigt und dies ungefähr ähnlich bei der 2D- und 3D-Darstellung - doch nur bis zur siebten Welle. Hier kommt es in beiden Fällen dazu, dass die Anzahl der Einschläge die Anzahl der Abschüsse übersteigt - nur leicht im 2D-Fall und stärker im dreidimensionalen. In den nachfolgenden Wellen bleibt die Anzahl der Einschläge in 2D immer unter der Anzahl der Abschüsse. Anders sieht es bei der 3D-Darstellung aus. Hier überholen sich die Werte der Abschüsse und Einschläge mehrmals, was darauf schließen lässt, dass die dreidimensionale Darstellung bei einer größeren Anzahl von Objekten den Benutzer überfordern kann. Womöglich wird die Szene unübersichtlicher, wenn der Benutzer sie räumlich wahrnimmt und er zu viele Eindrücke zur selben Zeit verarbeiten muss. Es könnte auch sein, dass sich die Augen des Nutzers im 3D-Fall und bei einer Eindrucksüberlast anders verhalten und somit das Eyetrackingsystem in seiner Funktion beeinträchtigt wird. Anders als die Abschuss- und Einschlagskurven verhält sich die Kurve der grünen Meteoriten in beiden Darstellungsarten fast gleich. Weder im 2D- noch im 3D-Raum ist ein signifikanter Unterschied zwischen den Darstellungsarten festzustellen. Auch nach der siebten Welle, welche offenbar einen Grenzwert für die Anzahl der Objekte, die noch sinnvoll vom Benutzer verarbeitet werden können, ändern sich die Werte der beiden Kurven kaum. Womöglich kann ein überlasteter Benutzer die verschiedenen Meteoritenarten kaum unterscheiden und somit werden die grünen Meteoriten mehr oder weniger gleichermaßen abgeschossen bzw. können passieren wie andere Meteoriten, welche wiederum auf dem Planeten einschlagen. Die Durchschnittswerte der erreichen Punkte lag für 2D bei 56,13 und für 3D bei 52,07. Die Spieler erreichten durchschnittlich etwas mehr Punkte, wenn sie im zweidimensionalen Modus spielte.

Im Vergleich kommt es im 3D-Fall zu mehr Einschlägen und weniger Abschüssen von Meteoriten, als im zweidimensionalen. Im Bezug auf die Unterscheidung von grünen und anderen Meteoriten konnte kein erheblicher Unterschied festgestellt werden.

Ein entscheidender Faktor, ob ein Meteorit einschlägt oder nicht, ist auch seine Geschwindigkeit. Die Durchschnittsgeschwindigkeit der abgeschossenen Meteoriten lag bei 35.539543 units/s und die der eingeschlagenen Meteoriten bei 40.09875 units/s, wenn 2D spielt wurde. Im 3D-Fall lag diese bei 34.508602 units/s (abgeschossene) und 38.52657 units/s (eingeschlagene). In Unity 3D kann units/s als m/s verstanden werden, da dies die Standardeinstellung von Unity 3D ist. In beiden Fällen, also egal ob 2D oder 3D ist die Durchschnittsgeschwindigkeit der eingeschlagenen Meteoriten höher, als die der abgeschossenen. Schnellere Objekte sind demnach vom Benutzer schwerer zu erfassen. Zusätzlich legen diese den Weg bis zum Planeten schneller zurück, was zur Verkleinerung des Abschusszeitintervalls führt. Unabhängig davon ob in 2D oder in 3D gespielt wurde - weniger Zeit für einen Abschuss und ein sich schneller bewegendes Objekt machen es dem Benutzer schwerer, solche Meteoriten zu erfassen und somit abzuschießen. Eine räumliche Darstellung mittels 3D-Technik hilft hier offenbar auch nicht weiter.

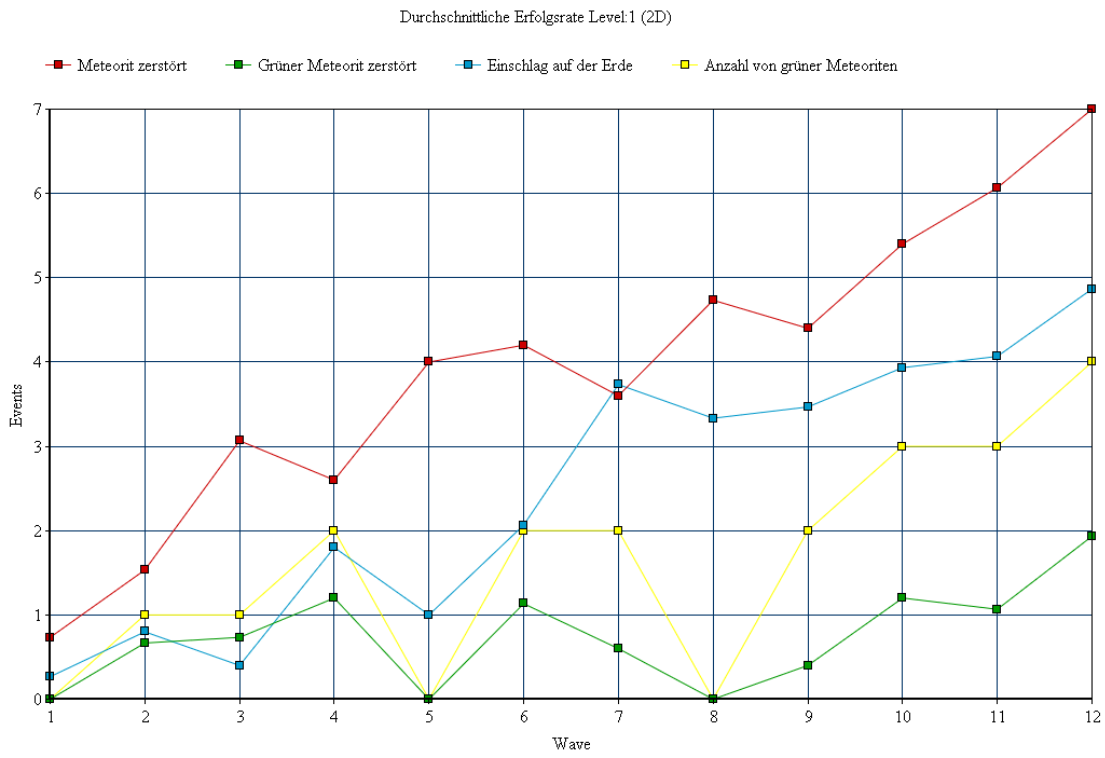


Abbildung 5.8: Erfolgsrate 2D. Die x-Achse gibt die aktuelle Welle an, was auch gleichzeitig die Anzahl der generierten Meteoriten festlegt. Auf der y-Achse werden verschiedene "Events" dargestellt. Je nach Farbe haben diese unterschiedliche Bedeutungen. Rot: Ein Meteorit wurde vom Benutzer abgeschossen. Grün: Ein grüner Meteorit wurde vom Benutzer abgeschossen. Blau: Ein Meteorit ist auf der Erde eingeschlagen. Gelb: Die Anzahl der grünen Meteoriten, welche in der aktuellen Welle generiert wurden. Man erkennt bei Welle 7, dass die Anzahl der Einschläge von Meteoriten die Anzahl der Abschüsse kurz überholt. Danach liegt diese Anzahl aber bis zum Spielende wieder unterhalb der Abschussanzahl.

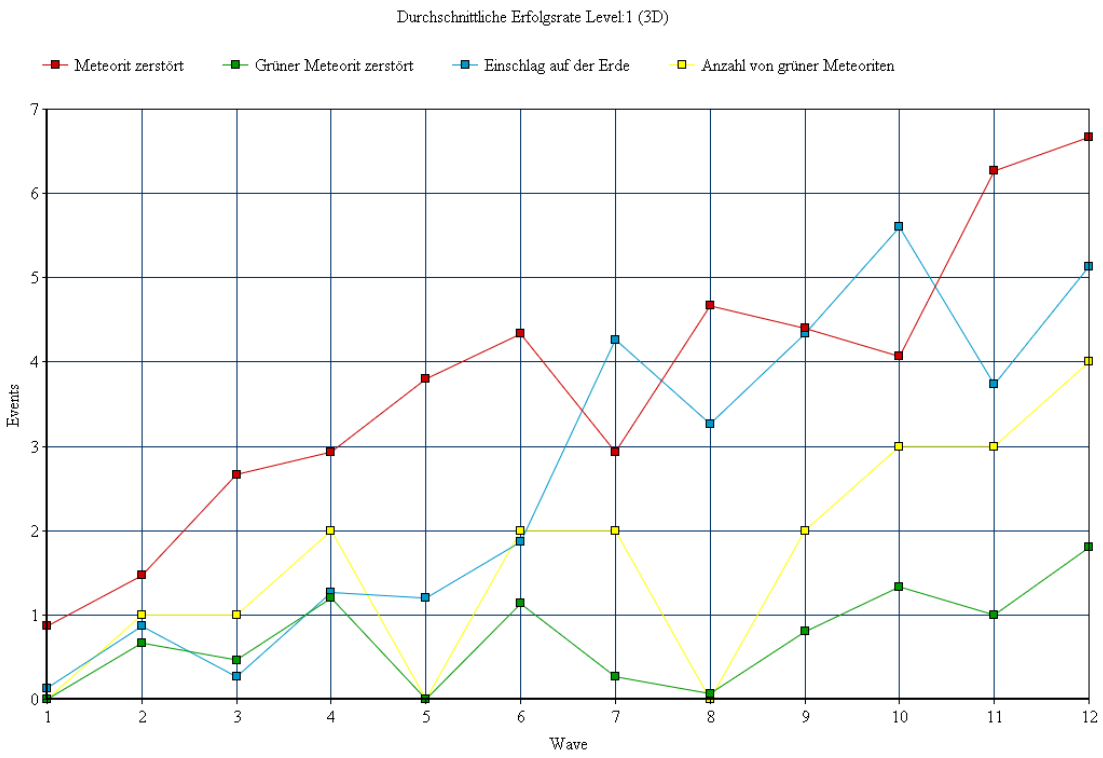


Abbildung 5.9: Erfolgsrate 3D. Diagrammbeschreibung entsprechend Abbildung 5.8. Es fällt auf, dass die grüne Kurve in beiden Diagrammen (5.8 und hier) sehr ähnlich verlaufen. Ab Welle 7 überholen sich die rote und blaue Kurve des öfteren gegenseitig. Ein deutlich sichtbarer Unterschied zum Verlauf der Kurven bei der 2D-Darstellung.

5.2.2 Auswertung der Tiefenerkennungsmethoden

Der Fokus des zweiten Levels lag mehr auf der Evaluierung der Zielberechnungsmethoden, als auf Usability oder Taskload, wie bei Level 1. NASA TLX und SUS kamen hier nicht zum Einsatz. Lediglich der PRC-Fragebogen (jeweils einen pro Berechnungsmethode) sollte von den Probanden ausgefüllt werden. Welche Berechnungsmethode verwendet wurde, wurde den Probanden weder mitgeteilt, noch wussten sie, wie diese funktionieren. Nur die Aufgabe wurde erläutert und das Ziel des Levels definiert. Da im zweiten Level der Fokus auf der Messung der Performanz der unterschiedlichen Zielmethoden lag, wurde hier nur im 3D-Raum gespielt, da bei einer 2D-Darstellung die Tiefenerkennung keinen Sinn ergibt. Auch hier wurden zusätzlich die aufgezeichneten Daten analysiert.

Auswertung der Fragebögen zu den Tiefenerkennungsmethoden

Die Ergebnisse der PRC-Fragebögen sind in Abbildung 5.10 dargestellt. Auch hier variieren einige Begriffe stark. Der Begriff "Fast" (dt. schnell) wurde für die Berechnung mit dem Pupillendurchmesser zwei mal negativ markiert und für die Berechnung mithilfe des Abstandes zwischen den Pupillen zwei mal positiv. Dies lässt grundsätzlich darauf schließen, dass eine der beiden Methoden in ihrer Leistung der anderen überlegen ist, da die Bewertungen sehr gegensätzlich ausgefallen sind.

"Stressful" (dt. stressig, belastend) erhielt in beiden Fällen genau gleich viele Markierungen. Somit kann man sagen, dass die Belastung des Benutzers für die beiden Berechnungsarten nicht sonderlich variierte.

Ein deutlicher Unterschied war bei dem Attribut "Controllable" (dt. kontrollierbar) zu erkennen. Nur eine Markierung bei der Verwendung des Pupillendurchmessers und vier bei der Distanz zwischen den Pupillen wurden erfasst. Dies lässt darauf schießen, dass die Nutzer diese Methode für beherrschbarer hielten, als die andere.

Auch eine hohe Anzahl an Markierungen erhielt - für beide Berechnungsarten - der Begriff "Creative" (dt. kreativ). Dies lässt sich vermutlich mit der Spielwelt in Szene 2 erklären. Die Aufgabe der Teilnehmer, Objekte hinter anderen transparenten Objekten anzuvisieren, war sehr speziell und ungewöhnlich. Die Vermutung liegt nahe, dass sich die Nutzer nicht genau erklären konnten, wie das Prinzip hinter dem Abschussvorgang funktionierte und die Aufgabe dadurch sehr ungewohnt oder nicht erklärbar erschien. Allgemein gab es Rückmeldungen der Teilnehmer, dass das Spielen per Eyetracking auf einem 3D-Display eine sehr ungewöhnliche Angelegenheit ist. Das Gesamtsystem wurde somit als sehr kreativ und neuartig bewertet. Dabei kam es weniger auf die verwendete Mathematik an, als auf die optischen Eindrücke und die Steuerung lediglich mit den Augen und einem Linksklick.

Ebenfalls eine hohe Anzahl an Markierungen erhielt der Begriff "Fun" (dt. Spaß). Vier mal wurde er für die Pupillendurchmessermethode angegeben und sechs mal für die Pupillendistanz. Zwar gibt es Unterschiede bezüglich der Anzahl, doch die Probanden hatten in beiden Fällen offensichtlich Spaß an der Aufgabe.

"Exciting" und "Entertaining" (dt. aufregend, spannend und unterhaltend) erhielten jeweils fünf Markierungen für die Pupillendurchmessermethode. Die Pupillendistanzmethode erhielt vier Markierungen bei "Exciting" und drei bei "Entertaining". Fünf Markierungen ist die zweithöchste Anzahl an Markierungen im gesamten Fragebogen. Daraus lässt sich ableiten, dass die Beispielapplikation und die gestellte Aufgabe einen relativ guten Unterhaltungswert liefern und somit eventuell die Aufmerksamkeit von Personen erregen könnten.

Ansonsten erhielten die restlichen Begriffe im Schnitt weniger als zwei Markierungen und wiesen keine signifikanten Unterschiede bezüglich der verwendeten Technik auf.

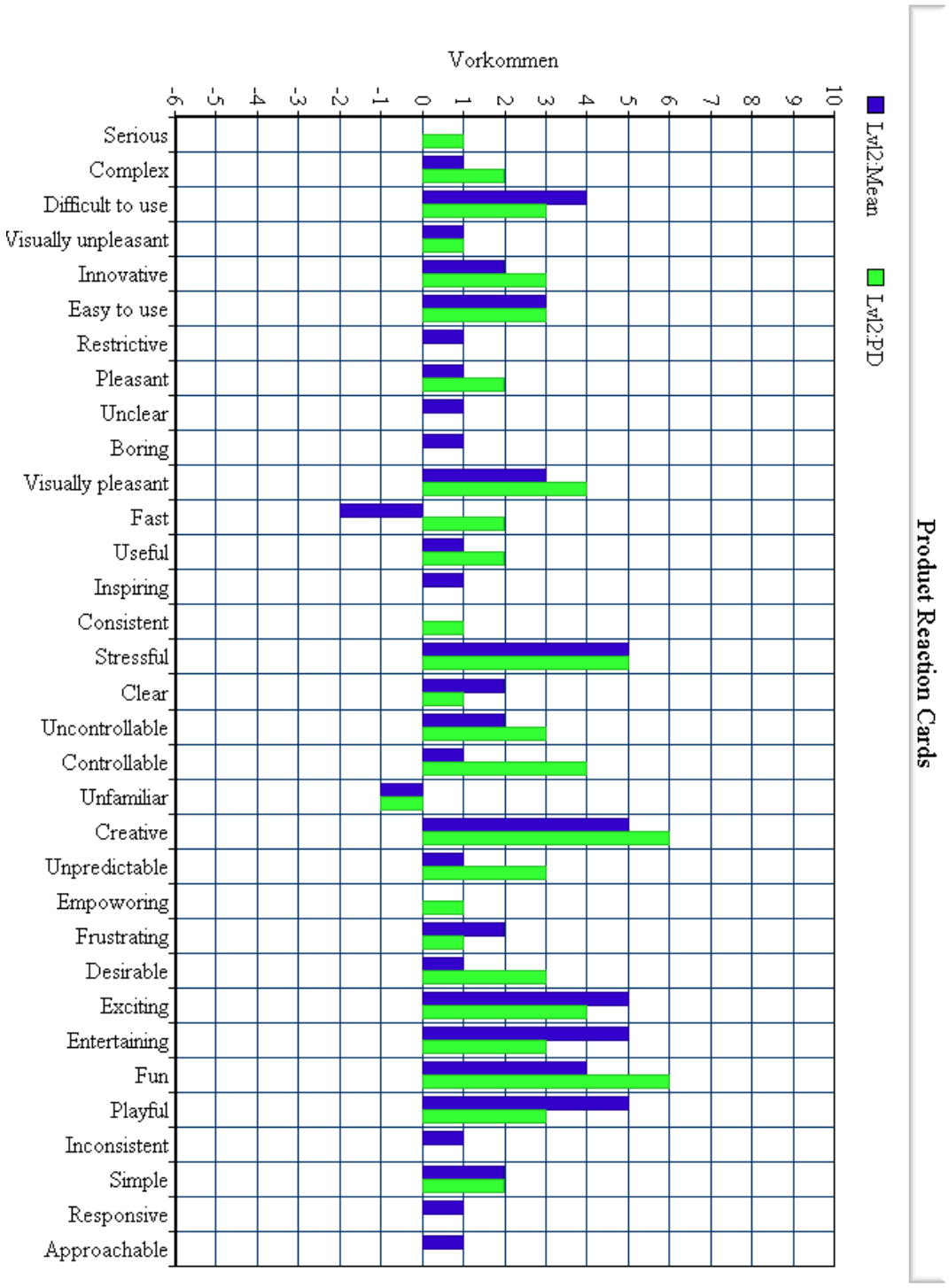


Abbildung 5.10: PRC Ergebnisse für den zweiten Level. Grün dargestellt die Vorkommen der Attribute, wenn die Distanz zwischen linker und rechter Pupille verwendet wurde und blau, wenn der Pupillendurchmesser bei der Zielbestimmung verwendet wurde.

Auswertung der aufgezeichneten Daten zur Tiefenerkennung

Wie in Abbildung 5.11 erkennbar, kommt es offensichtlich zu unterschiedlichen Durchschnittswerten des Pupillendurchmessers, wenn ein Nutzer Objekte unterschiedlicher Tiefe anvisiert. Die Werte für die unterschiedlichen Ebenen unterscheiden sich so stark, dass es möglich sein sollte, über den Pupillendurchmesser zu erkennen, ob ein Benutzer ein Objekt bestimmter Tiefe anschaut.

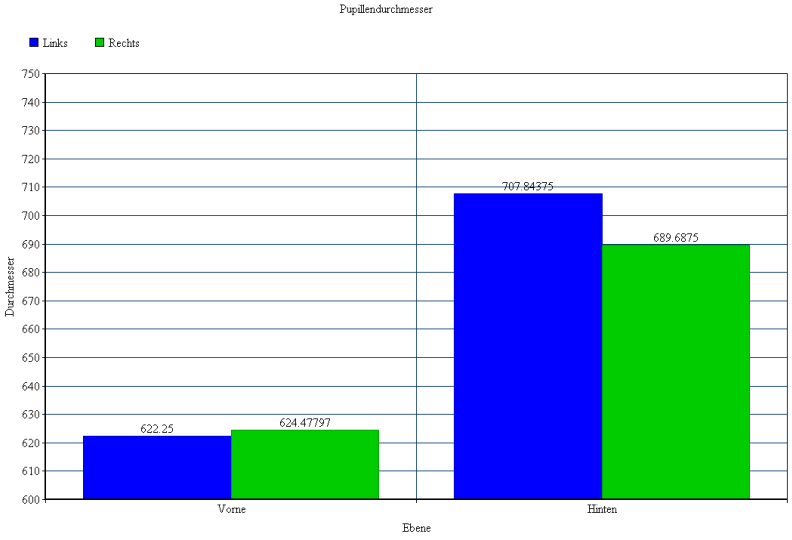


Abbildung 5.11: Durchschnittswerte der Pupillendurchmesser für Abschüsse auf vorderer Ebene (links) und auf hinterer Ebene (rechts). Der Pupillendurchmesser war im Durchschnitt größer, wenn ein Objekt auf der hinteren Ebene abgeschossen wurde.

Da in der Beispielanwendung lediglich zwei verschieden tiefe Ebenen getestet wurden, könnte man diesen Versuch auf n Ebenen erweitern, um die Genauigkeit zu bestimmen, mit welcher es möglich ist, richtige Aussagen bezüglich der Tiefe eines Objekts zu machen, auch wenn die Tiefe kontinuierlich variiert.

Grundsätzlich kann man hier zumindest festhalten, dass es zu einer Variation des Pupillendurchmessers kommt, fokussiert ein Benutzer Objekte mit unterschiedlichen Tiefenwerten.

Die zweite Methode, welche die Distanz zwischen linker zu rechter Pupille verwendet, wies einige Unstimmigkeiten auf. Während der Studie wurde von den Probanden bemerkt, dass auf einer bestimmten Bildschirmseite immer das Objekt auf der vorderen Ebene und auf der anderen Bildschirmseite immer das Objekt auf der hinteren Ebene abgeschossen wurde. Die genaue Ursache für dieses Phänomen konnte nicht bestimmt werden. Es ist aber davon

Pupillendurchmessermethode	Pupillendistanzmethode
2,28	7,5

Tabelle 5.2: Durchschnittliche Abschussrate der Raumschiffe in Level 2.

auszugehen, dass entweder Licht- oder andere Umwelteinflüsse als Störfaktoren in Frage kommen oder diese Methode einen starken Lernprozess voraussetzt.

Die zweite Methode konnte unter den Umständen der Benutzerstudie also nicht zuverlässig zur Tiefenerkennung genutzt werden. Zwar konnten unter Verwendung dieser Methode mehr Raumschiffe abgeschossen werden, wie man in Tabelle 5.2 sehen kann. Die Diskrepanz der Werte ist aber darauf zurückzuführen, dass es dem Nutzer möglich war auf einer speziellen Bildschirmseite immer das Raumschiff zu treffen und dieses somit zuverlässig abschießen zu können.

5.3 Ergebnisse und Diskussion

Die Ergebnisse für die unterschiedlichen Bereiche, die mithilfe der Beispielapplikation evaluiert wurden, sind nach Level aufgeteilt und werden einzeln besprochen.

5.3.1 Ergebnisse des ersten Levels

Im Bereich Usability erzielte die Beispielapplikation ein gutes Ergebnis (siehe Ergebnisse in Abbildung 5.5). Zwar verbesserte sich die Usability nicht, wenn die 3D-Darstellung zum Einsatz kam, aber sie wurde auch nicht schlechter. Insgesamt kann man behaupten, dass die dreidimensionale Darstellung keine Einbußen im Bereich Usability herbeiführen muss. Auch bei der Anforderung an den Benutzer konnte nicht festgestellt werden, dass durch dreidimensionalen Output die Taskload deutlich erhöht wird (siehe Ergebnisse in Abbildung 5.6). Aufgaben, welche mit 2D- oder 3D-Darstellung gelöst werden können, müssen nicht zwangsläufig in einer der beiden Anzeigetechniken schwieriger zu bewältigen sein.

Signifikante Unterschiede waren lediglich bei der Erfolgsrate zu erkennen (siehe Abbildung 5.8 und Abbildung 5.9). Hier leidet die Anzahl der Abschusserfolge ab der siebten Welle bei der dreidimensionalen Darstellung stärker, als bei der zweidimensionalen. Möglicherweise überfordert hier die 3D-Darstellung der Szene den Spielenden mehr, als bei der herkömmlichen zweidimensionalen Darstellungsart. Da sich ab der siebten Welle sehr viele Objekte in der Szene befinden, ist der Benutzer an sich schon etwas überlastet. Kommt nun noch der 3D-Effekt hinzu, könnte dies zu einer weiteren Überforderung des Benutzers führen, da noch mehr Eindrücke auf diesen zukommen.

Schlussendlich kann gesagt werden, dass eine 3D-Darstellung einer dreidimensionalen Spielwelt optisch sehr gut zur Geltung kommen kann, ohne die Usability einzuschränken oder die Taskload zu erhöhen. Es sollte aber dennoch darauf geachtet werden, den Benutzer

nicht mit Eindrücken zu überfluten, sodass die zu bewältigende Aufgabe nicht mehr (gut) erfüllbar ist.

5.3.2 Freitexte zum Vergleich 2D/3D

Einige Studienteilnehmer gaben Feedback in Form von Freitexten. Die folgenden Aussagen beziehen sich auf Level 1.

Teilnehmer mit Sehhilfe (Brille) klagten über ungenaue Blickverfolgung. Es wurde angenommen, dass die Brille in Kombination mit der Polarisationsbrille einen Störfaktor für den Eyetracker darstellt.

Zusätzlich war es für die Teilnehmer schwer ordnungsgemäß zu spielen, wenn die Anzahl der auf die Erde zufliegender Meteoriten zu groß wurde. Durch diese hohe Anzahl an Objekten, war es dem Spieler nicht mehr möglich, sich auf ein spezielles Objekt zu konzentrieren. Die meisten Teilnehmer konnten dann nicht mehr genau zielen und versuchten durch wildes Umschauen, die verbleibenden Meteoriten abzuschießen. Neben der großen Objektanzahl wurde die Aufgabe zwischen den Meteoritentypen zu unterscheiden als schwer eingestuft. Bei einer zu großen Anzahl an Meteoriten seien die grünen kaum noch von den anderen zu unterscheiden.

Des weiteren gab es Rückmeldungen über die Blickverfolgung an sich. Das genaue Anvisieren eines Punktes wurde als schwer bemängelt, während große Blickänderungen (beispielsweise Blick von linker Seite des Displays zur rechten) aus Sicht der Probanden vom System sehr gut erkannt wurden. Dies lässt sich auf die von Grund auf nicht perfekte Technik zurückführen. Auch durch die 3D-Brille wird das Eyetrackingsystem gestört, denn Reflexionen auf der Brille und ihre Tönung können die Fehlerrate des Eyetrackingsystems erhöhen.

Ein Nutzer gab an, dass er durch weites Öffnen der Augen eine bessere Kontrolle über das System erlangte. Durch weit geöffnete Augen ist das System eher in der Lage, das Auge des Probanden richtig zu erfassen und die Blickrichtung kann somit weniger fehlerhaft berechnet werden. Augen, die nicht weit genug geöffnet sind, werden nicht mehr genau erfasst und im schlimmsten Fall kann das Eyetrackingsystem keine Daten mehr liefern.

Da die Kalibrierung des Systems sehr wichtig ist und möglichst gut ausfallen sollte (was nicht immer der Fall ist), kam es bei manchen Benutzern zu genaueren Ergebnissen bei der Blickberechnung, bei manchen arbeitete das System unpräzise, was einigen Benutzern das Spielen erschwerte. Es muss also für jeden Benutzer ein Optimum gefunden werden, in dem die einzelnen Faktoren (Abstand zum Eyetracker, Höhe der Kopfstütze, Winkel des Eyetrackers) stimmen, um eine fehlerfreie Blickverfolgung zu ermöglichen. War dies nicht der Fall, arbeitete das System unpräzise und der Benutzer verlor das Gefühl der Korntrolle über das System.

Manche Benutzer empfanden die Aufgabe im ersten Level als sehr anstrengend. Ein Proband klagte über leichte Kopfschmerzen nach der Studie.

5.3.3 Ergebnisse des zweiten Levels

Durch die Erkenntnisse, welche durch die Evaluierung des zweiten Levels gewonnen wurden, sollte die Performanz der unterschiedlichen Berechnungsarten festgestellt werden. Deutliche Ergebnisse, wie zum Beispiel die Überlegenheit einer Methode gegenüber der anderen, konnten nicht ermittelt werden. Bei einzelnen Probanden versagten die Methoden gänzlich. Die Vermutung liegt nahe, dass ein gewisser Lernprozess durchlaufen werden muss, ehe der Benutzer in der Lage ist, die gestellte Aufgabe in Level 2 ordnungsgemäß zu bewältigen. Zusätzlich könnten Umweltfaktoren negativ zur Evaluierung der beiden Methoden beigetragen haben. Beobachtungen während der Studie und auch bei der Entwicklung der Beispielapplikation legen nahe, dass solche Störfaktoren eine sehr große Rolle bezüglich der korrekten Funktion der Berechnungsarten spielen. Solche Störgrößen könnten verantwortlich sein für die Ergebnisse bei der Verwendung der Pupillendistanz. Dieser Lösungsansatz könnte mit Übung und unter besseren Bedingungen möglicherweise zu besseren Ergebnissen führen.

Die Methode, welche sich als Auswertungsgrundlage auf den Pupillendurchmesser bezog, lieferte ein Ergebnis, welches auf eine bessere Kontrolle schießen lies, als bei der Pupillendistanzmethode. Hier war die Position auf dem Bildschirm nicht ausschlaggebender Faktor für die Tiefenbestimmung. Dennoch liegt der Verdacht nahe, dass ein Lernprozess nötig wäre, um die Aufgabe besser lösen zu können und eine bessere Kontrolle über die Anwendung zu erlangen.

Abschließend kann behauptet werden, dass sich beide Methoden unter bestimmten Voraussetzungen zur Tiefenerkennung eignen können - insbesondere die Pupillendurchmessermethode. Eventuell könnte die andere Methode in einem anderen Kontext unter anderen Bedingungen eine bessere Leistung erzielen.

5.3.4 Freitexte zu den Tiefenerkennungsmethoden

Für den zweiten Level gaben die Nutzer ebenfalls Feedback in Freitextform.

Manche Benutzer gaben an, Unterschiede bezüglich der Performanz der einzelnen Zielmethoden festgestellt zu haben. Hier wurde angegeben, dass die Methode, welche mit der Distanz zwischen linker und rechter Pupille arbeitet besser funktionierte, als die Pupillendurchmessermethode. Dies muss jedoch mit Vorsicht betrachtet werden, da die hier als besser bezeichnete Methode einige Probleme aufwies. Diese Methode führte teilweise dazu, das auf der linken Seite des 3D-Displays die Meteoriten vor den Raumschiffen abgeschossen wurden und auf der rechten Seite immer die Raumschiffe. Es muss also Störfaktoren gegeben haben, die die Auswertung dahingehend beeinflussten, dass die Benutzer keine Chance mehr hatten selbst zu entscheiden, was sie abschossen. Hier können Lichtverhältnisse oder Reflexionen auf der 3D-Brille eine große Rolle bezüglich der Performanz spielen.

Des weiteren wurde angegeben, dass ein langes Fokussieren des Raumschiffes nötig war, um es abzuschießen.

5.3.5 Fazit

Die Evaluierung erstreckte sich über viele verschiedene Bereiche der Beispielapplikation. Mit dem ersten Level sollten die Unterschiede von 2D- und 3D-Darstellung im Bezug auf Usability und Taskload untersucht werden. Der zweite Level sollte die Tiefenerkennung mittels Eyetracking und die hier entwickelten verschiedenen Ansätze und ihre Performanz durchleuchten.

Insgesamt gesehen eignet sich Eyetracking (wenn auch nicht immer optimal) für die Steuerung eines 3D-Spiels. Die Usabilty litt nicht unter der dreidimensionalen Darstellung (SUS). Auch die Schwere der Aufgaben erhöhte sich nicht signifikant (TLX). Dennoch gibt es Unterschiede im Bezug auf die Erfolgsrate (siehe Abbildung 5.8 und 5.9). Wie zuvor schon angesprochen, könnte im Falle der 3D-Darstellung die Anzahl der Eindrücke für den Benutzer steigen und ihn dadurch überfordern. Dies wurde auch von den Probanden in Freitexten erwähnt.

Im Bereich der Tiefenerkennung waren die Ergebnisse nicht so eindeutig. Zwar konnten teilweise Erfolge verzeichnet werden, aber durch die kurze Spielphase und den dadurch nur begrenzt stattgefundenen Lernprozess war die Aufgabe des zweiten Levels nur bedingt machbar. Trotz allem liegt es nahe, dass beide Methoden für eine Tiefenbestimmung genutzt werden können - insbesondere die Verwendung des Pupillendurchmessers scheint hierfür erfolgversprechend.

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

In dieser Arbeit wurden die Entwicklung einer 3D-Beispielapplikation mithilfe einer 3D-Engine und eines Eyetrackingsystems beschrieben und die im Rahmen der Entwicklung implementierten Auswertungsmethoden evaluiert. In Kapitel 4 wurde die Implementierung der Beispielapplikation und des dazugehörigen Kommunikationsservers beschrieben. Das zu Grunde liegende Konzept der genutzten 3D-Engine wurde verdeutlicht und die Prinzipien des blickbasierten Steuerns der Applikation wurden erläutert. Um die Performanz der implementierten Techniken zu evaluieren, wurde eine Benutzerstudie durchgeführt, in welcher Usability, Taskload und Blickdaten erfasst und ausgewertet wurden. Interessant war hier, dass beispielsweise im Bereich Usability im Rahmen der Beispielapplikation weder bei der 2D- noch bei der 3D-Darstellung Einbußen verzeichnet werden konnten. Auch die Taskload wurde nicht signifikant von der Darstellungsart beeinflusst. Diese Erkenntnisse lassen den Schluss zu, dass möglicherweise zukünftige 3D-Interfaces im Bereich Usability und Taskload 2D-Interfaces in nichts nachstehen werden. Für die meisten Probanden der Benutzerstudie war die blickbasierte Interaktion neuartig und die 3D-Welt optisch ansprechend. Derartige Applikationen könnten durch ihre Optik und die unkonventionelle Eingabeart großes Interesse wecken.

Die verschiedenen Auswerungsmethoden zur Tiefenerkennung in 3D-Räumen, die im Kapitel 4 beschrieben und in Kapitel 5 untersucht wurden, ergaben unterschiedliche Ergebnisse. Besonders die Methode, welche die Distanz zwischen linker und rechter Pupille als Berechnungsgrundlage nutzte, wies einige Ungereimtheiten auf - anders bei der Pupillendurchmessermethode. Diese wies klar erkennbare Unterscheidungskriterien bei der Evaluierung in Kapitel 5 auf. Möglicherweise kann auf Grundlage von Pupillendurchmessern eine auf n-Ebenen skalierbare Tiefenerkennung entwickelt werden, welche eine dynamischere Auswahl und Manipulation von Objekten in 3D-Umgebungen zulässt. Doch auch der Ansatz unter Verwendung der Pupillendistanz sollte nicht verworfen werden. Es konnte durch die Distanz zwischen den Pupillen, wenn auch nicht zuverlässig, eine Tiefenerkennung realisiert werden. Durch Elimination etwaiger Störfaktoren lässt sich diese Methode eventuell verbessern und in zukünftigen Systemen einsetzen.

Blickbasierte Interaktion mit 3D-Displays bietet eine Breite Palette an Möglichkeiten für zukünftige Technologien. Durch die Weiterentwicklung von Spielekonsolen und anderen Homeentertainmentsystemen bestünde eventuell die Möglichkeit, dass diese Art der Interaktion bald auch in Privathaushalten Einzug findet.

Somit ist die blickbasierte Interaktion ein sehr aktuelles und wichtiges Forschungsgebiet, das in Zukunft stark an Bedeutung gewinnen wird.

Ausblick

Blickbasierte Interaktion auf 3D-Displays ist ein Gebiet, welches sich aus zwei grundlegenden, schon länger bestehenden Bereichen entwickelt hat. Die schon sehr ausgereiften 3D-Techniken kombiniert mit der Blickverfolgung und -analyse können im Bereich Usability punkten. Diese stellen für den Menschen (Aussage vieler Probanden) eine neuartige Interaktionsmöglichkeit dar. Ein blickbasiertes Manipulieren eines dreidimensionalen Raums könnte, wenn vollständig ausgereift, in vielen Bereichen eingesetzt werden. Wenn Sachverhalte aus der Realität dreidimensional im Rechner abgebildet werden könnten, könnte auch Eyetracking herangezogen werden, um dem Benutzer einer 3D-Applikation ein zusätzliches Eingabeinstrument zur Verfügung zu stellen. Dies könnte in unterschiedlichsten Bereichen geschehen. Im Auto [Poi11], in der Computerspieleindustrie [LHC+04] oder im medizinischen Bereich (bei Operationen) [SVH+02].

Blickbasierte Interaktion auf 3D-Displays ist ein noch nicht sehr weit erforschtes Teilgebiet der Mensch-Computer-Interaktion. Durch immer neue und bessere Techniken können derartige Interaktionsmöglichkeiten weiter entwickelt und somit eventuell neue Erkenntnisse und Anwendungsbereiche gefunden werden. Unter Verwendung anderer, schon bestehender Systeme, wie beispielsweise optische Verfolgungssysteme, welche den Menschen im Raum erfassen können, kann eventuell auch die blickbasierte Interaktion noch weiter verfeinert werden. Die Integration von derartigen Techniken und die Verbesserung der schon bestehenden Methoden zur Blickauswertung können als Aufgaben für zukünftige Systeme angesehen werden. Solche Herausforderungen könnten folgendermaßen aussehen.

6.1.1 Herausforderungen

Im Bereich der blickbasierten Interaktion auf 3D-Displays gibt es viele Ideen, welche die hier besprochenen Verfahren noch weiter verbessern könnten. Im Folgenden werden einige Ansätze und Ideen diskutiert, die Potential für eine Verbesserung der Interaktionsmöglichkeiten haben könnten.

Autostereoskopische Displays

Die 3D-Brille, welche bei herkömmlichen 3D-Displays zum Einsatz kommt, behindert das Eyetracking stark. Aufgrund der Tönung einer Polatisationsbrille und Reflexionen auf den Brillengläsern wird der Eyetracker gestört und kann nicht mehr einwandfrei arbeiten. Durch die Verwendung von autostereoskopischen Displays [Dod05] könnte diesem Problem entgegen gewirkt werden. Ohne Brille kann der Eyetracker (wie bei der üblicher Verwendung) die Augen besser erfassen und ein besseres Ergebnis liefern. Das Gesamtsystem muss dann

so aufgebaut werden, dass der Proband sich mit dem Kopf auf der Kopfstütze genau im sogenannten "Sweetspot" (der Punkt, an dem der 3D-Effekt eines autostereoskopischen Displays am besten ist) befindet.

Dieser Punkt - der "Sweetspot" - könnte auch per Eyetracking lokalisiert werden, wodurch ein genaues "senden" des 3D-Bildes an einen bestimmten Betrachter möglich wäre. Dadurch könnten auch mehrere Betrachter gleichzeitig unterstützt werden. Um dies zu realisieren, muss ein System in der Lage sein, mehr als einen Betrachter zu erfassen [HNP11].

Headtracking

Die Kalibrierung des Eyetrackingsystems kann noch so perfekt sein, wenn sich ein Proband bewegt, ist ein genaues Tracking und die Bestimmung eines Punktes auf dem Bildschirm, welcher vom Nutzer angeschaut wird, sehr fehlerbehaftet. Dieses Problem könnte durch die Bestimmung der Kopfposition durch ein Headtrackingsystem [LCSA00] gelöst werden. Wenn bekannt ist wo sich der Kopf eines Benutzers relativ zum Eyetracker/Bildschirm befindet, wäre eine genauere Blickbestimmung möglich, selbst wenn der Benutzer sich bewegt.

Um den Komfort des Eyetrackings zu erhöhen, wäre ein kalibrierungsfreies System vorteilhaft. Dies würde den Nutzer entlasten und den Einsatz einer solchen Technologie vereinfachen [SWL00].

Mobile Endgeräte

Der Markt der mobilen Geräte und zugleich deren Leistung steigt rapide an [JYH11]. Autosterioskopische Smartphones sind schon auf dem Markt verfügbar und stellen somit die Grundlage für die blickbasierte Interaktion mit 3D-Inhalten auf mobilen Endgeräten dar. Blickbasierte Interaktion mit Smartphones kann möglich sein, ist aber noch mit einigen technischen Problemstellungen behaftet [DDLS07].

Danksagung

Ich bedanke mich bei Florian Alt und Stefan Schneegaß für das Vertrauen und die Hilfe, die nötig war, um diese Arbeit zu bewältigen. Außerdem bedanke ich mich bei Rufat Rzayev für die tolle Zusammenarbeit. Ohne einen Partner wäre das Bewältigen dieser Aufgabe nicht möglich gewesen. Ich bedanke mich bei allen Studienteilnehmern für ihre Teilnahme, obwohl es keine Belohnung dafür gab und bei allen Leuten die meine Arbeit auf Fehler korrigiert haben. Insbesondere bei Anna-Lena Janz und bei meiner Mutter Susanne Wendt. Vielen Dank!

A Ein Anhang

A.1 Verwendete Materialien

Die für diese Arbeit verwendeten Materialien wie die für die Evaluierung verwendeten Fragebögen sind in diesem Abschnitt zu finden.

A.1.1 Fragebögen

Auf den folgenden Seiten sind alle in der Studie verwendeten Fragebögen enthalten.

Fragebögen

1. Demographie und PCR
2. System Usability Scale (SUS)
3. NASA TLX

Interaction with 3D Content

(August/September 2013)

Universität Stuttgart

Fragebogen

A. Demographie

Geschlecht: o männlich o weiblich

Alter: _____ Beruf / Studiengang: _____

Besitzt du einen 3D-Fernseher?

Wenn ja, wie oft benutzt du die 3D-Funktionalität?

o täglich
o wöchentlich
o monatlich
o seltener / nie

B. User Experience

Choose <u>five</u> (5) words from the grid below which best describe your experience of using the system. If the word is followed by the sign (+/-), please mark whether you mean the word in a positive or negative sense.

Fast (+/-)	**Useless**	**Useful**	**Inspiring**
Rigid	**Consistent**	**Too technical**	**Stressful**
Clear	**Uncontrollable**	**Businesslike (+/-)**	**Controllable**
Unfamiliar (+/-)	**Creative**	**Undesirable**	**Predictable (+/-)**
Time-saving	**Unpredictable**	**Familiar (+/-)**	**Time-consuming**
Dated	**Empowering**	**Slow (+/-)**	**Frustrating**
Desirable	**Unpleasant**	**Exciting**	**Entertaining**
Dull	**Fun (+/-)**	**Playful**	**Ordinary**
Unapproachable	**Inconsistent**	**Simple (+/-)**	**Restful**
Responsive	**Approachable**	**Serious**	**Complex**
Difficult to use	**Visually unpleasant**	**Innovative**	**Easy to use**
Restrictive	**Pleasant**	**Unclear**	**Boring**
Visually pleasant	**Poor quality**	**Novel**	**High quality**

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

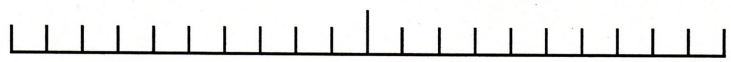
Figure 8.6

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

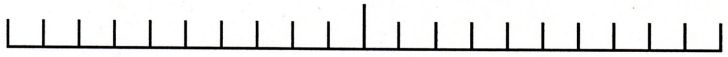
Name	Task	Date

Mental Demand — How mentally demanding was the task?



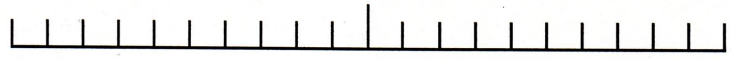
Very Low — Very High

Physical Demand — How physically demanding was the task?



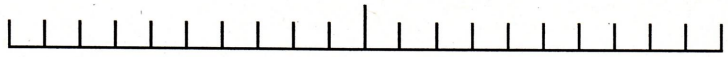
Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?



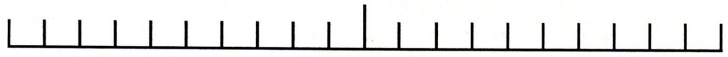
Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?



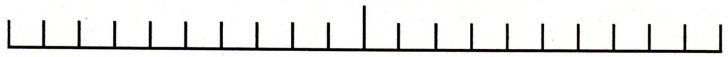
Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?



Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?



Very Low — Very High

Literaturverzeichnis

[AVRM11] S. Almeida, A. Veloso, L. Roque, Ó. Mealha. The Eyes and Games: A Survey of Visual Attention and Eye Tracking Input in Video Games. *Proceedings of SBGames*, 2011. (Zitiert auf Seite 14)

[BM02] J. Benedek, T. Miner. Measuring Desirability: New Methods for Evaluating Desirability in a Usability Lab Setting. In *Proceedings of UPA 2002 Conference*. 2002. (Zitiert auf Seite 42)

[Bro96] J. Brooke. SUS: A quick and dirty usability scale. In P. Jordan, B. Thomas, B. Weerdmeester, I. McClelland, Herausgeber, *Usability evaluation in industry*, S. 189–194. Taylor & Francis, London, 1996. (Zitiert auf Seite 42)

[Dah06] M. Dahm. *Grundlagen der Mensch-computer-interaktion*. Pearson Studium, 2006. (Zitiert auf Seite 21)

[DBE+10] O. Deussen, H. Bülthoff, T. Ertl, D. Keim, B. Lintermann, H. Reiterer, A. Schilling. Visualisierung auf Großbildschirmen. *Informatik-Spektrum*, 33(6):551–558, 2010. (Zitiert auf Seite 9)

[DBMB07] M. Dorr, M. Böhme, T. Martinetz, E. Barth. Gaze beats mouse: a case study. In *Proceedings of the 2nd Conference on Communication by Gaze Interaction*, S. 16–19. 2007. (Zitiert auf Seite 15)

[DDLS07] H. Drewes, A. De Luca, A. Schmidt. Eye-gaze interaction for mobile phones. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, S. 364–371. ACM, 2007. (Zitiert auf Seite 61)

[DLK12] F. Daiber, L. Li, A. Krüger. Designing gestures for mobile 3D gaming. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, S. 3:1–3:4. ACM, New York, NY, USA, 2012. doi:10.1145/2406367.2406371. URL http://doi.acm.org/10.1145/2406367.2406371. (Zitiert auf Seite 17)

[Dod05] N. A. Dodgson. Autostereoscopic 3D displays. *Computer*, 38(8):31–36, 2005. (Zitiert auf den Seiten 20 und 60)

[DPHW11] A. T. Duchowski, B. Pelfrey, D. H. House, R. Wang. Measuring gaze depth with an eye tracker during stereoscopic display. In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, APGV

'11, S. 15–22. ACM, New York, NY, USA, 2011. doi:10.1145/2077451.2077454. URL http://doi.acm.org/10.1145/2077451.2077454. (Zitiert auf Seite 20)

[DSIo9] A. De Santis, D. Iacoviello. Robust real time eye tracking for computer interface for disabled people. *Computer methods and programs in biomedicine*, 96(1):1–11, 2009. (Zitiert auf Seite 10)

[FMO12] K. Funes Mora, J.-M. Odobez. Gaze estimation from multimodal Kinect data. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, S. 25–30. IEEE, 2012. (Zitiert auf Seite 15)

[GHM99] L. Goulden, J. A. Herman, P. S. Moore. Remote with 3D organized GUI for a home entertainment system, 1999. US Patent 5,956,025. (Zitiert auf Seite 16)

[GJG04] L. A. Granka, T. Joachims, G. Gay. Eye-tracking analysis of user behavior in WWW search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, S. 478–479. ACM, 2004. (Zitiert auf Seite 13)

[GWB04] T. Grossman, D. Wigdor, R. Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, S. 61–70. ACM, New York, NY, USA, 2004. doi:10.1145/1029632.1029644. URL http://doi.acm.org/10.1145/1029632.1029644. (Zitiert auf Seite 18)

[HKB+12] P. Hoske, G. Kunze, K. Bürkle, M. Schmauder, M. Brütting, C. Böser. Interaktiver Simulator für mobile Arbeitsmaschinen-Virtuelle Prototypen im Einsatzkontext erleben. In *Konferenz Methoden und Werkzeuge in Produktentwicklung und Design, Dresden*. 2012. (Zitiert auf Seite 9)

[HNP11] K. Hopf, F. Neumann, D. Przewozny. Multi-user eye tracking suitable for 3D display applications. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, S. 1–4. IEEE, 2011. (Zitiert auf Seite 61)

[HO11] D. Hartescu, A. Oikonomou. Gaze tracking as a game input interface. In *Computer Games (CGAMES), 2011 16th International Conference on*, S. 126–133. IEEE, 2011. (Zitiert auf Seite 14)

[IM06] P. Isokoski, B. Martin. Eye tracker input in first person shooter games. In *Proceedings of the 2nd Conference on Communication by Gaze Interaction: Communication by Gaze Interaction-COGAIN 2006: Gazing into the Future*, S. 78–81. 2006. (Zitiert auf Seite 13)

[JNR08] S. A. Johansen, M. Noergaard, J. Rau. Can eye tracking boost usability evaluation of computer games. In *Proceedings of CHI*, Band 8. 2008. (Zitiert auf Seite 13)

[Jön05] E. Jönsson. If looks could kill–an evaluation of eye tracking in computer games. *Unpublished Master's Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden*, 2005. (Zitiert auf Seite 10)

[JYH11] J. Jie, K. Yang, S. Haihui. Research on the 3D game scene optimization of mobile phone based on the Unity 3D engine. In *Computational and Information Sciences (ICCIS), 2011 International Conference on*, S. 875–877. IEEE, 2011. (Zitiert auf Seite 61)

[KJK+06] Y.-M. Kwon, K.-W. Jeon, J. Ki, Q. M. Shahab, S. Jo, S.-K. Kim. 3D Gaze Estimation and Interaction to Stereo Dispaly. *IJVR*, 5(3):41–45, 2006. (Zitiert auf Seite 19)

[LCSA00] M. La Cascia, S. Sclaroff, V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(4):322–336, 2000. (Zitiert auf Seite 61)

[LHC+04] C.-S. Lin, C.-C. Huan, C.-N. Chan, M.-S. Yeh, C.-C. Chiu. Design of a computer game using an eye-tracking device for eye's activity rehabilitation. *Optics and Lasers in Engineering*, 42(1):91–108, 2004. (Zitiert auf Seite 60)

[LM04] J. Leyba, J. Malcolm. Eye tracking as an aiming device in a computer game. *Course work (CPSC 412/612 Eye Tracking Methodology and Applications by A. Duchowski), Clemson University*, 2004. (Zitiert auf Seite 14)

[LMW+11] T. Leyvand, C. Meekhof, Y.-C. Wei, J. Sun, B. Guo. Kinect identity: Technology and experience. *Computer*, 44(4):94–96, 2011. (Zitiert auf Seite 15)

[MCP+12] W. A. Mattingly, D.-j. Chang, R. Paris, N. Smith, J. Blevins, M. Ouyang. Robot design using Unity for computer games and robotic simulations. In *Computer Games (CGAMES), 2012 17th International Conference on*, S. 56–59. IEEE, 2012. (Zitiert auf Seite 23)

[nas] *NASA Task Load Index (NASA-TLX): A paper and pencil package, Version 1.0*. Human Performance Research Group, NASA Ames Research Center, Moffett Field, CA. (Zitiert auf Seite 42)

[OMY02] T. Ohno, N. Mukawa, A. Yoshikawa. FreeGaze: a gaze tracking system for everyday gaze interaction. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, ETRA '02, S. 125–132. ACM, New York, NY, USA, 2002. doi:10.1145/507072.507098. URL http://doi.acm.org/10.1145/507072.507098. (Zitiert auf Seite 21)

[PB06] A. Poole, L. J. Ball. Eye tracking in HCI and usability research. *Encyclopedia of human computer interaction*, S. 211–219, 2006. (Zitiert auf Seite 22)

[Poi11] T. M. Poitschke. *Blickbasierte Mensch-Maschine Interaktion im Automobil*. Dissertation, München, Technische Universität München, Diss., 2011, 2011. (Zitiert auf Seite 60)

[PPK00] K. Perlin, S. Paxia, J. S. Kollin. An autostereoscopic display. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, S. 319–326. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. doi:10.1145/344779.344933. (Zitiert auf Seite 10)

[RHFL10] S. Reichelt, R. Häussler, G. Fütterer, N. Leister. Depth cues in human visual perception and their realization in 3D displays. In *SPIE Defense, Security, and Sensing*, S. 76900B–76900B. International Society for Optics and Photonics, 2010. (Zitiert auf Seite 20)

[RW] M. Ruppert, J. Wörler. Aktuelle Entwicklungen der Mensch-Computer-Schnittstelle: eine Chance für die Raumgeometrie! (Zitiert auf den Seiten 22 und 23)

[SBG12] D. Slambekova, R. Bailey, J. Geigel. Gaze and gesture based object manipulation in virtual worlds. In *Proceedings of the 18th ACM symposium on Virtual reality software and technology*, VRST '12, S. 203–204. ACM, New York, NY, USA, 2012. doi:10.1145/2407336.2407380. URL http://doi.acm.org/10.1145/2407336.2407380. (Zitiert auf Seite 15)

[SD12] S. Stellmach, R. Dachselt. Look & touch: gaze-supported target acquisition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, S. 2981–2990. ACM, New York, NY, USA, 2012. doi:10.1145/2207676.2208709. URL http://doi.acm.org/10.1145/2207676.2208709. (Zitiert auf Seite 16)

[Seno9] SensoMotoric Instruments. *iView X ^{TM} System Manual*, 2009. (Zitiert auf Seite 36)

[SVH+02] T. Suthau, M. Vetter, P. Hassenpflug, H.-P. Meinzer, O. Hellwich. Konzeption zum Einsatz von Augmented Reality in der Leberchirurgie, 2002. (Zitiert auf Seite 60)

[SWL00] S.-W. Shih, Y.-T. Wu, J. Liu. A calibration-free gaze tracking technique. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, Band 4, S. 201–204. IEEE, 2000. (Zitiert auf Seite 61)

[TJ00] V. Tanriverdi, R. J. Jacob. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, S. 265–272. ACM, 2000. (Zitiert auf Seite 15)

[VDSF97] S. Volbracht, G. Domik, K. Shahrbabaki, G. Fels. How effective are 3D display modes? In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, S. 540–541. ACM, New York, NY, USA, 1997. doi:10.1145/258549.259022. URL http://doi.acm.org/10.1145/258549.259022. (Zitiert auf Seite 16)

[Wit10] E. Wittl. *Die Wertigkeit der stereokopischen 3D-CT-Angiographie in der Darstellung und Beurteilung von Nierenarterienstenosen: eine ROC-Analyse*. 2010. (Zitiert auf Seite 23)

[WMZ+10] S. Wang, Z. Mao, C. Zeng, H. Gong, S. Li, B. Chen. A new method of virtual reality based on Unity3D. In *Geoinformatics, 2010 18th International Conference on*, S. 1–5. IEEE, 2010. (Zitiert auf Seite 23)

Alle URLs wurden zuletzt am 20. 10. 2013 geprüft.

Erklärung

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

Ort, Datum, Unterschrift