LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Masterarbeit

User Interaction in Mobile WebVR

Manuel F. Graf
grafm@cip.ifi.lmu.de

Bearbeitungszeitraum:	01. 12. 2016 bis 15. 07. 2017
Betreuer:	Dipl.-Ing. Martin Pfannenstein
Verantw. Hochschullehrer:	Prof. Dr.-Ing. Eckehard Steinbach

Zusammenfassung

Virtual Reality (VR) hat in den letzten Jahren an Popularität gewonnen, besonders im Computerspielbereich, bei Simulationen und im Gebiet der Telepräsenz. Die hierzu benötigten "'Head Mounted Displays"' sind teuer und stellen sehr hohe Anforderungen an die dazu nötige Computersysteme. Daneben gibt es einige VR-Anwendungen speziell für Smartphones, die sich durch ein simples Headset (z.B. Google Cardboard) betrachten lassen. Diese Anwendungen sind meist nativ als Anwendung für die jeweilige Geräteplattform umgesetzt. Neue Standards des W3-Konsortiums erlauben jedoch die Erkennung von VR-Systemen wie der "'HTC Vive"' innerhalb des Browsers. In der dieser Arbeit wird untersucht, welche Möglichkeiten geboten werden VR-Inhalte für moderne Webbrowser zu erzeugen und darzustellen.

Anhand von verschiedenen Eingabemethoden, die in traditionellen VR-Umgebungen genutzt werden, wurden verschiedene Eingabemethoden entwickelt. Als Ein- und Ausgabegeräte werden hier ausschließlich Smartphones verwendet, um die Notwendigkeit von zusätzlicher Peripherie zu vermeiden. Der Vorteil dabei ist vor allem, dass diese Geräte bereits in den meisten Haushalten vorhanden sind. Untersucht wird vor allem die Selektion von Objekten in einer Virtuellen Umgebung mittels Blickrichtung und eines virtuellen Zeigegeräts. Es wurde eine Nutzerstudie (n=29) durchgeführt, die die Unterschiede in der Benutzerfreundlichkeit entwickelter Eingabemethoden überprüft. Die Ergebnisse der Nutzerstudie zeigten eine Präferenz der Nutzer für die Selektion mittels Blickrichtung. Zur Ermittlung der Effizienz der Eingabemethoden wurden Kennzahlen wie die benötigte Zeit und begangene Fehler gemessen. Die Studie zeigt keine signifikanten Unterschied für die benötigte Zeit, Objekte mittels Blickrichtung oder eines zweiten Smartphones als virtuellen Zeigegeräts auszuwählen.

Abstract

Virtual Reality (VR) has gained popularity in the last few years, especially in use for video games, simulations and in the field of telepresence. The head mounted displays (HMDs), which are needed to display virtual reality content, are quite expensive and usually require high-end computers to efficiently render the graphics output. There are also VR applications, which are made specifically for the use with smartphones, which can be experienced by the use of simple HMDs, like the Google Cardboard. These applications are mostly native applications for specific operating systems. The W3-Consortium presented a draft to enable the detection of VR-Systems like the HTC Vive within modern web browsers and to allow their usage in browser based 3D applications.

This thesis describes possibilities to create and display VR content within web browsers. Multiple interaction techniques have been developed, based on traditional research about 3D interaction techniques, which serve especially the selection of objects within immersive virtual environments. The developed techniques require no additional periphery except for modern smartphones — since those are already available in most households. The implemented interaction techniques feature selection via gaze and via a second smartphone, which serves as a virtual pointing device. A user study has been conducted (n=29) to evaluate the usability of the developed interaction methods. The users preferred gaze-based selection over the virtual pointing device, but no significant differences in efficiency were observed.

Aufgabenstellung

Virtual Reality (VR) hat in den letzten Jahren an Popularität gewonnen, besonders im Computerspiel- Bereich. Die hierzu benötigten VR Headsets sind durchweg teuer und stellen sehr hohe Anforderungen an die Hardware. Daneben gibt es einige VR-Anwendungen speziell für Smartphones, die sich durch ein simples Headset (z.B. Google Cardboard) betrachten lassen. Diese Anwendungen sind meist nativ als App für die jeweilige Geräteplattform umgesetzt. In dieser Arbeit soll untersucht werden, welches Potential Web-basierte VR-Anwendungen haben. Hierzu sollen Frameworks evaluiert werden, die den Browser als Plattform für VR nutzen. Zudem gibt es einen ersten Entwurf für eine Standardisierung einer VR-Schnittstelle im Browser (WebVR), die hinsichtlich ihrer Anwendbarkeit evaluiert werden soll. Der Fokus der Arbeit liegt im Speziellen auf Bedienkonzepten, auch jenen, die sich bei nativen VR Anwendungen etabliert haben. Ziel der Arbeit ist es, eine Web-basierte VR-Anwendung zu implementieren. Darauf aufbauend soll ein Bedienkonzept erarbeitet und implementiert werden, mit dem sich durch virtuelle Räume navigieren lässt.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, 9. Juli 2017

. . . . . .

Inhaltsverzeichnis

1 Einleitung

1.1 Arten von Virtuellen Umgebungen

Virtuelle Umgebungen (*engl.* „Virtual Environments“, VE) und Virtuelle Realitäten (VR) sind kein neues Forschungsgebiet. Schon seit Jahrzehnten ist die Erforschung von solcher Virtuellen Umgebungen im Gange. Die Interaktion von Nutzern mit virtuellen 3D-Umgebungen ist bereits seit den 1960er Jahren von Interesse. Sutherland stellte 1965 das erste HMD vor, bei dem die Position des Kopfes verfolgt werden konnte. Jedoch war er seiner Zeit voraus, denn erst Ende der 1980er Jahre war die Entwicklung von Miniatur-Röhrenmonitoren und Systemen zur Messung von deren Position so weit, dass man sie effizient zur Darstellung von immersiven VEs nutzen zu konnte.

Es gibt diese Systeme heute in den unterschiedlichsten Varianten. Vom geschlossenen System der kompletten Virtualisierung über Mixed- und Augmented-Reality-Ansätze, die danach streben, virtuelle und reale Welten miteinander zu verschmelzen.

Unter geschlossenen, immersiven Systemen werden solche Systeme verstanden, die dem Nutzer eine Umgebung präsentieren, während die Außenwelt komplett ausgeschlossen wird. Die Außenwelt kann vollständig hinter der virtuellen versteckt werden, z.B. durch ein blickdichtes Display und Kopfhörer. Offene oder halboffene Systeme lassen hingegen meist zu, dass Reize der realen Welt sich mit denen der virtuellen vermischen, z.B. durch durchsichtige Displays. Auch in geschlossenen Systemen lässt sich dieser Effekt simulieren, indem der Durchblick mittels Videoaufnahmen realisiert wird. Eine Kamera zeichnet die reale Welt auf und projiziert sie dann auf das Display. Das Bild wird daraufhin analysiert und anschließend virtuelle 3D-Objekte passend auf diesem Bild platziert. Diese Systeme werden oft auch als Augmented-Reality (AR) oder Mixed-Reality (MR) Systeme bezeichnet.

- **Virtuelle Realität (*engl.* „Virtual Reality“, VR)**: Als *Virtuelle Realität* bezeichnet man die Darstellung und Wahrnehmung von vollständig computergenerierten Umgebungen und gleichzeitiger Ausgrenzung der Außenwelt. Hierbei wird die gesamte wahrnehmbare Welt generiert und dem Nutzer interaktiv präsentiert. Insbesondere setzen VR-Umgebungen auf **hohe Immersion und Präsenz**. Immersion beschreibt den Grad der Wahrnehmung, vollständig in eine virtuelle Welt einzutauchen und dabei sensorisch von der Außenwelt abgeschnitten zu sein. Dies ist vom Begriff der Präsenz zu unterscheiden der beschreibt bis zu welchem Grad der Nutzer denkt, wirklich an einem anderen Ort zu befinden [5] (S. 4).

- **Erweiterte Realität (*engl.* „Augmented Reality“, AR)**: Typischerweise wird ein System als *Augmented Reality* bezeichnet, wenn es Objekte der realen Welt mit digitalen Informationen anreichert. Beispielsweise, indem ein bestimmtes reales Objekt durch eine Kamera erkannt wird, kann auf einem (halb-)offenen System zusätzliche Information zu jenem Objekt dargestellt werden.

- **Gemischte Realität (*engl.* „Mixed Reality“, MR)**: Der Begriff ist oft nur schwer von AR zu unterscheiden. Hierbei handelt es sich um eine glaubhafte Vermischung von Informationen aus der realen Welt (wie bei AR) mit virtuellen Objekten (wie bei VR). Insbesondere ist hier auch die Realitätstreue ein entscheidender Punkt. Die virtuellen Objekte müssen in MR im Gegensatz zu AR einen festen Platz in der abgebildeten realen Welt einnehmen. [24]

In den letztes Jahren haben die Weiterentwicklung von Ausgabegeräten und die immer geringer werdenden Platzanforderung für Sensoren und Hardware dafür gesorgt, dass eine neue Welle an Virtual-Reality-Headsets für Endkunden verfügbar gemacht wurde. Deren Anzeigequalität zeichnet sich z.B. durch eine deutlich höhere Auflösung aus, als bei jenen, die bereits Ende letzten Jahrhunderts verfügbar waren [49].

1.1.1 Raumskalierte Systeme

VR-Systeme können weiter kategorisiert werden als „raumskalierte“ oder „nicht raumskalierte“ Systeme. Ein Beispiel für raumskalierte VR-Systeme ist beispielsweise die (im Sektor des Industriedesigns genutzte) CAVE [9] oder das Endnutzersystem HTC Vive. Diese Systeme sind mit Sensorik ausgestattet, die es erlaubt, die Position und Orientierung des Nutzers im Raum zu erkennen. Meist sind dies Kameras, die fest im Raum installiert sind und anhand von Markern an den Head Mounted Displays (HMDs) und Controllern deren Position und Drehung im Raum bestimmen.

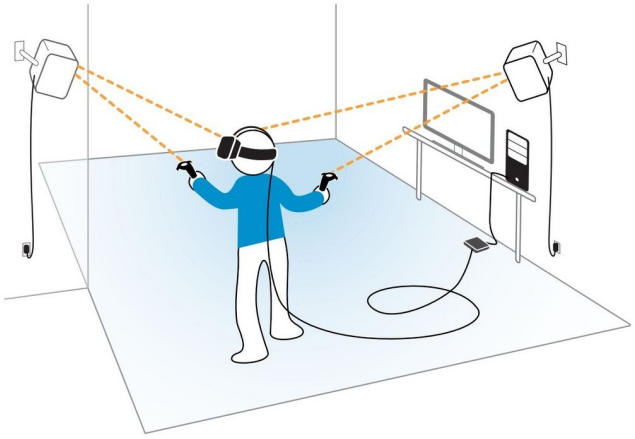


Abbildung 1.1: *Beispiel für den Aufbau eines raumskalierten Virtual-Reality-Systems. Sensoren in den Ecken des Raumes erkennen hierbei Position und Orientierung des Nutzers im Raum. (Quelle: https://www.vrheads.com/least-painful-way-set-htc-vive-lighthouses)*

Der Nachteil bei der Nutzung für Endnutzer jedoch ist einerseits, dass die Sensoren fest installiert sind, im Falle der CAVE wegen der fünf großen Projektionsflächen. Dank der kleinen und leichten Sensoren von Systemen wie der HTC Vive ist es jedoch kein Hindernis mehr, das System zu bewegen. Ein Vive-Aufbau benötigt maximal eine Stunde Einrichtungszeit und einen Raum von mindestens 1,5m auf 2m, damit der Nutzer ein gesichertes Minimum an Bewegungsfreiheit hat. Da sich der Nutzer bei Benutzung eines solchen VR-Systems physisch bewegen muss, ist dies ein entscheidender Aspekt. Die Sicht auf diese Fläche darf dabei nicht verdeckt werden, da sonst die visuellen Sensoren die Verbindung verlieren. Dies sorgt dafür, dass besonders beengten Bereiche nicht geeignet sind, um ein solches System zu nutzen.

1.1.2 Nicht raumskalierte Systeme

Es sind auch VR-Headsets erhältlich, die kein Tracking physischer Fortbewegung für Navigation innerhalb der virtuellen Umgebung erlauben. Sie sind jedoch - durch weniger benötigte Hardware - preisgünstiger. Ein namhaftes Beispiel, insbesondere für die Anwendung in Videospielen, ist das HMD Oculus Rift. Auch für die Spielkonsole Sony Playstation gibt es bereits ein eigenes VR-HMD. Diese Systeme besitzen bis jetzt keine Sensoren zur Erkennung des Nutzers im Raum und sind dafür ausgelegt, sitzend vor einem PC oder einer Konsole genutzt zu werden. Jedoch gibt es für das Oculus Rift mittlerweile einen Sensor, der auf dem Tisch platziert werden kann, um Position und Orientierung der proprietären Controller (Oculus Touch) zu erkennen, um somit eine natürlichere Interaktion in 3D-Umgebung zu ermöglichen.

1.2 Motivation

Ein weiterer Nachteil für den Endnutzer sind die Kosten, die beim Kauf solcher Systeme entstehen. Ihre Preise liegen zwischen 900 und 1200 Euro. In diesem Preis sind aber noch nicht die Kosten für ein High-End-Computersystem eingerechnet, welches benötigt wird, um Inhalte auf beispielsweise der HTC Vive abzuspielen. Laut Angaben von HTC sollte mindestens eine Grafikkarte vom Typ nVidia GeForce GTX 970 oder ATI Radeon R9 290 vorhanden sein, welche zum aktuellen Zeitpunkt noch einmal ca. 200 Euro kostet. Auch für die günstigere Alternative, das Oculus Rift VR-System, muss man mit Kosten von 600 Euro rechnen (und zusätzliche 150 Euro für die Oculus Touch Controller).

Ein Nachteil dieser Systeme ist auch, dass sie meist nur plattformspezifisch einsetzbar sind. Während die HTC Vive z.B. hauptsächlich auf Windows-Betriebssystemen zusammen mit der Plattform für Digitalen Rechteverwaltung (engl. „digital rights management“, DRM) „Steam“ genutzt werden kann, stellt dies für Nutzer anderer Betriebssysteme ein Problem dar.

Um diese beiden Nachteile zu umgehen, hat sich diese Arbeit zum Ziel gemacht, sich neue Entwicklungen im Web zu nutze zu machen, um mit Geräten Virtual Reality für Endnutzer zu ermöglichen, die heute in fast jedem Haushalt vorhanden sind. Da die meisten Haushalte heutzutage im Besitz eines oder mehrerer Smartphones sind, sind diese Endgeräte von besonderem Interesse. Zunächst wird diese Arbeit darauf eingehen, welche Interaktionsmöglichkeiten in virtuellen Umgebungen häufig verwendet werden. Anschließend wird beschrieben, welche Möglichkeiten das Web heute schon bietet, um diese Funktionalität plattformübergreifend und unabhängig von spezifischen VR-Systemen zu implementieren. Hierbei liegt der Fokus der Arbeit besonders auf den Schnittstellen moderner Browser und der Entwicklungen im Bereich der Web-Entwicklung und Erstellung von 3D-Anwendungen mit JavaScript.

Nachfolgend werden in dieser Arbeit zunächst einige Möglichkeiten beschrieben, die zur Erstellung und Darstellung von 3D-Inhalten in Browsern genutzt werden können. Anschließend werden mögliche Eingabegeräte, die in traditionellen VR-Systemen genutzt werden, aufgezählt und diskutiert, ob und wie die Daten dieser Eingabegeräte in Browsern genutzt werden können. Daraufhin folgt eine Beschreibung der genutzten 3D-Umgebung, welche als VE umgesetzt werden sollte, sowie die Beschreibung der implementierten Eingabemethoden für die (im Zuge dieser Arbeit durchgeführten) Nutzerstudie. Abschließend werden die gesammelten Daten zur Bestimmung der Benutzerfreundlichkeit der entwickelten Eingabemethoden statistisch untersucht.

2 Annahmen und Systemspezifikation

Für das Erstellen von 3D-Szenen, die als Virtuelle Umgebung genutzt werden können, gibt es einige gängige Systeme. Da jedoch die Darstellung von VR-Inhalten im Browser noch nicht sehr gängig ist, werden im folgenden Kapitel Technologien beschrieben, die es ermöglichen sollen, VR-Inhalte in modernen Browsern darzustellen. Diese umfassen sowohl eigenständige Software, die auch in der professionellen Computerspielentwicklung verwendet wird, als auch neue Web-Technologien, die speziell für die Anzeige von VEs im Browser entwickelt wurden.

2.1 Erstellung von 3D-Inhalten fürs Web

Weit verbreitete Werkzeuge zur Erstellung von 3D-Inhalten, z.B. in Videospielen, sind *Engines* und Autorenwerkzeugen (*engl.* Authoring Tools) wie „Unity3D“ oder das „Unreal Development Kit“ (UDK). Diese erlauben Entwicklern, 3D-Szenen zu erstellen und dort Objekte zu platzieren, sowie Logik und Verhalten für diese zu hinterlegen. In der Regel bauen diese Systeme auf Entitäten-Komponenten-Systeme (*engl.* Entity-Component-Systems, ECS) auf. Hierbei werden Objekte (Entitäten) definiert, die mit einzelnen Komponenten, wie z.B. einer geometrischen Form, Kollisionskörpern oder Skripts erweitert werden können. Die grundlegenden Bausteine einer solchen Software-Architektur sind:

- **Entitäten:** Sie beschreiben zunächst leere Container-Objekte, denen mit Hilfe von wiederverwendbaren Komponenten z.B. eine äußere Form oder ein Verhalten zugewiesen werden kann.
- **Komponenten:** SIe sind wiederverwendbare Module, die einer Entität zugewiesen werden können, um sie auf bestimmte Art zu verändern. Diese können meist pro nutzender Entität konfiguriert werden und beinhalten sämtliche Logik, die für die Lösung eines spezifischen Problems relevant ist. Diese können von der Änderung des Erscheinungsbilds eines Objekts bis zur Deklaration des Verhaltens eines Objekts über den Zeitverlauf reichen.
- **Systeme:** Sie bieten in ECSs einen globalen Handlungsspielraum und Hilfsfunktionalität für das Zusammenspiel von einzelnen Komponenten und ermöglichen so die Verwaltung von Entitäten und Komponenten.

Anwendungen, die mit *Software Development Kits* (SDK) wie Unity3D oder dem UDK erstellt werden, können für die gängigsten Betriebssysteme (*engl.* „Operating System“, OS) exportiert werden. Die resultierenden Anwendungen nutzen dann die für das jeweilige System vorherrschende Standard-Grafik-Bibliotheken, z.B. DirectX für Windows oder OpenGL auf Mac OS und Linux.

2.1.1 WebGL and three.js

Auch für den Browser existiert eine Schnittstelle zur Erzeugung von dreidimensionalen Grafiken, die ebenfalls Funktionen der Grafikkarte nutzen kann. **WebGL** (Web Graphics Library) ist eine, in den meisten modernen Browsern vorhandene, JavaScript-Schnittstelle. Sie baut auf *OpenGL ES* 2.0 auf, welches ein Teil der OpenGL-Spezifikation ist und für eingebettete Systeme verwendet wird. Es ist plattformübergreifend einsetzbar und kann somit verwendet werden, um interaktive 2D- und 3D-Inhalte in Canvas-Elementen der HTML5-Spezifikation zu nutzen.

Die Schnittstelle bietet dabei tiefgreifenden Zugriff auf Befehle zur Berechnung und Erstellung von Grafiken, ähnlich wie OpenGL. Um jedoch komplexe Inhalte zu erstellen, bietet es sich meist an, fertige SDKs und Entwicklungswerkzeuge, wie die oben beschriebenen, zu nutzen. Auch für eine direkte Benutzung im Browser gibt es bereits Bibliotheken, die Funktionalität der WebGL-Schnittstelle abstrahieren und Entwicklern somit Werkzeuge bieten, 3D-Inhalte ohne Kenntnisse von WebGL zu erstellen.

Die gängigste Bibliothek um 3D-Inhalte für Browser mit JavaScript zu erstellen ist *three.js* [7]. Sie erlaubt das Anzeigen von 3D-Szenen im Browser sowie die Erstellung und Manipulation von 3D-Objekten. Die Bibliothek bietet für die Programmierung die meisten Funktionalitäten, die auch in Autorenwerkzeugen wie Unity3D verfügbar sind. Jedoch fehlt eine grafische Benutzeroberfläche (*engl.* „Graphical User Interface“, GUI), die als Autorenwerkzeug genutzt werden kann. Zwar besitzt die Bibliothek ein Editor-Plugin, jedoch dient dies meist nur der Analyse von Szenen für Entwickler. Es können keine komplexen Szenen erstellt und exportiert werden. Der Arbeitsablauf beim Erstellen von 3D-Inhalten in three.js besteht somit aus der rein programmatischen Erstellung und Verwaltung aller Objekte.

Dies bringt manche Vorteile mit sich, wie z.B. die einfachere Versionierung der Inhalte mittels eines Versionsverwaltungssystems, jedoch ist evtl. zunächst mehr Einarbeitungszeit und mehr Programmieraufwand nötig, da keine komfortable GUI zur Erstellung und Modifikation von Inhalten vorhanden ist.

2.1.2 Unity und WebGL

Unity3D ist ein gängiges Entwicklungswerkzeug zur Erstellung von plattformübergreifenden 3D-Inhalten. Neben der Möglichkeit, das erstellte Projekt für Betriebssysteme wie Windows, Mac OS und Linux zu exportieren, ist es auch möglich, dieses für die Anzeige im Browser zu exportieren. Unity3D nutzt intern ASM [21], eine (im Funktionsumfang reduzierte) JavaScript-Spezifikation, die insbesondere für die Nutzung mit Compilern Anwendung findet. Die Unity3D-Engine versteht sowohl Programmcode, der in C# verfasst ist, als auch die interne Version von JavaScript, die meist als *UnityScript* bezeichnet wird. Auf einige Funktionen des Unity3D-SDKs können jedoch in dem für WebGL exportierten Endprodukt nicht zugegriffen werden. Ein Beispiel hierfür ist das Paket für Netzwerkfunktionen, welches beispielsweise Verbindungen zu anderen Geräten über *Sockets* ermöglichen. Es existiert aber aus diesem Grund eine Schnittstelle zu Scripts im Browser [47], um Funktionen des Browsers und somit auch von externen Webdiensten zu nutzen. Die fehlenden Netzwerkfunktionen bedingen, dass auf Browser-Funktionalitäten zurückgegriffen werden muss, um Konnektivität zu anderen Diensten oder Geräten im Netzwerk zu gewährleisten. Insbesondere als Ersatz für Socket-Verbindungen muss hierbei auf Technologien wie *WebSockets* oder *WebRTC* zurückgegriffen werden. Auf diese Technologien wird in dieser Arbeit später im Abschnitt 3.2.3 noch genauer eingegangen.

2.1.3 WebSockets und WebRTC

Die *WebSocket*-Spezifikation beschreibt den Aufbau einer interaktiven bidirektionalen Kommunikationssitzung zwischen einem Browser und einem über das Netzwerk erreichbaren Server. Die Benutzung von WebSockets entspricht dabei jener von üblichen Socket-Verbindungen. Ein WebSocket-Client kann sich zu Beginn via HTTP-Protokoll bei einem Server anmelden. Wie bei regulären HTTP-Anfragen wird hierbei zwischen Client und Server das *Transmission Control Protokoll* (TCP) für die Übertragung der Daten genutzt. Im Gegensatz zum normalen HTTP-Protokoll bleibt die zugrundeliegende TCP-Verbindung in einer WebSocket-Sitzung jedoch bestehen, so dass diese weiter verwendet werden kann, um Daten an den Client zu senden.

WebRTC unterscheidet sich von Websockets vor allem durch die Nutzung des Real-Time Transport Protokolls (RTP). Dieses wird vor allem genutzt, um kontinuierlich Audio-, Video- oder Rohdaten zu übermitteln. Der Leistungsgewinn liegt hierbei vor allem in der Nutzung des *User Datagram Protokolls* (UDP) statt TCP. Da UDP ein verbindungsloses Protokoll ist, können hierbei zwar Daten verloren gehen, jedoch ist auch der Datendurchsatz höher. Ein besonderes Merkmal von WebRTC ist, dass damit sog. Peer-to-Peer Verbindungen aufgebaut werden, die zwei Endgeräte erlaubt, Daten über einen direkte Verbindung auszutauschen.

2.1.4 WebVR Entwurf des W3-Konsortiums

Das Konsortium W3C hat im Zuge der aktuellen Entwicklungen bereits einen Entwurf für eine zukünftige Unterstützung von VR-fähigen Displays und Eingabegeräten veröffentlicht [51]. Dieser Entwurf unterliegt einer ständigen Weiterentwicklung, weswegen auch das W3C im momentanen Stand davor warnt, Komponenten davon in eigenen Anwendungen zu verwenden. Jedoch gibt WebVR bereits eine Richtung für einen Standard vor und dient somit als Vorlage für Software, die es bereits ermöglicht, Funktionalitäten des zukünftigen Standards zu verwenden (sog. *Polyfill*) [18].

Der W3C-Standard umfasst die Benutzung von VR-fähigen Ein- und Ausgabegeräten im Browser. Das W3C geht vor allem ein auf die Abfrage und Verwendung der Sensordaten von verschiedenen HMDs im Browser mit der Programmiersprache JavaScript. Er befasst sich aber auch mit Themen des Datenschutzes, des Komforts und der Sicherheit der Nutzer.

Die einheitliche Schnittstelle für VR-fähige Displays wird im Standard als *VR Device* beschrieben. Dieses kann z.B. ein stereoskopisches Display sein, wie bei namhaften HMDs, ein Smartphone mit passender Halterung zur stereoskopischen Ansicht oder auch nur ein Gerät, welches zwar keine stereoskopischen Inhalte anzeigen kann, jedoch über die Fähigkeit verfügt, die eigene Orientierung in sechs Freiheitsgraden zu messen.

Desweiteren definiert der WebVR-Entwurf, wie Webseiten mit dem Gerät kommunizieren dürfen. Dies geschieht meist über die Erkennung von Ereignissen, die vom Browser gesendet werden. Eine Webseite muss sich zunächst jedoch am Gerät anmelden, um eine Sitzung zu erhalten. Erst nach der Bestätigung der Sitzung kann die Webseite die sog. *Pose* vom Gerät lesen. Diese enthält Daten zur Orientierung und/oder Position des Gerätes - je nachdem, welche Daten für das jeweilige Gerät verfügbar sind. Ebenfalls erlaubt der Standard die Konfigurierung der Anwendung für unterschiedliche Ausgabegeräte. Zum Beispiel beinhaltet er Optionen für die Linsenkorrektur oder (im Falle der Nutzung von Smartphones) eine Einstellung für die Fehlerkorrektur von Gyroskop-Abweichungen.

2.1.5 Mozilla A-Frame

Im Zuge der Weiterentwicklung der WebVR-Spezifikation entstanden Frameworks und Bibliotheken, die es Entwicklern vereinfachen sollen, VR-Inhalte im Browser zu erstellen. Ein interessanter Ansatz ist hier vor allem der von *Mozilla A-Frame*. Die Bibliothek erlaubt es (ähnlich wie bei bekannten ECSs) Objekte mit HTML-Befehlen zu erstellen und diese mit Logik auszustatten. A-Frame verfolgt hier einen deklarativen, Dokumentbasierten Ansatz. Das heißt, Basis-Objekte wie geometrische Formen können durch hinzufügen des entsprechenden HTML-Elements zum Quellcode der Website in einer 3D-Szene platziert werden. Nach dem Laden des A-Frame-Frameworks mittels folgendem Code können die A-Frame HTML-Elemente innerhalb von HTML-Dokumenten genutzt werden:

```
<head>
  <script src="https://aframe.io/releases/0.5.0/aframe.min.js"></script>
</head>
```

Eine 3D-Szene, die WebVR unterstützt, kann nun also mit wenigen HTML-Befehlen erstellt werden. Folgendes Code-Beispiel aus der A-Frame Dokumentation [34] erzeugt beispielsweise das Ergebnis in Abb. 2.1.

```
<body>
  <a-scene>
    <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
    <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
    <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5"
        color="#FFC65D"></a-cylinder>
    <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4"
        color="#7BC8A4"></a-plane>
  </a-scene>
</body>
```

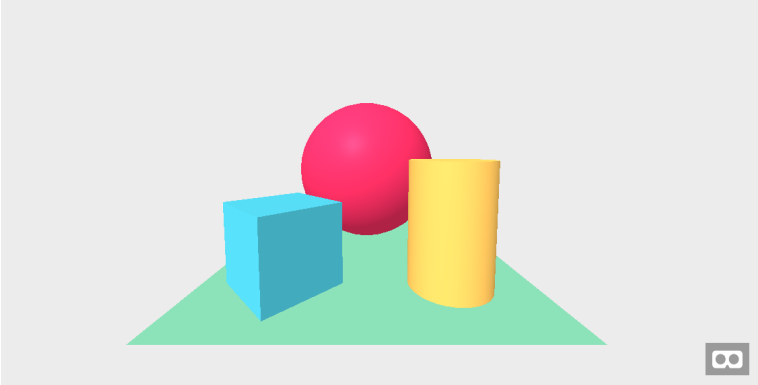


Abbildung 2.1: *Die Standard-Szene einer A-Frame-Anwendung*

Grundsätzliche Information wie Position, Rotation und Farbe vom Objekten kann direkt als Attribut dieses HTML-Elements angegeben werden. Im obigen Beispiel werden beispielsweise Position und Rotation in der 3D-Szene direkt mittels HTML-Attribut übergeben. Es können auch benutzerdefinierte Attribute angelegt werden. Dies geschieht mittels des Komponenten-Systems. Es können Schemata angelegt werden, die bestimmen, welchen Datentyp Attribute haben und wie sich das Vorhandensein einer Komponente auf den Lebenszyklus eines Objekts auswirkt.

A-Frame ermöglicht es, 3D-Inhalte für den Browser zu erstellen und diese unabhängig vom Ausgabegerät zu präsentieren. Doch für die Interaktion des Nutzers bietet A-Frame meist keine vorgefertigte Lösung, insbesondere nicht für die Nutzung in immersiven VEs. Deswegen wird die Arbeit anschließend auf generelle Konzepte der Nutzerinteraktion in 3D-Umgebungen eingehen.

2.2 Nutzerinteraktion in VEs

Die Interaktion von Nutzern mit virtuellen 3D-Umgebungen ist bereits seit den 1960er Jahren von Interesse. Sutherland stellte 1968 das erste HMD vor, bei dem die Position des Kopfes verfolgt werden konnte [46]. Jedoch war er seiner Zeit voraus, denn erst Ende der 1980er Jahre war die Entwicklung von CRT-Displays und Systemen zur Positionserkennung im Raum (*engl.* „tracking devices“, im nachfolgenden *Tracker*) so weit, um sie effizient zur Darstellung von immersiven VEs nutzen zu können. Nachfolgend werden einige Konzepte von Eingabegeräten beschrieben, die in 3D-Umgebungen Anwendung finden. Zudem wird jeweils angemerkt, ob und welche Möglichkeiten moderne Browser bieten, um die Daten solcher Geräte zu empfangen, um sie in Web-Anwendungen nutzen zu können.

2.2.1 Eingabegeräte und Charakteristiken

Eingabegeräte lassen sich laut [5] (S. 88ff) anhand einiger Charakteristiken klassifizieren:

- Anzahl der Freiheitsgrade (*engl.* „degrees of freedom“, DOF)
- Frequenz der Datenübermittlung und Typ der Daten
- benötigte Aktivität zur Auslösung von Funktionalität

Insbesondere für die Manipulation von Objekten relevant sind die Freiheitsgrade des Eingabegeräts sowie die Anzahl der DOF, die mit einer Bewegung verändert werden können. Einer reguläre Computermaus gilt als 2-DOF-Eingabegerät, da sie zwei Freiheitsgrade besitzt — die freie Bewegung (Translation) auf der X- und Y-Achse einer Ebene. Diese werden als integriert bezeichnet, da mit einer Bewegung sowohl die X- als auch die Y-Position mit einer Bewegung verändert werden kann. Im Raum frei bewegliche Eingabegeräte, wie z.B. die Controller der HTC Vive, besitzen sechs Freiheitsgrade. Sie können auf drei Achsen eines dreidimensionalen kartesischen Koordinatensystems bewegt (drei Translationsfreiheitsgrade, vgl. Abb. 2.2) und um drei Achsen gedreht werden (drei Rotationsfreiheitsgrade, vgl. Abb. 2.2) und gelten somit als 6-DOF-Eingabegeräte.

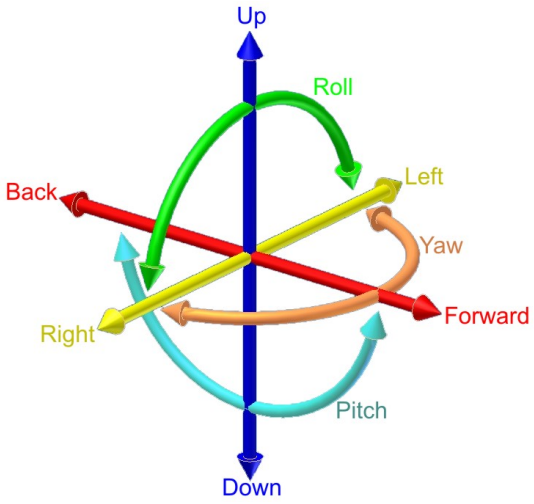


Abbildung 2.2: *Grafische Darstellung der sechs Freiheitsgrade. Die drei Translationsfreiheitsgrade sind dargestellt durch die rote, gelbe und blaue Achse. Die Rotationsfreiheitsgrade sind dargestellt durch „Roll“, „Pitch“ und „Yaw“. (Quelle: https://en.wikipedia.org/wiki/Six_degrees_of_freedom)*

Die Daten, die vom Eingabegerät übermittelt werden, unterscheiden sich sowohl in ihrer Zusammensetzung als auch in der Frequenz, in der sie übermittelt werden. Meist übermitteln diese entweder diskrete oder kontinuierliche Werte oder eine Kombination aus beiden. Diskrete (oder digitale) Werte sind hierbei meist boolesche Werte Beispielsweise geben sie an, ob eine bestimmte Taste am Eingabegerät gedrückt wurde. Kontinuierliche (oder analoge) Daten sind oft reelle Zahlenwerte, die von der momentanen Position oder Orientierung des Geräts abhängig sind und kontinuierlich ohne weitere Notwendigkeit einer Aktion durch den Nutzer übermittelt werden.

Desweiteren unterscheiden sich Eingabegeräte in der benötigten Aktivität, um die Datenübermittlung einzuleiten. Während *rein aktive* Eingabegeräte eine Interaktion vom Nutzer voraussetzen, um eine Aktion einzuleiten, übermitteln *rein passive* Eingabegeräte einen konstanten Datenstrom — ohne, dass der Nutzer aktiv handeln muss.

Ein Beispiel für ein rein aktives Eingabegerät, das sowohl diskrete als auch kontinuierliche Daten übermittelt ist ein Trackball. Dieser sendet nur bei Rotation des Balls kontinuierliche Daten, die der aktuellen relativen Änderung der Rotation entsprechen. Er bietet jedoch auch die Möglichkeit, durch Druck einer Taste diskrete Daten zu übermitteln. Ein Beispiel für ein rein passives

Eingabegerät ist ein optischer Tracker, welcher kontinuierlich Daten übermittelt, solange das Objekt, welches verfolgt werden soll, sich im Aufnahmebereich befindet.

Bowman et. Al. [5] teilen zudem Eingabegeräte in folgende Kategorien ein und bewerten sie anhand ihrer Gebrauchstauglichkeit für die Interaktion in immersiven VEs:

- Desktop-Eingabegeräte und 3D-Computermäuse
- Tracker
- direkte menschliche Eingabe

Desktop-Eingabegeräte sind Geräte, die ursprünglich für den sitzenden Gebrauch vor Desktop-Computern entwickelt wurden. Sie können jedoch mittels einer sinnvollen Übersetzung auch in 3D-Umgebungen genutzt werden. Gängig sind hierbei vor allem Tastatur, 2D-Computermäuse und Trackballs.

Desktop-Eingabegeräte: Die *Tastatur* ist ein weitläufig verbreitetes, rein aktives Eingabegerät. Durch die Menge an unterschiedlichen diskreten Datenwerten, die von einer Tastatur übermittelt werden können, ist sie ein vielseitiges Eingabegerät. Jedoch sind die Daten, die von Tastaturen übermittelt werden, von diskreter Natur. Für eine Eingabe von kontinuierlichen oder analogen Daten sind sie ungeeignet, weswegen eine Computermaus als zusätzliches Eingabegerät von großem Nutzen ist. Zwar sind Tastaturen, mit geeigneter Übersetzung von bestimmten Tasten zu Funktionen, für die Nutzung in immersiven VEs benutzbar. Jedoch sind sie dafür gedacht, sitzend vor einem Desktop-Computer genutzt zu werden und sie sind nicht ohne weiteres geeignet, sie zusammen mit einem HMD zu benutzen, das keine Sicht auf die Tastatur ermöglicht. Durch eine sinnvolle Verteilung der Funktionen auf der Tastatur lässt sich dieser Effekt jedoch mildern. Beispielsweise können die sehr nah beieinander liegenden Pfeiltasten oder die Tasten W, S, A und D zur Bewegung in einer 3D-Umgebung benutzt werden, wie es oft in Videospielen der Fall ist. Dadurch, dass hierbei die Hand meist an derselben Position auf der Tastatur verharrt und nicht nach anderen Tasten gesucht werden muss ist sie ein Eingabegerät, das zumindest für sitzende Interaktion in einer VE geeignet ist. Die Eingabe von Tastaturbefehlen kann über die Browser API *KeyBoardEvent* abgefragt werden. Wird eine Taste der Tastatur gedrückt, wird ein sog. *Event* vom Browser gesendet, der neben dem entsprechenden Bezeichner für die gedrückte Taste auch weitere Informationen enthält, beispielsweise ob die Taste für längere Zeit gedrückt wurde, oder ob modifizierende Tasten wie die Strg-, Alt- oder Shift-Taste währenddessen betätigt waren. Eine Auflistung aller vorhandenen Informationen zum *KeyBoardEvent* kann in der Dokumentation des Mozilla Developer Network eingesehen werden [36].

2D-Computermäuse sind ebenfalls ein klassisches Eingabegerät für moderne Desktop-Computer. Sie bieten sowohl kontinuierliche Daten über die relative Veränderung der Position des Geräts, als auch diskrete Eingabewerte durch die einzelnen Maustasten. Der Nachteil für die Nutzung in immersiven VEs ist bei Computermäusen genau wie bei der Tastatur jedoch, dass sie nur auf einer Ebene benutzt werden können. Die Abfrage von Aktionen einer 2D-Computermaus können im Browser ebenfalls abgefangen werden. Für die Erkennung der Bewegung der Maus kann das *MouseEvent* genutzt werden, welches bei der Bewegung des Mauszeigers über die Webseite vom Browser gesendet wird. Desweiteren kann erkannt werden, ob der Nutzer Maustasten oder das Mausrad betätigt.

Es existieren auch Mäuse, die sechs Freiheitsgrade bieten und somit besonders gut für die Interaktion in dreidimensionalen Umgebungen geeignet sind. Diese werden vor allem von professionellen Autoren von 3D-Inhalten benutzt. Ein Beispiel für eine solches Eingabegerät ist die *SpaceMouse* von 3Dconnexion [1](Abb. 2.3). Für die Abfrage der Daten solcher Geräte im Browser muss sie jedoch meist erst die Zuweisung der Tasten und Achsen für das Betriebssystem angepasst werden. Hierfür ist meist zusätzliche Treiber-Software für das Betriebssystem nötig. Jedoch können diese dann über die Gamepad API [35] des Browsers ausgelesen werden.

Trackballs hingegen können jedoch auch in stehender Position in eine Hand genommen werden und die kontinuierlichen Daten werden übermittelt, sobald die im Gerät integrierte Kugel rotiert wird. Dabei wird die Rotation der Kugel ähnlich übersetzt wie eine reguläre Bewegung einer 2D-Computermaus.

Abbildung 2.3: *Beispiel einer Computermaus mit sechs Freiheitsgraden: Die SpaceMouse von 3Dconnexion. (Quelle: http://www.3dconnexion.de/products/spacemouse/spacemousepro.html)*

Gamepads und *Joysticks* sind weitere Eingabegeräte, die ursprünglich für den Gebrauch mit Desktop-Computern und Spielkonsolen gedacht sind. Sie sind heute sowohl kabelgebunden als auch kabellos erhältlich und sind somit sowohl für den sitzenden Gebrauch vor einem Desktop-Computer als auch frei stehend im Raum gleichermaßen einfach zu bedienen. Das Aussehen unterscheidet sich je nach System, jedoch sind auf modernen Gamepads meist ähnliche Komponenten zu finden. Ein bis zwei analoge Schalthebel, welche je zwei Freiheitsgrade bieten, sowie einige Tasten und ein Steuerkreuz. Gamepads von namhaften Spielekonsolen wie der *Sony PlayStation*(s. Abb. 2.4) besitzen auch verbaute Motoren, die Vibration erzeugen können. Sie fungieren somit neben ihrer Funktion als Eingabegerät auch als haptisches Ausgabegerät. Einige Gamepads besitzen ebenfalls verbaute Gyroskope um die Rotation des Geräts zu übermitteln. Somit sind sie prinzipiell geeignet um in VEs genutzt zu werden. Jedoch müssen sie üblicherweise in beiden Händen gehalten werden, was beidhändige Interaktion verhindert. Über die Gamepad-API [35] moderner Browser kann bereits auf die Daten dieser Eingabegeräte zugegriffen werden.

Tracker: sind rein passive Geräte oder Aufbauten, welche die reale Umgebung analysieren und Informationen über die Position und Orientierung von bestimmten Objekten im Raum errechnen, zum Beispiel für ein HMD. Es gibt magnetische wie optische Tracker, die von einer Basisstation aus für das menschliche Auge unsichtbares Licht aussenden. Dieses wird dann von Empfängern erkannt, um somit dessen Position des Empfängers im Raum zu errechnen (s. Abb. 1.1). Somit eignen sie sich besonders gut für die Nutzung in immersiven VEs, da ihre digitale Repräsentation realitätsnah in der virtuellen Welt platziert werden kann.

Die Controller sowie das HMD der HTC Vive nutzen diese Technologie, um die Position sowie die Orientierung der einzelnen Geräte im Raum zu erkennen. Die Abfrage der Daten im Browser wird erst mit Veröffentlichung des WebVR-Standards in allen Browsern verfügbar sein. Bis dahin muss jedoch ein Polyfill [18] zur Substitution der Funktionalität benutzt werden. Auf die Informationen der Controller der HTC Vive können über die Gamepad-API des Browsers zugegriffen werden. Zu diesen optischen Bewegegungssensoren gehören beispielsweise Kameras, welche mit Infrarot-Lichtquellen sowie Infrarot-Kameras ausgestattet um neben dem aufgezeichneten Bild noch Tiefeninformation zu erhalten. Aktuelle Beispiele für solche Bewegungssensoren sind die *Microsoft Kinect*, *Intel RealSense* oder *Leap Motion*. Während die Kinect zur Erkennung ganzer Körper genutzt wird, dienen RealSense und Leap Motion hauptsächlich zur Erkennung einer oder beider Hände des Nutzers im Raum (s. Abb. 2.5). Die Treibersoftware dieser Produkte bieten meist

Abbildung 2.4: *Beispiel eines aktuellen Gamepads: Sony Dualshock 4 Wireless Gamepad. Neben den beiden analogen Schalthebel bietet er durch die Nutzung eines Gyroskops auch noch die Erkennung der Rotation. (Quelle: https://www.playstation.com/en-us/explore/accessories/dualshock-4-wireless-controller-ps4/)*

eine Lösung zur automatischen Errechnung von Skelett-Modellen, um Positionen einzelner Gliedmaßen und Gelenke des erkannten Nutzers im Programmcode nutzen zu können. Um diese jedoch auch im Browser nutzen zu können, müssen die kontinuierlichen Daten der Aufzeichnung von der Treibersoftware über einen eigenen Kanal, wie beispielsweise WebSockets oder WebRTC, erfolgen. Genauere Erfahrungsberichte zur Benutzung der Leap Motion-Kamera können im Abschnitt 3.2.2 gefunden werden.

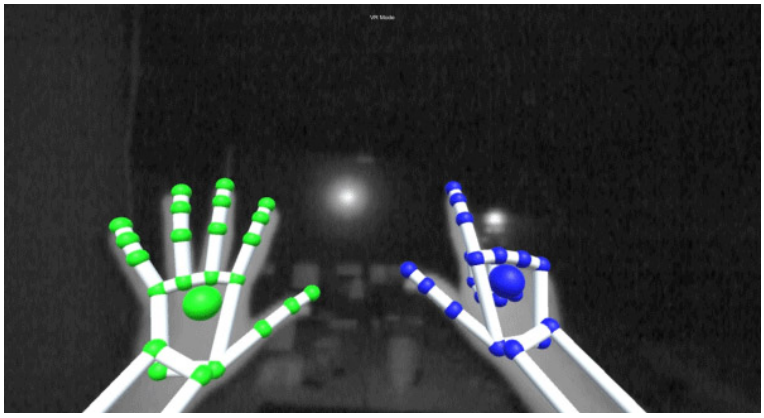


Abbildung 2.5: *Beispiel der Erkennung der Hände eines Nutzers und Verwendung in einer VE mit der Leap Motion-Kamera. (Quelle: https://gallery.leapmotion.com/)*

Ebenfalls gehören *Trägheitssensoren* in Geräten, wie z.B. das Gyroskop in modernen Smartphones zu solchen Trackern. Bei diesen ist besonders zu beachten, dass nach einiger Benutzungszeit Fehler in der Messung auftreten können. Besonders das Gyroskop ist für einen solchen *Sensordrift* sehr anfällig. Moderne Browser in Mobilgeräten nutzen jedoch zum Ausgleich bereits ebenfalls die Daten des Magnetometers. Auch der WebVR-Standard bietet die Konfiguration eines Filters an, mit dem diese Fehler reduziert werden können [18].

Besonders interessant für die Nutzung im Browser und in Smartphones ist hier der Ansatz von *Google Tango*[16]. Die Technologie erlaubt es, mithilfe der in Smartphones verbauten Kameras,

Räume anhand des Kamerabildes zu analysieren und die Position des Gerätes im Raum zu ermitteln. Somit kann ein raumskaliertes System ermöglicht werden, ohne die Notwendigkeit fest installierter Emitter oder Sensoren.

Direkte menschliche Eingabe: Darunter verstehen Bowman et al. [5] Eingaben, die direkt auf Signalen beruhen, die der menschliche Körper produziert. Beispiele hierfür wären etwa Messungen von physiologischen Signalen wie Puls, Hautleitwiderstand oder Gehirnwellen. Ebenso zählen Mikrofone zur Aufnahme vom Sprachbefehlen zu dieser Kategorie. Mittels Webdiensten, wie die *Google Spracherkennung* [17], *IBM Watson* [23] oder *Amazons Alexa Voice Service* [2] ist es im Browser bereits möglich, mit dem Mikrofon aufgenommene Sprachaufnahmen in Echtzeit zu analysieren und diese in Text umwandeln zu lassen. Die erhaltene Nachschrift kann dann mittels JavaScript analysiert werden, um Funktionen der eigenen Webanwendung aufzurufen. Die Dienste unterscheiden sich jedoch in der Anzahl der verfügbaren Sprachen. Während IBM Watson z.B. nur die Erkennung von Portugiesisch, Französisch, Japanisch, Mandarin, Arabisch und Englisch unterstützt, bieten die Schnittstellen von Amazon und Google auch die Erkennung deutscher Sprache an. Für die Abfrage dieser Dienste ist es jedoch nötig, das aufgenommene Audiosignal zunächst vom Browser an den eigenen Webserver zu senden, der wiederum eine Anfrage an den jeweiligen Dienst stellt und das Ergebnis zurück an den Browser sendet. Die Dienste unterscheiden sich weiterhin in ihrer Genauigkeit, Sprache in Text umzuwandeln. Die beste Qualität bot bei ersten Vergleichen während der Entwicklung eines Prototyps jedoch die Spracherkennung von Google.

Eine interessante Implementierung bietet Googles Browser *Google Chrome*. Im Chrome-Browser ist bereits eine Schnittstelle verfügbar, die es mit wenigen Befehlen erlaubt, direkt vom Browser aus eine Anfrage an den Spracherkennungsdienst von Google zu senden. Jedoch gilt hier besonders zu beachten, dass dies vor allem in Browsern von mobilen Endgeräten (wie Smartphones) nicht fehlerfrei funktioniert. Besonders die kontinuierliche Aufnahme von Sprache sorgte hierbei für Probleme. Normalerweise wird vorausgesetzt, dass der Aufnahme von Audiosignalen im Browser eine Nutzerinteraktion vorausgeht, wie beispielsweise ein Klick auf die Oberfläche der Webseite. Zwar gibt es die Option, nach dieser Interaktion kontinuierlich Audiosignale aufzunehmen und an den Spracherkennungsdienst zu senden. Jedoch war dies bei ersten Tests mit Google Chrome auf Smartphones nicht möglich, da die Sprachaufzeichnung immer nach wenigen Sekunden unterbrochen wurde und neu gestartet werden muss. Somit ist die Spracherkennung für die Unterstützung der Interaktion in immersiven VEs zwar eine sehr interessantes Funktionalität, jedoch noch nicht für jede Konfiguration von WebVR-fähigen Endgeräten problemlos einsetzbar.

Grundlegende Interaktionen in VEs: Bowman et. Al [5] geben in ihrem Nachschlagewerk über Nutzerinteraktion in immersiven 3D-Umgebungen Beispiele für grundlegende Aufgaben, die in solchen Systemen üblicherweise bearbeitet werden müssen. Sie identifizieren vor allem drei Aufgaben, von denen in jeder interaktiven VR-Anwendungen i.d.R mindestens eine Anwendung findet:

- Selektion von Objekten,
- Manipulation von selektierten Objekten,
- Fortbewegung und Navigation in der virtuellen Umgebung.

Im Folgenden werden einige Möglichkeiten beschrieben, die Bewältigung dieser Aufgaben in einer VE zu ermöglichen und es wird erläutert, ob und wie es möglich ist, diese Funktionalität auch in einer Web-basierten Anwendung zu implementieren.

2.2.2 Manipulation von Objekten

Bowman et al. identifizieren die grundlegenden Aufgaben der Manipulation von Objekten in einer 3D-Umgebung [5] (S. 142). Sie weisen zwar darauf hin, dass dies nicht alle Anwendungsspezifischen Interaktionen abdeckt, jedoch für grundlegende Interaktion mit Objekten in virtuellen Umgebungen gilt:

- **Selektion** ist der Vorgang, ein bestimmtes Objekt aus einer Vielzahl von Objekten zu identifizieren und auszuwählen. Das Pendant in der realen Welt wäre ein Objekt mit der Hand aufzunehmen.
- **Positionierung** ist der Vorgang der aktiven Veränderung der räumlichen Position eines Objekts. Das Pendant in der realen Welt wäre es, das aufgenommene Objekt an einem anderen Ort im Raum abzulegen.
- **Rotation** ist der Vorgang, die Orientierung eines Objekts im Raum zu ändern. Das Pendant in der realen Welt wäre es, das aufgenommene Objekt an einem anderen Ort im Raum zu drehen.

Für jeden dieser grundlegenden Interaktionen gibt es einige entscheidende Parameter, die die Gebrauchstauglichkeit der Interaktion und somit den erfolgreichen Abschluss der Aufgabe beeinflussen können[14]. Ein Beispiel hierfür ist die Entfernung des Zielobjekts zum Nutzer. Eine Auflistung dieser Parameter befindet sich in Tabelle 2.1.

Vorgang	**beeinflussende Parameter**
Selektion	Entfernung zum Objekt, Größe des Objekts, Dichte der umliegenden Objekte und Verdeckung, Anzahl der zu selektierenden Objekte
Positionierung	Entfernung und Richtung vom Nutzer zum Zielpunktes und der Ausgangsposition des Objekts, benötigte Präzision beim Platzieren.
Rotation	Entfernung zum Objekt, Ausrichtung des Objekts, Ausmaß der Drehung und benötigte Präzision

Tabelle 2.1: *Grundlegende Manipulationsschritte und beeinflussende Parameter*

Zudem lässt sich die Manipulation weiter durch die dahinterstehende Metapher kategorisieren. Bowman et. Al. [5] unterscheiden diese hierbei in *exozentrische* und *egozentrische* Metaphern, wobei exozentrische Szenarien beschreiben, in denen der Nutzer in Vogelperspektive die Welt betrachten kann, die er zu manipulieren versucht. Als egozentrische Interaktion werden Szenarien beschrieben, in denen der Nutzer die zu manipulierende Umgebung aus der Ich-Perspektive wahrnimmt.

Im weiteren wird sich die Arbeit vor allem mit der egozentrischen Interaktion auseinandersetzen, da diese für die nachfolgende Fallstudie und Benutzerstudie am naheliegendsten ist. Für diese Art der Interaktion werden besonders Konzepte wie eine virtuelle Repräsentation der Hand oder virtuelle Zeigegeräte empfohlen. Für die Fallstudie des im Zuge dieser Arbeit erstellten Bedienkonzeptes ist besonders die benutzerfreundliche und effiziente Selektion von Objekten zur Navigation durch eine VE von besonderem Interesse. Aus diesem Grund wird vor allem die Selektion untersucht.

2.2.3 Selektion

Die Selektion von Objekten als Teilaufgabe der Manipulation von Objekten kann wiederum in Teilaufgaben unterteilt werden, für die ein Bedienkonzept Lösungen bieten muss:

- Das Anzeigen eines Objekts aus einer Menge verfügbarer Objekte
- Die Bestätigung der Auswahl
- Die Rückmeldung des Systems über erfolgreiche Auswahl eines Objektes

Das Anzeigen eines Objekts kann hierbei durch verschiedene Interaktionen vorgenommen werden. Eine Möglichkeit ist die Auswahl durch Spracheingabe, welche besonders bei vielen gleichartigen Objekten problematisch sein kann. Sind die Objekte in der virtuellen Welt dem Nutzer räumlich nahe und wird ein geeigneter Tracker verwendet, der die Position der Hand oder eines anderen Eingabegerätes im Raum erkennen kann, so können Objekte direkt mit der virtuellen Repräsentation des Gerätes oder der Hand berührt werden, um diese zu selektieren. Dies bedingt jedoch zunächst für weit entfernte Objekte, dass der Nutzer sich zunächst zu dem Objekt bewegt, welches selektiert werden soll. Dies ist jedoch nicht in allen Szenarien gegeben und manchmal wenig praktikabel. Um das zu umgehen, kann ein virtuelles Zeigegerät genutzt werden.

Die Bestätigung der Auswahl erfolgt laut [5] üblicherweise über eine diskrete Eingabe, wie einen Tastendruck, eine Geste oder ein Sprachbefehl. Jedoch kann auch kein gesonderter Befehl nötig sein, um die Eingabe zu bestätigen. Die Rückmeldung über eine erfolgreiche Selektion eines Objekts kann dem Nutzer über verschiedene Arten erfolgen, beispielsweise durch visuelle, auditive oder haptische Signale.

In der Literatur beschrieben werden einige Selektionsmechanismen, von denen im Nachfolgenden einige vorgestellt werden.

- Der einfache Raycast von der Position des Eingabegeräts [5],
- volumetrische Ansätze wie *SpotLight* [31], und *Aperture* [15],
- Selektion mit einer virtuellen Hand [5], oder
- Selektion über die Bildebene [5].

Raycast: Die einfachste Art eines virtuellen Zeigegeräts ist der sog. *Raycast*. Dabei wird vom virtuellen Eingabegerät ein Strahl in eine bestimmte Richtung ausgesandt, der beim Auftreffen auf ein Objekt dieses auswählt. Die Richtung dieses Strahls hängt dabei von der Ausrichtung des verwendeten Eingabegerätes ab. Die Länge des Strahls kann hierbei frei gewählt werden. Die Nutzung eines regulären Raycast stößt man jedoch auf einige Probleme hinsichtlich der Gebrauchstauglichkeit zur egozentrischen Selektion von Objekten in VEs. Bei weit entfernten Zielobjekten muss beachtet werden, dass dies eine hohe Präzision benötigt, da eine kleine Abweichung des Winkels bereits große Veränderungen in der Bahn des Strahls bewirkt. Außerdem ist die Form und die Position des zu selektierenden Objekts entscheidend bei der Selektion auf weite Entfernung. Im Beispiel des, in Abschnitt 3.1 beschriebenen, Systems sind die Objekte flache Kreise, die sich auf dem virtuellen Boden befinden. Sind diese weit entfernt, sind sie mit einem Strahl sehr schwer zu selektieren, da die zu treffende Fläche sehr klein ist. Die große Abweichung der Endposition des Strahls bei kleiner Änderung der Rotation des Eingabegeräts erschwert dies weiter. Sind die Objekte dicht beieinander oder überlappen sich diese, wird die Aufgabe weiterhin erschwert. Auch die Verdeckung von Objekten untereinander oder durch andere Objekte in der VE, können eine erfolgreiche Selektion mit einem regulären geraden Strahl verhindern. Zur Milderung einiger dieser Probleme kann bereits die Veränderung der selektierenden Körpers, z.B. die Nutzung eines Kegels statt eines dünnen Strahls oder die Veränderung der Bahn des Strahls beitragen. Es existieren viele unterschiedliche Methoden zur Selektion von Objekten mittels Raycast [3]. Methoden wie der sog. *Depth-Ray* [48] erlauben dabei die Spezifizierung bestimmter Objekte entlang des Strahls. Andere Methoden nutzen mehrere Strahlen zur genauen Bestimmung eines Objekts [54]

oder verformte Strahlen zur Indikation von Objekten, die teilweise verdeckt sind [11]. Zur Selektion von Objekten auf einem hohen Absatz der VE kann beispielsweise ein Strahl mit parabolischer Flugkurve genutzt werden, wie bei der Teleport-Komponente der HTC Vive (s. Abb. 2.6).

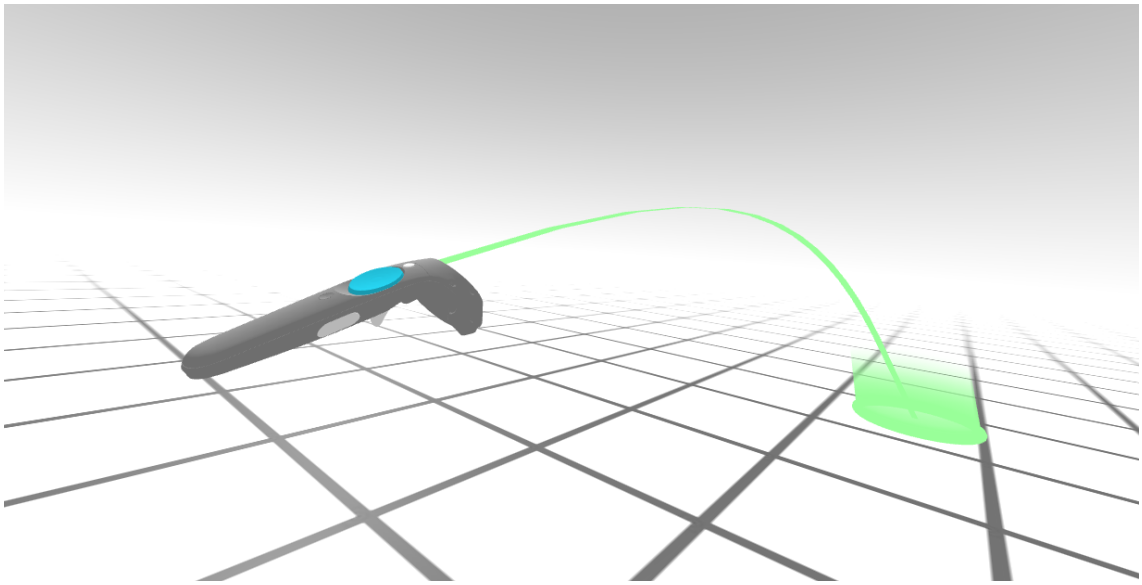


Abbildung 2.6: *Raycast zur Bestimmnung eines Punkts im Raum bei der Verwendung eines HTC Vive-Controllers in einer webbasierten VE, erstellt mit A-Frame (Quelle: https://aframe.io/blog/teleport-component/)*

Volumetrische Ansätze: Aufgrund dieser Probleme gibt es auch Ansätze, bei denen der einfache Strahl durch ein volumetrisches Objekt, wie ein Kegel oder eine Kugel ersetzt wird. Dies reduziert die Anforderung an die Präzision des Nutzers um Objekte zu selektieren. Bei der sog. „Taschenlampen"-Methode (engl. *Flashlight*)[31] wird statt des einfachen Strahls ein Kegel an der Position des virtuellen Zeigegeräts platziert, dessen Spitze sich am Eingabegerät befindet und dessen Mittelachse an die Richtung des Eingabegeräts angepasst wird. Dadurch wird es aber notwendig, zwischen den Objekten, die sich innerhalb des so aufgespannten Kegels befinden, zu unterscheiden. Für die Selektion einzelner Objekte aus dieser Menge ist also weitere Interaktion notwendig. Dies kann entweder ein einfacher Ansatz sein, bei dem jeweils das Objekt selektiert wird, welches der Mittelachse am nächsten liegt [31]. Die *Aperture*-Methode (s. Abb. 2.7) ermöglicht es dem Nutzer durch eine weitere Interaktion einzustellen, welche Objekte innerhalb des Kegels selektiert werden sollen. Hierbei kann eine Ebene, die parallel zur Grundfläche des Kegels verläuft, entlang der Mittelachse des Kegels verschoben werden, um anzuzeigen, in welcher Distanz zum Nutzer Objekte selektiert werden sollen. Eine Verbindung dieser Methode mit Raycasts wurde auch von Vanacken et. al untersucht [48]. Sie verglichen hierbei einen Raycast, bei dem man, wie bei der Aperture-Methode, die Entfernung der zu selektierenden Objekte angeben kann mit einem volumetrischen Ansatz, bei dem eine virtuelle Kugel als Körper dient. Ihre Ergebnisse legen jedoch nahe, dass der Raycast zur Selektion von Objekten jedoch erfolgreicher war.

Selektion mit einer virtuellen Hand: Ist ein geeigneter Tracker zur Positionserkennung der Hand eines Nutzers verfügbar, so können diese zur Selektion benutzt werden. Befinden sich die Objekte nah genug im virtuellen Umfeld des Nutzers, so kann dies durch einfache Berührung der Objekte mit der virtuellen Hand geschehen. Für weiter entfernte Objekte ist es jedoch nötig, ebenfalls Konzepte anderer Methoden, wie beispielsweise des Raycasts, zu verwenden. Lösungen bieten hierbei verschiedene Konzepte, wie das *beidhändige Zeigen*. Werden beide Hände des

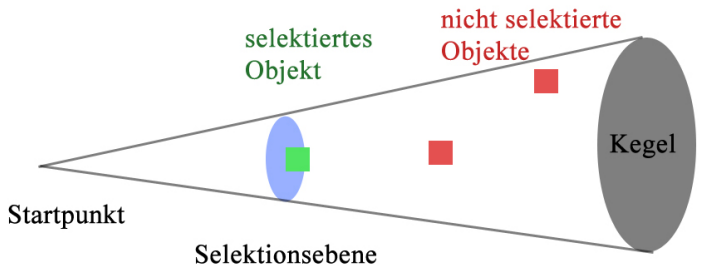


Abbildung 2.7: *Abstrakte Darstellung eines konischen Selektionsbereichs mit Abstandsbestimmung zur Selektion bestimmter Objekte nach [15]*

Nutzers vom System erkannt, kann die Position einer Hand als Ursprung des Strahls bestimmt werden. Die Richtung des Strahls wird schließlich durch die Position der anderen Hand bestimmt. Kann nur eine Hand erkannt werden, ist die sog. *Go-Go*-Methode[5] (S. 160) geeignet. Bei dieser Methode kann der virtuelle Arm des Nutzers verlängert werden, um weit entfernte Objekte zu erreichen. Die Streckung des virtuellen Arms kann dabei vom Nutzer durch die Vergrößerung der Distanz seiner Hand zum HMD vergrößert werden. Zur Manipulation und Selektion von nahe gelegenen Objekten wird die virtuelle Hand analog zur gemessenen Position in der 3D-Szene platziert. Ab einem gewählten Schwellenwert streckt sich der virtuelle Arm. Dabei steigt die Distanz der virtuellen Hand zum Nutzer nichtlinear. Experimente haben für diese Methode zwar gezeigt, dass sie leicht verständlich ist, jedoch weniger effektiv als reguläre Raycasts [4].

Selektion über die Bildebene: Methoden, die die Bildebene nutzen, reduzieren die Komplexität der Selektion auf eine Ebene, womit vom Eingabegerät nur noch zwei DOF benötigt werden. Bei der sog. *Sticky Finger*-Variante (s. Abb. 2.8) wird von der virtuellen Kopfposition des Nutzers ein Strahl in Richtung des Eingabegeräts oder des Fingers des Nutzers gesendet. Letzeres zumindest dann, wenn Tracker genutzt werden, die die Position der Hand erkennen. Der Raycast wird dann ins Unendliche verlängert. Selektiert werden dann jene Objekte, die von diesem Vektor geschnitten werden.

Eine weitere Abwandlung dieser Methode ist die sog. *Head Crusher*-Methode [5] (S. 157). Diese bedingt die Nutzung eines Trackers, der die Position der einzelnen Finger einer Hand erkennt. Objekte können dann selektiert werden, indem aus der Perspektive des Zeige- und Mittelfingers des Nutzers das gewünschte Zielobjekt sich genau zwischen den beiden Fingern befindet. Dies bietet vor allem im Vergleich zum vorigen Beispiel den Vorteil, dass das Eingabegerät bzw. der Finger des Nutzers das Zielobjekt nicht verdeckt. Da bei Nutzung der Bildebene zur Selektion nur zwei DOF benötigt werden, wäre auch eine Methode vorstellbar, die eine reguläre Computermaus nutzt. Hierbei könnte die Bewegung der Maus ein Objekt auf der Bildebene verschieben, der die Richtung des ausgesendeten Strahls bestimmt. Auch die Selektion von Objekten mittels der Blickrichtung des Nutzers könnte dieser Kategorie zugeordnet werden, da i.d.R. nur zwei DOF der Kopfrotation benötigt werden, um die eindeutige Blickrichtung des Nutzers zu erkennen.

2.3 Fortbewegung und Navigation in VEs

Wenn nachfolgend von Fortbewegung in einer VEs gesprochen wird, so beschreibt dies in Wirklichkeit nur eine Veränderung des präsentierten Bildausschnitts der VE bzw. der Veränderung der Kamerakoordinaten im virtuellen Raum. Dadurch hat der Nutzer den Eindruck, er würde sich innerhalb der VE fortbewegen. Laut Bowman [5] (S. 188ff) lassen sich die Verfahren dafür in *physische* und *virtuelle* sowie *aktive* und *passive* Methoden kategorisieren.

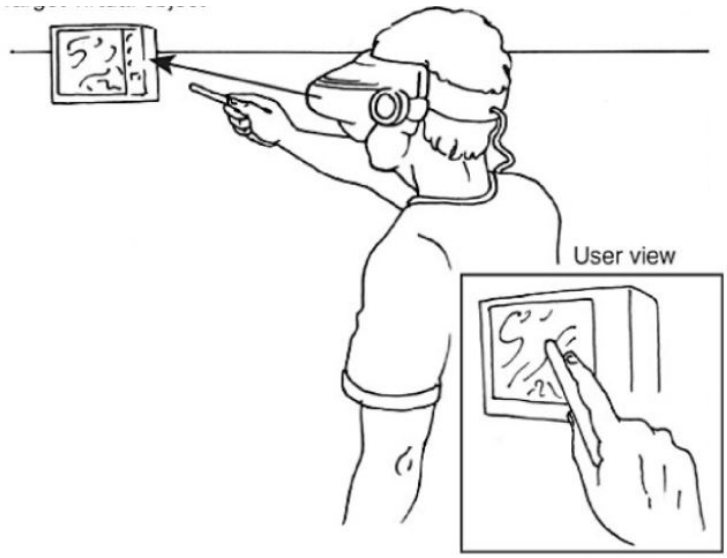


Abbildung 2.8: *Erkennung von selektierten Objekten über die Bildebene mit der Sticky-Finger-Methode nach Variante[5] (S. 157)*

- *Physische Fortbewegungsmethoden* verlangen vom Benutzer körperliche Aktivität, um sich wahlweise wirklich im Raum zu bewegen oder Geräte zu nutzen, die es ihm erlauben, auf einer Stelle zu laufen. HMDs, deren Position im Raum gemessen werden kann und deren Informationen genutzt werden, um den wahrgenommenen Bildausschnitt der VE zu verändern, gehören zu dieser Kategorie.

- *Virtuelle Fortbewegungsmethoden* erlauben es dem Nutzer, seine Position innerhalb der VE zu verändern, obwohl er in Wirklichkeit ortsgebunden ist.

- Bei *aktiven Fortbewegungsmethoden* steuert der Nutzer selbst die Bewegung durch die VE. Unter Steuerung ist hierbei die Einleitung der Bewegung, Steuerung der Geschwindigkeit sowie Richtung der Bewegung und deren Abbruch zu verstehen.

- Bei *passiven Fortbewegungsmethoden* nimmt ein System die Positionierung des Nutzers in der VE vor.

2.3.1 Ansätze zur Bewegungsmetapher

Doug Bowman stellt dazu einige Ansätze vor, die es Nutzern ermöglichen sollen, sich durch VEs zu bewegen. Er klassifiziert diese dabei nach der dahinterliegenden Interaktionsmetapher für die Fortbewegung mittels

- physischer Bewegung,

- Lenkung eines virtuellen Vehikels,

- Angabe eines konkreten Ziels oder

- Manipulation der Umgebung.

Fortbewegung durch physische Bewegung: Die aktive Fortbewegung anhand physischer Bewegung hat den Vorteil, dass die Sinneseindrücke der Augen und des Gleichgewichtssinns synchronisiert werden können. Diese Art der Fortbewegung setzt jedoch voraus, dass wahlweise Tracker vorhanden sind, mit denen sich die Position des Nutzers im Raum verfolgen lässt, oder dass Geräte vorhanden sind, mit denen sich das Laufen simulieren lässt, wie z.B. Laufbänder. Im Falle von raumskalierten Systemen wie der HTC Vive ist dies möglich, hat jedoch auch den Nachteil, dass die Fläche, auf der man sich real bewegen kann, meist sehr viel kleiner ist, als die VE. Deswegen müssen die verschiedenen Methoden zur Fortbewegung oft kombiniert werden.

Fortbewegung durch Lenkung eines virtuellen Vehikels: Die virtuelle Bewegung innerhalb einer VE mittels ein virtuelles Vehikel erlaubt es dem Nutzer, sich innerhalb der VE auch über weite Strecken zu bewegen. Dabei ist es notwendig, falls eine aktive Fortbewegung möglich sein soll, dem Nutzer Einflussmöglichkeiten auf die Parameter der Bewegung zu gestatten. Die Richtung der Bewegung kann beispielsweise durch die Blickrichtung oder das Anzeigen mittels eines virtuellen Zeigegeräts angegeben werden. Der Nachteil bei der Nutzung der Blickrichtung ist, dass sie während der Ausführung der Bewegung zwangsläufig mit der Bewegungsrichtung gekoppelt ist. Dem Nutzer ist es so also nicht möglich, sich umzusehen, während er sich fortbewegt. Zur Steuerung der Bewegungsrichtung können jedoch auch andere Eingabegeräte genutzt werden. Um der Metapher eines üblichen Lenkrads zu folgen, wird lediglich ein Freiheitsgrad benötigt, die Rotation um eine Achse.

Fortbewegung durch Angabe eines konkreten Ziels: Ein Ansatz ist auch das Anzeigen eines Zielpunktes oder -objektes und anschließender passiver Fortbewegung zu jenem Punkt. Da die VE meist viel größer ist als der tatsächlich bei raumskalierten Systemen vorhandene Raum, wird diese Methode auch in vielen Anwendungen für die HTC Vive genutzt. Durch das Betätigen einer Taste wird hier ein parabolisch geformter Strahl in die virtuelle Szene gesendet. An dem Ort, an dem der Strahl den Boden der VE oder ein Objekt schneidet, wird ein Indikator angezeigt, ob sich der Nutzer zu diesem Punkt bewegen darf. Wird die Taste nun erneut betätigt, wird der Nutzer zu diesem Punkt in der VE bewegt. Für diese Art der Fortbewegung ist jedoch nötig, die Selektion eines bestimmten Punktes oder Objekts in der VE vornehmen zu können. Hierfür eignen sich beispielsweise die Konzepte aus Abschnitt 2.2.3. Ein ähnliches Konzept wurde auch in dem im Zuge dieser Arbeit erstellten Prototypen für die Nutzerstudie verwendet (s. Abschnitt 3.2.3).

Fortbewegung durch Manipulation der Umgebung: EIne weitere Möglichkeit zur aktiven Fortbewegung in VEs ist die Manipulation der Umgebung. Diese kann z.B. durch Gesten vorgenommen werden. Entweder kann die VE durch Handgesten „verschoben“ werden (*grabbing the air* [32]), oder die VE kann anhand eines Referenzobjekts manipuliert werden. Wie bei der Manipulation von Objekten kann hierbei das Referenzobjekt selektiert werden. Durch Rotation oder Verschiebung des Objektes ändert sich dann nicht dessen Position, sondern die Position des Nutzers innerhalb der VE.

Die Fortbewegung innerhalb einer VE kann Auswirkungen auf das Wohlbefinden der Nutzer haben. Im Folgenden Abschnitt wir kurz erklärt, was Gründe sind für das Auftreten der *Simulatorkrankheit* (auch *Cyber Sickness* oder *Virtual Reality Sickness* genannt).

2.3.2 Cyber Sickness

Nutzer einer immersiven VE laufen Gefahr, Opfer der sogenannten „Cyber Sickness“ (CS) zu werden, welche sich in Symptomen äußert, die denen der See- bzw. Reisekrankheit ähnlich sind. Beispiele hierfür sind Augen-, und Kopfschmerzen, Schweißausbrüche, Desorientierung oder Übelkeit. LaViola et al. [29] untersuchten gängige Theorien über die Entstehung der CS, insbesondere

die Theorie über den Konflikt der Sinneswahrnehmungen und die Theorie der Haltungsinstabilität. Die Theorie über den Konflikt der Sinneswahrnehmungen besagt, dass die Unterschiede der visuellen Wahrnehmung und des Gleichgewichtssinns bezüglich der eigenen Fortbewegung Konflikte erzeugen, die zu den genannten Symptomen führen können [39]. Die Theorie der Haltungsinstabilität basiert auf der Idee, dass der menschliche Körper stets versucht, im Vergleich zur Umgebung eine stabile Haltung einzunehmen [40]. Ändert sich die Wahrnehmung der Umgebung jedoch schlagartig, so weiß der Körper nicht, was die korrekte Reaktion auf diese Änderung ist.

LaViola et al. [29] identifizieren zudem weitere technische und individuelle Faktoren, die Einfluss auf das Auftreten und die Intensität der Symptome haben können. Zu den technischen Faktoren zählen für sie:

- Die Fehler beim Messen der korrekten Orientierung des HMD bzw. eine unkontrollierbare Bewegung des Bildausschnitts. Eine fehlerhafte Messung in der Position oder Orientierung des HMD kann zu Konflikten in der Sinneswahrnehmung führen und somit die Symptome hervorrufen.
- Zeitliche Verzögerung zwischen einer Aktion des Nutzers und dessen Darstellung, wie z.B. zwischen Kopfdrehung und Aktualisierung des angezeigten Bilds im HMD kann Symptome hervorrufen.
- Flimmern des Bilds, besonders bei großem Sichtfeld (engl. „field of view“, FOV) kann ebenfalls Symptome hervorrufen.

Zu den individuellen Faktoren, die Einfluss auf die Anfälligkeit für die Symptome haben, zählen sie Geschlecht, Alter, den aktuellen Gesundheitszustand und die Position des Nutzers im realen Raum. Befindet sich der Nutzer in sitzender Position, können die Symptome nach der Theorie zur Haltungsinstabilität reduziert werden.

Weitere Vorschläge, um das Auftreten von Symptomen der Cyber Sickness zu verhindern bedingen laut [29] eine künstliche Stimulation des Gleichgewichtsinns, oder die Verwendung von Objekten, die innerhalb der VE als Fixpunkt dienen. Dies macht hinsichtlich der Theorie zur Haltungsinstabilität Sinn. Es ist vorstellbar, dass z.B. die Bewegung durch eine VE weniger Symptome der CS hervorruft, wenn der Nutzer immer fixe Objekte im Blickfeld hat.

Da für den Prototypen dieser Arbeit besonders die Fortbewegung in VEs von Bedeutung war, sind auch die Ergebnisse von Fernandes et. al. [12] von Interesse. Sie fanden bei ihrer Studie heraus, dass eine geringfügige Verkleinerung des Sichtfelds während der passive Bewegung durch eine VE Nutzern erlaubt hat, länger in dieser zu verweilen, ohne an Symptomen der CS zu leiden .

3 Methodik

Die praktische Aufgabe dieser Arbeit bestand in der Entwicklung eines Bedienkonzeptes für den *NavVis IndoorViewer* (IV). Dafür mussten zunächst typische Bedienkonzepte für Virtuelle Umgebungen aus der Literatur auf ihre Anwendbarkeit im Web analysiert werden. Einige dieser Konzepte wurden bereits im letzten Kapitel beschrieben. Da die aktuelle Implementierung des IV die Navigation durch die VE nur mittels der Selektion von diskreten Ortspunkten erlaubt, wurde besonderer Fokus auf die Selektion von Objekten gelegt. Nachfolgend wird der IV und seine Funktionsweise grob umschrieben.

Anschließend werden die drei Konzepte zur Selektion von Objekten beschrieben, die mit Hilfe von three.js und aktuell verfügbaren Web-Technologien integriert wurden:

- Blickrichtung
- Raycast
- Raycast (parabolische Kurve)

Als Ein- und Ausgabegeräte wurden bei dieser Arbeit ausschließlich Smartphones genutzt, um die Notwendigkeit von zusätzlicher Peripherie zu eliminieren. Das Ziel dieser Beschränkung sollte vor allem sein, nur Geräte zu nutzen, die bereits in nahezu jedem Haushalt vorhanden sind. Laut Bundesamt für Statistik besitzen 95% der Haushalte in Deutschland mindestens ein Smartphone und oder Mobiltelefon. Desweiteren sind durchschnittlich pro Haushalt 1,765 Smartphones vorhanden [45].

Zur Nutzung eines Smartphones als HMD ist eine Halterung notwendig, die mit einem Gummiband am Kopf des Nutzers befestigt werden kann. Namhafte Beispiele sind unter anderem Headsets wie das *Google Cardboard*, *Google Daydream* oder *Samsung Gear VR*. Es gibt jedoch auch einige dieser Halterungen von Drittherstellern, die preisgünstiger zu erwerben sind. Für die Entwicklung wurde als Testgerät die Halterung aus Abb. 3.1 genutzt. Sie bietet ein Stellrad zum Einstellen des Augenabstands sowie für den Abstand zwischen Augen und Display.

Abbildung 3.1: *Die für die Durchführung der Nutzerstudie genutzte Halterung zur Nutzungs eines Smartphones als HMD.*

3.1 Systembeschreibung: NavVis Indoor Viewer

Der *NavVis Indoor Viewer (IV)* ist eine Software vom in München ansässigen Unternehmen NavVis, welches sich auf die Vermessung von Innenräumen spezialisiert hat. Mit eigens entworfenen Trolleys werden Punktwolken von Innenräumen von Gebäuden erzeugt. An diskreten Ortspunkten im Gebäude werden zudem von Kameras Panoramabilder aufgezeichnet, die 360°-Bildmaterial zu den jeweiligen Koordinaten der Ortspunkte liefern. Die diskreten Ortspunkte werden in einem Datensatz gespeichert und können über die NavVis JavaScript-API abgerufen werden. Die Darstellung erfolgt mit JavaScript und three.js und ist somit im Browser verfügbar. Für den Datenabgleich und die Konnektivität zur API von NavVis wird im IV Angular.js genutzt.

Zu jeder Panorama-Aufnahme ist zudem die Information vorhanden, an welcher Position im Raum die jeweilige Aufnahme aufgezeichnet wurde. Die Koordinaten werden sowohl im lokalen Koordinatensystem der 3D-Szene als auch in *WGS 84*-Koordinaten angegeben, um auch anhand von GPS-Signalen die richtigen Panoramen auswählen zu können.

Das System ist vor allem für die Nutzung an Desktop-Rechnern gedacht und bietet die Möglichkeit, sich mit einer Computermaus innerhalb des Raumes von einem Panorama zum nächsten zu bewegen oder Informationen über speziell ausgezeichnete Sehenswürdigkeiten (engl. „points of interest“, POI) einzusehen, die im Datensatz an festen Koordinaten vermerkt sind. Ein solcher POI ist in Abb. 3.2 d) zu erkennen.

Wird die Maus über das Browserfenster bewegt, wird von der Kamera in der 3D-Szene ein Raycast ausgeführt, der beim Auftreffen auf einen diskreten Ortspunkt (Abb. 3.2 a) und b)) diesen markiert (Abb. 3.2 b)). Durch die Betätigung der Maustaste wird eine Animation ausgelöst. Hierbei wird das Sichtfeld des Nutzers kurzzeitig reduziert, wodurch ein Zoom-Effekt entsteht der dem Nutzers eine Bewegung durch den Raum suggeriert. Die Verwendung dieses Effekts wirkt sich laut [12] lindernd auf Symptome der „Cyber Sickness“ aus. Während der Laufzeit der Animation wird im Hintergrund das neue Panorama geladen, welches am Ende der Animation angezeigt wird. Desweiteren bietet das System die Möglichkeit, sich über die Benutzeroberfläche eine Route zu bestimmten POIs oder Ortspunkten anzeigen zu lassen.

Die Interaktion erfolgt hauptsächlich über eine grafische Benutzeroberfläche (Abb. 3.2 c)), die im Browser mittels HTML über die bestehende 3D-Umgebung gelegt wird. Dies erlaubt jedoch nur die Benutzung von Desktop-Eingabegeräten wie einer Computermaus oder die Touch-Eingabe auf Smartphones. Für eine Nutzung als immersive VE ist der IV im momentanen Stand jedoch deswegen nicht optimiert. Vor allem die Tatsache, dass innerhalb der 3D-Umgebung keine grafische Benutzeroberfläche verfügbar ist, reduziert die Funktionalität enorm, die vom Nutzer innerhalb der VE ausgelöst werden kann. Für die meisten Interaktionen, wie das Bewegen zu einem bestimmten Punkt im Raum anhand dessen Koordinaten oder das Abrufen der Information eines POI, ist jedoch in der API von NavVis bereits Funktionalität vorgesehen, welche mittels JavaScript aufgerufen werden kann.

Zudem muss beachtet werden, dass das Bildmaterial der Panorama-Aufnahmen nicht stereoskopisch aufgenommen wird. Also existiert pro diskretem Ortspunkt nur ein monoskopisches Panorama-Bild zur Anzeige. Diese Tatsache ist ein weiterer Punkt, der die Anzeige des IV auf HMDs erschwert, die in der Regel dafür gedacht sind stereoskopisches Bildmaterial anzuzeigen. Der IV bietet bereits die Möglichkeit der dreidimensionalen Anzeige der Inhalte. Die Daten der aufgenommenen Punktwolke werden dazu genutzt, realistisch wirkende dreidimensionale Szenen zu erstellen und anzeigen zu können, welche auch bei Benutzung mit stereoskopischen Ausgabegeräte eine realistische Tiefenwahrnehmung ermöglichen kann. Die Darstellung ist jedoch wenig detailgetreu und wurde deswegen in der Arbeit nicht verwendet.

Es gab bereits erste Versuche, den IV mit stereoskopischen Ausgabegeräten zu nutzen. In der Dokumentation des IV [38] war hierfür bereits ein Beispiel gegeben. Dieses Beispiel besteht aus einer Website, die zwei Iframes zeigt, in denen jeweils eine Instanz desselben IVs anzeigt wird. Dies ist neben dem zusätzlichen Aufwand, für jede Interaktion oder Kopfrotation zwei kom-

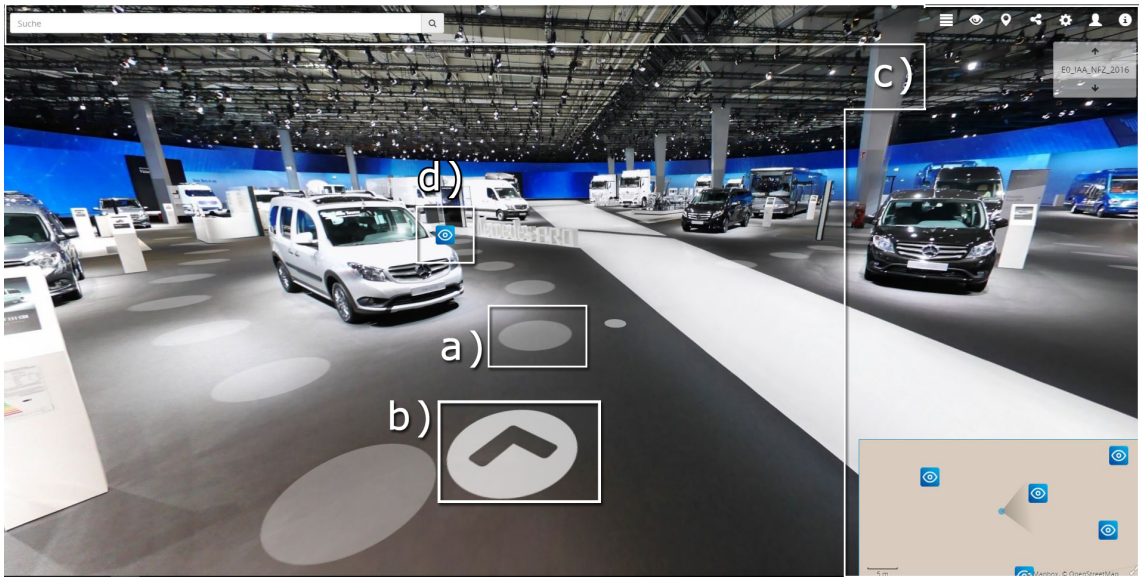


Abbildung 3.2: *Ansicht des NavVis IndoorViewers auf einem Desktop-PC. Der Ausschnitt zeigt a) die diskreten Ortspunkte, zu denen ein Nutzer navigieren kann, b) einen Ortspunkt, wenn der Nutzer den Mauscursor über den Punkt bewegt, c) die grafische Benutzeroberfläche, die erlaubt bestimmte POIs zu finden oder Einstellungen für die Darstellung vorzunehmen, sowie eine kleine Übersichtskarte des Geländes und d) ein Symbol, welches die Position eines POI anzeigt.*

plett unabhängige Instanzen des IV zu aktualisieren, ein enormer Mehraufwand an Rechenleistung. Dieser Aufwand ist insbesondere für Smartphones, die nicht ganz aktuell sind, kaum zu bewältigen. Zudem müssen durch die simultane Nutzung von zwei Angular.js-Anwendungen pro Aktualisierungszyklus Daten doppelt abgeglichen werden, was weitere Rechenleistung benötigt. Änderungen in der Blickrichtung oder Ortswechsel könnten durch die nicht synchronisierte Aktualisierung der unabhängigen Instanzen zu ungewollten Effekten in der Anzeige führen.

Im Zuge dieser Arbeit wurden diese Nachteile, wie nachfolgend beschrieben, umgangen: Three.js bietet bereits die Möglichkeit, aus einer bestehenden 3D-Szene stereoskopische Bilder zu errechnen und diese schließlich separat anzuzeigen. Hierfür müssen in der 3D-Umgebung zwei Kameras erstellt werden, deren Ausgabe daraufhin auf je einer Seite des Bildschirms angezeigt werden kann (vgl. Abb. 3.3). Für die Verwendung dieses Effekts existieren Beispiele in der Dokumentation der Bibliothek [8].

Zur Darstellung der virtuellen Umgebung auf dem Smartphone muss zudem die Adresszeile des Browsers ausgeblendet werden. Dafür kann jedoch eine Vollbild-Anzeige des Canvas-Elements im Browser angefordert werden. Das kann in aktuellen Browsern mittels der *requestFullScreen()*-Methode von HTML-Elementen erfolgen. Hierbei ist zu beachten, dass dieser immer eine Nutzerinteraktion vorausgehen muss, beispielsweise die Betätigung einer Schaltfläche. Im Beispiel des entwickelten Prototyps musste hierfür ein beliebiger Punkt innerhalb der Webseite geklickt werden.

Die Umgebung, die für die Untersuchungen dieser Arbeit betrachtet wird, zeigt die Ausstellungshalle eines namhaften Automobilherstellers. Sie besteht aus einer großen Halle, in der die Automobile ausgestellt sind sowie eine Treppe, die eine Verbindung zu einer kleineren Halle darstellt.

Als wichtigste Interaktion innerhalb des IV wurde für den Prototyp zunächst die Navigation identifiziert. Da diese vor allem über die Selektion von Ortspunkten ausgelöst wird, die in der VE verteilt sind, musste vor allem dieses Problem gelöst werden. Im nachfolgenden Kapitel werden

Abbildung 3.3: *Ansicht des NavVis IndoorViewers mit dem verwendeten Stereo-Effekt*

die umgesetzten Bedienkonzepte vorgestellt, die innerhalb des IV genutzt werden können um dessen Funktionen auch zu nutzen, wenn dieser als immersive VE verwendet wird.

3.2 Eingabemethoden

3.2.1 Eingabe mittels Blickrichtung

Die einfachste Methode, das Auswählen eines Objektes mit einem HMD zu gewährleisten, ist die Selektion durch die Blickrichtung, da diese keinerlei zusätzliche Peripherie oder Eingabegeräte erfordert. Somit wurde für diese Eingabemethode auch nur ein Smartphone inkl. einer passenden Halterung benötigt (vgl. Abb. 3.1).

Die Orientierung des Smartphones kann mittels der *DeviceOrientation API* des Browsers ausgelesen werden. Die ausgelesenen Daten beinhalten *Eulersche Winkel* [1] (wie veranschaulicht in in Abb. 3.4). Die drei Winkel bilden eine Menge von intrinsischen Tait-Bryan-Winkeln mit der Drehreihenfolge $Z'X'Y''$ [50][53]. Das bedeutet, bei jeder Rotation anhand der drei übergebenen Winkel wird zunächst davon ausgegangen, dass das Koordinatensystem vom Gerät und der Welt exakt gleich sind. Danach werden die Rotationen nacheinander (1. um die Z-Achse, 2. um die X-Achse und 3. durch die nun verschobene Y-Achse des internen Koordinatensystems des Geräts) ausgeführt, was jedesmal eine Veränderung des Koordinatensystems des Gerätes zur Folge hat.

Es ist anzumerken, dass die Reihenfolge der Drehungen sich jedoch von der unterscheidet, die normalerweise in Bibliotheken wie three.js genutzt wird. Das bedingt, dass bei der direkten Nutzung Werte in 3D-Umgebungen die Drehreihenfolge vom Standard $X-Y-Z$ auf $Z-X-Y$ geändert werden muss. Three.js bietet jedoch einen eigenen Konstruktor für Eulersche Winkel, bei der die Rotationsreihenfolge angegeben werden kann. Im nachfolgenden Codebeispiel wird mit three.js ein *Euler*-Objekt erstellt, welches die Rotation eines 3D-Objekts anhand von Eulerschen Winkeln beschreibt. Da three.js intern mit

[1] Eulersche Winkel beschreiben die Drehung eines Objekts anhand dreier sequentieller Rotationen um die x-, y- und z-Achsen um jeweils die Winkel α, β und γ — Im Falle von Tait-Bryan-Winkeln auch Roll-, Nick- und Gierwinkel (*engl.* „roll“, „pitch“ und „yaw“) genannt.

Bogenmaß (Radian) arbeitet und die Winkel der DeviceOrientation API jedoch in Bogengrad angegeben sind, müssen die Winkel zudem zunächst ins Bogenmaß umgewandelt werden. Auch hierfür hat die Bibliothek jedoch bereits Hilfsfunktionen bereitgestellt.

```
var z = THREE.Math.degToRad( deviceOrientationEvent.alpha ); // Z
var x = THREE.Math.degToRad( deviceOrientationEvent.beta ); // X'
var y = THREE.Math.degToRad( deviceOrientationEvent.gamma ); // Y''

var deviceRotation = new THREE.Euler(x,y,z,'ZXY');
```

Desweiteren kann die aktuelle Orientierung des Geräts (also ob dieses hochkant (*engl.* „portrait mode“) oder quer (*engl.* „landscape mode“) gehalten wird) mittels der *ScreenOrientation API* abgefragt werden. Die Schnittstelle liefert die aktuelle Orientierung des Ausgabegeräts in Textform und kann die Ausprägungen aus der Spalte „Orientierung“ in Tabelle 3.1 annehmen. Dort sind in den Spalten α, β und γ jeweils vermerkt, welche Drehungen die jeweilige Orientierung repräsentieren.

Orientierung	Erklärung	α	β	γ
portrait-primary	Die obere Kante des Geräts zeigt nach oben	0°	90°	0°
portrait-secondary	Die obere Kante des Geräts zeigt nach unten	180°	-90°	0°
landscape-primary	Die obere Kante des Geräts zeigt nach links	0°	90°	-90°
landscape-secodary	Die obere Kante des Geräts zeigt nach rechts	0°	90°	90°

Tabelle 3.1: *Ausgabewerte der Browserschnittstelle ScreenOrientation und deren Bedeutung, sowie der Eulerschen Winkeln die zur gewünschten Drehung führen.*

Ein Nachteil bei der Benutzung von Eulerschen Winkeln ist das Auftreten des sogenannten *Gimbal Lock* [10]: Ist ein Winkel so gewählt (beispielsweise 90°), dass sich in der resultierenden Repräsentation nach der ersten Rotation zwei Achsen decken, so sind diese nicht mehr unterscheidbar. Anschaulicher lässt sich dies anhand eines Flugzeugs beschreiben. Beträgt der Nickwinkel 90° und das Flugzeug zeigt mit der Spitze direkt nach oben, so sind Roll- und Gierwinkel bei der nächsten Drehung nicht mehr unterscheidbar. Man verliert also einen Freiheitsgrad für die Drehung des Objekts. Verhindern kann man diesen Effekt durch die Benutzung von *Quaternionen* [2] zur Repräsentation von der Orientierung eines Objekts im dreidimensionalen Raum [10][37]. Die *WebVR API* nutzt aus diesem Grund Quaternionen für die Repräsentation der Orientierung des HMD und der Eingabegeräte.

Der IV arbeitet für die Wahl des Bildausschnittes jedoch weder mit Quaternionen noch Eulerschen Winkeln, sondern mit *sphärischen Koordinaten*. Diese benötigen nur 2-DOF, und werden mit zwei Winkeln, Polar- und Azimutalwinkel (engl. „latitude“ und „longitude“), angegeben. Intern wird ein konstanter Radius gewählt, der eine Kugel um die Position der Kamera bestimmt. Anhand des Polar- und Azimutalwinkels wird ein Punkt auf der so aufgespannten Kugel ermittelt. Die Blickrichtung der Kamera in der 3D-Szene wird daraufhin in Richtung diesen Punktes gedreht. Die Umrechnung der Drehwinkel um die einzelnen Achsen den Gerätes wir anhand folgenden Codebeispiels aus der Dokumentation des NavVis IndoorViewers [38] vorgenommen:

[2] Quaternionen werden zur Beschreibung der Orientierung eines Objekts im Raum verwendet. Ihre Parameter beschreiben eine beliebige Achse im Raum sowie den Drehungswinkel des Objekts um diese Achse.

```
RAD_OFFSET = THREE.Math.degToRad(90);

if (orient === 'portrait-primary' || orient === 'portrait-secondary') {
   beta = beta - RAD_OFFSET;

   lon = alpha;
   lat = beta;
} else if (orient === 'landscape-secondary') {
   // Winkel gamma ausgleichen
   gamma = gamma < 0 ? -gamma + RAD_OFFSET : gamma - RAD_OFFSET;

   // Grenzfälle wenn direkt nach oben oder unten gesehen wird
   gamma = Math.sign(gamma) ==
       Math.sign(scope.deviceMotion.accelerationIncludingGravity.z) ?
       (-1) * gamma : gamma

   lon = alpha;
   lat = gamma;
} else //if (orient === 'landscape-primary')
{
   // Winkel gamma ausgleichen
   gamma = gamma > 0 ? -gamma + RAD_OFFSET : gamma - scope.RAD_OFFSET;

   // Grenzfälle wenn direkt nach oben oder unten gesehen wird
   gamma = Math.sign(gamma) ==
       Math.sign(scope.deviceMotion.accelerationIncludingGravity.z) ?
       (-1) * gamma : gamma;

   lon = alpha;
   lat = gamma;
}
```

Für den Azimutalwinkel wird also stets α-Wert verwendet, anhand dessen — bei aufrechter Orientierung des Smartphones — die Drehung des Displays nach links und rechts erkannt werden kann. Für die Neigung des Displays nach oben und unten muss hierbei im Hochformat β verwendet werden, im Querformat γ. Daraus ergibt sich, dass eine Neigung um die X-Achse im Welt-Koordinatensystem aus Abb. 3.4 den Nutzer in der VE nach oben bzw. unten sehen lässt, eine Drehung um die Z-Achse hingegen nach links oder rechts.

Die Indikation eines Objektes wird mit dem Blickrichtungs-Vektor des Nutzers bzw der Orientierung des HMDs im Raum gesteuert. Für die Bestätigung der Auswahl kann, im Falle von Halterungen wie dem *Google Cardboard*, durch die Betätigung eines eingebauten Schalters erfolgen, der bei Betätigung mittels einen kleinen Hebels auf die Oberfläche des Display drückt. Bei einigen HMDs, die für Smartphones entworfen wurden, gibt es auch Schalter, die bei Betätigung eine Änderung des Magnetfelds bewirken und somit eine Interaktion anstoßen können. Jedoch sind diese Schalter bei verschiedenen Herstellern unterschiedlich. Im Falle der Magnetschalter gibt es zudem keine einfache Methode, dieses Ereignis im Browser zu erkennen. Zwar kann der, vom Schalter des Google Cardboard ausgelöste Klick im Browser wie ein normaler Druck auf das Display erkannt werden, doch besitzen nicht alle HMDs eine solche Vorrichtung. Die universellste Methode erscheint also die Nutzung einer automatischen Betätigung, die unabhängig vom HMD-Modell funktioniert. Deswegen wurde bei der Entwicklung des Prototypen darauf verzichtet, solche Vorrichtungen zu unterstützen.

Statt dessen wurde als erste Eingabemethode für den Prototypen der Arbeit ein Fadenkreuz auf Basis der Bibliothek *Reticulum* (https://github.com/skezo/Reticulum) implementiert, das sich

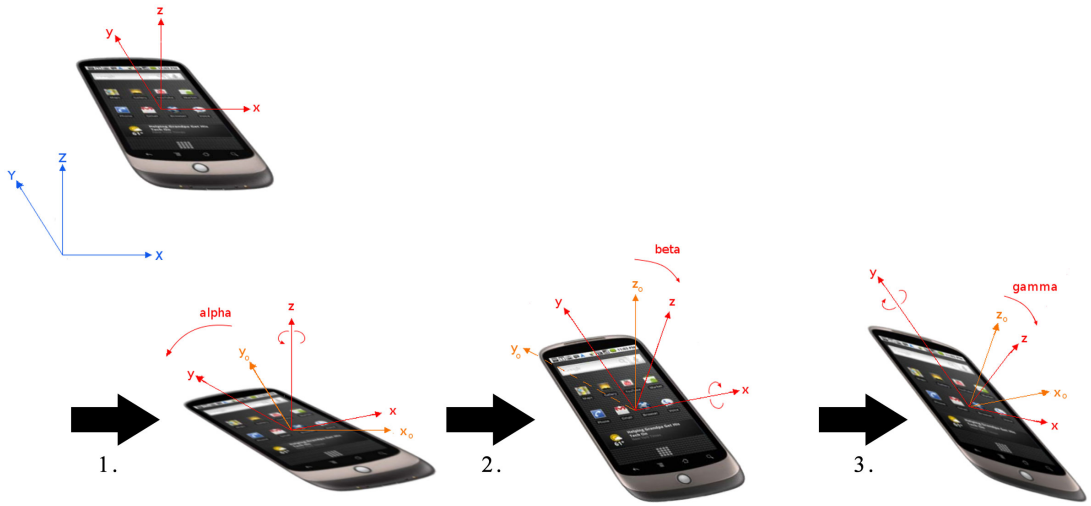


Abbildung 3.4: *Referenzmodell der DeviceOrientation API zur Bestimmung der Drehung des Ausgabegeräts im Browser [50]. Blaue Achsen bezeichnen das Referenz-Koordinatensystem in der realen Welt. Rote und orangefarbene Achsen beziehen sich auf die Achsen des Geräte-Koordinatensystems und sind als x, y, z nach und x_0, y_0, z_0 vor der jeweiligen Rotation angegeben.*

stets in der Mitte des Sichtfeldes des Nutzers befindet. Sieht der Nutzer auf ein selektierbares Objekt, so erscheint ein runder Fortschrittsbalken um den Zeiger (s. Abb 3.5). Abhängig von der Verweildauer (engl. „dwell time“, [20]) wird die dahinterliegende Aktion ausgeführt. Die Vorteile dieser Methode liegen darin, dass

- das Konzept intuitiv ist, beispielsweise eine Schaltfläche zu betätigen, indem man sie lange ansieht,
- keinerlei extra Peripherie außer dem HMD nötig ist und somit
- alle Interaktionen freihändig durchgeführt werden können, was zusätzliche Interaktion für die Hände ermöglicht.

Ein großer Nachteil dieser Methode ist jedoch, dass der Nutzer in seiner Freiheit beschränkt wird, sich umzusehen. Auch in der durchgeführten Benutzerstudie bemängelte ein Proband, dass er sich sorgte, ungewollt Funktionalität auszulösen, indem er virtuelle Objekte ansah.

3.2.2 Freihändige Interaktion mit dem Leap Motion-Controller

Als zweite Eingabemethode wurde untersucht, ob Kameras mit Tiefeninformation genutzt werden können, um die Hände des Nutzers zu erkennen und in der VE darzustellen. Besonders naheliegend war hier die Benutzung der Leap Motion-Kamera, da diese durch die Nutzung von Websockets bereits eine Anbindung an Browser basierte Systeme bietet. Zwar bestand die Aufgabenstellung dieser Arbeit daraus, für die Eingabemethoden möglichst keine zusätzliche Peripherie außer Smartphones zu verwenden, jedoch lassen aktuelle Entwicklungen darauf schließen, das solche Hardware bereits in naher Zukunft in Mobilgeräten verfügbar sein wird. Es wurde bereits angekündigt, dass Intel an der Integration von 3D-Kameras in Smartphones arbeitet [43] und diese bereits zur Vorbestellung verfügbar sind.

Für die Tests von Eingabemethoden, die eine virtuelle Hand nutzen, wurde im Zuge dieser Arbeit jedoch nur die Leap Motion-Kamera verwendet. Diese muss jedoch an einen Desktop-PC

Abbildung 3.5: *Die für den Prototypen genutzte Anzeige zur Steuerung mittels Blickrichtung. Der graue Kreis in der Mitte des Blickfelds zeigt den Cursor, wenn kein Objekt vom dahinterliegenden Raycast geschnitten wurde (links) und den Ladebalken zur Auslösung der Bewegung, falls ein Objekt geschnitten wurde (rechts).*

angeschlossen werden, um schließlich über einen lokalen Webserver eine Websocket-Verbindung aufbauen zu können. Über diese können die Daten über aufgezeichneten Handpositionen an das Smartphone gesendet werden. Dazu muss jedoch zunächst das *Leap Motion SDK* installiert sowie konfiguriert werden, damit eine Websocket-Verbindung zu einem anderen Gerät (in diesem Fall dem Smartphone) aufgebaut werden kann. Das SDK liefert dann eine Lösung für die Implementierung der virtuellen Hand in einer 3D-Szene (s. Abb. 3.6).

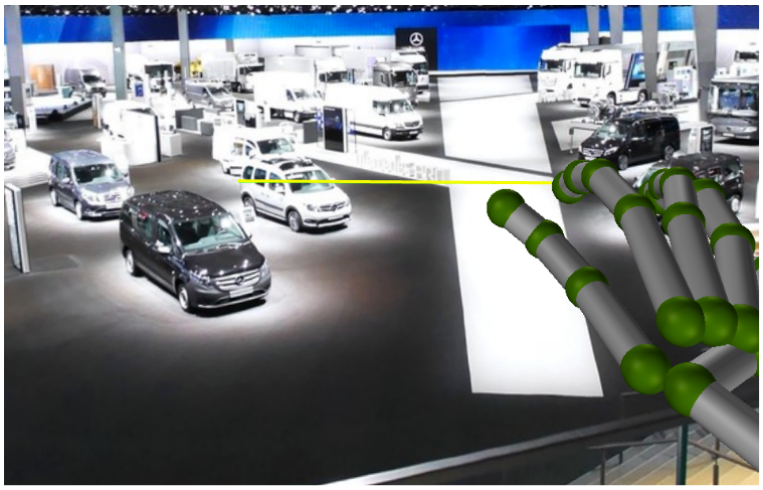


Abbildung 3.6: *Die für den Prototypen genutzte Anzeige zur Steuerung mittels virtueller Hand.*

In den ersten Tests war jedoch ersichtlich, dass mehrere Probleme bei der Verwendung aufkommen können:

- Es ist keine direkte Verbindung der Leap Motion-Kamera mit dem Smartphone möglich, so dass diese nur mit einem Desktop-PC verwendet werden kann.
- Die Verdeckung der Finger untereinander sorgte für eine ungenaue Erkennung der Hand. Die fehleranfällige Erkennung der Hand schien nicht geeignet für Aufgaben, die eine hohe Präzision erfordern. Da die Selektion der Ortspunkte in der vorgegebenen VE, besonders auf hohe Distanz, eine hohe Präzision erfordert, erschien die Verwendung nicht sinnvoll.

- Neben der fehlerhaften Erkennung traten auch hohe Latenzen auf, die dafür sorgten, dass die Repräsentation der Hand in der VE oft erst wenige Sekunden nach der eigentlichen Bewegung der Hand aktualisiert wurde. Ein Grund dafür könnte die Verwendung von WebSockets sein. Für die hochfrequente Aktualisierung kontinuierlicher Daten ist der häufige Auf- und Abbau von TCP-Verbindungen eine Quelle für Verzögerungen. Ein kontinuierlicher Datenfluss wie bei der Verwendung von WebRTC könnte Abhilfe schaffen, jedoch bieten die Softwarepakete der Leap Motion keine Option eine WebRTC-Verbindung statt einer WebSocket-Verbindung zu nutzen.

Die Aufgabenstellung setzte vor allem die Evaluation von Eingabemethoden voraus, die ohne zusätzliche Peripherie genutzt werden können, um die Navigation im IV zu erlauben. Da die Interaktion mittels virtueller Repräsentation der Hände eher für Aufgaben wie die Manipulation von Objekten geeignet sind (und aufgrund der beschriebenen Probleme bei der Implementierung) wurde die virtuellen Hand als Eingabemethode nicht weiter verfolgt.

3.2.3 Zweites Smartphone als virtuelles Zeigegerät

Da virtuelle Zeigegeräte eine verbreitete Eingabemethode in aktuellen VEs sind und aus Studien des statistischen Bundesamts hervorgeht, dass in den meisten Haushalten mehr als ein Smartphone vorhanden ist [45], wurde für den Prototyp besonderen Wert auf die Nutzung eines zweiten Smartphones als Eingabegerät gelegt. Der Vorteil hierbei liegt vor allem darin, dass in Smartphones bereits Tracker vorhanden sind, um dessen Orientierung im Raum zu messen. Die Hauptaufgabe bei der Implementierung eines virtuellen Zeigegeräts ist die Erkennung der Orientierung des Geräts. wie in Abschnitt 3.2.1 beschrieben. Dafür kann die *DeviceOrientation API* des Browsers genutzt werden, um diese zu erkennen.

In Hinsicht auf die Verwendung der entwickelten Eingabemethoden mit unterschiedlichen VR-Systemen, die vom WebVR-Standard unterstützt werden (u.a. Systeme wie HTC Vive oder Google Daydream), ist die Selektion durch einen Raycast naheliegend, da diese in vergleichbaren Systemen auch oft verwendet werden, um Objekte auszuwählen.

Für die Implementierung des zweiten Smartphones als virtuelles Zeigegerät wurde innerhalb der Anwendung eine weitere Adresse angelegt. Mit dem Smartphone, welches als HMD fungiert, muss zunächst auf die Webadresse navigiert werden, die die virtuelle Umgebung lädt und anzeigt. Mit dem zweiten Smartphone kann nun die Adresse *<Webadresse>/control* im Browser aufgerufen werden. Dort wird eine simple Bedienungsanleitung angezeigt, die kurz die Benutzung des virtuellen Zeigegeräts beschreibt (Abb 3.7).

Abbildung 3.7: *Ausgabe auf dem Smartphone, welches als virtuelles Zeigegerät genutzt wird*

Sowohl das Smartphone, welches zur Anzeige benutzt wird, als auch das Smartphone welches als virtuelles Zeigegerät genutzt wird, verbinden sich daraufhin mittels einer WebSocket-Verbindung zum angegebenen WebServer, der die Parameter für die Peer-to-Peer Verbindung zwischen ihnen aushandelt (*Signaling*). Ein vereinfachter Ablauf ist in Abb. 3.8 dargestellt. Für dieses Protokoll wurde die Bibliothek *socket.io* (https://socket.io/blog/socket-io-p2p/) genutzt, die bereits die Funktionalität zur Herstellung einer WebRTC-Verbindung anbietet. Die Bibliothek socket.io muss dafür zunächst im genutzten

Webserver (im Falle dieses Prototypen ein lokaler *Node.js*-Server) initialisiert werden werden. Daraufhin haben alle Endgeräte, die auf eine Adresse des Servers navigieren, Zugriff auf die Funktionalität der Bibliothek und somit auf WebSocket-Funktionen. Über den so aufgebauten WebSocket-Kanal muss nun eines der verbundenen Geräte eine Peer-to-Peer-Sitzung beantragen, woraufhin der Server an die anderen verbundenen Geräte die ausgehandelten Parameter sendet, um schließlich eine direkte Verbindung zwischen den Teilnehmern aufzubauen.

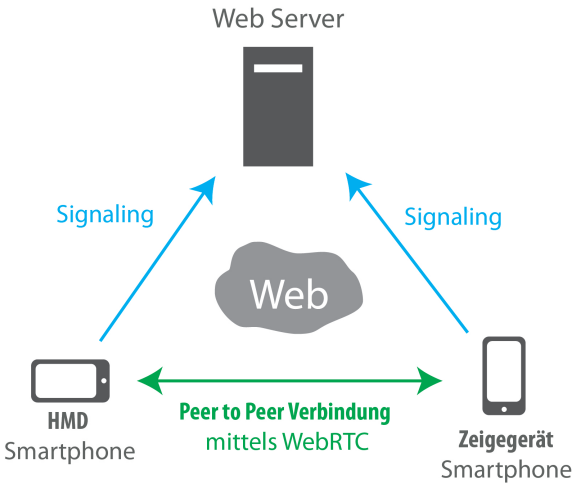


Abbildung 3.8: *Verbindungen zwischen den beiden genutzten Smartphones*

Das Smartphone, welches als Zeigegerät fungiert sendet daraufhin jedesmal, wenn es sich bewegt oder gedreht wird, seine aktuelle Orientierung an das HMD-Smartphone. Dort werden die empfangenen Daten genutzt, um ein virtuelles Zeigegerät analog zur Rotation des Smartphones in der Hand des Nutzers zu drehen.

Da die Position des Smartphones in der Hand des Nutzers jedoch nicht ermittelt werden kann, wird dessen Repräsentation in der 3D-Szene immer mittig unter der Kamera dargestellt. Von dieser Position aus wird nun ein sichtbarer Strahl in die VE gesendet, der beim Auftreffen auf den Boden der VE oder einen Ortspunkt einen runden Indikator an der Stelle zeigt, an der der Strahl auf das jeweilige Objekt getroffen ist. Dies soll der besseren Orientierung der Nutzer beim präzisen Zielen dienen. Wird ein Ortspunkt direkt geschnitten, ändert sich die Farbe des Strahls sowie des Indikators. Das dient dazu, um dem Nutzer zu signalisieren, dass ein Punkt getroffen wurde (s. Abb. 3.9). Wird einmalig auf das Display des Zeigegerät-Smartphones gedrückt, so sendet es zusätzlich zu den Orientierungsdaten auch noch einen konkreten Befehl, der dem HMD-Smartphone signalisiert, dass eine Bewegung zum ausgewählten Ortspunkt erfolgen soll.

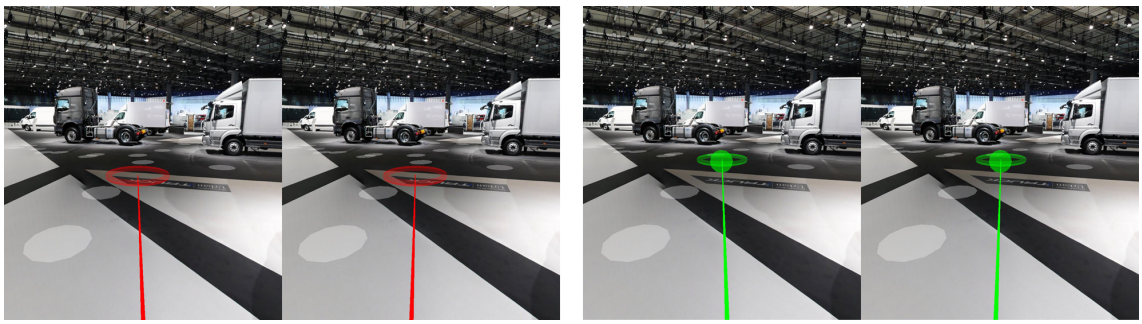


Abbildung 3.9: *Die für den Prototypen genutzte Anzeige zur Steuerung mittels virtuellem Zeigegerät (gerader Raycast). Links sieht man den roten Strahl, wenn kein Ortspunkt vom Raycast geschnitten wurde und rechts die Anzeige, falls doch.*

Da sich die Nutzer während dem Umsehen in der VE unter Umständen drehen müssen und nicht garantiert werden kann, dass es keine Abweichung zwischen den gemessenen Werten der Orientierung des Zeigegeräts und des HMDs gibt, kann es passieren, dass sich der Strahl nicht mehr im Blickfeld des Nutzers befindet. Dafür wurde ein weiterer Befehl implementiert, der ausgelöst werden kann, indem das Display des Zeigegeräts länger als 2s gedrückt wird. Daraufhin

wird das Objekt, welches das Zeigegerät in der VE repräsentiert, automatisch wieder parallel zur Blickrichtung des Nutzers gedreht.

Neben dem normalen Raycast, der vom digitalen Zeigegerät in die VE gesendet wird, wurde insbesondere für Ortspunkte, die auf höheren Punkten in der VE liegen, noch eine weitere Art des Raycasts implementiert. Diese weist in Anlehnung an die Teleport-Funktion von Anwendungen der HTC Vive eine parabolische Kurve auf (s. Abb. 3.10). Dies soll ermöglichen, Objekte zu selektieren, die mit einem geraden Raycast schwer zu erreichen sind. Hierfür werden schrittweise entlang einer parabolischen Kurve Raycasts ausgesendet, die beim ersten Auftreffen auf ein Hindernis den genauen Punkt der Kollision liefern.

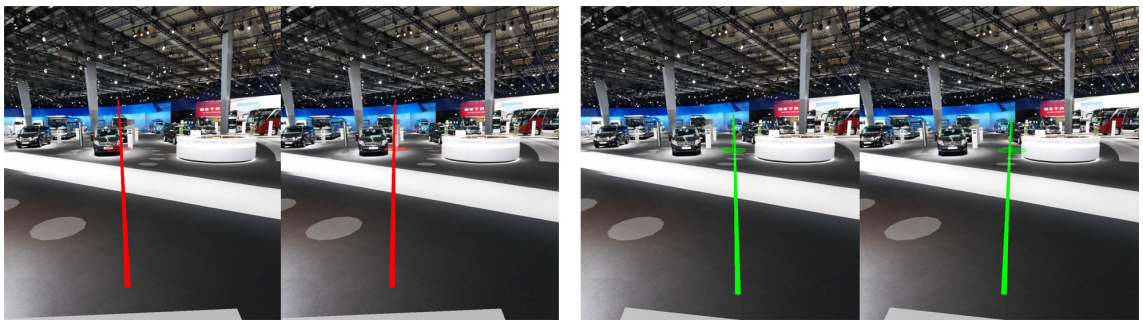


Abbildung 3.10: *Die für den Prototypen genutzte Anzeige zur Steuerung mittels virtuellem Zeigegerät (parabolischer Raycast). Links sieht man den roten Strahl, wenn kein Ortspunkt vom Raycast geschnitten wurde und rechts die Anzeige, falls doch.*

Da jedoch in der genutzten Version des IV (bis auf eine große Bodenplatte, die auf Höhe der Ortspunkte mit der kleinsten z-Koordinate liegt, und den Ortspunkten selbst) keine echte Geometrie vorhanden war, gab es keine Objekte, die den Strahl blockieren können.

Die Vorteile des implementierten virtuellen Zeigegeräts liegen darin, dass es (ähnlich wie die Steuerung mittels Blickrichtung) vollständig Browser basiert implementiert werden konnte, ohne die Notwendigkeit, weitere Peripherie zu verwenden. Der erwartete Vorteil gegenüber der Selektion zur Blickrichtung wurde vor allem darin gesehen, dass die Interaktion entkoppelt von der Blickrichtung möglich ist. Das erlaubt dem Nutzer, sich umzusehen, ohne ungewollt die Navigation zu Ortspunkten auszulösen. Insbesondere in Hinblick auf die Betrachtung und Informationsgewinnung über die in der VE verteilten POIs erschien dies als eine geeignete Basis für die Weiterentwicklung des Prototypen.

Im folgenden Kapitel wird beschrieben, auf welche Kriterien der erstellte Prototyp untersucht wurde und wie diese Kriterien mit Hilfe einer Nutzerstudie erhoben wurden. Von besonderem Interesse war die Untersuchung auf die Gebrauchstauglichkeit (engl. „Usability“) der implementierten Eingabemethoden.

3.3 Evaluationskriterien

Gebrauchstauglichkeit (*engl.* „Usability“): Die Gebrauchstauglichkeit einer Anwendung wird von der Norm ISO 9126 als Gütekriterium von Softwareprodukten genannt. Insofern war die Benutzerfreundlichkeit des Prototypen für die Evaluation der entwickelten Bedienkonzepte von Interesse. Die ISO Norm 9241 bestimmt drei Leitkriterien für die Gebrauchstauglichkeit einer Software:

- Effektivität zur Lösung einer Aufgabe,
- Effizienz der Handhabung des Systems,

- Zufriedenheit der Nutzer einer Software. [52]

Aus diesen Leitkriterien können Ziele für die Gebrauchstauglichkeit abgeleitet werden, zum Beispiel Fehlertoleranz, schnelle Erledigung bestimmter Aufgaben, leichte Erlernbarkeit oder die reduzierte Fehlerrate bei der Bedienung. Justin Mifsud empfiehlt auf seiner Website „UsabilityGeek“ das Vorgehen zum Messen wichtiger Usability-Kenngrößen[33]. Somit wurden als abhängige Variablen für den erstellten Prototyp die Anzahl der Fehler beim Navigieren von einem Ortspunkt zum nächsten sowie die Zeit zum erfolgreichen Navigieren aufgenommen.

Am Ende jeder Aufgabe musste von jedem Probanden ein Fragebogen ausgefüllt werden, um die Zufriedenheit des Nutzers mit der Methode aufzunehmen. Standardbeispiele für solche Fragebögen sind laut [33] z.B. folgende:

- ASQ - After Scenario Questionnaire (3 Fragen) [30]
- NASA TLX - NASA Task Load Index (5 Fragen) [6]

Für die Durchführung der Nutzerstudie wurde aufgrund der einfachen Durchführung der ASQ gewählt. Der Vorteil dieses Fragebogen besteht in der kurzen Beantwortungszeit, da dieser anhand dreier kurzer Fragen Daten über die Zufriedenheit des Nutzers mit dem System erhebt. Jede der drei Fragen konnte damit auf einer Skala von 1 (stimme überhaupt nicht zu) bis 7 (stimme voll zu) beantwortet werden. Aus den daraus resultierenden Werten kann der Wert für die Nutzerzufriedenheit errechnet werden, indem das arithmetische Mittel aller Einzelantworten gebildet wird. Da durch das Design der Studie jede Sitzung bereits ca. 25-40 Minuten in Anspruch nahm, sollten die Fragebögen so schnell wie möglich zu beantworten sein.

Normalerweise besteht der NASA-TLX Frgaebogen aus zwei Schritten. Zunächst muss auf einer fünf- oder siebenstufigen Likert-Skala der benötigte mentale wie physische Aufwand zur Lösung einer Aufgabe bewertet werden. Der zweite Schritt besteht aus einer paarweisen Gewichtung der einzelnen Faktoren. Um die Zeit für die Beantwortung der Fragebögen so kurz wie möglich zu halten, wurde aber die kurze Version des NASA-TLX Fragebogens, der sog. *Raw TLX*, verwendet [6]. Bei diesem wird die Gewichtung der einzelnen Faktoren ausgelassen.

Da die Cyber Sickness ein relevanter Punkt bei der Evaluierung von immersiven VR-Systemen ist, war die Erhebung von Indikatoren dafür auch Bestandteil der durchgeführten Nutzerstudie. Jeweils vor und nach den Navigationsaufgaben werden den Nutzern ebenfalls Fragebögen zu Ihrem Wohlbefinden vorgelegt. Um Veränderungen des Wohlbefindens zu erkennen, die aus der Benutzung des VR-Systems resultieren, muss der Fragebogen jeweils vor der erstmaligen Benutzung und erneut nach Abschluss des Experiments beantwortet werden.

Ein Fragebogen, der zu diesem Zweck verwendet wird, ist der Fragebogen zur Quantifizierung von Effekten der Simulatorkrankheit (*engl.* „Simulator-Sickness-Questionnaire“, SSQ). [25] Wir verwendeten dabei eine leicht abgewandelte Version der Universität Hong Kong [22] für die Nutzung in VR-Szenarien. Erhoben wurden hierbei die Intensität verschiedener Symptome die zu drei verschiedenen Kategorien zugeordnet werden können:

- Übelkeit
- Beeinträchtigung des Augenapparates
- Desorientierung

In Tabelle 3.2 sind die drei berechneten Teilwerte aufgeführt, sowie die Symptome des Fragebogens, welche für die Berechnung genutzt wurden und deren Gewichtung für den Gesamtwert laut [22]. Die Probanden sollten pro Symptom jeweils markieren, ob und in welcher Intensität die Symptome auftreten. Die einzelnen Intensitäten konnten angegeben werden als *gar nicht*(0), *ein wenig*(1), *moderat*(2) und *schwerwiegend*(3).

Teilwert	Symptome	Gewichtung
Übelkeit	Generelles Unwohlsein, erhöhter Speichelfluss, Schweißbildung, Übelkeit, Konzentrationsschwierigkeiten, Magenprobleme, Aufstoßen	9,54
Augenapparat	Generelles Unwohlsein, Müdigkeit, Beanspruchung der Augen, Schwierigkeiten beim Fokussieren, Konzentrationsschwierigkeiten, verschwommene Sicht	7,58
Desorientierung	Schwierigkeiten beim Fokussieren, Übelkeit, „Fullness of Head“, verschwommene Sicht, Schwindel (Augen geschlossen / offen), Vertigo	13,92

Tabelle 3.2: *Erhobene Messwerte für die einzelnen Teilwerte der Simulatorkrankheit in einer VE und deren Gewichtung für den Gesamtwert nach [22].*

3.3.1 Studiendesign

Üblicherweise werden in Nutzerstudien zur Wahrung von interner und externer Validität Gruppen von Probanden verschiedenen Konditionen ausgesetzt und mit einer Kontrollgruppe verglichen. Um jedoch mit einer kleineren Population gute Ergebnisse für die Gebrauchstauglichkeit des entwickelten Prototypen zu ermitteln, benutzen wir einen sog. Innersubjekt-Ansatz (engl. „Within-Subjects“)[41]. Das bedeutet, dass jede Interaktionsmethode von jedem Probanden einzeln getestet wird, statt Gruppen zu bilden, die die Aufgaben je unter verschiedenen Konditionen lösen müssen. Dieser bietet gegenüber Gruppen-basierten Ansätzen folgende Vorteile :

- Es ist eine kleinere Gesamtpopulation nötig.
- Es ist eine einfachere Bewertung der einzelnen Eingabemethoden für Nutzer möglich, da sie alle verfügbaren Eingabemethoden benutzen.

Diese Methode birgt jedoch auch Risiken für die Validität der Studie.

- Lerneffekt beeinflusst Messergebnisse
- eine relative Bewertung lässt unbeliebte Methode in den Zahlen schlechter erscheinen (zumindest bei qualitativem Feedback)
- erhöhte Sitzungsdauer pro Proband

Am wahrscheinlichsten beeinflusst der Lerneffekt die Validität der quantitativen Datenerhebung. Der Effekt kann jedoch durch einen Gewichtungsausgleich (engl. „counterbalancing“) geschmälert werden. Die Nutzer werden nacheinander alle Methoden testen, jedoch in unterschiedlicher Reihenfolge, sodass Lerneffekte im Umgang mit dem IV sich gleichermaßen auf die einzelnen Sitzungen verteilt.

In jeder Sitzung wird dem Nutzer ein HMD aufgesetzt und die Aufgabe gestellt, sich nacheinander in der VE zu festgesetzten Punkten zu bewegen. Der jeweils nächste Zielpunkt wurde dem Nutzer durch Markierung eines Ortspunktes im Umfeld angezeigt (s. Abb. 3.11) Sowohl die Umgebung als auch der Ablauf bleibt hierbei gleich, um möglichst viele Einflüsse auf die Messung durch veränderte Bedingungen auszuschließen. Die unabhängige Variable soll nur die Eingabemethode sein, welche folgende Ausprägungen haben:

- Selektion von Ortspunkten durch Blickrichtung (*engl.* „Gaze“)
- Selektion von Ortspunkten durch ein zweites Smartphone, welches in der 3D-Szene wie ein Laserpointer fungiert

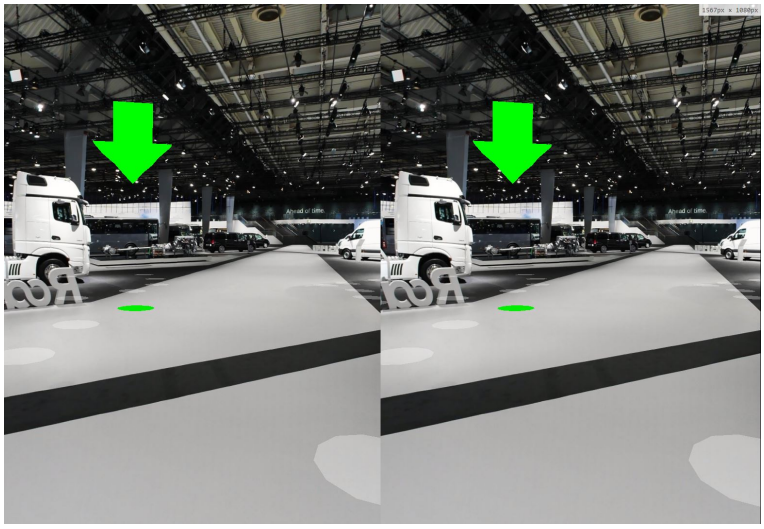


Abbildung 3.11: *Die für die Nutzerstudie genutzte Markierung des jeweils nächsten Zielpunktes*

- Selektion von Ortspunkten durch ein zweites Smartphone, welches in der 3D-Szene wie ein Laserpointer fungiert, der eine parabolischen Strahl aufweist.

Die abhängigen Variablen zur Erhebung der quantitativen Daten waren die Abschlussrate und die benötigte Zeit, um sich anhand der vorgeschriebenen Route durch die VE zu navigieren. Die Zeiten wurden jeweils pro Navigationsschritt erhoben. Da sich die einzelnen Ortspunkte der Route durch den Ablauf und die genutzte Instanz des IV bedingt in Distanz und Höhengefälle unterschieden, diente dies vor allem der detaillierteren Problemerkennung, falls in den erhobenen Daten Inkonsistenzen auftreten. Vor allem, da sich auf der Route eine Treppe befand, sollte anhand der Daten noch erkennbar sein, wie viel Zeit der Nutzer benötigte, um dieses Hindernis zu bewältigen.

Weitere Messungen, die für die Evaluation von Interesse sind, sind die Fehler, die beim Lösen der Aufgabe vom Probanden begangen werden. Diese Fehler lassen sich in zwei Kategorien aufteilen: in sog. *Ausrutscher* (engl. „slips“) und *Fehler* (engl. „mistakes“)[42].

- *Mistakes* sind hier zu verstehen als Aktionen, die eine falsche Intention des Nutzers beinhalten. D.h. der Nutzer versucht Aktionen durchzuführen, die nicht direkt der gestellten Aufgabe entsprechen. Ein Beispiel hierfür wäre auf einer herkömmlichen Website, dass ein Nutzer versucht, eine Überschrift zu klicken, hinter der sich jedoch gar kein Link verbirgt. Diese Art von Fehler wurde auf den Kontext der Selektion von Ortspunkten so übersetzt, dass jede Navigation zu einem Ortspunkt, der nicht als Zielpunkt angegeben war, als Fehler gewertet wird.

- *Slips* werden als solche Fehler bezeichnet, die zwar auf einer korrekte Intention des Nutzers im Bezug auf die Aufgabenstellung basieren. Jedoch wird bei der Durchführen einer eigentlich korrekten Aktion ein Fehler begangen, der verhindert, dass die Aufgabe erfolgreich bewältigt werden kann. Am Beispiel einer 2D-GUI wäre dies, wenn ein Nutzer versucht, eine gekennzeichnete Schaltfläche zu betätigen, jedoch durch fehlerhafte Betätigung nicht in der Lage ist, die dahinter liegende Funktionalität auszulösen. Im Fall dieser Studie wurde diese Art Fehler so übersetzt, dass jeder Verlust des Fokus auf das zu selektierende Objekt bei der Selektion von Ortspunkten als ein solcher Fehler gezählt wird. Das bedeutet, es wurde jedes Mal ein Fehler gezählt, wenn der Proband zwar den richtigen Punkt fokussiert hat, jedoch bevor er diesen aktivieren konnte, den Fokus auf das Objekt wieder verloren hat (beispielsweise durch leichte Kopfbewegungen im Falle der Steuerung durch Blickrichtung).

Um die benötigte Zeit der Nutzersitzungen auf ein Minimum zu beschränken, wurde die Zahl der Navigationsschritte pro Ausprägung der unabhängigen Variable auf zehn Schritte festgelegt. Die einzelnen Schritte unterschieden sich hier in Distanz, Dichte der umliegenden Punkte und Höhengefälle. Um dennoch einzelne Abschnitte untereinander vergleichen zu können, wurden die Messungen hierbei jeweils pro Navigationsschritt erhoben.

4 Resultate

Insgesamt nahmen 29 Nutzer an der Studie teil (n = 29). Davon waren 15 männlich und 14 weiblich. Der jüngste Teilnehmer war 14 Jahre alt, der älteste 60 (Mittelwert: 34,62). Neun Sitzungen fanden bei einem Software-Unternehmen in München statt, weitere neun Sitzungen in einem in München ansässigen Verlag für interaktive Medien und elf Sitzungen am Lehrstuhl für Medientechnik der Technischen Universität München. Die durchgeführte Nutzerstudie wurde an den drei verschiedenen Orten mit den dort verfügbaren Smartphones durchgeführt. Für alle Sitzungen an einem Ort wurden jeweils dieselben Geräte benutzt.

4.1 Ergebnisse der Nutzerstudie

Proband #5 musste im voraus von der Datenanalyse ausgeschlossen werden, da dieser bereits nach sehr kurzer Benutzungszeit das Experiment unterbrechen musste. Laut eigenen Angaben reagierte er sehr empfindlich auf die Verzögerung, die zwischen der Kopfbewegung und der resultierenden grafischen Ausgabe auf dem HMD bestand. Auch im vorgelegten Fragebogen zur Simulatorkrankheit erzeugte dieser Proband einen Extremwert. In Abb. 4.13 ist dessen Wert als größter Ausreißer mit einer Differenz von 500 Punkten auf der SSQ-Skala ersichtlich.

Um die Effizienz der implementierten Methoden zu überprüfen, wurden die Zeiten gemessen, die jeder Nutzer pro Navigationsschritt benötigte. Um herauszufinden, ob sich die einzelnen Abschlusszeiten statistisch signifikant unterscheiden, wurden, soweit möglich, sog. ANOVA (engl. „Analysis of Variance“)-Tests mit wiederholten Messungen durchgeführt. Dieser findet vor allem Anwendung, wenn ein Innersubjekt-Studiendesign (s. Kapitel 3.3) verwendet wird. Hierbei wird jedem Probanden jede mögliche Ausprägung der unabhängigen Variable präsentiert.

Zur Untersuchung der statistischen Signifikanz von Auswirkungen einer unabhängigen auf eine abhängige Variable ist es hierbei zunächst nötig, dass die Daten folgenden Ansprüchen genügen:

- Es existiert nur eine abhängige Variable, dessen Werte stetiger Natur sind. Im Falle der Analyse der erstellten Bedienkonzepte sind dies hier Variablen wie die Abschlusszeit, die Fehler bei der Bedienung, oder die Messung der erhobenen qualitativen Daten wie der Nutzerzufriedenheit.
- Es existiert eine nominale unabhängige Variable, hier die Eingabemethode.

Da der ANOVA-Test ein parametrischer Test ist, sind nachfolgende Annahmen über den Datenbestand zu analysieren. Sollten diese alle nicht zutreffen, so wird von [27] empfohlen, einen nicht-parametrischen Test zur Bestimmung der Signifikanz der Unterschiede zwischen Eingabemethoden zu wählen. Hier wurde dafür Friedmans zweifaktorielle Varianzanalyse nach Rang bei verbundenen Stichproben [26] genutzt:

- Es existieren für keine Ausprägung der unabhängigen Variable (Eingabemethode) extreme Ausreißer in den Messungen der abhängigen Variable.
- Die Ausprägungen der unabhängigen Variable sollte annähernd normalverteilt sein. Der ANOVA-Test ist jedoch laut [27] recht robust, auch wenn die Daten nicht normal verteilt sind. Es muss angemerkt werden, dass durch die geringe Stichprobenzahl, die Tests für die Prüfung auf Normalität unter Umständen nicht aussagekräftig sind. Der gewählte statistische Test zur Bestimmung der Normalverteilung der Daten war hier der Test nach Shapiro-Wilk.
- Es existiert *Homoskedastizität* (auch: *Sphärizität*) zwischen den einzelnen Stufen der unabhängigen Variable. Ist diese gegeben, sind die Varianzen der Differenzen aller Stufen der

unabhängigen Variablen gleich. Zum Prüfen der Sphärizität kann der Test von Mauchly [13] (S. 160, 186) verwendet werden.

4.1.1 Analyse der quantitativen Daten zur Effizienz

Zunächst wird von [27] empfohlen, den Datensatz auf Ausreißer hin zu überprüfen. Idealerweise sollten sich in den Daten für einen ANOVA-Test keinerlei Ausreißer befinden. Um eine valide ANOVA zu gewährleisten, müssen solche Ausreißer aus den Datensätzen entfernt werden. Durch die Entfernung der Datensätze, welche in mindestens einer der Szenarien als Ausreißer erkannt werden, sinkt die Zahl der analysierten Datensätze jedoch auf n=22, was der Reduzierung der Gesamtpopulation von ca. 25% entspricht. Das bedeutet: 25% der Probanden haben bei mindestens einer der Eingabemethoden ungewöhnlich viel Zeit benötigt. Um die geringe Gesamtpopulation nicht weiter zu verkleinern, werden die Methoden beschrieben, die bei der Datenanalyse genutzt wurden, um so viele Datensätze wie möglich in die Analyse einfließen zu lassen.

Erfolgreicher Abschluss: Zur Messung der Effektivität der einzelnen Eingabemethoden wurde für jede Ausprägung der Eingabemethode notiert, ob der Proband mit allen Eingabemethoden die Route durch die VE vollständig absolvierte. An der Stelle der Route, an der die Ortspunkte auf einer Treppe lagen und die Probanden diese mit der jeweiligen Eingabemethode erklimmen mussten, scheiterte ein Proband bei der Selektion mit dem parabolischen Raycast und zwei Probanden bei der Selektion mit dem regulären Raycast. Ein Proband war, wie bereits zuvor beschrieben, nicht in der Lage, auch nur eine einzige Aufgabe zu bewältigen, da er das Experiment schon nach der ersten Minute in der VE abbrechen musste. Somit ergeben sich die in Tabelle 4.1 aufgeführten Abschlussraten der einzelnen Aufgaben. Da sich alle Abbrüche außer dem im Falle von Proband #5 an dem Treppenabsatz der Route ereigneten, werden im Folgenden die Abschlusszeiten sowohl für die benötigte Gesamtzeit, als auch für die Zeit zur Selektion aller Ortspunkte aufgeführt, die sich auf ebenem Grund befanden. Da die Daten erst zum Ende jeder Eingabemethode gesammelt in eine Datenbank geschrieben wurden, fallen Probanden, die die gestellte Aufgabe mit einer der drei Eingabemethoden nicht lösen konnten, komplett aus der Analyse der Gesamtzeiten heraus.

Eingabemethode	Abschlussrate
Blickrichtung	96,5%
Raycast	89,65%
Raycast (gebogen)	92,1%

Tabelle 4.1: *Anteil der Probanden, die mit der gegebenen Eingabemethode erfolgreich die komplette vorgegebene Route abschließen konnten.*

Abschlusszeiten: Die Zeit, die benötigt wurde, um mit einer bestimmten Eingabemethode die vorgegebenen Ortspunkte zu selektieren und sich somit durch die VE zu bewegen, ist für die Evaluation der Eingabemethoden eine der wichtigsten Metriken. Im Fall der erhobenen Abschlusszeiten gab es in den Daten einige, mitunter extreme, Ausreißer. Für die grafische Analyse und Erkennung der Ausreißer kann ein Boxplot genutzt werden. In Abb. 4.1 ist erkennbar, dass gerade für die Selektion mit der Blickrichtung einige Probanden als Ausreißer klassifiziert werden können. Die Y-Achse zeigt die Abschlusszeiten in Millisekunden (ms). Auch bei dem Bedienkonzept mit dem zweiten Smartphone als virtuelles Zeigegerät mit gebogenen Strahl benötigte einer dieser Probanden ungewöhnlich viel Zeit.

Im Falle des Probanden #11 war dies mit Desorientierung zu erklären. An einem Punkt benötigte er geraume Zeit um zum nächsten Zielpunkt zu navigieren. dies war auch erkennbar an der Zahl an Fehlern, die er beim Navigieren machte. Er bewegte sich zunächst zu fünf anderen

Ortspunkten, bevor er den Indikator für den richtigen Zielpunkt fand und der festgesetzten Route folgte. Dies gilt ebenfalls für den Probanden #13.

Für den gewählten ANOVA-Test ist die Normalverteilung der Daten ebenfalls eine wichtige Annahme. Wie in Tabelle 4.2 zu erkennen ist, wurde ein Shapiro-Wilk-Test durchgeführt, um die Normalverteilung der Daten zu überprüfen. Mit einem p-Wert[3] $<$ 0.05 wird die Nullhypothese der Normalverteilung der Daten abgelehnt — das bedeutet, die Daten sind nicht genügend normalverteilt. Die Verteilung der Daten ist auch in der Histogrammdarstellung in Abb. 4.2 erkennbar.

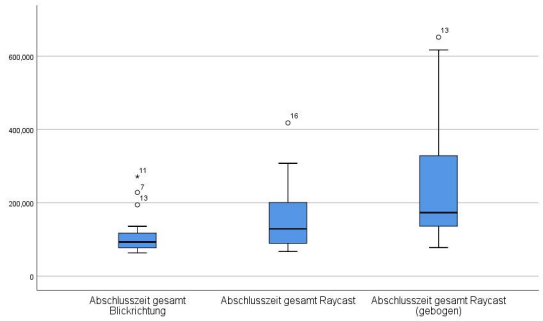


Abbildung 4.1: *Abschlusszeiten (in ms) der Probanden (n=28) nach Eingabemethode für die gesamte Route.*

Jedoch muss beachtet werden, dass bei einer kleinen Anzahl an Probanden die Verteilung oft nicht genau erkennbar ist [13] (S. 160). Zwar herrscht für die beiden linken Histogramme, also die Verteilungen der Zeiten für die Bedienung mit der Blickrichtung und dem gebogenen Raycast durchaus Ähnlichkeit zu einer Normalverteilung, jedoch wirkt das Histogramm für die Abschlusszeiten unter der Bedingung des normalen Raycasts eher linear monoton fallend.

Der Mauchly-Test auf Sphärizität war für die Abschlusszeiten der gesamten Route signifikant($p = 0,003 < 0,5$) und somit wird die Annahme auf Sphärizität der Daten abgelehnt. Nach [13] (S. 186) können zwar Korrekturen vorgenommen werden, um sicherzustellen, dass die Ergebnisse nicht zu falschen Schlussfolgerungen führen, jedoch wurde aufgrund der Verletzung zweier wichtiger Annahmen für den ANOVA-Test für die Abschlusszeiten der gesamten Route ein anderer nicht-parametrischer Test gewählt.

Da für parametrische Tests eine Normalverteilung vorausgesetzt wird, wird von [27] empfohlen einen nicht-parametrischen Test, wie Friedmans zweifaktorielle Varianzanalyse nach Rang bei verbundenen Stichproben [26] zu verwenden. Die Analyse mit diesem Test ergibt, dass die Abschlusszeiten mit der Nutzung verschiedener Eingabemethoden sich statistisch signifikant unterscheiden, $\chi^2(2) = 24000, p < 0,0005$. Es wurden paarweise Vergleiche mit einer Bonferroni-Korrektur für mehrere Vergleiche angestellt (IBM SPSS Statistics, 2017). Die Abschlusszeiten unterschieden sich signifikant für die Selektion von Objekten mittels der Blickrichtung und der Selektion mit einem zweiten Smartphone mit gebogenem Raycast ($p < 0.005$) und einem regulärem Raycast ($p = 0,003$).

	Shapiro-Wilk		
	Statistik	df	Signifikanz
Abschlusszeit gesamt Blickrichtung	0.742	25	0.000
Abschlusszeit gesamt Raycast	0.873	25	0.005
Abschlusszeit gesamt Raycast (gebogen)	0.845	25	0.001

Tabelle 4.2: *SPSS-Ausgabe des Shapiro-Wilk-Tests auf Normalverteilung*

Die Untersuchung der Standardabweichungen und Mittelwerte der Abschlusszeiten in Tabelle 4.3 zeigt, dass die Abschlusszeit bei der Bedienung mit der Blickrichtung durchschnittlich etwa

[3]Der Wert p beschreibt bei statistischen Tests der Inferenzstatistik die Wahrscheinlichkeit, dass ein Effekt zufällig auftritt. I.d.R kann ein Test als statistisch signifikant angesehen werden, wenn $p < 0,05$. [13] (S. 150). Bei den meisten durchgeführten statistischen Test in SPSS wird dieser Wert als Signifikanz ausgegeben.

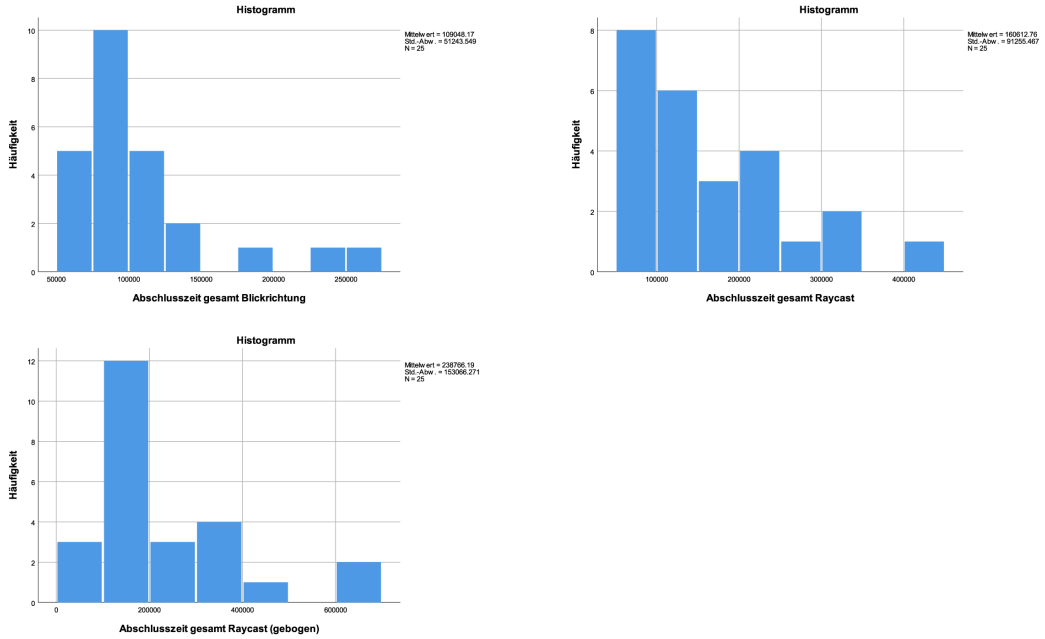


Abbildung 4.2: *Histogramm-Ansicht der Verteilung Abschlusszeiten-Verteilung aller Probanden.*

109s $\pm$ 51s betrug. Die Zeiten für die Selektion mit dem virtuellen Zeigegerät, welches einen normalen Raycast nutzte, lagen bei ca. 160s $\pm$ 91s. Mit dem gebogenen Strahl stiegen diese nochmals auf durchschnittlich 238s $\pm$ 153s. Die Standardabweichungen lassen bereits erkennen, dass die Daten sehr unterschiedlich gestreut sind. Wegen des Rauschens im Datensatz müsste dieser erst bereinigt werden, um die gewünschten Informationen aus den Daten korrekt herauslesen zu können. Ein Grund für die massiven Abweichungen in der Zeiten können ebenfalls durch die Schwierigkeit der Aufgabe erklärt werden, das Höhengefälle der Treppe in der VE zu überwinden. Besonders bei der Nutzung eines Strahls mit parabolischer Form war dies eine Herausforderung. Wie in Abschnitt 3.1 beschrieben, war in der 3D-Szene keine echte Geometrie vorhanden, d.h. der Strahl konnte nicht neben dem Zielpunkt auf dem Treppenabsatz landen, sondern die Nutzer mussten den Strahl so positionieren, dass dieser den Ortspunkt direkt schneidet. Durch die Kommentare der Nutzer während der Studie wurde klar, dass vielen Probanden die nötige Genauigkeit fehlte um mit dem virtuellen Zeigegerät die Punkte auf dem Treppenabsatz zu treffen.

	Mittelwert	Std.-Abweichung	N
Abschlusszeit gesamt Blickrichtung	109048.18	51243.549	25
Abschlusszeit gesamt Raycast	160612.76	91255.467	25
Abschlusszeit gesamt Raycast (gebogen)	238766.19	153066.271	25

Tabelle 4.3: *SPSS-Ausgabe: Mittelwerte und Standardabweichungen der Abschlusszeit (gesamt) in ms für alle Probanden, die mit jeder Eingabemethode erfolgreich waren (n=25).*

Aufgrund dieser Tatsache ist die Analyse der Zeiten für die Selektion von Ortspunkten auf ebenem Grund von besonderem Interesse und wird im Nachfolgenden genauer untersucht. Es muss jedoch angemerkt werden, dass ein Datensatz invalide gespeichert wurde, weswegen hier nur noch die Gesamtpopulation von n=27 untersucht wird. Auch für die gemessenen Zeiten der mutmaßlich einfach zu selektierenden Punkte gab es einige Ausreißer, wie in Abb. 4.3 zu entnehmen ist.

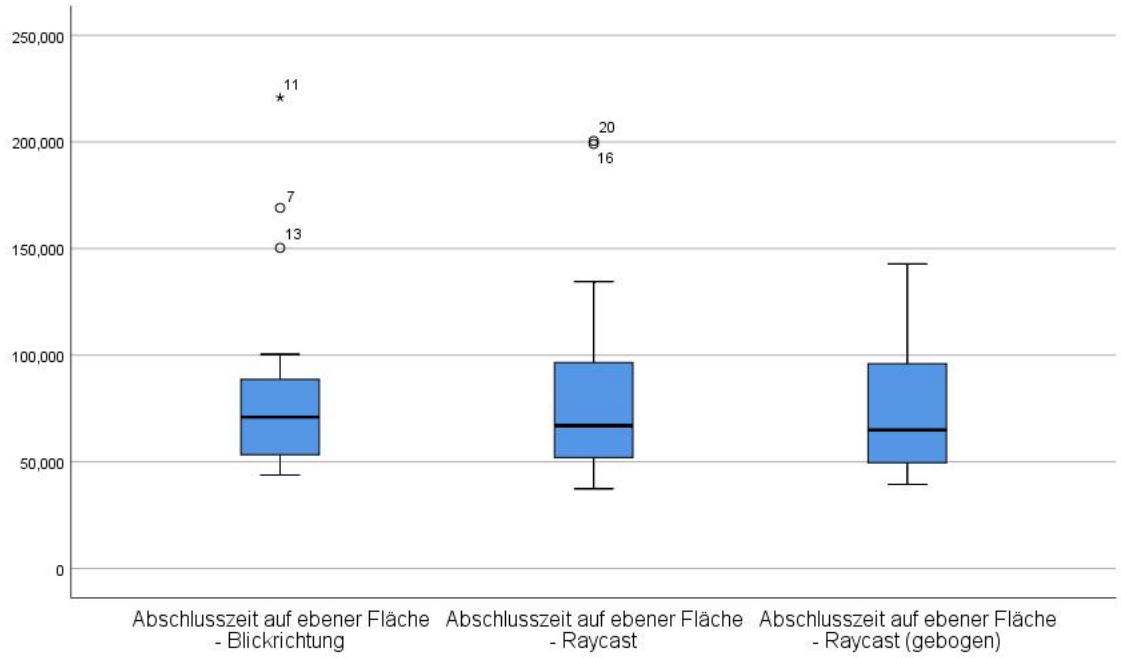


Abbildung 4.3: *Abschlusszeiten (in ms) der Probanden auf ebenem Grund.*

Im Falle des Probanden #11 wurde bereits erklärt, dass dieser sich innerhalb der VE „verlief“. Auch bei Proband #20 ist aus der Menge an Bewegungen erkenntlich, dass dieser zunächst den Indikator für den nächsten Zielpunkt aus den Augen verlor und zu sieben anderen Ortspunkten sprang, bevor er wieder auf den richtigen Weg zurück fand. Bei Proband #7 ist in den Daten vor allem zu erkennen, dass die Zahl der Fokusverluste besonders hoch war. Er hatte also Schwierigkeiten, einen Ortspunkt mittels der Blickrichtung so lange zu fokussieren, dass die Bewegung zum Punkt ausgelöst werden konnte. Die restlichen Ausreißer können anhand der erhobenen Daten nicht nachvollzogen werden.

Auch bei der Betrachtung der Abschlusszeiten zur Selektion von Objekten auf ebenem Grund wurde ein Shapiro-Wilk-Tests auf Normalverteilung durchgeführt. Mit $p < 0.05$ (wie in Tabelle 4.4 ersichtlich) wird die Nullhypothese der Normalverteilung der Daten abgelehnt. die Daten sind also nicht normal verteilt. Ein Blick auf die Histogramme in Abb. 4.4 zeigt ebenfalls, dass diese Messungen keine Normalverteilung aufweisen.

	Shapiro-Wilk		
	Statistik	df	Signifikanz
Abschlusszeit gesamt Blickrichtung	0.742	25	0.000
Abschlusszeit gesamt Raycast	0.873	25	0.005
Abschlusszeit gesamt Raycast (gebogen)	0.845	25	0.001

Tabelle 4.4: *SPSS-Ausgabe des Shapiro-Wilk-Tests auf Normalverteilung*

Vergleicht man die gemessenen Abschlusszeiten aus Tabelle 4.5 mit denen aus Tabelle 4.3, so ist schnell erkenntlich, dass sich Mittelwerte und Standardabweichungen sehr stark unterscheiden. Während bei der Betrachtung der gesamten Abschlusszeit der Median für die Zeiten bei der Selektion zwischen den einzelnen Eingabemethoden stark unterscheidet, ist dies für die Punkte auf ebenem Grund nicht der Fall. Dies ist auch in Abb. 4.5 ersichtlich.

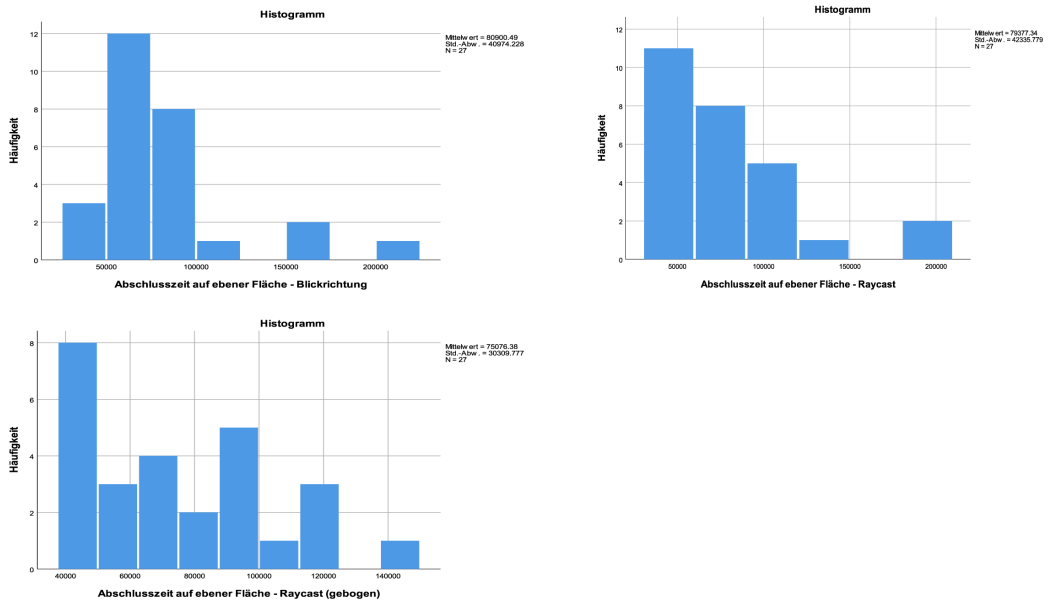


Abbildung 4.4: *Histogramm-Ansicht der Verteilung der kumulierten Selektionszeiten für Punkte auf ebener Fläche*

	Mittelwert	Std.-Abweichung	N
Abschlusszeit auf ebener Fläche - Blickrichtung	80900.49	40974.228	27
Abschlusszeit auf ebener Fläche - Raycast	79377.34	42335.779	27
Abschlusszeit auf ebener Fläche - Raycast (gebogen)	75076.38	30309.777	27

Tabelle 4.5: *SPSS-Ausgabe: Mittelwerte und Standardabweichungen der Abschlusszeit (nur Punkte auf ebener Fläche)*

Die Annahme auf Sphärizität ist bei den Zeiten auf ebener Fläche jedoch nicht verletzt, da der Mauchly-Test auf Sphärizität nicht statistisch signifikant ist, $\chi^2(2) = 5049, p = 0,80$.

Jedoch kann anhand der Ergebnisse kein statistisch signifikanter Unterschied in den Mittelwerten der Zeit erkannt werden, $F(2,52) = 0,249, p = 0,309$. Zusammenfassend können sich aus den gemessenen Daten zur Abschlusszeit folgende Ergebnisse ableiten lassen:

- Für die Zeit, die benötigt wurde, um die gesamte Route zu absolvieren können signifikante Unterschiede in den Mittelwerten erkannt werden. Durchschnittlich waren die Probanden mit der Selektion mittels Blickrichtung am schnellsten, gefolgt vom virtuellen Zeigegerät mit einfachem Raycast und schließlich dem parabolischen Raycast. Die schlechte Performance bei der Selektion mit gebogenem Raycast bei Punkten, die nicht auf dem virtuellen Boden lagen, sondern ein Höhengefälle aufwiesen, ist mit der benötigten hohen Anforderung an Präzision zu erklären, diese direkt mit dem gebogenen Strahl zu schneiden.

- Zwischen den Mittelwerten der Zeit für die Selektion von Punkten, die alle auf dem Boden der VE lagen, lassen sich keine signifikanten Unterschiede feststellen.

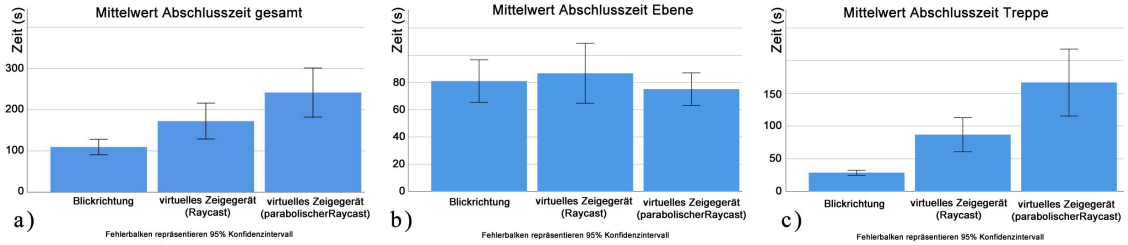


Abbildung 4.5: *Mittelwerte der gemessenen Abschlusszeiten a) gesamt, b) für die Selektion von Punkten auf dem Boden der VE und c) für die Selektion von Punkten auf der virtuellen Treppe*

Fehler: Wie in Abschnitt 3.3 beschrieben, wurden bei der Durchführung der Benutzerstudie die begangenen Fehler bei der Interaktion pro Navigationsschritt erhoben. Einerseits wurde zur Messung der Zielgenauigkeit der Probanden mit den einzelnen Eingabemethoden die Zahl der Fokusverluste („slips“) gezählt, andererseits die Zahl der Punkte, die ein Proband besuchte, bevor er zum jeweiligen Zielpunkt gelangt ist. Für die Evaluation der Eingabemethode ist die Analyse der Fokusverluste von Bedeutung, da diese Einsicht darüber gewähren kann, wie einfach es für Probanden war, mit einer bestimmten Eingabemethode die Ortspunkte zu selektieren. Die Untersuchung wird hier separat für die Punkte auf ebenem Grund (n=27) sowie für die auf der virtuellen Treppe (n=25) angestellt. Zur Erkennung von Ausreißern wurde der Boxplot in Abb. 4.6 erstellt. Zur besseren Vergleichbarkeit zwischen den Fehlern die auf ebenem Grund gemacht wurden und den begangenen Fehlern bei dem Versuch, die Punkte auf der Treppe zu selektieren, sind diese in einer Grafik zusammengefasst.

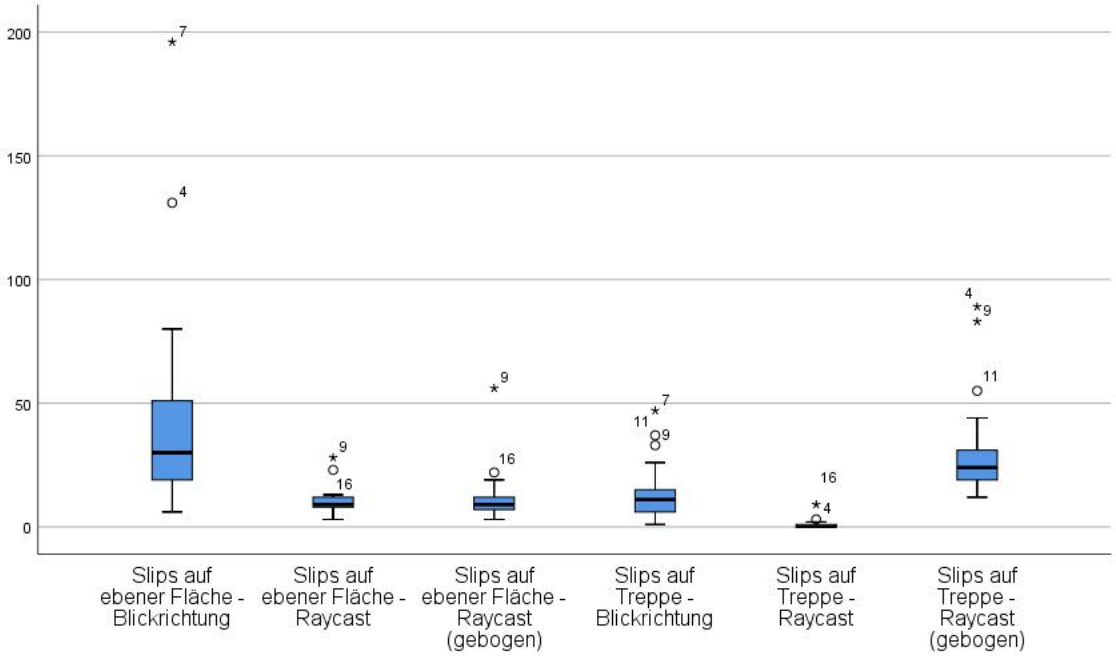


Abbildung 4.6: *Anzahl Fokusverluste (engl. „Slips“) für Punkte auf ebenem Grund und bei dem Versuch, die Treppe in der VE zu erklimmen.*

Auch hier sind vor allem extreme Ausreißer bei der Selektion mittels Blickrichtung zu erkennen. Dies ist höchstwahrscheinlich bedingt durch die Implementierung der Eingabemethode. Wie in Abschnitten 3.2.1 und 3.2.3 beschrieben, wurden die Ortspunkte (sobald der Nutzer diese ein-

mal erfolgreich fokussiert hat) so gedreht, dass ihre Normale (parallel zur Blickrichtung) auf ihn zeigt. Bei der Selektion mittels Blickrichtung kam es hierbei jedoch zu sehr vielen falsch-positiven Messungen der Fokusverluste. Der Grund dafür konnte nicht gefunden werden. Da sowohl die Eingabemethode Blickrichtung, als auch die mittels virtuellem Zeigegerät (Raycast) im Hintergrund einen regulären Raycast ausführen und somit die selbe Logik für die Erkennung eines solchen Fokus-Verlusts haben, kann der Unterschied in der Anzahl der gemachten Fehler nur damit erklärt werden, dass es den Nutzern einfacher fiel, das virtuelle Zeigegerät nach erfolgreicher Fokussierung eines Punktes ruhig zu halten, um den Fokus nicht wieder zu verlieren. Da jedoch für die Implementierung des Zeigers für die Steuerung mittels Blickrichtung eine vorgefertigte Lösung verwendet wurde, sind fehlerhafte Erkennungen in der genutzten Softwarekomponente nicht auszuschließen. Die Mittelwerte und Standardabweichungen für die Anzahl der Fokusverluste bei der Selektion von Punkten ebenem Grund und denen auf der virtuellen Treppe können den Tabellen 4.6 bzw. 4.7 entnommen werden. Eine grafische Repräsentation der gemessenen Daten ist in Abb. 4.7 zu sehen.

	Mittelwert	Std.-Abweichung	N
Slips auf ebener Fläche - Blickrichtung	41.44	40.764	27
Slips auf ebener Fläche - Raycast	9.59	5.556	27
Slips auf ebener Fläche - Raycast (gebogen)	11.04	10.105	27

Tabelle 4.6: *Mittelwerte und Standardabweichungen für die aufgetretenen Fokus-Verluste bei der Selektion von Punkten auf ebener Fläche*

Für die Fehler, die bei der Selektion von Ortspunkten auf einer Ebene, begangen wurden wurde ein Shapiro-Wilk-Test durchgeführt. Mit $p < 0,05$ wird die Nullhypothese der Normalverteilung der Daten abgelehnt. Auch der Mauchly-Test auf Sphärizität war nicht signifikant, $\chi^2(2) = 66,065, p < 0,05$, somit wird die Annahme auf Sphärizität der Daten abgelehnt. Deswegen wurde ein Friedman-Test durchgeführt, um zu bestimmen, ob es signifikante Unterschiede bei den Mittelwerten der begangenen Fokus-Fehlern zwischen den unterschiedlichen Eingabemethoden gab. Es wurden paarweise Vergleiche mit einer Bonferroni-Korrektur für mehrere Vergleiche angestellt (IBM SPSS Statistics, 2017). Die Anzahl dieser Fehler unterschied sich mitunter signifikant, $\chi^2(2) = 31.981, p < 0,05$. Die Fehler beim Fokussieren von Punkten auf einer geraden Ebene unterschieden sich signifikant für die Eingabemethode Blickrichtung (Median: 30) und mit gebogenem Raycast (Median: 9, $p < 0.005$) und einem regulärem Raycast (Median: 9, $p < 0.005$). Unterschiede zwischen einem normalen Raycasts und einem parabolischen waren folglich nicht signifikant ($p = 1$).

	Mittelwert	Std.-Abweichung	N
Slips auf Treppe - Blickrichtung	13.28	11.649	25
Slips auf Treppe - Raycast	1.00	1.893	25
Slips auf Treppe - Raycast (gebogen)	29.96	19.709	25

Tabelle 4.7: *Mittelwerte und Standardabweichungen für die aufgetretenen Fokus-Verluste bei der Selektion von Punkten auf der virtuellen Treppe*

Für die Selektion der Ortspunkte auf der Treppe wurde ein Shapiro-Wilk-Test durchgeführt. Mit $p < 0,05$ wird die Nullhypothese auf Normalverteilung der Daten abgelehnt. Auch der Mauchly-Test auf Sphärizität war statistisch signifikant, $\chi^2(2) = 7,325, p = 0,026$. Somit wird die Annahme auf Sphärizität der Daten abgelehnt. Deswegen wurde ein Friedman-Test durchgeführt, um zu bestimmen, ob es signifikante Unterschiede in den Mittelwerten der begangenen Fokus-Fehler zwischen den unterschiedlichen Eingabemethoden gab. Es wurden paarweise Ver-

gleiche mit einer Bonferroni-Korrektur für mehrere Vergleiche angestellt (IBM SPSS Statistics, 2017). Die Anzahl dieser Fehler unterschied sich mitunter signifikant, $\chi^2(2) = 46,08 p < 0,05$. Die Fehler beim Fokussieren von Punkten auf einer geraden Ebene unterschieden sich signifikant für die Eingabemethode Blickrichtung (Median: 11) und virtuellem Zeigegerät mit parabolischem Raycast (Median: 24, $p = 0.002$) sowie regulärem Raycast (Median: 0, $p = 0.002$). Auch die Unterschiede in den Mittelwerten der Fehler zwischen den beiden Raycasts waren statistisch signifikant ($p < 0.005$).

Zusammenfassend lassen sich für die aufgetretenen Fokus-Fehler folgende Ergebnisse festhalten:

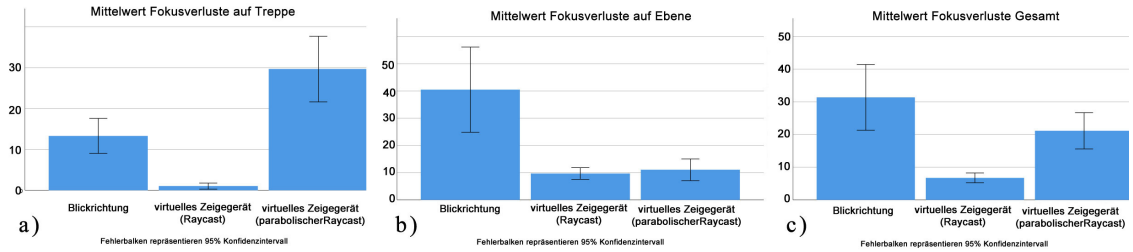


Abbildung 4.7: *Mittelwerte der aufgetretenen Fokusverluste bei der Selektion von a) Punkten auf der virtuellen Treppe, b) Punkten auf dem Boden der VE und c) gesamt*

- Für die Selektion von Ortspunkten auf dem Boden der VE wurde der Fokus bei Steuerung mit der Blickrichtung am häufigsten verloren. Die Begründung dafür ist jedoch nur auf eine fehlerhafte Softwareverhalten zurückzuführen. Durch die Drehung der Ortspunkte in Richtung des Probanden, welche die Selektion vereinfachen sollte, begann der runde Ladebalken viel öfter sich neu zu füllen als erwartet. In der Implementierung wurde der Ortspunkt gedreht, sobald der Raycast in Blickrichtung diesen schnitt. Eine mögliche Fehlerquelle ist, dass während dieser Drehung von der verwendeten Softwarekomponente der Schnitt vom gesendeten Raycast mit dem Ortspunkt nicht mehr erkannt wurde. Warum dies bei der Selektion von Ortspunkten auf der Treppe nicht so oft auftrat, konnte jedoch nicht erklärt werden. Zwischen den beiden Varianten des virtuellen Zeigegeräts bestand kein signifikanter Unterschied hinsichtlich dieser Metrik, da diese auf der gleichen Implementierung basierten.

- Für die Selektion von Ortspunkten auf der Treppe in der genutzten VE, verloren die Nutzer mit dem virtuellen Zeigegerät, das einen parabolischen Raycast benutzte, am häufigsten den Fokus auf das zu selektierende Objekt. Die war auch bei der Beobachtung der Probanden während der Studien bereits erkenntlich. Das Hauptproblem war die benötigte Präzision, um mit dem parabolischen Strahl genau den Ortspunkt zu treffen. Durch die fehlende Geometrie der Treppe in der 3D-Szene war die Position des Schnittpunktes von Strahl und Treppe für den Nutzer nur ersichtlich, wenn der Ortspunkt direkt geschnitten wurde. Verfehlte der Strahl den Ortspunkt, so war nur zu sehen, wo er auf der Bodenfläche der VE auftraf. Diese Bodenfläche befand sich jedoch nicht auf Höhe der Ortspunkte auf der Treppe, sondern darunter. Dies erschwerte das genaue Adjustieren des virtuellen Zeigegeräts. Ein überraschendes Ergebnis ist, dass der Fokus auf getroffene Objekte bei der Nutzung vom virtuellen Zeigegerät mit geradem Strahl fast nie verloren wurde. Aufgrund der Beobachtung der Probanden und den Erkenntnissen über die Abschlusszeiten, lässt dies zwar nicht darauf schließen, dass diese Methode sich besser zum Selektieren geeignet hat. Doch war es hier einfacher als bei den anderen Eingabemethoden, den Fokus auf das selektierte Objekt zu erhalten, sobald dieses getroffen wurde.

4.1.2 Analyse der erhobenen qualitativen Daten

Ergbnisse des NASA-TLX und ASQ: Zur Messung der Nutzerzufriedenheit und der Einschätzung, wie einfach es den Nutzern fiel, anhand der unterschiedlichen Eingabemethoden durch die VE zu navigieren, wurden den Probanden nach Abschluss eines Durchgangs jeweils zwei Fragebögen vorgelegt. Wie in Abschnitt 3.3 beschrieben, mussten dabei, im Falle des ASQ, drei Einschätzungen abgegeben werden, wie leicht die Aufgabe mit der jeweiligen Eingabemethode fiel und wie zufrieden die Nutzer mit ihrer Leistung waren. Der NASA-TLX-Fragebogen diente der subjektiven Einschätzung des mentalen wie physischen Aufwands, der benötigt wurde, um die Aufgabe zu bewältigen.

Im Falle des ASQ wurde hierbei der Gesamtwert dem Mittelwert der beiden ersten Fragen berechnet. Die dritte Frage zur Zufriedenheit der Nutzer mit der Dokumentation des Systems wurde hierbei aus der Analyse ausgeschlossen, da bis auf die Instruktionen zu Beginn jedes Versuchs keinerlei Dokumentation vorhanden war. Ein hoher Gesamtwert entspricht somit einer hohen Zufriedenheit mit der Eingabemethode. In Abb. 4.8 ist vor allem bei der Selektion mittels Blickrichtung ein Ausreißer zu erkennen. Proband #11 galt bereits bei der Analyse der Abschlusszeiten für die Selektion mittels Blickrichtung als extemer Ausreißer, was die Unzufriedenheit mit der Eingabemethode und der eigenen Leistung erklären kann. Proband #2 brach den Versuch, die Treppe zu erklimmen mit dem virtuellen Zeigegerät mit geradem Strahl vorzeitig ab, da er nicht in der Lage war mit dem regulären Raycast die höher liegenden Puntke erfolgreich zu selektieren.

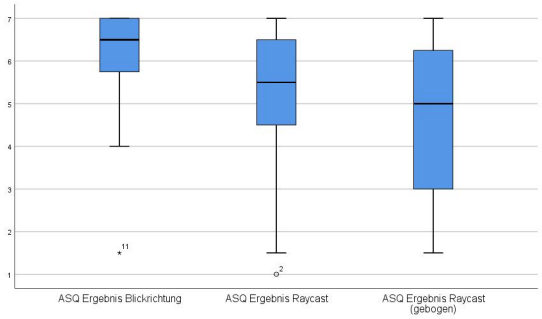


Abbildung 4.8: *Mittelwerte und Ausreißer des errechneten ASQ-Gesamtwertes nach Eingabemethode*

Für die Gesamtwerte zur Zufriedenheit wurde ein Shapiro-Wilk-Test durchgeführt. Mit einem p-Wert < 0.05 wird die Nullyhypothese der Normalverteilung der Daten abgelehnt. Da sich die meisten Antworten bei in der Likert-Skala jedoch bei den Extrempunkten befanden, ist die Normalverteilung der Daten nicht anzunehmen. Der Mauchly-Test auf Sphärizität war jedoch nicht statistisch signifikant, $\chi^2(2) = 5,058, p = 0,08$, somit kann die Sphärizität der Daten angenommen werden.

Der erhobene Gesamtwert zur Nutzerzufriedenheit unterschied sich zwischen den verschiedenen Eingabemethoden statistisch signifikant, $F(2,54) = 7,332, p = 0,02$. Die Mittelwerte der Nutzerzufriedenheit in Tabelle 4.8 und Abb. 4.9 lassen erahnen, dass die Nutzer mit der Eingabe durch die Blickrichtung mit $6,14 \pm 1,27$ am zufriedensten waren, gefolgt vom virtuellen Zeigegerät mit geradem Strahl ($5,07 \pm 1,66$) und schließlich dem virtuellen Zeigegerät mit parabolischem Strahl ($4,60 \pm 1,78$).

Nachträgliche paarweise Vergleiche mit Bonferroni-Korrektur zeigen jedoch, dass sich nur die Nutzerzufriedenheit zwischen der Eingabemethode Blickrichtung und virtuellem Zeigegerät mit parabolischem Strahl signifikant um 1,54 Punkte unterschied (95% Konfidenzintervall: $0,71$ bis $2,36$), $p < 0,005$. Der Vergleich vom Unterschied der Nutzerzufriedenheits-Mittelwerte zwischen Selektion mittels Blickrichtung und dem virtuellen Zeigegerät mit regulärem Raycast lag bei 1,07 (95% Konfidenzintervall: -0,005 bis 2,15) und war nicht statistisch signifikant, $p = 0,051$. Ebenso waren die Differenzen bei den zwei Varianten des virtuellen Zeigegeräts untereinander mit einer Differenz des Mittelwertes von 0,47 (95% Konfidenzintervall: $-0,75$ bis 1,68) nicht statistisch signifikant, $p = 1$.

Zusammenfassend lässt sich also sagen:

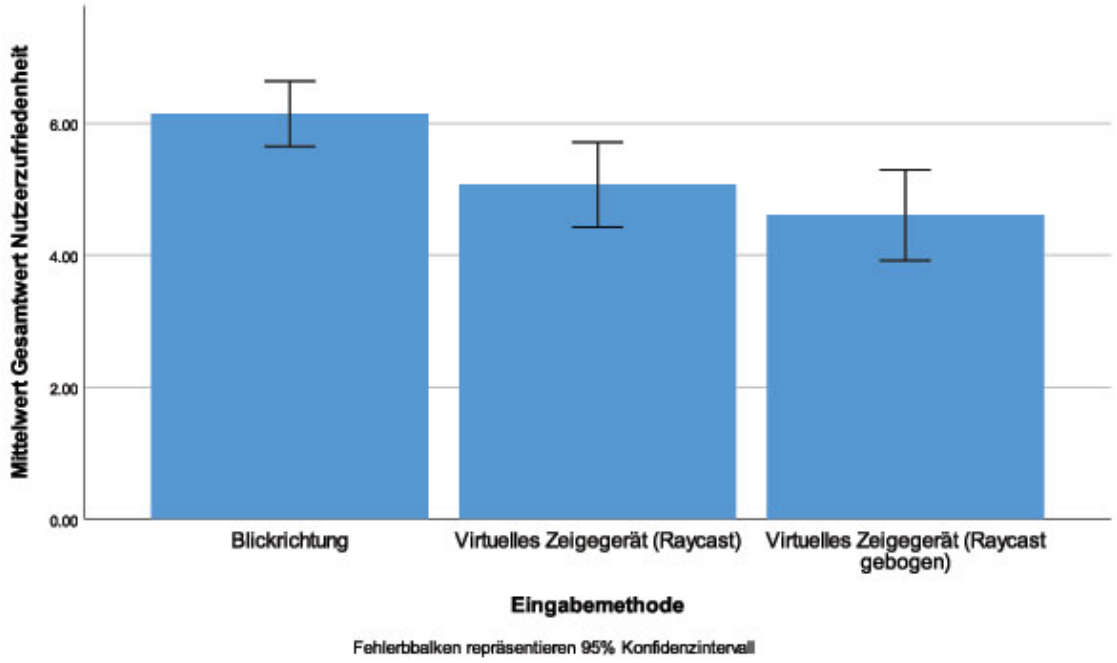


Abbildung 4.9: *Grafische Darstellung der Mittelwerte der erhobenen Daten zur Nutzerzufriedenheit nach Eingabemethode*

- Die Nutzer waren mit der Eingabemethode *Blickrichtung* im Schnitt zufriedener als mit einem virtuellen Zeigegerät, jedoch statistisch signifikant nur im Vergleich zu der parabolischen Variante. Da jedoch die Ausreißer nicht aus den Datensätzen entfernt wurden, kann dieses Ergebnis verzerrt sein.
- Zwischen den beiden Varianten des virtuellen Zeigegeräts gab es bezüglich der Nutzerzufriedenheit keine statistisch signifikante Präferenz.

	Mittelwert	Std.-Abweichung	N
ASQ Ergebnis Blickrichtung	6.1429	1.26825	28
ASQ Ergebnis Raycast	5.0714	1.65951	28
ASQ Ergebnis Raycast (gebogen)	4.6071	1.77616	28

Tabelle 4.8: *Mittelwerte und Standardabweichungen in der Nutzerzufriedenheit nach Eingabemethode.*

Bei der Bewertung des NASA-(R)TLX wurde die Methode nach [6] verwendet, und alle verfügbaren Teilwerte aufsummiert um die gesamte kognitive Belastung zu ermitteln. Der Gesamtwert kann so interpretiert werden, dass ein hoher Wert einer hohen (mentalen wie physischen) Anstrengung zur Lösung der Aufgabe entspricht. Der Boxplot in Abb. 4.10 zeigt insbesondere bei der Nutzung der virtuellen Zeigegeräte Ausreißer, von denen jedoch keiner als extrem klassifiziert wurde.

Aus den Einzelantworten der Fragebögen lässt sich erkennen, dass Proband #7 vor allem viel Zeitdruck bei der Lösung der Aufgabe verspürte und auch den Höchstwert bei der Frage angab, ob er während der Aufgabe viel Irritation oder Stress verspürte. Proband #15 empfand die Steuerung mit dem virtuellen Zeigegerät (gerader Strahl) als körperlich wie mental sehr anstrengend. Vor allem ist dies auf die hohe benötigte Präzision beim Treppenaufstieg zu erklären, da dieser Proband

die Aufgabe an dieser Stelle abbrach. Proband #14 gab bei allen Fragen zur Anstrengung der Benutzung des virtuellen Zeigegeräts mit parabolischem Strahl den Maximalwert an, da dieser ebenfalls an der selben Stelle den Versuch abbrechen musste. Um die Stichprobenzahl so groß wie möglich zu halten, wurden die Datensätze jedoch nicht aus der Analyse entfernt.

Für die Gesamtwerte zur benötigten Anstrengung konnte mit dem Shapiro-Wilk-Test keine Normalverteilung festgestellt werden ($p < 0,05$ für alle Eingabemethoden). Wie bereits erwähnt, verzerren geringe Stichprobenzahlen die Ergebnisse dieses Tests. Die grafische Analyse der Werteverteilung anhand von Histogrammen in Abb. 4.11, lassen jedoch die Annahme von nahezu normalverteilten Werten zu.

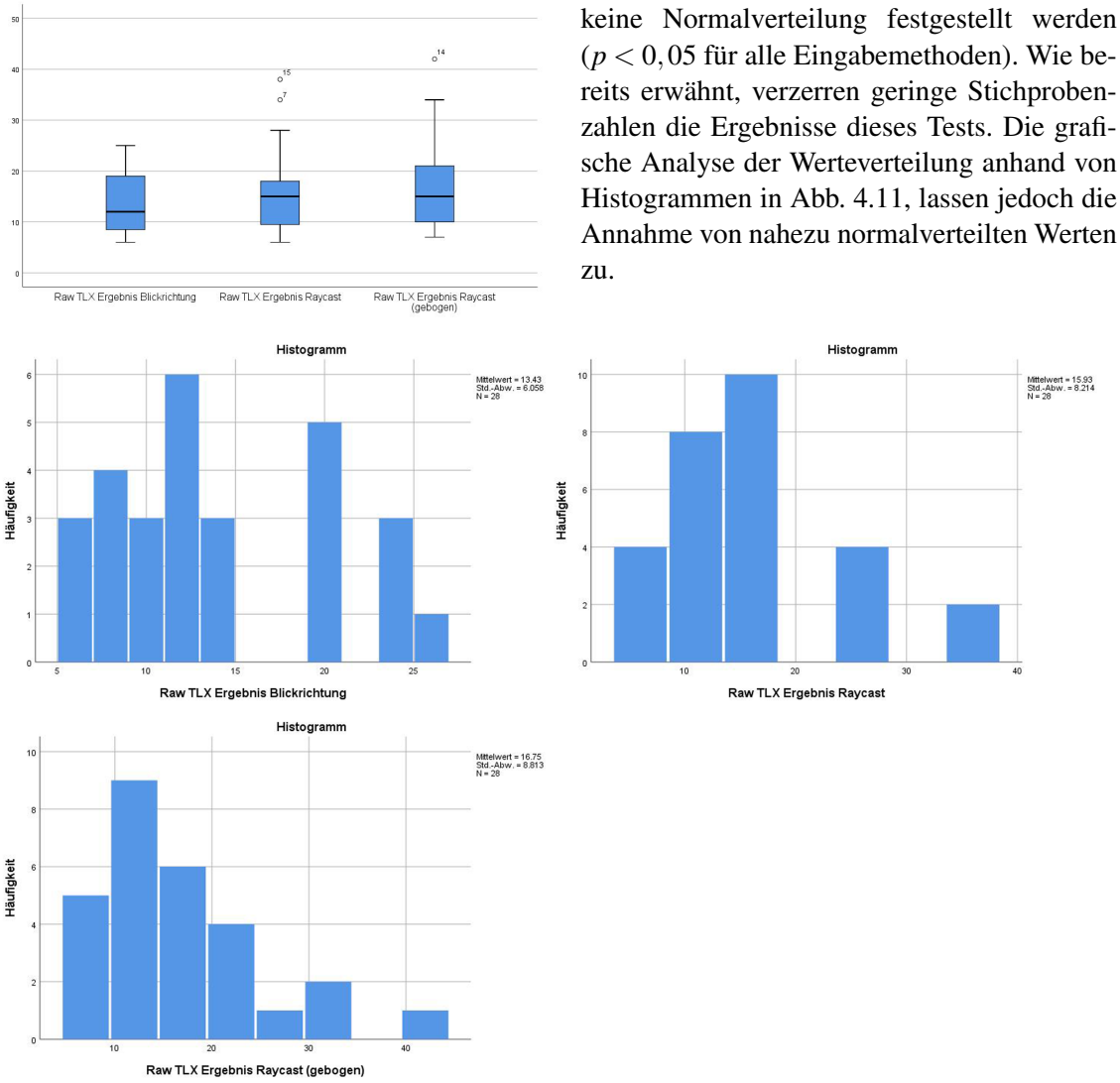


Abbildung 4.11: *Histogramme der Verteilung des errechneten NASA-(R)TLX Gesamtwertes nach Eingabemethode*

Der Mauchly-Test auf Sphärizität war jedoch statistisch nicht signifikant, $\chi^2(2) = 0,73, p = 0,694$, d.h. die Annahme der Sphärizität der Daten kann bestätigt werden. Der erhobene Gesamtwert zur benötigten Anstrengung unterschied sich zwischen den verschiedenen Eingabemethoden jedoch nicht statistisch signifikant, $F(2,54) = 2,408, p = 0,1$. Die Mittelwerte der ermittelten Werte zur benötigten Anstrengung in Tabelle 4.9 und Abb. 4.12 lassen zwar den Schluss zu, dass die Nutzer die Eingabe durch die Blickrichtung mit $13,43 \pm 6,06$ am wenigsten anstrengend empfanden, gefolgt vom virtuellen Zeigegerät mit geradem Strahl ($15,93 \pm 8,21$) und

schließlich dem virtuellen Zeigegerät mit parabolischem Strahl ($16,75 \pm 8,83$), jedoch kann dafür mit den vorhandenen Daten kein statistischer Nachweis erbracht werden.

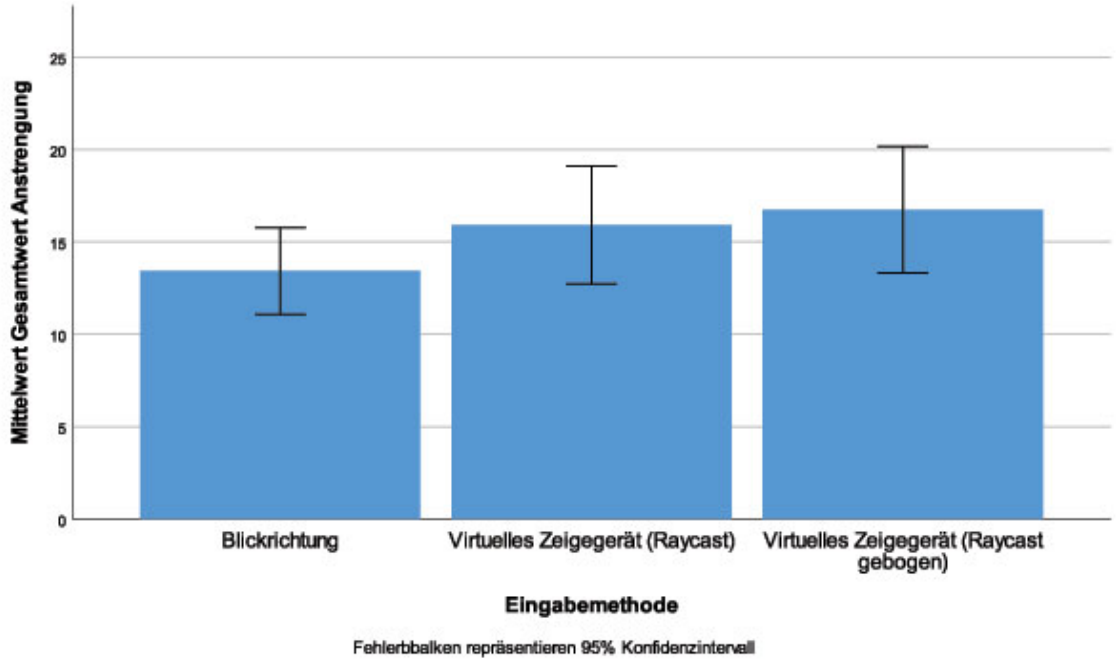


Abbildung 4.12: *Mittelwerte der erhobenen Daten zur Nutzerzufriedenheit nach Eingabemethode*

	Mittelwert	Std.-Abweichung	N
Raw TLX Ergebnis Blickrichtung	13.43	6.058	28
Raw TLX Ergebnis Raycast	15.93	8.214	28
Raw TLX Ergebnis Raycast (gebogen)	16.75	8.813	28

Tabelle 4.9: *Mittelwerte und Standardabweichungen in der benötigten Anstrengung nach Eingabemethode.*

Cyber Sickness: Jeder Proband musste im Zuge des Experiments eine Reihe von Fragebögen beantworten. Zur Erhebung der Unterschiede im Wohlbefinden durch den SSQ wurde dieser jeweils vor dem und nach dem Experiment vorgelegt. Da nur zwei Messzeitpunkte vorliegen, wird von [28] die Verwendung eines T-Tests für paarweise Stichproben empfohlen. Auch bei diesem Test müssen zunächst die Ausreißer überprüft werden. Abb. 4.13 zeigt die Verteilung der Differenzen für den Hauptwert des SSQ. Der Proband #5, welcher aus der Analyse der quantitativen Daten ausgeschlossen werden musste, ist hier als Extremwert zu erkennen. Interessant ist vor allem, dass bei ungefähr 50% der Probanden nach dem Experiment sogar eine Linderung der Symptome zu erkennen war. Das ist statistisch sehr schwer nachzuvollziehen. Die Gewichtung der Faktoren kann hier jedoch mitunter extreme Unterschiede und Verzerrungen in den Daten bewirken. Proband #25 hat im Fragebogen vor dem Experiment bereits sehr hohe Werte angegeben. Das führt dazu, dass die Differenz seines Gesamtwertes zu Effekten der Simulatorkrankheit einen hohen negativen Wert annimmt und er somit als extremer Ausreißer deklariert werden kann. Bei einer so extremen Verbesserung des Gesamtwertes ist es anzunehmen, dass dieser beim zweiten Fragebogen die Antworten relativ abgab, also ein Wert von 0 bei einem Symptom als „keine Veränderung zu vorher“

interpretierte und nicht als „Symptom tritt zum jetzigen zeitpunkt nicht auf“. Deswegen wurde der Proband aus der Analyse entfernt. Somit verringert sich die Stichprobengröße auf $n = 28$.

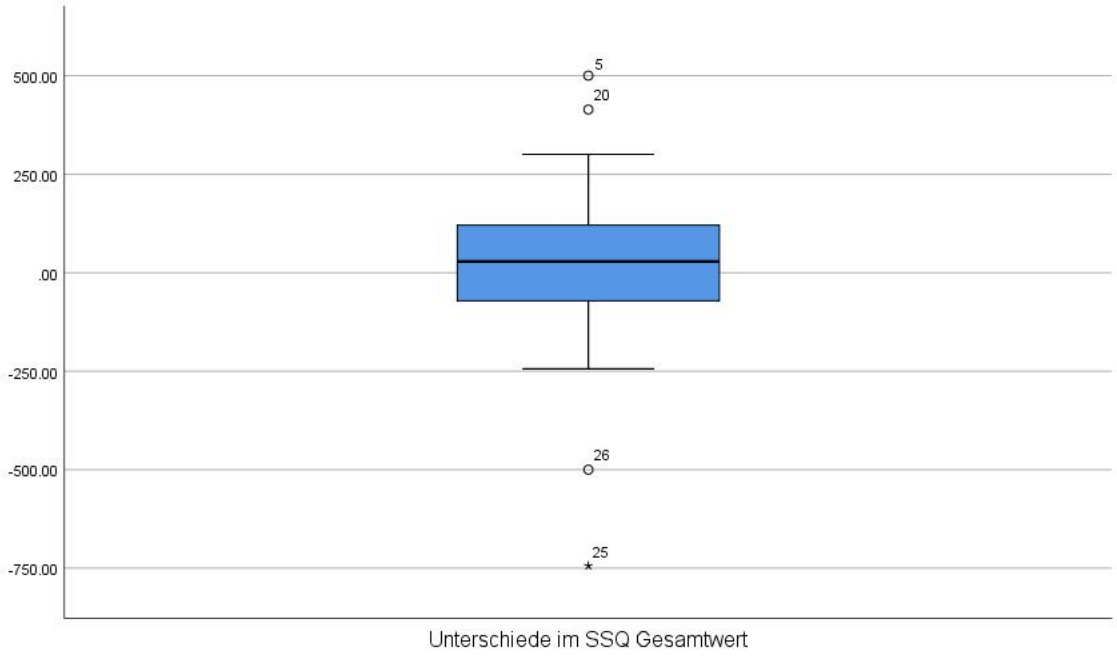


Abbildung 4.13: *Differenzen in den Gesamtwerte des SSQ. Die Y-Achse zeigt die Differenz des Gesamtwertes zwischen der Erhebung nach und vor dem Experiment.*

Ebenso müssen die Daten auf Normalverteilung hin überprüft werden — obwohl der Test, ebenso wie der ANOVA-Test, recht robust gegenüber der nicht-Normalverteilung der Daten ist. Durch die Entfernung des Probanden #25 kann durch das Ergebnis des Shapiro-Wilk-Tests Normalverteilung angenommen werden ($p = 0,242$). Anhand der Mittelwerte in Tabelle 4.10 lässt sich erkennen, dass der Mittelwert des gewichteten SSQ-Gesamtwertes von der Erhebung vor dem Experiment ($214,29 \pm 216,61$) im Vergleich zu der nach dem Experiment auf $259,53 \pm 196,29$ stieg. Der Anstieg betrug also im Schnitt 45,22 Punkte (95% Konfidenzintervall: -30,25 bis 120,72). Durch die hohe Varianz der Werte ist dieser Anstieg im Mittelwert jedoch nicht statistisch signifikant, $t(27) = 1,23, p = 0,229$.

	Mittelwert	N	Standard Abweichung	Standardfehler des Mittelwertes
Übelkeit nachher	15.332143	28	19.1398466	3.6170910
Übelkeit vorher	18.057857	28	20.9878853	3.9663375
Anstrengung des Augenapparates nachher	22.740000	28	17.3832708	3.2851294
Anstrengung des Augenapparates vorher	20.845000	28	17.9405195	3.3904395
Desorientierung nachher	31.320000	28	25.3082121	4.7828025
Desorientierung vorher	18.394286	28	23.9768968	4.5312076
SSQ Gesamtwert nachher	259.526614	28	196.2946279	37.0961978
SSQ Gesamtwert vorher	214.291314	28	216.6054799	40.9345880

Tabelle 4.10: *Mittelwerte und Standardabweichungen der, anhand der Antworten des SSQ ermittelten, Werte der Intensität von Symptomen der Simulatorkrankheit*

Die Gewichtung nach [22] (Tabelle 3.2) einzelner Teilwerte und die Zusammenfassung zu einem Gesamtwert erschwert die Gewinnung von Erkenntnissen über die tatsächliche Veränderung bestimmter Symptome und verzerrt die Ergebnisse von Vergleichen untereinander. Aufgrund dessen werden die einzelnen Teilwerte von Symptomen der Übelkeit, der Anstrengung des Augenapparates und der Desorientierung einzeln untersucht. Abb. 4.14 zeigt die erstellten Boxplots zur Erkennung von Ausreißern. Bei keiner Symptomatik können extreme Ausreißer erkannt werden, jedoch sind Ausreißer zu erkennen. Proband #5, der das Experiment sofort abbrechen musste, ist hier besonders bei den Symptomen, die Übelkeit indizieren, als solcher zu erkennen. Auch bei Symptomen, die den Augenapparat betreffen gibt, es sowohl Ausreißer, bei denen sich die Symptome gebessert sowie verschlechtert haben. Bei den Symptomen die mit Desorientierung zusammenhängen, war nur ein Ausreißer zu erkennen. Da keiner dieser Ausreißer extrem galt, wurden diese in der Analyse mit berücksichtigt.

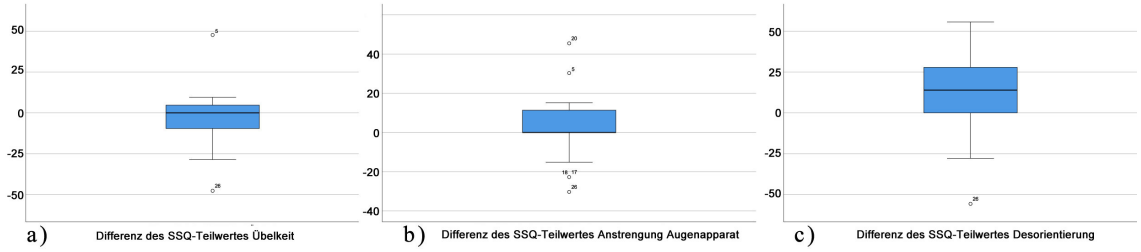


Abbildung 4.14: *Differenzen in den einzelnen Teilwerten des SSQ, wobei a) Symptome berücksichtigt, die mit Übelkeit assoziiert werden, b) Symptome der Anstrengung des Augenapparates und c) Symptome der Desorientierung. Die Y-Achse zeigt die Differenz des Gesamtwertes zwischen der Erhebung nach und vor dem Experiment.*

Die Veränderung der ermittelten Teilwerte für Symptome, die mit Übelkeit assoziiert werden, waren laut Shapiro-Wilk-Test nicht normalverteilt, $p = 0,02$. Es lässt sich anhand der Mittelwerte in Tabelle 4.10 erkennen, dass der Mittelwert der Intensität von Übelkeits-Symptome vor dem Experiment $(18,06 \pm 20,99)$ im Vergleich zu dem nach dem Experiment auf $15,33 \pm 19,14$ abfiel. Die Linderung der Symptome betrug also im Schnitt 2,73 Punkte (95% Konfidenzintervall: -9,24 bis 3,79). Jedoch waren auch die Änderungen des Mittelwerts nicht statistisch signifikant, $t(27) = -0,859, p = 0,398$.

Die Differenzen der ermittelten Teilwerte für Symptome, die mit der Anstrengung des Augenapparates assoziiert werden, waren laut Shapiro-Wilk-Test nicht normalverteilt, $p = 0,036$. Es lässt sich anhand der Mittelwerte in Tabelle 4.10 erkennen, dass der Mittelwert der Intensität von Symptom-Intensität vor dem Experiment $(20,85 \pm 17,94)$ im Vergleich zu dem nach dem Experiment auf $22,74 \pm 17,38$ anstieg. Die Verschlimmerung der Symptome betrug also im Schnitt 1,89 Punkte (95% Konfidenzintervall: -4,28 bis 8,07). Jedoch waren auch die Änderungen des Mittelwerts nicht statistisch signifikant, $t(27) = 0,692, p = 0,534$.

Die Differenzen der ermittelten Teilwerte für Symptome, die mit Desorientierung assoziiert werden, waren laut Shapiro-Wilk-Test nicht normalverteilt, $p = 0,43$. Es lässt sich anhand der Mittelwerte in Tabelle 4.10 erkennen, dass der Mittelwert der Intensität von Desorientierungs-Symptomen vor dem Experiment $(18,39 \pm 23,97)$ im Vergleich zu dem nach dem Experiment auf $31,32 \pm 25,31$ anstieg. Die Verschlimmerung der Symptome betrug also im Schnitt statistisch signifikante 12,93 Punkte (95% Konfidenzintervall: -9,24 bis 3,79), $t(27) = 2,608, p = 0,015$.

Zusammenfassend lässt sich also sagen, dass keine statistisch signifikanten Änderungen im Wohlbefinden der Probanden festgestellt haben. Trotz Probanden #5, der wegen Übelkeit das Experiment vorzeitig abbrechen musste, wurden in den erhobenen Daten keine statistisch signifikanten Unterschiede finden lassen, die suggerieren, dass es den Probanden nach dem Experiment

schlechter ging als davor. Einzig der Teilwert für Symptome der Desorientierung haben sich statistisch signifikant verschlechtert. Dieser wurde wie in Tabelle 3.2 ersichtlich. Die Ergebnisse decken sich mit denen von Stanney [44], die ebenfalls herausfanden, dass die Verwendung des SSQ bei der Evaluation von immersiven VEs oft höhere Werte in der Desorientierung liefert. Es kann also davon ausgegangen werden, dass es den Probanden nach dem Experiment zunächst schwer fiel, scharf zu sehen oder sie sich benommen bzw. schwindlig fühlten.

Präferierte Methode: Abschließend wurde jeder Proband nach seiner subjektiven Einschätzung gefragt, welche der Eingabemethoden für ihn am angenehmsten zu bedienen war. Zudem durfte zusätzlich noch angegeben werden, warum dies der Fall war. Abb. 4.15 zeigt eine kläre Präferenz der Probanden. 70,4% der Probanden empfanden demnach die Selektion der Ortspunkte mittels Blickrichtung am angenehmsten, 18,5% der Probanden bevorzugten das virtuelle Zeigegerät mit geradem Raycast und lediglich 11,1% das virtuelle Zeigegerät mit parabolischem Raycast.

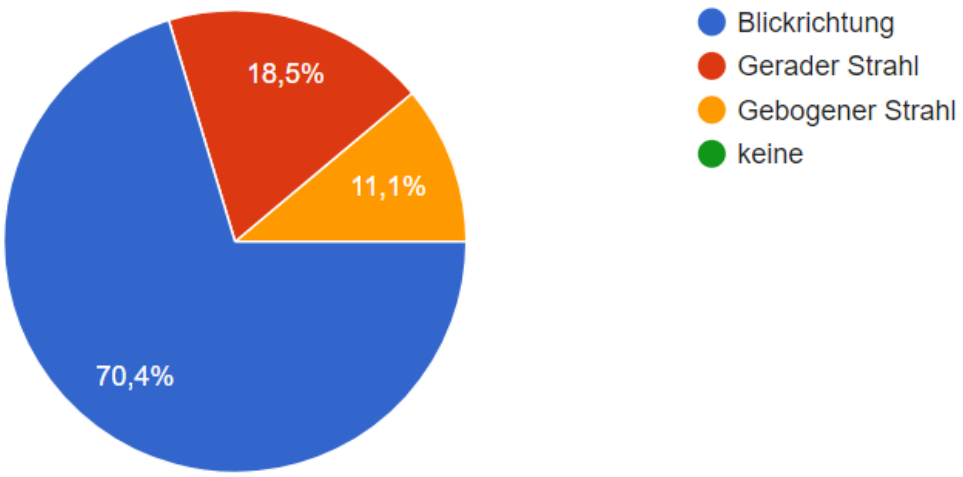


Abbildung 4.15: *Kuchendiagramm zur prozentualen Verteilung der präferierten Eingabemethode*

Es kann davon ausgegangen werden, dass das Ergebnis stark durch die Schwierigkeit beeinflusst wurde, die Ortspunkte auf der virtuellen Treppe zu selektieren. Dies war auch anhand den Kommentaren der Probanden während der Studie zu erkennen. Auch die abgegebenen Freitext-Kommentare waren aufschlussreich. Einige der Probanden, die die Selektion mittels Blickrichtung bevorzugten, gaben als Grund an, dass die Eingabemethode präziser, einfacher und intuitiv war. Probanden, die das virtuelle Zeigegerät bevorzugten, merkten zumeist an, dass sie es als angenehm empfanden, dass sie Objekte unabhängig von der Kopfbewegung selektieren konnten und sich somit frei in der VE umsehen konnten. Ein Proband bemerkte ebenfalls, dass er sich sorgte „ungewollte Aktionen auszulösen“, während er sich umsieht.

Bildwiederholfrequenz: Die Geräteleistung der Smartphones, welche als Ausgabegeräte benutzt wurden wurde anhand der gemessenen durchschnittlichen Bildwiederholfrequenz (*engl.* „frames per second“, FPS) gemessen. Die verwendeten Geräte waren für die ersten neun Sitzungen ein *Samsung Galaxy S7*, für die nächsten neun Sitzungen ein *Sony Xperia Z5* und für die letzten Sitzungen am Lehrstuhl für Medientechnik ein *Samsung Galaxy S5*. Der Wert sollte, um eine für den Nutzer möglichst angenehme Darstellung zu gewährleisten, nicht unter 60 Bildern pro Sekunde liegen. Beispielsweise werden VR-Spiele für Sony Playstation VR nicht zertifiziert, wenn sie diesen Wert unterschreiten [19].

Im entwickelten Prototypen war es nicht möglich, diesen Wert zu erreichen. Da insbesondere der parabolische Raycast mehr Berechnungen pro Zyklus erfordert, wurde angenommen, dass dieser den größten Leistungsabfall erzeugt. Die gemessenen Mittelwerte der FPS in Tabelle 4.11 zeigen, dass das aktuellste Gerät eine konstante mittlere FPS Zahl für alle Eingabemethoden aufwies. Jedoch muss hier angemerkt werden, dass bei den ersten acht Sitzungen die FPS-Zahlen nicht korrekt in der Datenbank gespeichert wurden. Somit beruht der Wert der Leistung des *Samsung Galaxy S7* auf nur auf der Messung bei einer einzigen Sitzung. Bei den Sitzungen mit dem *Sony Xperia Z5* wurden nicht alle Reiter des Browsers geschlossen, was eine Reduzierung der Geräteleistung bewirkte, die erst zu spät während der Studie bemerkt wurde. Aufgrund dieser Fehler wird dieses Ergebnis nicht weiter untersucht. Da die FPS-Zahl des *Samsung Galaxy S7* bei den verschiedenen Eingabemethoden kaum abweicht, wird geschlossen, dass bei aktuellen und zukünftigen Geräten der Mehraufwand des parabolischen Raycast sich nicht stark auf die Geräteleistung auswirkt. Bei älteren Geräten kann wegen der Fehler bei der Durchführung keine repräsentative Aussage getroffen werden. Bei der Nutzung des *Samsung Galaxy S5* wurden zur Vermeidung der fehlerhaften Messungen jedoch alle Reiter des Browser nach Abschluss einer Aufgabe geschlossen. Es lässt sich also durchaus die Aussage formulieren, dass sich die zahl der FPS zwischen den Geräten *Samsung Galaxy S8* und *Samsung Galaxy S5* drastisch unterscheiden, weswegen für die Nutzung des IV in mobilen Browsern ein aktuelles Gerät empfohlen wird. Auf dem älteren Samsung-Gerät wirkte sich auch der Mehraufwand durch die Nutzung des parabolischen Raycast negativ auf die Bildwiederholungsrate aus.

	Blickrichtung	Raycast	Raycast (parabolisch)
Samsung Galaxy S7	53.54230678	54.93514323	53.81064064
Sony Xperia Z5	30.64293085	27.88347102	19.02489016
Samsung Galaxy S5	38.46004919	37.96376576	28.0744409

Tabelle 4.11: *Bildwiederholungsrate der verwendeten Ausgabegeräte bei unterschiedlichen Eingabemethoden*

5 Zusammenfassung und Ausblick

Im Zuge der Arbeit wurden drei unterschiedliche Bedienkonzepte für die Navigation in einer webbasierten VR-Anwendung entwickelt und mit einer Benutzerstudie auf ihre Gebrauchstauglichkeit hin evaluiert. Die Nutzerstudie evaluierte die benötigte Zeit und begangenen Fehler in der Selektion von Objekten in einer virtuellen Umgebung. Desweiteren wurden qualitative Daten zur Nutzerzufriedenheit und der benötigten Anstrengung erhoben und statistisch analysiert. Zudem wurden Effekte der VR-Anwendung auf Symptome der *Cyber Sickness* ermittelt.

Die erste Eingabemethode zur Selektion von Objekten war die freihändige Eingabe mittels Blickrichtung. Die beiden anderen Eingabemethoden benutzten ein zweites Smartphone als virtuelles Zeigegerät. Die beiden virtuellen Zeigegeräte unterschieden sich in der Form des ausgesendeten Strahls zur Selektion von Objekten. Bei der ersten Methode wurde ein gerader Strahl ausgesandt, bei der zweiten ein Strahl mit parabolischer Form.

Die Nutzerstudie ergab keine signifikanten Unterschiede bei der Selektion von Objekten auf der Bodenfläche der virtuellen Umgebung. Bestand ein Höhengefälle und lagen die Punkte auf einer höheren Position als der Nutzer, so waren die Ergebnisse jedoch nicht erwartungsgemäß. Bei der Entwicklung wurde davon ausgegangen, dass der parabolische Strahl zur Selektion von Objekten auf einem höheren Niveau eine schnellere Selektion ermöglicht. Die genutzte virtuelle Umgebung bot jedoch keine echte Geometrie, die den geraden Strahl des virtuellen Zeigegeräts oder der Steuerung mit der Blickrichtung hätte blockieren können. Vielmehr bestand die Umgebung nur aus einer großen Sphäre, auf der ein 360°-Panoramabild dargestellt wurde, sowie die zu selektierenden Ortspunkte. Eine große Zahl der Nutzer empfand es als kompliziert, mit dem parabolischen Strahl die Objekte exakt zu treffen. Dies führte dazu, dass die Selektion mittels Blickrichtung von einer Großzahl der Nutzer als angenehmer empfunden wurde als die Selektion mittels virtuellem Zeigegerät. Die Aufgabe bestand jedoch nur aus der Selektion von Objekten in der VE und verlangt keine weitere Interaktion wie dem Informationsgewinn über einzelne Orte der virtuellen Umgebung. Es sollte also zukünftig untersucht werden, welche Eingabemethoden sich als effizient erweisen, wenn der Nutzer außer der Selektion noch weitere Arbeitsschritte zu absolvieren hat. Beispielsweise die Wegfindung in Innenräumen oder Informationsgewinnung über die Umgebung.

Somit sind die Ergebnisse der Nutzerstudie nicht repräsentativ für die Nutzung in einer virtuellen Umgebung mit echter Geometrie. Vielversprechend ist die Option des IV anhand der Daten der Punktwolken die tatsächliche physische Beschaffenheit der Innenräume zu analysieren. Anhand dieser Analyse kann dann eine korrekte virtuelle Repräsentation des Raums erstellt werden, statt nur die aufgenommenen 360°-Panoramen zu verwenden. Mit Voranschreiten der Entwicklung dieser Version könnten die Vorteile der virtuellen Zeigegeräte besser aufzeigbar sein.

Nachdem durch die Beobachtung und Gespräche mit den Probanden der Nutzerstudie klar wurde, dass vor allem der Aufstieg der virtuellen Treppe das Hauptproblem für die meisten Probanden darstellte, wurde auch der Prototyp dahingehend geändert, dass die im NavVis IV vorhandene Funktionalität zum Wechseln des Stockwerks in der VE durch Betätigen einer Schaltfläche im dreidimensionalen Raum ermöglicht wird. Für die aktuelle Version von three.js existiert momentan keine Komponenten, die eine einfache Erzeugung von dreidimensionalen grafischen Benutzeroberflächen ermöglichen. Nachträglich wurden deswegen einfache dreidimensionale Schaltflächen implementiert, die in der virtuellen Umgebung platziert werden können. Für das Wechseln des Stockwerkes kann statt der Selektion höher liegender Ortspunkte auch eine solche Schaltfläche ausgewählt werden, was das Wechseln von Stockwerken vereinfachen soll. Auch weitere Funktionalitäten des IV können so genutzt werden, beispielsweise die Routenplanung.

Für die Smartphone basierte Interaktion in Browser basierten virtuellen Umgebungen können noch viele Ansätze untersucht werden. Während sich diese Arbeit im wesentlichen mit der Bedienung einer VE mit ein oder zwei Smartphones befasst hat, können Entwicklungen in der Hardwaretechnologie jedoch auch weitere Eingabekonzepte ermöglichen.

Nicht untersucht wurden Eingabemethoden, die die Hände des Nutzers mittels einer Tiefenkamera in der VE darstellen können. Solche Kameras sind bereits fest in Mobilgeräten wie Notebooks verbaut und neueste Entwicklungen lassen annehmen, dass diese unter Umständen bald schon in handelsüblichen Smartphones verbaut sein werden [43]. Da diese Art der Eingabe sich vor allem für die Manipulation von Objekten eignet, wird davon ausgegangen, dass andere Eingabemethoden für die Selektion vielversprechender sind. Für die Manipulation von 3D-Objekten oder -Welten kann die Verwendung einer virtuellen Hand jedoch vorteilhaft sein und sollte somit weiter untersucht werden.

Auch direkte menschliche Eingabe, wie Spracheingabe, wurde nicht auf Gebrauchstauglichkeit für die Nutzung im NavVis IndoorViewer untersucht. Diese könnte vor allem für die Steuerung mittels Blickrichtung eine Möglichkeit bieten, Eingaben zu bestätigen. Somit kann die Notwendigkeit einer automatisierten Bestätigung mittels des *dwell-timers* [20] eliminiert werden.

Für die Navigation innerhalb einer IV-Instanz kann auch eine Karte genutzt werden (s. Abb. 3.2 c) rechts unten). Die Nutzung einer solchen Karte kann für Navigation in Gebieten hilfreich sein, von denen man in der Ich-Perspektive nur einen kleinen Ausschnitt sieht — vor allem, wenn der Nutzer sich über weite Strecken bewegen muss. Die Selektion der Ortspunkte auf einer virtuellen Karte kann, insofern die Repräsentation der Karte mit den Eindrücken der Umgebung übereinstimmt, ebenfalls eine geeignete Navigationsmethode sein, auch in immersiven VEs [5](S. 211). Die weitere Forschung sollte also Eingabemethoden überprüfen, die exozentrische Interaktion mit der gegebenen VE zulassen.

Diese Arbeit befasst sich hauptsächlich mit der Selektion von Objekten mittels Blickrichtung eines HMDs und der Nutzung eines zweiten Smartphones als virtuelles Zeigegerät. Multimodale Eingabekonzepte wie die Verbindung von Blickrichtung und Spracheingabe im Browser können mit der steigenden Zahl von Webdiensten und Browserfunktionen zukünftig jedoch von hohem Interesse sein. Auch Projekte wie *Google Tango* werden förderlich für den Grad der Immersion von Smartphone basierten VR-Anwendungen sein können. Gerade die hohe Verfügbarkeit dieser Endgeräten in Haushalten bestimmen ihr großes Potenzial. Es wird also empfohlen, die Entwicklungen in diesem Bereich zu verfolgen und weitere bestehende Interaktionskonzepte aus traditionellen VR-Anwendungen auf Anwendbarkeit mit Smartphones als Ein- und Ausgabegeräte zu untersuchen.

Anhang

Verwendete Fragebögen

SSQ zur Erhebung der Symptome der „Cyber Sickness“

Der folgende Fragebogen wurde dabei jeweils vor und nach dem Experiment vorgelegt.

Bitte gib hier noch einmal auf dich zutreffende Symptome an, ob und in welcher Schwere sie im Moment zutreffen.

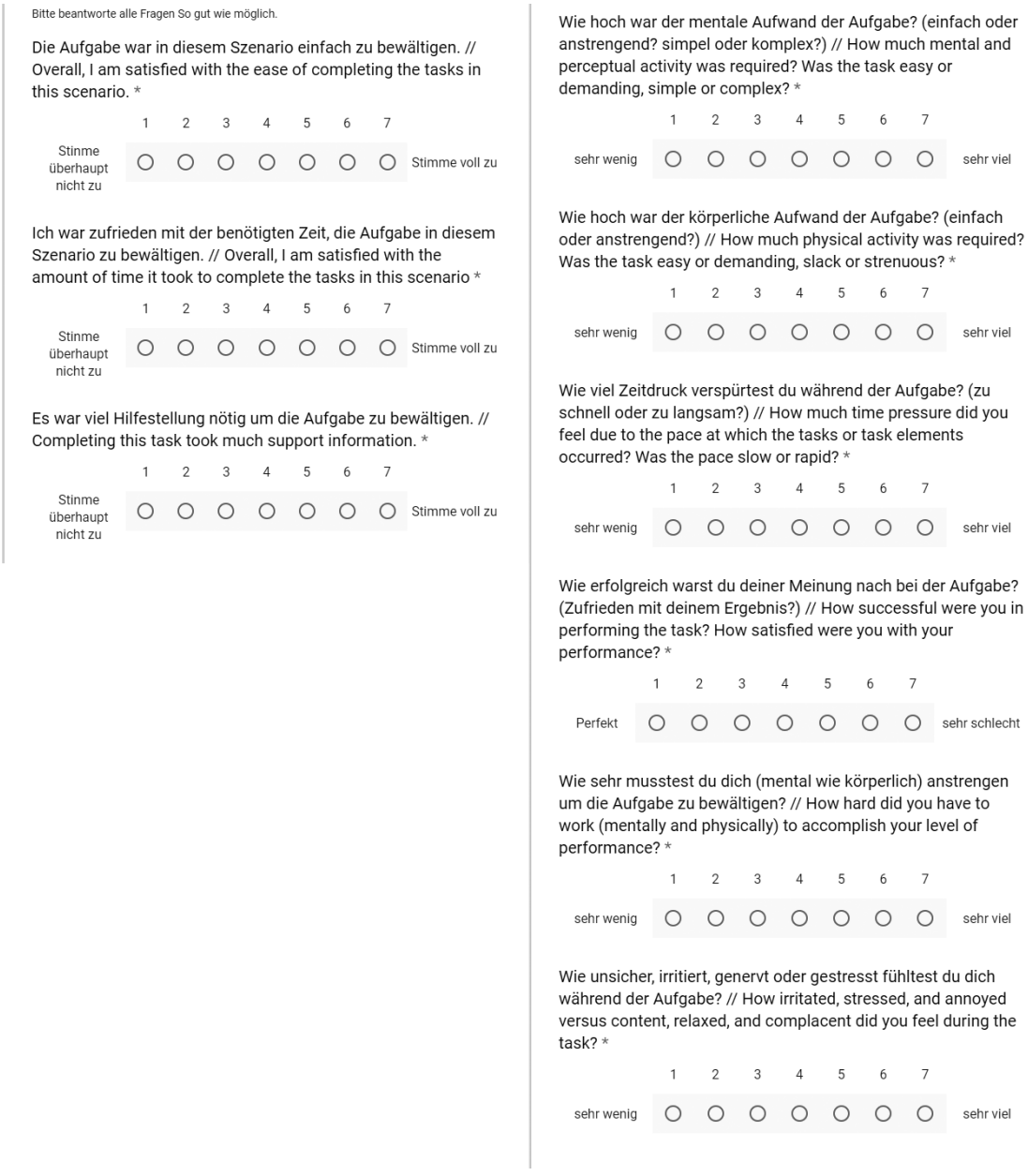
Please fill in this questionnaire. Circle below if any of the symptoms apply to you now.

Generelles Unwohlsein

	gar nicht / none	ein wenig / slight	moderat	schlimm / severe
Generelles Unwohlsein / General discomfort	○	○	○	○
Erschöpfung / Fatigue	○	○	○	○
Langweile / Boredom	○	○	○	○
Benommenheit / Drowsiness	○	○	○	○
Kopfschmerzen / Headache	○	○	○	○
Augen angestrengt / Eyestrain	○	○	○	○
Fokus-Probleme / Difficulty focusing	○	○	○	○
Speichelfluss erhöht / Salvation Increase	○	○	○	○
trockener Mund / Salvation Decrease	○	○	○	○
Schwitzen / Sweating	○	○	○	○
Übelkeit / Nausea	○	○	○	○
Konzentrationsprobleme / Difficulty concentrating	○	○	○	○
Depression	○	○	○	○
"Kopf voll" / "Fullness of Head"	○	○	○	○
Sicht unscharf / Blurred vision	○	○	○	○
Schwindelgefühl (Augen zu) / Dizzyness (eyes open)	○	○	○	○
Schwindelgefühl (Augen offen) / Dizzyness (eyes closed)	○	○	○	○
Schwindel, Drehen / Vertigo	○	○	○	○
Mattheit / Faintness	○	○	○	○
aktive Wahrnehmung der Atmung / Aware of breathing	○	○	○	○
Magenprobleme / Stomach Awareness	○	○	○	○
Appetitlosigkeit / Loss of Appetite	○	○	○	○
Harndrang / Desire to move bowels	○	○	○	○
Verwirrung / Confusion	○	○	○	○
Aufstoßen / Burping	○	○	○	○
Übergeben / Vomiting	○	○	○	○
Sonstiges / other	○	○	○	○

Post-Task-Fragebogen zur Nutzerzufriedenheit und Anstrengung

Der Fragebogen wurde den Probanden nach der Benutzung jeder einzelnen Eingabemethode vorgelegt.



Bitte beantworte alle Fragen So gut wie möglich.

Die Aufgabe war in diesem Szenario einfach zu bewältigen. // Overall, I am satisfied with the ease of completing the tasks in this scenario. *

1 2 3 4 5 6 7

Stinme überhaupt nicht zu ○ ○ ○ ○ ○ ○ ○ Stimme voll zu

Ich war zufrieden mit der benötigten Zeit, die Aufgabe in diesem Szenario zu bewältigen. // Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario *

1 2 3 4 5 6 7

Stinme überhaupt nicht zu ○ ○ ○ ○ ○ ○ ○ Stimme voll zu

Es war viel Hilfestellung nötig um die Aufgabe zu bewältigen. // Completing this task took much support information. *

1 2 3 4 5 6 7

Stinme überhaupt nicht zu ○ ○ ○ ○ ○ ○ ○ Stimme voll zu

Wie hoch war der mentale Aufwand der Aufgabe? (einfach oder anstrengend? simpel oder komplex?) // How much mental and perceptual activity was required? Was the task easy or demanding, simple or complex? *

1 2 3 4 5 6 7

sehr wenig ○ ○ ○ ○ ○ ○ ○ sehr viel

Wie hoch war der körperliche Aufwand der Aufgabe? (einfach oder anstrengend?) // How much physical activity was required? Was the task easy or demanding, slack or strenuous? *

1 2 3 4 5 6 7

sehr wenig ○ ○ ○ ○ ○ ○ ○ sehr viel

Wie viel Zeitdruck verspürtest du während der Aufgabe? (zu schnell oder zu langsam?) // How much time pressure did you feel due to the pace at which the tasks or task elements occurred? Was the pace slow or rapid? *

1 2 3 4 5 6 7

sehr wenig ○ ○ ○ ○ ○ ○ ○ sehr viel

Wie erfolgreich warst du deiner Meinung nach bei der Aufgabe? (Zufrieden mit deinem Ergebnis?) // How successful were you in performing the task? How satisfied were you with your performance? *

1 2 3 4 5 6 7

Perfekt ○ ○ ○ ○ ○ ○ ○ sehr schlecht

Wie sehr musstest du dich (mental wie körperlich) anstrengen um die Aufgabe zu bewältigen? // How hard did you have to work (mentally and physically) to accomplish your level of performance? *

1 2 3 4 5 6 7

sehr wenig ○ ○ ○ ○ ○ ○ ○ sehr viel

Wie unsicher, irritiert, genervt oder gestresst fühltest du dich während der Aufgabe? // How irritated, stressed, and annoyed versus content, relaxed, and complacent did you feel during the task? *

1 2 3 4 5 6 7

sehr wenig ○ ○ ○ ○ ○ ○ ○ sehr viel

Inhalt der beigelegten CD

- Programmcode der für die Nutzerstudien verwendeten Prototypen inkl. aller Eingabemethoden, sowie der experimentellen Implementierung der Sprachsteuerung sowie der Implementierung der Handerkennung mit dem Leap Motion Controller:
  /Prototyp/

- Bedienungsanleitung zum Starten der lokalen Webserver und der Anwendung:
  /Prototyp/README.txt

- SPSS-Datensätze die zur Analyse der qualitativen und quantitativen Daten genutzt wurde. Vorhanden als SPSS-Datei (.sav) und csv:
  *Daten/data/scores.**: Daten aller ermittelten Gesamtwerte für Zeiten und Fragebögen
  *Daten/data/scores-completed.**: Daten aller ermittelten Gesamtwerte für Zeiten und Fragebögen, mit Kennzeichnung der Probanden, die eine Methode nicht erfolgreich abgeschlossen haben
  *Daten/data/scores-ssq.**: Daten des Simulator-Sickness-Fragebogens
  Daten/data/Rohdaten-quantitativ.csv: Alle gemessenen, nicht aufsummierten qualitativen Daten (Zeiten, Fehler, benötigte Schritte, FPS) pro Navigationsschritt sowie die Ausprägungen jeder Frage der vorgelegten Fragebögen und deren aufsummierte Gesamtwerte. Eine Zeile entspricht somit allen erhobenen Daten eines Probanden.
  Daten/data/Rohdaten-qualitativ.csv: Abgegbene Antworten der Nutzer für alle Fragebögen in Textform. Eine Zeile entspricht also allen erhobenen qualitativen Daten eines Probanden. Beinhaltet auch die Freitext-Antworten zur präferierten Methode.

- Die genutzten Fragebögen zur Erhebung qualitativer Daten, zu PDFs konvertierte Webformulare.
  /Fragebogen/

- Alle Dateien die zur Herstellung der schriftlichen Ausarbeitung genutzt wurden, alle LaTEX-Dateien, Bibliografie-Dateien, sowie Abbildungen
  /Schriftlich/
  /Schriftlich/figures: alle verwendeten Abbildungen

- Die gesamte Ausgabe der durchgeführten SPSS Analysen als SPSS Output-Viewer-Datei (.spv) als auch HTML.
  */Daten/output/output.**

- PDFs der Literatur die frei zum Download stand und keine Web-Quelle war
  /Literatur/

Literatur

[1] 3DCONNEXION. SpaceMouse Pro Produktseite. http://www.3dconnexion.de/products/spacemouse/spacemousepro.html, 2017. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[2] AMAZON. Alexa voice service. https://developer.amazon.com/de/alexa-voice-service, 2017. Quelle aus dem Web. (abgerufen am: 23.06.2017).

[3] ARGELAGUET, F., AND ANDUJAR, C. A survey of 3d object selection techniques for virtual environments. *Computers & Graphics 37*, 3 (2013), 121–136.

[4] BOWMAN, D. A., AND HODGES, L. F. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), I3D '97, ACM, pp. 35–ff.

[5] BOWMAN, D. A., KRUIJFF, E., LAVIOLA, J. J., AND POUPYREV, I. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[6] BYERS, J. C., BITTNER, A. C., AND HILL, S. Traditional and raw task load index (tlx) correlations: Are paired comparison necessary? In *Advances in Industrial Ergonomics and Safety* (1989), A. Mital, Ed.

[7] CABELLO, R. three.js documentation. https://threejs.org/, 2016. Quelle aus dem Web. (abgerufen am: 01.06.2017).

[8] CABELLO, R. three.js docs - stereocamera. https://threejs.org/docs/#api/cameras/StereoCamera, 2017. Quelle aus dem Web. (abgerufen am: 20.06.2017).

[9] CRUZ-NEIRA, C., SANDIN, D. J., AND DEFANTI, T. A. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, pp. 135–142.

[10] DIEBEL, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors.

[11] FEINER, A. O. S. The flexible pointer: An interaction technique for selection in augmented and virtual reality. In *Proc. UIST'03* (2003), pp. 81–82.

[12] FERNANDES, A. S., AND FEINER, S. K. Combating vr sickness through subtle dynamic field-of-view modification. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on* (2016), IEEE, pp. 201–210.

[13] FIELD, A., AND HOLE, G. *How to Design and Report Experiments*. 2003.

[14] FOLEY, J. D., WALLACE, V. L., AND CHAN, P. The human factors of computer graphics interaction techniques. *IEEE Computer Graphics and Applications 4*, 11 (Nov 1984), 13–48.

[15] FORSBERG, A., HERNDON, K., AND ZELEZNIK, R. Aperture based selection for immersive virtual environments. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (1996), ACM, pp. 95–96.

[16] GOOGLE. Google tango produktseite. https://get.google.com/tango/, 2017. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[17] GOOGLE. Speech api - speech recognition | google cloud platform. https://cloud.google.com/speech/, 2017. Quelle aus dem Web. (abgerufen am: 23.06.2017).

[18] GOOGLEVR TEAM. googlevr / webvr-polyfill. https://github.com/googlevr/webvr-polyfill, 2017. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[19] HALL, C.

[20] HANSEN, J. P., JOHANSEN, A. S., HANSEN, D. W., ITOH, K., AND MASHINO, S. Command without a click: Dwell time typing by mouse and gaze selections. In *Proceedings of Human-Computer Interaction–INTERACT* (2003), pp. 121–128.

[21] HERMAN, D., WAGNER, L., AND ZAKAI, A. asm.js working draft - 18 august 2014. http://asmjs.org/spec/latest/, 2014. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[22] HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY. A brief explanation of the simulator sickness questionnaire (ssq). http://www.cybersickness.org/Simulator_Sickness_Questionnaire.htm. Quelle aus dem Web. (abgerufen am: 01.05.2017).

[23] IBM. Speech to text | about speech to text | ibm watson developer cloud. https://www.ibm.com/watson/developercloud/doc/speech-to-text/index.html, 2017. Quelle aus dem Web. (abgerufen am: 23.06.2017).

[24] JOHNSON, E. What are the differences among virtual, augmented and mixed reality? https://www.recode.net/2015/7/27/11615046/whats-the-difference-between-virtual-augmented-and-mixed-reality. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[25] KENNEDY, R. S., LANE, N. E., BERBAUM, K. S., AND LILIENTHAL, M. G. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology 3*, 3 (1993), 203–220.

[26] LAERD STATISTICS. Friedman test in spss statistics. https://statistics.laerd.com/premium/spss/ft/friedman-test-in-spss-1.php, 2013. Quelle aus dem Web. (abgerufen am: 29.06.2017).

[27] LAERD STATISTICS. One-way repeated measures anova using spss statistics. https://statistics.laerd.com/premium/spss/owrma/one-way-repeated-measures-anova-in-spss-10.php#carry-on, 2013. Quelle aus dem Web. (abgerufen am: 29.06.2017).

[28] LAERD STATISTICS. Statistical test selector. https://statistics.laerd.com/premium/sts/index.php, 2013. Quelle aus dem Web. (abgerufen am: 29.06.2017).

[29] LAVIOLA JR, J. J. A discussion of cybersickness in virtual environments. *ACM SIGCHI Bulletin 32*, 1 (2000), 47–56.

[30] LEWIS, J. R. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: The asq. *SIGCHI Bull. 23*, 1 (Jan. 1991), 78–81.

[31] LIANG, J., AND GREEN, M. Jdcad: A highly interactive 3d modeling system. *Computers & graphics 18*, 4 (1994), 499–506.

[32] MAPES, D. P., AND MOSHELL, J. M. A two-handed interface for object manipulation in virtual environments. *Presence: Teleoperators & Virtual Environments 4*, 4 (1995), 403–416.

[33] MIFSUD, J. Usability metrics: A guide to quantify the usability of any system. http://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/, Juni 2015. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[34] MOZILLA A-FRAME. https://aframe.io/docs/0.5.0/introduction/. https://aframe.io/docs/0.5.0/introduction/, 2016. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[35] MOZILLA DEVELOPER NETWORK. Gamepad api. https://developer.mozilla.org/en-US/docs/Web/API/Gamepad, 2017. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[36] MOZILLA DEVELOPER NETWORK. Keyboardevent spezifikation. https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent, 2017. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[37] MUKUNDAN, R. Quaternions: From classical mechanics to computer graphics, and beyond. In *Proceedings of the 7 th Asian Technology Conference in Mathematics, 2002* (2002).

[38] NAVVIS. IndoorViewer Documentation. https://www.navvis.com/docs/indoorviewer/, 2017. Quelle aus dem Web. (abgerufen am: 10.01.2017).

[39] REASON, J. T., AND BRAND, J. J. *Motion sickness.* Academic press, 1975.

[40] RICCIO, G. E., AND STOFFREGEN, T. A. An ecological theory of motion sickness and postural instability. *Ecological psychology 3*, 3 (1991), 195–240.

[41] SAURO, J. Comparing between- and-within-subjects studies. https://measuringu.com/between-within/, August 2015. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[42] SAURO, J. Measuringu: Measuring errors in the user experience. https://measuringu.com/errors-ux/, August 2015. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[43] SHAH, A. Intel's smartphone with integrated realsense 3d camera to ship for $399 | pcworld. http://www.pcworld.com/article/3019937/hardware/intels-smartphone-with-integrated-realsense-3d-camera-to-ship-for-399.html, Januar 2016. Quelle aus dem Web. (abgerufen am: 01.05.2017).

[44] STANNEY, K. M., AND KENNEDY, R. S. The psychometrics of cybersickness. *Communications of the ACM 40*, 8 (1997), 66–68.

[45] STATISTISCHES BUNDESAMT DESTATIS. Ausstattung privater Haushalte mit Informations- und Kommunikationstechnik. https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsguetern/Tabellen/Infotechnik_D.html. Quelle aus dem Web. (abgerufen am: 26.06.2017).

[46] SUTHERLAND, I. E. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I* (1968), ACM, pp. 757–764.

[47] UNITY TECHNOLOGIES. Webgl: Interacting with browser scripting. https://docs.unity3d.com/Manual/webgl-interactingwithbrowserscripting.html, 2017. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[48] VANACKEN, L., GROSSMAN, T., AND CONINX, K. Exploring the effects of environment density and target visibility on object selection in 3d virtual environments. In *3D User Interfaces, 2007. 3DUI'07. IEEE Symposium on* (2007), IEEE.

[49] VIRTUAL REALITY SOCIETY. History of virtual reality. https://www.vrs.org.uk/virtual-reality/history.html, 2016. Quelle aus dem Web. (abgerufen am: 10.06.2017).

[50] W3-KONSORTIUM. Deviceorientation event specification. https://www.w3.org/TR/2016/CR-orientation-event-20160818/, 2016. Quelle aus dem Web. (abgerufen am: 23.06.2017).

[51] W3-KONSORTIUM. Webvr editor's draft, 17 april 2017. https://w3c.github.io/webvr/spec/latest/, 2017. Quelle aus dem Web. (abgerufen am: 18.06.2017).

[52] WIKIPEDIA. EN ISO 9241. https://de.wikipedia.org/wiki/EN_ISO_9241, Oktober 2016. Quelle aus dem Web. (abgerufen am: 10.04.2017).

[53] WIKIPEDIA. Euler angles. https://en.wikipedia.org/wiki/Euler_angles, 2017. Quelle aus dem Web. (abgerufen am: 23.06.2017).

[54] WYSS, H. P., BLACH, R., AND BUES, M. isith-intersection-based spatial interaction for two hands. In *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on* (2006), IEEE, pp. 59–61.

Abbildungsverzeichnis

Tabellenverzeichnis