Kollaboratives Erkunden von Software mithilfe virtueller Realität in ExplorViz

Bachelorarbeit

Timm Häsemeyer

28. September 2017

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Arbeitsgruppe Software Engineering

Betreut durch: Prof. Dr. Wilhelm Hasselbring
M.Sc. Christian Zirkelbach

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel, 28. September 2017

Zusammenfassung

Die Architektur des Softwareüberwachungs- und Visualisierungstools *ExplorViz* befindet sich im Umbruch. Hierbei wird wird von einer monolithischen zu einer verteilten Architektur mit *Frontend* und *Backend* gewechselt. Die monolithische Implementierung von *ExplorViz* verfügt bereits über die Möglichkeit die Applikationsansicht in virtueller Realität darzustellen. Diese Visualisierung ist auf Grundlage der *Oculus Rift Development Kit 1 (Dk1)* realisiert worden, welche nicht über eine Positionserkennung verfügt. Hierdurch ist nur das Umschauen und Interagieren mit einem Controller oder durch Gesten möglich. Auf das Raumkonzept der virtuellen Realität, welches die Navigation und Fortbewegung durch Gehen im realen Raum ermöglicht, wurde aus diesem Grund nicht zurückgegriffen. Durch die Nutzung des Raumkonzepts wird beim Nutzer der Raumsinn aktiviert, durch welchen erkundete Sachverhalte besser in Erinnerung behalten werden. Die aktuellen Head-Mounted Displays (HMDs) ermöglichen die Positionsbestimmung des HMDs und damit des Benutzers im Raum.

Diese Abschlussarbeit widmet sich der Entwicklung eines Plugins für die neue *ExplorViz*-Architektur, welches das vollständige Erkunden einer Software in virtueller Realität ermöglichen soll. Hierzu wird die bestehende Landschaftsansicht der neuen Architektur dreidimensional dargestellt und mit der Applikationsansicht verschmolzen. Dafür werden Teile der bestehenden VR-Visualisierung der monolithischen Architektur übernommen und die neue Landschaftsansicht um das Raumkonzept sowie eine kollaborative Nutzbarkeit erweitert. Im Rahmen dieser Arbeit werden dazu zunächst die notwendigen Grundlagen vermittelt, verwendbare Ansätze identifiziert, ein Konzept des Plugins erstellt, dieses Implementiert und im Anschluss evaluiert. Letzteres ist besonders wichtig, da uns derzeit kein ähnlicher Ansatz bekannt ist, der das Raumkonzept der virtuellen Realität für Softwareüberwachungs- und Visualisierungstools nutzt. Aus diesem Grund dient die Evaluation der Verifizierung der Nutzbarkeit des Raumkonzepts bei Softwareüberwachungs- und Visualisierungstools hinsichtlich Navigation und Empfinden beim Erkunden der Softwarelandschaft.

Im Rahmen der Evaluation hat sich gezeigt, dass die Nutzer das Erkunden der virtuellen Umgebung positiv und die Navigation in dieser sehr intuitiv empfunden haben. Die Interaktion mithilfe der Controller bietet noch Verbesserungspotenzial und wurde deshalb als eher praktisch vom Benutzer eingestuft. Des Weiteren hat die Studie ergeben, dass der Nutzer durch die Kombination aus 3D-Visualisierung und intuitiver Navigation in der virtuellen Umgebung, sehr einfach zusätzliche Informationen aus der Landschaft gewinnen kann. Deshalb stellt das Erkunden einer Software mit *ExplorViz* in virtueller Realität eine praktische Alternative zum Erkunden am Bildschirm dar, die vom Benutzer gut angenommen wird.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Virtuelle Realität (VR) simuliert physische Anwesenheit in virtuellen Welten, bietet Interaktionsmöglichkeiten mit dieser und hat in den letzten Jahren zunehmend an Bedeutung gewonnen [Brill 2009; Pérez Fernández und Alonso 2015]. Die Reduzierung der Anschaffungskosten und gleichzeitige Leistungssteigerung der entsprechenden Hardware hat dazu geführt, dass *VR* massentauglich geworden ist [Bozgeyikli et al. 2016]. Aufgrund dieser Tatsachen, haben sich viele namenhafte Hersteller mit der Nutzbarkeit von *virtueller Realität* beschäftigt und entsprechende Produkte entwickelt.

Beispielsweise hat Sony ein Produkt namens *Playstation VR*[1] für die aktuelle Playstation auf den Markt gebracht, welches das Erleben von Videospielen in der *virtuellen Realität* ermöglicht. *Oculus*[2] sowie *HTC*[3], in Zusammenarbeit mit *Valve (Steam VR)*[4], bieten eine computerbasierte Lösung für das Nutzen von VR-Anwendungen. Des Weiteren hat *Samsung*[5] eine handybasierte Lösung für das Nutzen von VR-Anwendungen entwickelt. Die Produkte und Lösungen dieser Firmen haben dazu beigetragen, dass viele Haushalte Zugang zur *virtuellen Realität* haben.

Virtuelle Realität findet oft in Bereichen Anwendung, in denen eine Simulation viele Vorteile bietet oder notwendig ist. Das Proben von Szenarien im Alltag, wie eine Feuerevakuierung oder das korrekte Verhalten bei Erdbeben, sind kostspielig und schwer Umzusetzen. Des Weiteren sind solche Szenarien oft wenig realistisch, da aus Sicherheitsgründen auf Feuer und Erderschütterungen verzichtet wird. Eine 3D-Visualisierung, die *VR* nutzt, bietet hier einen alternativen Ansatz. Das Szenario wird virtuell dargestellt und die Teilnehmer betreten dieses mithilfe entsprechender Hard- und Software. Feuer, Rauch und Erderschütterungen können nun mit eingebunden werden, ohne dass sich ein Teilnehmer in echter Gefahr befindet [Sharma et al. 2014; Li et al. 2017].

Virtuelle Realität kann ebenfalls bei der Entwicklung von Software von Vorteil sein. UML-Diagramme sind in der Softwareentwicklung sehr wichtige Werkzeuge, um Strukturen, Daten- und Kontrollflüsse darzustellen. 3D- Visualisierungen von UML-Diagrammen, die

[1] https://www.playstation.com/de-de/explore/playstation-vr/?utm_medium=Paid_Search&utm_campaign=&utm_source=&utm_term=pa-co-103636&utm_content=&emcid=pa-co-103636

[2] https://www.oculus.com

[3] https://www.vive.com/de/

[4] http://store.steampowered.com/steamvr?l=german

[5] http://www.samsung.com/de/wearables/gear-vr-r324/SM-R324NZAADBT/?cid=de_ppc_google_im-smartphones-gearvr_20170428_samsungvr-phrase&tmcampid=7&tmad=c&tmplaceref=c_DCO_FY17_Smartphones_GearVR_Brand+Product_P&tmclickref=p_samsung%20vr

in der virtuellen Realität erkundet werden, bleiben dem Benutzer besser in Erinnerung, da der dieser von der realen Welt abgeschirmt und somit konzentrierter ist [Zhang und sho Chen 2005]. Ein weiterer Grund dafür liegt darin, dass für die räumliche Wahrnehmung im menschlichen Körper das räumliche Gedachtnis zuständig ist. Es speichert räumliche Informationen und wird von speziellen Nervenzellen unterstützt, die aktiviert werden, wenn sich der Mensch durch den Raum bewegt. Bei der Navigation durch die virtuelle Umgebung wird diese durch die Kopf- oder Körperbewegung des Nutzers aktualisiert, wodurch diese Nervenzellen aktiviert werden [Elliott et al. 2015]. Resultierend können zusätzliche Informationen aus diesem Speicher abgerufen werden.

Des Weiteren bietet *virtuelle Realität* die Möglichkeit standortunabhängige Versammlungen von Personen, indem die virtuelle Umgebung als gemeinsamer Ort betreten wird. Das bietet sowohl privat als auch geschäftlich enormen Mehrwert. Diese Aspekte haben zur Motivation dieser Arbeit beigetragen. Im folgenden Abschnitt werden weitere Aspekte vorgestellt, die einen Beitrag zur Motivation geleistet haben.

1.1. Motivation

Große Softwarelandschaften bestehen oft aus vielen Servern, Anwendungen, Komponenten, Paketen und Klassen. Dies macht es schwierig einen Überblick über die Kommunikation und Datenflüsse innerhalb dieser Softwarelandschaft zu erhalten. *ExplorViz*[6] wird derzeit für die Überwachung und Leistungsanalyse von Softwarelandschaften verwendet [Zirkelbach et al. 2015]. Dabei stellt *ExplorViz* die Kommunikation zwischen den Servern und Anwendungen in einer Landschaftsansicht, sowie die Kommunikation innerhalb dieser in einer Applikationsansicht dar. Diese Ansichten werden in einem Webbrowser bereitgestellt. Eine mit *ExplorViz* überwachte Software wird momentan auf einem einzelnen Bildschirm betrachtet und kann nicht mit anderen Personen geteilt werden.

Diese Art der Darstellung profitiert nicht von den Vorteilen, welche beispielsweise die *virtuelle Realität* bietet. Einer dieser Vorteile ist das Raumkonzept, welches die Navigation in der virtuellen Welt durch intuitives Gehen, und Bewegen des Kopfes ermöglicht. Studien haben gezeigt, dass Informationen und Sachverhalte besser in Erinnerung behalten werden, wenn sie durch den Raumsinn unterstützt worden sind [Elliott et al. 2015]. Durch das Raumkonzept der virtuellen Realität wird genau dieser angesprochen.

ExplorViz befindet sich in einem architektonischen Umbruch. Die Architektur wechselt dabei von einer monolithischen [Fittkau et al. 2017; 2015, b] zu einer verteilten mit Frontend und Backend. Dabei werden die darzustellenden Daten der überwachten Software vom Backend an das Frontend gesendet. Die verteilte Architektur bezeichnen wir im Folgenden als *ExplorViz*. Abbildung 1.1 zeigt die monolithische und die verteilte Architektur mit *Frontend* und *Backend*. Des Weiteren ist geplant *ExplorViz* um eine Datenbankmonitoring, -Analyse und -Visualisierung zu erweitern [Zirkelbach 2017]. Die Umsetzung der alten

[6] https://github.com/ExplorViz

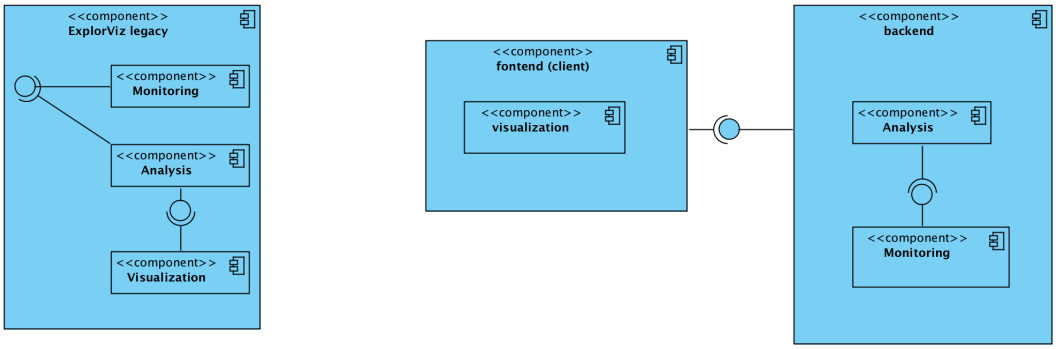

Abbildung 1.1. Architekturvergleich von *ExplorViz legacy* (links) und *ExplorViz* (rechts) in Form eines Komponentendiagramms

Architektur wird als *ExplorViz legacy* bezeichnet und verfügt bereits über eine Visualisierung in virtueller Realität [Fittkau et al. 2015a]. Diese Visualisierung ist jedoch nur in der Anwendungsansicht verfügbar und nutzt nicht das Raumkonzept der virtuellen Realität.

Aus diesem Grund widmet sich diese Arbeit der Erstellung eines Plugins für *ExplorViz*, sodass hier neben der standard 2D- und 3D-Visualisierung, auch eine für virtuelle Realität verfügbar ist. Diese Visualisierung soll das vollständige Erkunden einer Softwarelandschaft in virtueller Realität ermöglichen und Applikations- und Landschaftsansicht vereinen, damit ein nahtloses Nutzererlebnis möglich ist. Hierzu ist eine dreidimensionale Landschaftsansicht notwendig, in der eine 3D-Applikation aus der Applikationsansicht dynamisch erzeugt werden kann. Dazu wird die Landschaftsansicht um eine z-Koordinate erweitert, um das Raumkonzept der virtuellen Realität ergänzt und das Rendering entsprechend angepasst, sowie Teile des bestehenden VR-Modus übernommen. Diese Visualisierung soll, mithilfe intuitiverer Navigation und räumlicherer Darstellung, ein realeres Bild der Softwarelandschaft ermöglichen. Zudem soll eine standortunabhängige Zusammenarbeit mit anderen Personen möglich sein, sodass mehrere Benutzer - als Beobachter (*Spectator*) - die Sicht eines Benutzers, der das Modell manipuliert, teilen und nachvollziehen können. Im Rahmen dieser Zusammenarbeit soll die gemeinsame Manipulation der Landschaft in einer gemeinsamen, virtuellen Umgebung möglich sein, sodass mehrere Personen standortunabhängig eine Software mit *ExplorViz* erkunden können.

1.2. Ziele

1.2.1. Z1: Identifikation verwandter Arbeiten

Das Prinzip der Wissenschaft ist es neues Wissen zu schaffen und nicht bereits erwiesenes oder erforschtes erneut zu erarbeiten. Bereits gewonnenes Wissen kann dazu genutzt werden bestehende Ansätze zu erweitern oder zu vervollständigen. Des Weiteren ist es dadurch

möglich, Herausforderungen früher zu erkennen und dementsprechende Lösungsansätze zu finden. Deshalb ist es wichtig bestehende, verwandte Arbeiten zu identifizieren und vorzustellen, sowie zu analysieren und Schlüsse daraus für diese Arbeit zu ziehen. Ziel ist es somit, Arbeiten oder Anwendungen zu identifizieren, die einen *Multiuser*-Ansatz bieten, das Raumkonzept der virtuellen Realität unterstützen oder Ähnlichkeiten zu *ExplorViz* aufweisen.

1.2.2. Z2: Konzept/Entwurf

Die Entwicklung eines Plugins beinhaltet mehr als nur die Modifikation bestehender Methoden oder Klassen des Hauptprogramms. Die Erweiterung von *ExplorViz* um einen VR-Modus erfordert neue Komponenten, Relationen, Routen, Services und eine Anpassung des Layouts. Die Implementierung dessen kann schnell unübersichtlich werden, sodass ein Konzept für eine systematische Implementierung des Plugins unerlässlich ist. Des Weiteren erleichtert ein Entwurf das Erledigen und Markieren von Meilensteinen, sodass der aktuelle Zustand ablesbar ist. Das Ziel ist es, ein Konzept zur Erweiterung von *ExplorViz* zu entwerfen, sodass der Benutzer im VR-Modus das Raumkonzept in *ExplorViz* nutzen kann. Weiterhin soll eine standortunabhängige und kollaborative Nutzung möglich sein. Dieses Konzept setzt sich aus den beiden Konzepten der folgenden Unterkapitel zusammen.

Z2.1: Migration

Auf Grundlage der in Abschnitt 1.2.1 gewonnen Ergebnisse, sollen verwendbare Ansätze in das Konzept übernommen werden. Beispielsweise verfügt *ExplorViz legacy* bereits über einen VR-Modus [Fittkau et al. 2015a]. Das Head-Mounted Display wird für die Navigation und die Controller für die Interaktion in der virtuellen Umgebung genutzt. Dies birgt den Vorteil, dass wir auf Grundlage bestehender Ansätze, neue Funktionalitäten entwickeln und implementieren können.

Z2.2: Erweiterung

Das Ziel ist es, ein Konzept zu entwerfen, welches das Konzept aus Abschnitt 4.1 um die Realisierung des Raumkonzepts, eine 3D-Visualisierung der bisherigen Landschaftsansicht und deren Fusion mit der Applikationsansicht ergänzt. Abschließend soll das Konzept einen VR-Modus beschreiben, der das Raumkonzept nutzt, Landschafts- sowie Applikationsansicht vereint und optional kollaborativ genutzt werden kann.

1.2.3. Z3: Implementierung

Damit *ExplorViz* die gewünschten Eigenschaften zur Verfügung stellt, müssen diese implementiert werden. Ziel ist es die, im Entwurf genanten, Funktionalitäten zu implementieren und damit *ExplorViz* um diese zu erweitern.

1.2.4. Z4: Evaluation

Nach der Implementierung des Plugins ist es wichtig dieses zu bewerten. Ziel ist es, den entwickelten VR-Modus anhand einer Studie darauf zu überprüfen, ob er von den Benutzern angenommen wird und eine praktische Alternative zum Erkunden am Bildschirm darstellt.

1.3. Aufbau

In Kapitel 2 werden Grundlagen erläutert, welche für diese Arbeit eine wichtige Rolle spielen. Kapitel 3 beinhaltet die Analyse und Bewertung verwandter Arbeiten, Ansätze und Anwendungen hinsichtlich der Verwendbarkeit für diese Ausarbeitung. Kapitel 4 präsentiert den Entwurf des Plugins, welcher sich aus einem Migrations- und Erweiterungskonzept zusammensetzt. Der Entwurf definiert welche Funktionalitäten aus bestehenden Ansätzen übernommen und welche neu entwickelt werden. Die Implementierung der entworfenen Funktionalitäten und deren Integration in *ExplorViz* ist in Kapitel 5 dokumentiert und dargestellt. Die Studie zur Ermittlung des Mehrwerts dieser Implementierung für Softwareüberwachungs- und Visualisierungstools wird in Kapitel 6 vorgestellt und deren Ergebnisse bewertet. In Kapitel 7 werden die Schlussfolgerungen dargelegt und mögliche Verbesserungen sowie Ergänzungen des Plugins vorgestellt.

Kapitel 2

Grundlagen und Technologien

2.1. Virtuelle Realität

Die *virtuelle Realität* ist eine vom Computer erzeugte und simulierte 3D-Umgebung, die der Benutzer betritt und mit der er interagieren kann [Brill 2009; Desai et al. 2014]. Diese Umgebung ermöglicht sowohl visuelle, akustische, als auch tastbare Wahrnehmungen des Benutzers, sodass dieser das Gefühl hat, in der Umgebung körperlich anwesend zu sein. Diese Auswirkung wird als *Immersion* bezeichnet [Brill 2009]. Resultierend kann jede physische Anwesenheit, sei es in realen oder imaginären Welten, simuliert werden [Pérez Fernández und Alonso 2015]. Die Interaktion mit der virtuellen Umgebung geschieht mithilfe technischer Hilfsmittel, sowie entsprechender Software, wodurch der Benutzer wie in seiner gewohnten, realen Umgebung handeln und sich bewegen kann [Brill 2009].

2.2. Erzeugung virtueller Realität

Ein Computer erschafft und aktualisiert die virtuelle 3D-Umgebung und stellt diese mithilfe einer VR Software stereoskopisch dar [Pérez Fernández und Alonso 2015]. Hierbei kann es sich beispielsweise um eine WebVR[1] handeln, die den Zugang zum Head-Mounted Display (HMD) im Browser ermöglicht. Damit der Benutzer dort dreidimensional sehen kann, erzeugt der Computer hierbei zwei stereoskopische Bilder der Umgebung, eins für das linke und eins für das rechte Auge [Crespo et al. 2015]. Die zwei Bilder werden dann auf dem HMD, das der Nutzer trägt, abgebildet [Desai et al. 2014]. Das HMD erkennt Kopfbewegungen und Positionsveränderungen beim Bewegen im Raum. Durch das HMD und die verwendete Hardware zur Erkennung der restlichen Körperbewegungen des Benutzers, werden Bewegungsinformationen an den Computer weitergegeben. Dieser verarbeitet diese Informationen wiederum mit der VR Software und aktualisiert die virtuelle Umgebung [Pérez Fernández und Alonso 2015].

2.2.1. Verwendete Hardware

Zum Betreten, Visualisieren und Aktualisieren der virtuellen Umgebung verwenden wir ein *HTC VIVE System*, welches aus aus einen Computer, einer *HTC VIVE* (HMD), zwei

[1] `https://webvr.info`

Controllern und zwei Basisstationen besteht [Zhou et al. 2016]. Der Computer erzeugt die Bilder der virtuellen Umgebung für das linke, sowie rechte Auge und sendet diese via Kabel an die *HTC VIVE*. Dieses stellt die Bilder auf dem internen Bildschirm dar, sodass der Benutzer die virtuelle Umgebung wahrnehmen kann. Die Interaktion mit der virtuellen Umgebung wird durch die kabellosen Controller und deren Knöpfe ermöglicht. Die Basisstationen tasten mithilfe von Laserstrahlen einen abgesteckten Bereich ab und können dort die Position, Neigung, sowie Richtung der *HTC VIVE* und der Controller ermitteln. Dieses Verfahren wird *Lighthouse* bezeichnet. Die Basisstationen und die Controller übermitteln die gewonnen Daten kabellos an die *HTC VIVE* und diese via Kabel an den Computer. Die *HTC VIVE* ermöglicht die Nutzung des Raumkonzepts der virtuellen Realität, da die Navigation durch Gehen, und Bewegen des Kopfes möglich ist.

2.2.2. Verwendete Software

Steam VR[2] ist die zugrundeliegende Software des *HTC VIVE Systems* [Dempsey 2016]. Sie verwaltet die gesamte Hardware und stellt über eine GUI sicher, dass alle Hardwarekomponenten verbunden und einsatzbereit sind. *Steam VR* ist eine Schnittstelle zwischen Computer und *HTC VIVE System*, welche Konfigurationen an der verwendeten Hardware ermöglicht. Des Weiteren verarbeitet die Software, die durch die Basisstationen, Controller und *HTC VIVE* gesammelten Daten und berechnet die neuen Daten für die neuen Bilder. Der Computer erzeugt aus diesen Daten die aktualisierten stereoskopischen Bilder der virtuellen Umgebung.

2.3. ExplorViz

ExplorViz ist eine Webapplikation, die zur Überwachung und Visualisierung großer Softwarelandschaften geeignet ist. Diese bestehen oft aus vielen Servern, Anwendungen, Paketen, Komponenten und Klassen. *ExplorViz* sammelt im Rahmen der Überwachung Informationen über die Kommunikation zwischen den Servern, Anwendungen, Paketen, Komponenten und Klassen. Diese Informationen beinhalten beispielsweise die Anzahl der Aufrufe einer Methode oder der Instanzen einer Klasse. Die Live-Visualisierung stellt die gesammelten Daten dar und kann jederzeit pausiert werden [Fittkau et al. 2017; 2015, b; Krause 2015]. *ExplorViz* verfügt über eine zwei- und eine dreidimensionale Visualisierung Die zweidimensionale Visualisierung, die Landschaftsansicht beinhaltet die Kommunikation zwischen den Servern sowie Anwendungen und ist in Abbildung 2.1 dargestellt. Die oberste Stufe der Hierarchie stellen die Systeme dar, die als graue Boxen dargestellt sind. Sie können durch einen Doppelklick geschlossen oder geöffnet werden, um den Inhalt ein- oder auszublenden. Ein System beinhalt einen Server (hellgrüne Box) oder einen Zusammenschluss mehrerer Server, welcher als dunkelgrüne Box dargestellt ist. Die dunkelgrünen Boxen können, durch einen Doppelklick, einen oder alle enthaltenen Server

[2] http://store.steampowered.com/steamvr?l=german

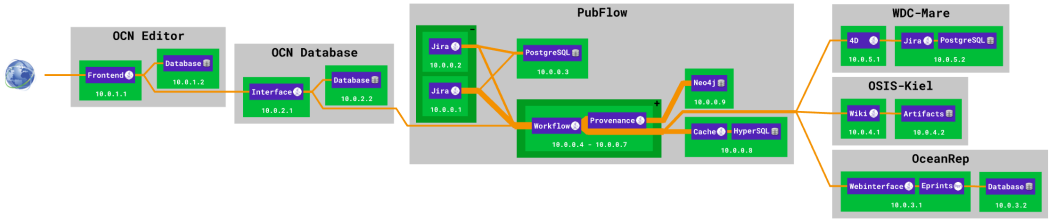

Abbildung 2.1. Darstellung der Landschaft in der Landschaftsansicht von *ExplorViz*

anzeigen. Die orangenen Linien zwischen den Komponenten stellt die Kommunikation dar, links die eingehende und rechts die ausgehende Kommunikation. Die Breite dieser wird durch die Anzahl der Aufrufe skaliert. Auf jedem Server kann eine oder mehrere Anwendungen laufen, die als blaue Boxen dargestellt sind.

Eine solche Anwendung kann durch einen Doppelklick in der Applikationsansicht dreidimensional dargestellt werden. In Abbildung 2.2 ist zu erkennen, dass diese aus grünen sowie blauen Boxen und orangenen Linien besteht. Die grünen Boxen stellen die

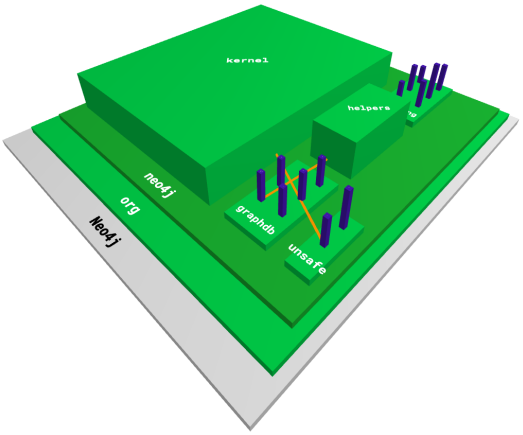

Abbildung 2.2. Darstellung einer Applikation in der Applikationsansicht von *ExplorViz*

Pakete und die blauen Boxen die Klassen dar. Die Boxen der Klassen sind unterschiedlich hoch, da sich deren Höhe an der Anzahl der aktiven Instanzen der Klasse bemisst [Krause 2015]. Ein geschlossenes Paket wird als Box dargestellt, deren Inhalt ausgeblendet ist. Ein geöffnetes Paket wird als grüne Fläche abgebildet, auf welcher der Inhalt des Pakets dargestellt wird. Die unterschiedlichen Grüntöne dienen lediglich dazu die verschiedenen hierarchischen Stufe zu verdeutlichen. Wie in der Landschaftsansicht wird durch orangene Linien die Kommunikation innerhalb der Anwendung dargestellt.

2.4. Programmiersprachen und Entwicklertools

Die folgende Auflistung erläutert kurz einige Programmiersprachen, -Dialekte und Tools, die für das Projekt verwendet werden.

- Das *Document Object Model (DOM)* wird vom Browser, aus einem HTML-Dokument erzeugt. Das DOM ist eine lokale hierarchische Baumstruktur, welche der Struktur des HTML-Dokuments entspricht. Jedes Element des Baums ist mit einer *ID* versehen und durch diese zugreifbar.
- *jQuery*[3] ist eine JavaScript Bibliothek, welche die DOM Manipulation und Navigation erleichtert.
- *JavaScript* ist eine ereignisgesteuerte Programmiersprache, sodass Komponenten durch Ereignisse aufeinander reagieren. JavaScript wird für Webanwendungen auf Benutzer- und Serverseite verwendet wird und erfreut sich großer Beliebtheit, da Programme nicht mit Typen annotiert werden müssen und somit schneller programmiert werden kann [Gallaba et al. 2015; Pradel et al. 2015]. Ein JavaScript wird vom Browser im HTML-Dokument aufgerufen und ermöglicht damit den direkten Zugriff auf das DOM, sodass ohne Umwege auf Benutzereingaben reagiert werden kann. Der Zugriff auf das DOM bietet die Möglichkeit eine angezeigte HTML-Seite, durch ein JavaScript, lokal zu veränderen, sodass das Dokument auf dem Server unberührt bleibt [Kyriakou et al. 2015]. *JavaScript* arbeitet mit Referenzen, sodass sich lokale Änderungen global auswirken. Ein als Parameter übergebenes Objekt wird somit in jeder Klasse und Methode manipuliert, in der dieses Objekt verwendet wird. Dies kann verhindert werden, indem die Methode *clone()* beim Übergeben auf das Objekt angewendet wird.

  JavaScript spielt für die alte und neue Architektur von *ExplorViz* eine wichtige Rolle, da *ExplorViz* vollständig und *ExplorViz legacy* partiell in JavaScript implementiert ist.
- *WebVR*[4] ist eine JavaScript API, welche den Zugriff auf VR Hardware, wie das HMD oder die Controller, im Browser ermöglicht. Hierdurch wird die Nutzung von virtueller Realität in webbasierten Anwendungen möglich.
- *WebGL*[5] ist eine JavaScript API, die das Rendern von interaktiven 2D- und 3D-Grafiken innerhalb eines kompatiblen Browsers ermöglicht. Der Vorteil von *WebGL* besteht darin, dass keine Plugins für die Umsetzung notwendig sind [Li et al. 2016].
- Das *Google Web Toolkit (GWT)*[6] ist ein javabasiertes Toolkit zum Erstellen dynamischer Webanwendungen [Larson et al. 2014]. Es kompiliert Java zu JavaScript, sodass die Webanwendung in Java geschrieben und in JavaScript ausgeführt werden kann [Larson

[3] https://jquery.com
[4] https://webvr.info
[5] https://developer.mozilla.org/de/docs/Web/API/WebGL_API
[6] http://www.gwtproject.org

et al. 2014]. Der zugrundeliegende Gedanke besteht darin, dass die meisten Entwickler mehr mit Java und seinem Objektmodell, als mit JavaScript vertraut sind [Larson et al. 2014].

- Das *JavaScript Native Interface (JSNI)*[7] wird vom *GWT* implementiert. Die JSNI-Methoden werden mit *native* deklariert und bieten die Möglichkeit JavaScripts direkt in den Java-Code einzubinden.

  JavaScript Native Interface (JSNI) spielt für *ExplorViz legacy* eine wichtige Rolle, da dort JavaScripts in den Java-Code eingebunden werden.

- *Ember*[8] ist eine clientseitiges JavaScript-Framework zum Erstellen einseitiger Webanwendungen. Inhalte können auf verschiedene Routen verteilt werden, welche durch die URL adressiert werden. Diese Routen wirken wie eigenständige HTML-Dokumente, jedoch werden die verschiedenen Inhalte lediglich in die aktuelle Seite geladen. Eine derartige Webanwendung besteht im Kern aus Templates und Routen. Die Templates geben die Darstellung und die Routen das Verhalten eines bestimmten Inhalts vor. Die Änderung eines Template oder einer Route hat eine unverzügliche Aktualisierung des Browsers zur Folge. Ein Ember Projekt hat die Softwarestruktur des *Model View Controller (MVC)* und ist entsprechend gekapselt. Eine Ember Komponente stellt in diesem Zusammenhang das *Model*, die Templates die *View* und die Ember Controller die *Controller* dieser Struktur dar. Methoden können in sogenannten *Ember Utils* ausgelagert werden und von mehreren Ember Komponenten genutzt werden. Dafür erzeugt jede dieser Ember Komponenten eine Instanz des *Ember Utils* und kann auf die hinterlegten Funktionen zugreifen. Diese *Ember Utils* können mehrfach instanziiert werden und sind daher zustandslos. Ember Services ähneln den *Ember Utils*, jedoch stellen diese ein *Singleton* dar und können deshalb einen Zustand haben. Diese Ember Services werden für geteilte Daten, wie das Datenmodel, verwendet.

  Ember wird verwendet, um *ExplorViz* als einseitige Webanwendung zu gestalten.

- *Handlebars*[9] ist eine Template-Engine für JavaScript, mit der das Aussehen des Templates gestaltet werden kann.

  Handlebars dient unter anderem dazu das Aussehen von *ExplorViz*, in Form von Templates, zu gestalten. Des Weiteren werden diese Templates verwendet, um bestimmte Aktionen oder Parameter weiterzuleiten.

- *Node.js*[10] ist eine Plattform für serverseitige Netzwerkanwendungen. Sie ermöglicht die Benutzung von JavaScript auf Serverseite, indem diese auch außerhalb eines Browsers ausgeführt werden können [Kyriakou et al. 2015; Alimadadi et al. 2016].

[7] http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSNI.html
[8] https://www.emberjs.com
[9] http://handlebarsjs.com
[10] https://nodejs.org/en/

Diese Eigenschaft wird für die Realisierung des *Backends* verwendet. Des Weiteren wird *Node.js* als Grundlage für den Paketmanager *NPM* benötigt, welcher für das *Frontend* verwendet wird.

- *NPM*[11] ist ein Paketmanager für *Node.js*, durch den alle Paketabhängigkeiten automatisch hergestellt werden [Kyriakou et al. 2015] und der für das *Frontend* verwendet wird. Des Weiteren können hiermit externe Module in Ember eingebunden werden, indem sie durch den Paketmanager installiert und in den Ordner *node_modules* des Projekts abgelegt werden.

  Es gibt eine weitere die Möglichkeit externe Bibliotheken in das Ember Projekt einzubinden. Diese Bibliotheken werden dazu in den Ordner *vendor* des Ember Projekts kopiert und die Datei *ember-cli-build.js* um einen entsprechenden Eintrag erweitert. Hierdurch können externe Module in das Projekt eingebunden werden, die beispielsweise nicht mit dem Modulmanager *npm* installiert werden können oder mit diesem nicht korrekt funktionieren. Für die Nutzung der Inhalte dieser Module ist kein zusätzlicher Import notwendig, da die Funktionen, Konstruktoren oder Variablen einmalig in die Datei *.jshintrc* eingetragen werden und damit global bekannt sind.

2.5. Three.js

Three.js[12] ist eine JavaScript Library, die *WebGL* nutzt. Sie ermöglicht das Erstellen und Rendern von 2D- und 3D-Objekten direkt im Webbrowser. Zu Beginn wird eine *Scene* erstellt, die als *Container* fungiert und alle 2D- und 3D-Objekte enthält, die gerendert werden sollen. Im Anschluss wird die *Camera* definiert, die festlegt was visualisiert wird [Dey et al. 2016]. . Zum Rendern wird der *WebGLRenderer*[13] verwendet, der in einer Rendering-Schleife 60 mal pro Sekunde aufgerufen wird und damit das derzeitige Bild aktualisiert. Dieses wird dabei in ein sogenanntes *Canvas* gezeichnet, ein HTML-Element das im Browser angezeigt wird. Die 2D- und 3D-Objekte setzen sich aus einer Geometrie und einem Material zusammen. Diese werden ebenfalls von *Three.js* implementiert und in Abschnitt 2.5.3 und Abschnitt 2.5.2 genauer beschrieben. *Three.js* stellt eine eigene API namens *WebVR.js*[14] bereit, welche die Nutzung der allgemeinen *WebVR.js*[15] API erleichtert. Des Weiteren stellt *Three.js* eine API namens *THREE.ViveController.js*[16] für die Realisierung und Nutzung der *HTC VIVE Controller* zur Verfügung. Diese werden wir im Verlauf der Arbeit mit einem Strahl ausstatten, der im Allgemeinen als *Ray* oder *Raycaster* bezeichnet wird.

[11] https://www.npmjs.com
[12] https://threejs.org
[13] https://threejs.org/docs/#api/renderers/WebGLRenderer
[14] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/WebVR.js
[15] https://webvr.info
[16] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/ViveController.js

Das Erstellen und Rendern der Grafiken in der neuen Architektur von *ExplorViz* wird durch *Three.js* bewerkstelligt.

2.5.1. 2D- und 3D-Objekte

In der neuen Architektur von *ExplorViz* wird zur Darstellung der 2D- und 3D-Objekte die Klasse *THREE.Mesh*[17] verwendet. Diese bildet die Objekte durch Aneinanderreihung vieler kleiner Dreiecke ab. Die 2D-Objekte werden dabei direkt der *Scene* hinzugefügt. Die 3D-Objekte der Applikation werden in einer Art Gruppe, dem *THREE.Object3D*[18] gesammelt und erst dann der *Scene* hinzugefügt. Diese Klasse stellt Methoden zur Manipulation des gesamten Inhalts der Gruppe bereit, sodass nicht jedes einzeln manipuliert werden muss. Unter anderem kann mit diesen Methoden die Position, Rotation, oder Skalierung der gesamten Gruppe verändert werden.

2.5.2. Aussehen von Objekten

Das Material[19] eines Objekts legt dessen Aussehen fest. *Three.js* stellt viele verschiedene Klassen zur Erzeugung Unterschiedlicher Materialien bereit. Abhängig von der Klasse können dort beispielsweise die Farbe, Schatten, Lichteinflüsse oder Bilder als Textur festgelegt werden. Bevor das Material eines Objektes überschrieben oder das Objekt selber gelöscht wird, muss das bestehende Material korrekt entfernt werden. Dies wird durch das Anwenden der Methode *dispose()* erreicht. Ansonsten kann es dazu kommen, dass belegter Speicher der Grafikkarte nicht richtig freigegeben wird und diese damit mehr Aufwand hat.

2.5.3. Geometrie von Objekten

Die geometrischen Eigenschaften eines Objekts werden durch die Geometrie[20] realisiert. *Three.js* stellt viele verschiedene Klassen zur Erzeugung Unterschiedlicher Geometrien bereit. Abhängig von der Klasse können dort beispielsweise die Position, Rotation oder Skalierung festgelegt werden. Eine weiteres wichtiges Atribut, das alle Geometrien bereitstellen, ist die *boundingBox*. Sie wird durch die Methode *computeBoundingBox()* erzeugt und beschreibt die die tatsächlichen Ausmaße eines Objekts im Raum. Durch sie sind wichtige Informationen wie Länge, Breite, Höhe oder Zentrum eines Objektes zugänglich. Für die Geoemetrie sollte der gleiche Löschvorgang, wie in Abschnitt 2.5.2 beschrieben, eingehalten werden.

[17] https://threejs.org/docs/index.html#api/objects/Mesh
[18] https://threejs.org/docs/index.html#api/core/Object3D
[19] https://threejs.org/docs/#api/materials/Material
[20] https://threejs.org/docs/#api/core/Geometry

Kapitel 3

Identifikation verwandter Arbeiten

In diesem Kapitel werden verwandte Arbeiten vorgestellt und auf Verwendbarkeit für diese Arbeit untersucht. Im Anschluss werden derzeit bestehende Herausforderungen geschildert und mögliche Lösungsansätze für diese dargelegt.

3.1. Vorstellung verwandter Arbeiten

Für uns relevante Arbeiten gliedern sich in die Kategorien Raumkonzept, Multiuser-Lösung sowie verwandte Monitoring Anwendungen und werden in den entsprechenden Unterabschnitten vorgestellt.

3.1.1. Raumkonzept

Die Arbeit von [Li et al. 2017] präsentiert eine Lösung zur Umsetzung eines virtuellen Sicherheitstrainings für Erdbeben-Szenarien. Dieses Sicherheitstraining soll dem Benutzer das korrekte Verhalten während eines Erdbebens vermitteln, ohne diesen dabei in Gefahr zu bringen. Die Umsetzung in einem virtuellen Szenario bietet den Vorteil, dass es besonders realistisch ist, da nicht auf herabstürzende Trümmer oder Erderschütterungen verzichtet werden muss. Für die technische Realisierung wird auf ein *HTC VIVE System* zurückgegriffen. Der Benutzer betritt das Erdbeben-Szenario durch die *HTC VIVE* und kann durch die Controller mit der virtuellen Umgebung interagieren. Die Position des Kopfes sowie der Controller werden mithilfe der Basisstationen ermittelt und dienen der Navigation und Fortbewegung in der fiktiven Umgebung. Während eines Erdbebens ist die Körperhaltung einer Person sehr wichtig, da sie das Verletzungsrisiko beeinflusst. Zu deren Erkennung wird ein spezieller Algorithmus angewandt, der die Körperhaltung eines Benutzers anhand der Position des Kopfes und der beiden Controller berechnet.

[Schepper et al. 2015] haben ein Spiel entwickelt, in dem zwei Spieler gegeneinander in einer virtuellen Umgebung spielen können. Ziel ist es dabei innerhalb eines Labyrinths die Flagge für sich zu erobern. Die virtuelle Umgebung wird durch eine *Oculus Rift* betreten und ermöglicht es dem Spieler sich dort umzusehen, da diese durch die Sensoren auf dem HMD aktualisiert wird. Beide Spieler werden dort als Avatar dargestellt und können durch eine Maus mit der Umwelt interagieren. Diese ermöglicht das Schießen auf den Gegner und das Rotieren des Körpers des Avatars. Die Fortbewegung im Spiel wird durch Gehen

und Bewegen in der realen Welt verwirklicht. Hierzu wird ein festgelegter Bereich von vier Kameras abgesteckt, in welchem sich die Spieler bewegen sollen. Mithilfe eines speziellen Algorithmus wird aus den Daten der vier Kameras die Position des Spielers berechnet. Aufgrund der Gößе des Labyrinths muss eine entsprechend große Fläche in der realen Welt zur Verfügung stehen und überwacht werden. In diesem Fall werden, wie in der rechten Darstellung von Abbildung 3.1 zu sehen, die Kameras auf einer Wiese installiert. Des Weiteren wird die zugrunde liegende Hardware, wie in der linken Darstellung von Abbildung 3.1 abgebildet, weitestgehend von Kabeln befreit. Der Benutzer wird dazu mit

Abbildung 3.1. Darstellung der kabellosen Umsetzung der Hardware links und Abtasten des realen Bereichs durch Kameras rechts [Schepper et al. 2015]

einem Laptop auf dem Rücken ausgestattet und dieses mit der *Oculus Rift* verbunden. Hierdurch wird neben einer größeren Bewegungsfreiheit ebenfalls eine Nutzbarkeit von großen realen Flächen zur Fortbewegung und Navigation ermöglicht. Das Raumkonzept ermöglicht die Fortbewegung in der virtuellen Welt durch Gehen in der realen Welt. Eine Herausforderung hierbei besteht darin, dass die Wände in der virtuellen Umgebung solide sind und der Benutzer beim Versuch hindurchzugehen auf der selben Position verbleibt. In der realen Welt bewegt er sich jedoch weiter und läuft Gefahr den vorher definierten Bereich zu verlassen.

Die Arbeit von [Roth et al. 2016] beschäftig sich mit der Nutzung von virtueller Realität im Hinblick auf soziale Interaktion. Hierbei werden die Unterschiede zwischen der virtuellen und der realen sozialen Interaktion herausgestellt. Der Nutzer wird in der virtuellen Umgebung als Avatar dargestellt und betritt diese durch eine *Oculus Rift*. Die Darstellung

des Avatars geschieht durch *Unity3D*[1]. Dem Benutzer ist es möglich sich mithilfe der HMD umzuschauen sowie mit der Umgebung durch Körperbewegungen zu interagieren. Die Körperbewegungen werden dabei mit *OptiTrack*[2] festgestellt, ein System das sowohl die Position, Neigung und Richtung der HMD als auch des restlichen Körpers ermitteln kann. Der Nutzer kann beispielsweise in der virtuellen Umgebung einen Ball mit dem Fuß zum Rollen bringen oder sich durch Gehen fortbewegen. Des Weiteren kann die Anwendung von zwei Personen genutzt werden, dies wird in Abschnitt 3.1.2 vorgestellt.

3.1.2. *Multiuser*-Lösung

Das in Abschnitt 3.1.1 erwähnte Spiel von [Schepper et al. 2015] bietet zwei Spielern die Möglichkeit eine gemeinsame virtuelle Umgebung zu betreten und aufeinander zu reagieren. Dies ist beispielhaft im linken Teil von Abbildung 3.2 dargestellt. Für die Erstellung

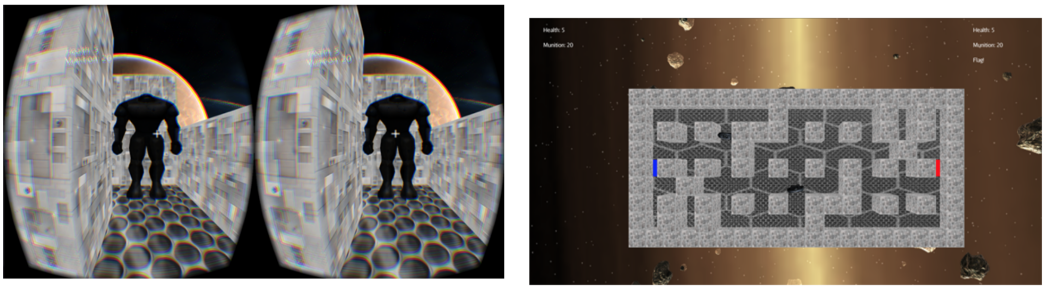

Abbildung 3.2. Darstellung der Spielersicht links und der Beobachtersicht rechts [Schepper et al. 2015]

des Spiels wurde eine Java-basierte Engine names *JMonkeyEngine*[3] verwendet. Zur Realisierung des Spiels wurde ein Sternnetzwerk verwendet, sodass ein Server die Daten global aktualisiert und an die *Clients* sendet. Es ist vorgesehen, dass sich zwei *Clients* als Spieler und ein weiterer *Client* als *Spectator* (Beobachter) mit dem Server verbinden können. Wie in der rechten Darstellung von Abbildung 3.2 zu sehen, ist es dem Beobachter möglich das Labyrinth von oben zu betrachten und den Spielverlauf zu verfolgen. Mithilfe einer API namens *SpiderMonkey* wird die Kommunikation zwischen den einzelnen Komponenten realisiert, da es das Senden von UDP- und TCP-Paketen unterstützt und je nach Präferenz besser mit verloren Paketen umgegangen werden kann. Der Spieler sendet seine Position als UPD-Paket - da hier die Geschwindigkeit der Datenübertragung wichtig für das Spielempfinden ist - an den Server und dieser aktualisiert die Position des Spielers global. Alle weiteren Daten werden mit einem TCP-Paket an den Server gesendet, da hier die korrekte Übermittlung vorrangig ist.

[1] `https://unity3d.com/de`
[2] `http://optitrack.com/motion-capture-virtual-reality/`
[3] `http://jmonkeyengine.org`

Die in Abschnitt 3.1.1 beschriebene Anwendung von [Roth et al. 2016] bietet die Möglichkeit einen virtuellen Raum zu zweit zu betreten und miteinander zu interagieren. Die Nutzer können dabei, wie in Abbildung 3.3 links zu sehen, gemeinsam Ball spielen. Weiterhin sind sie in der Lage, wie in Abbildung 3.3 rechts dargestellt, einander die Hand zu geben oder anders zu gestikulieren.

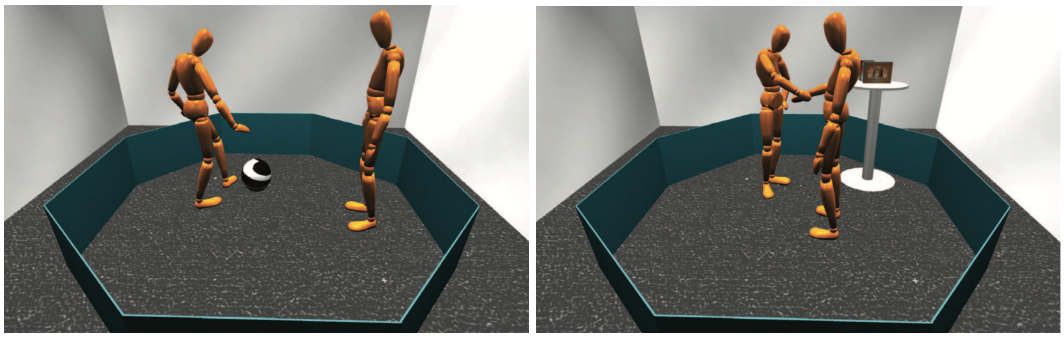

Abbildung 3.3. Darstellung eines virtuellen Ballspiels zweier Personen links und eines virtuellen Handschlags rechts [Roth et al. 2016]

[Sharma et al. 2014] haben eine Anwendung entwickelt, die das Proben einer U-Bahn Evakuierung in einer virtuellen Umgebung ermöglicht. Der Benutzer betritt das virtuelle Szenario durch eine *Cloud* und kann sich dort mithilfe einer *Oculus Rift* umschauen. Die Nutzung des Sensors der HMD zur Erkennung der Rotation des Kopfes ermöglicht dem Nutzer eine 360 Grad Perspektive der Umgebung. Die Interaktion und Fortbewegung geschieht durch eine Tastatur oder einen Joystick. Alternativ kann der Benutzer einen Bildschirm nutzen, um das virtuelle Szenario zu betreten. Die Interaktion, Fortbewegung und das Umschauen wird dann komplett von der Tastatur oder dem Joystick übernommen. Der Nutzer wird in der virtuellen Umgebung als Avatar dargestellt und Veränderungen der Position entsprechend aktualisiert. Die Umsetzung der Avatars wird ebenfalls mit *Unity3D* gestaltet und ist in Abbildung 3.4 dargestellt. Es können mehrere Benutzer an der Evakuierung teilnehmen, um eine realistische Ausgangssituation zu schaffen. Des Weiteren können computergesteuerte Avatars das virtuelle Szenario vervollständigen. Die Mehrfachnutzung dieser Anwendung wird durch *Photon*[4] - eine Engine für Multiplayer-Spiele - auf dem Cloud Server realisiert. Führt der Benutzer eine Aktion aus, wie beispielsweise eine Vorwärtsbewegung, wird diese an den *Cloud Server* geschickt. Dieser verarbeitet die Aktion und berechnet den neuen Zustand des virtuellen Szenarios. Die *Clients* werden vom Server darüber informiert, dass sich der Zustand aktualisiert hat, woraufhin dieser bei jedem Benutzer neu gerendert wird.

SIDEKIQ[5] ist eine von [Huang et al. 2015] entwickelte Software, die das Football Training

[4] https://www.photonengine.com/en-US/Photon

[5] https://eonsportsvr.com/product/sidekiq-7-on-7-firstdown-playbook/

Abbildung 3.4. Darstellung der Avatars und der Ubahnumgebung [Sharma et al. 2014]

erleichtern soll. Der Trainer kann dazu virtuelle Spielszenarien erstellen, welche Spieler und Spielpositionen enthalten. Die zu trainierende Person kann dieses Szenario durch einen Bildschirm oder eine HMD betreten und die Spielsituation aus einer Vogel-, Third-Person- oder direkt aus der Helm-Perspektive jedes einzelnen Spielers auf dem Feld betrachten. Der Ablauf des Trainings gestaltet sich so, dass die zu trainierende Person in einem gesetzten Zeitfenster entscheiden muss wohin der Ball gepasst wird. In Abbildung 3.5 ist zu sehen, dass der Trainer dabei hinter der Person steht und das Szenario mit einem Controller steuert. Dabei kann er die Perspektive der Person auf einer Leinwand verfolgen.

Abbildung 3.5. Der virtuelle Spielverlauf aus Sicht des Footballtrainers [Huang et al. 2015]

3.1.3. Verwandte Monitoring Anwendungen

ExplorViz legacy ist ein Softwareüberwachungs- und Visualsierungstool, welches die alte, monolithische Architektur von *ExplorViz* bezeichnet. [Fittkau et al. 2017; 2015, b]. Sie verfügt weitestgehend über die gleichen Hauptfunktionaliäten wie die neue Architektur. *ExplorViz legacy* verfügt bereits über einen Modus, der Visualisierungen in virtueller Realität

ermöglicht. Diese Visualisierung beschränkt sich auf die Anwendungsansicht und nutzt nicht das Raumkonzept der virtuellen Realität. Die virtuelle Umgebung wird mit einer *Oculus Rift DK1* visualisiert und mithilfe des integrierten Orientierungssystems, durch die Ausrichtung des Kopfes, aktualisiert. Damit auf die *Oculus Rift* im Webbrowser zugegriffen werden kann, wird die JavaScript API *WebVR*[6] verwendet. Die API ist bereits in den Entwicklerversionen von *Google Chrome* und *Mozilla Firefox* enthalten. Mithilfe dieser API ist die Verwendung neuerer Versionen der *Oculus Rift* oder anderer HMDs durch kleinere Anpassungen möglich [Fittkau et al. 2015a]. Im Rahmen dieser dynamischen Anpassungsmöglichkeit wird derzeit eine *HTC VIVE* für die Visualisierung verwendet. Des Weiteren werden die Controller der *HTC VIVE* zur Interaktion mit virtuellen Umgebung verwendet. Sie ermöglichen das Zoomen, Verschieben, Rotieren, Markieren und Auswählen von Objekten. Alternativ können für das Zoomen, Verschieben, Rotieren, Auswählen und Zurücksetzen der Perspektive Handgesten verwendet werden. Zur Erkennung dieser Gesten wird der *Leap Motion Controller*[7] verwendet, ein Gerät das auf einer Oberfläche platziert wird und den Bereich über ihm abtastet. Dabei können Handbewegungen und die Hand selbst sehr genau erfasst werden [Potter et al. 2013]. Des Weiteren ist es möglich sich Felder mit Informationen in der virtuellen Umgebung darstellen zu lassen.

3.2. Verwendbare Ansätze

Die Arbeit von [Li et al. 2017] greift auf ein *HTC VIVE System* zurück, um das Raumkonzept der virtuellen Realität zu nutzen. Durch die beiden Basisstationen steht ein relativ großer Bereich zur Verfügung, der die Erfassung der Position des Benutzers erlaub. Die *HTC VIVE* ist damit für die Umsetzung dieser Arbeit geeignet. Die *Oculus Rift DK2* verfügt als Nachfolger der *Oculus Rift DK1* über ein Head-tracking-System, das es ermöglicht die räumliche Position der HMD zu erfassen [Desai et al. 2014]. Eine externe Kamera wird dazu im Raum aufgestellt und erfasst die Infrarot-LED, die sich auf dem HMD befindet [Li et al. 2014]. Das Feld, in dem die Position des HMD erfasst werden kann, wird vom Erfassungsbereich dieser Kamera eingeschränkt, sodass nur kleinere Bereiche in der virtuellen Umgebung zur Verfügung stehen. Aus diesem Grund verwenden die Schöpfer verwandter Arbeiten, wie [Schepper et al. 2015; Roth et al. 2016], zwar eine *Oculus Rift*, nutzen sie jedoch nicht zur Verwirklichung des Raumkonzepts. Die *Oculus Rift Consumer Version 1* ist der Nachfolger der *Oculus Rift DK2*, dessen Positionserfassung über eine verbesserte Infrarotkamera erfolgt. Sofern dieser Nachfolger dadurch den erfassten und nutzbaren Bereich vergrößert, könnte er eine Alternative zur *HTC VIVE* darstellen.

Für die Modellierung der Avatars haben [Sharma et al. 2014; Roth et al. 2016] auf die Multiplattform-Spiel-Engine *Unity3D* zurückgegriffen. Diese Engine könnte für die Realisierung der Avatars in einer *Multiuser*-Lösung von *ExplorViz* geeignet sein, sofern sie

[6] https://webvr.info

[7] https://www.leapmotion.com/

in *Ember* eingebunden werden kann. [Sharma et al. 2014; Schepper et al. 2015] haben für die Realisierung des *Multiuser*-Ansatzes ein Sternnetzwerk verwendet. Sobald ein Nutzer eine Aktion, wie beispielsweise eine Positionsveränderung ausübt, wird diese an den Server übermittelt und dort die neuen Daten des virtuellen Szenarios inklusive der neuen Position des Benutzers berechnet. Im Anschluss wird diese Änderung bei allen *Clients* bekannt gemacht, sodass deren Darstellung neu gerendert wird. In Abbildung 3.6 ist die Kommunikation innhalb eines Sternnetzwerks dargestellt, welches sich für die Realisierung der kollaborativen Nutzbarkeit von *ExplorViz* im VR-Modus eignen könnte.

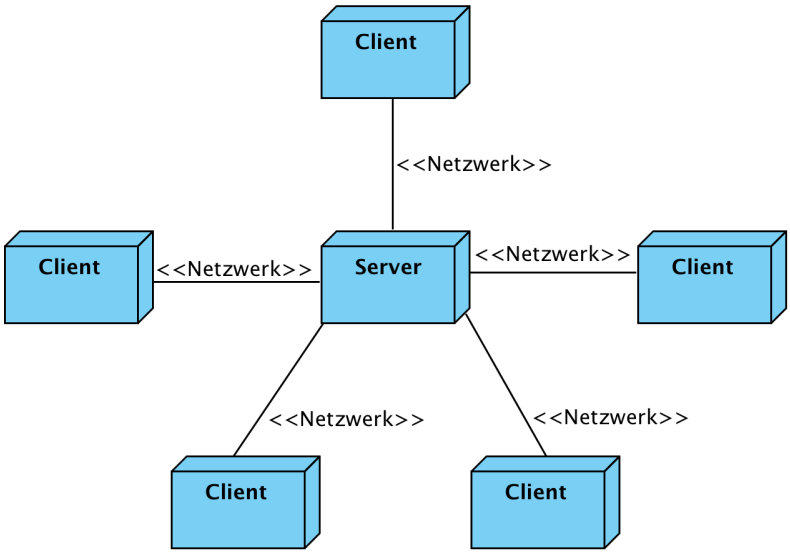

Abbildung 3.6. Die Kommunikation innerhalb eines Sternnetzwerks

Aufgrund der dynamischen Anpassungsfähigkeit an das HMD, kann die Umsetzung der Visualisierung aus Abschnitt 3.1.3 für die Integration in die neue Architektur verwendet werden. Die Realisierung ist teilweise in JavaScript implementiert, sodass wir in Abschnitt 4.1 verwendbare Programmabschnitte identifizieren. Die Verwirklichung der Interaktion durch die Controller eignet ebenfalls für die Übernahme in die neue Architektur. Die Umsetzung der *HTC VIVE Controller* geschieht durch eine von *Three.js* bereitgestellte API. In Listing 3.1 sind Teile einer möglichen Verwendung der Controller dargelegt, die aus einem von *Three.js* veröffentlichtem Beispiel[8] stammen. In Zeile 1 und 5 werden die Controller mithilfe der API erzeugt. Als nächstes werden sie in Zeile 2 und 6 an die Position im virtuellen Raum angepasst. Hierzu werden die Controller relativ zum virtuellen Fußboden ausgerichtet, auf dem der Benutzer steht. Abschließend werden sie in Zeile 3 und 7 der Szene hinzugefügt, damit sie in der virtuellen Umgebung sichtbar sind.

[8] https://github.com/mrdoob/three.js/blob/master/examples/webvr_vive.html

```
controller1 = new THREE.ViveController( 0 );
controller1.standingMatrix = renderer.vr.getStandingMatrix();
scene.add( controller1 );

controller2 = new THREE.ViveController( 1 );
controller2.standingMatrix = renderer.vr.getStandingMatrix();
scene.add( controller2 );
```

Listing 3.1. Verwendung der HTC VIVE Controller mithilfe von Three.js

Des Weiteren eignet sich die Realisierung der Darstellung von Informationen für die Übernahme in die neue Architektur. Die neue Visualisierung vereint und erweitert die Landschafts- und Applikationsansicht. Deshalb können Programmteile aus der bestehenden, verteilten Architektur von *ExplorViz* übernommen werden und als Grundlage für die neue Visualisierung dienen.

3.3. Bestehende Herausforderungen

Die Umsetzung des Raumkonzepts in der virtuellen Umgebung erfordert einen abgesteckten, realen Bereich, in dem die Überwachung der Position des Benutzers möglich ist. Sofern das virtuelle Szenario nicht exakt auf den realen Bereich abgebildet wird - da dieser beispielsweise größer ist - besteht die Gefahr den überwachten, realen Bereich zu verlassen [Schepper et al. 2015]. Eine Lösung kann darin bestehen, die Koordinaten der Grenzen des realen Bereichs mit der aktuellen Position des Benutzers zu vergleichen und bei Annäherung an diese, eine entsprechende Warnung auszugegeben. Als mögliche Realisierung dieser Warnung wäre ein virtueller Pfeil vorstellbar, der in die Richtung eines sicheren Bereichs zeigt.

Kapitel 4

Konzept/Entwurf

Der Entwurf des Plugins beinhaltet die Festlegung von Funktionalitäten, welche *ExplorViz* im Rahmen einer Visualisierung in virtueller Realität bereitstellen soll. Dazu wird in Abschnitt 4.1 die Übernahme bestehender Funktionalitäten aus verwandten Ansätzen geplant. Dieser Entwurf wird in Abschnitt 4.2 um zusätzliche Funktionalitäten erweitert, die neu entwickelt werden müssen.

4.1. Migrationskonzept

Die Analyse in Kapitel 3 hat ergeben, dass bestehende Programmteile aus *ExplorViz legacy* in die bestehende Architektur übernommen werden können. Im Rahmen dieser Übernahme werden wir im Folgenden die verwendbaren Programmabschnitte identifizieren.

Eine Visualisierung in virtueller Realität erfordert zwei verschiedene, angepasste Bilder des darzustellenden Szenarios. Die Grund dafür liegt darin, dass der Benutzer ein Bild für das linke und eins für das rechte Auge benötigt, um dreidimensional sehen zu können. Nachforschungen haben ergeben, dass diese Berechnungen bereits von der API namens *WebVR*[1] von *Three.js*[2] übernommen werden. Diese stellt ebenfalls einen Knopf zum Erzeugen und Betreten der virtuellen Realität bereit.

Die Umsetzung der Informationsanzeige im bestehenden VR-Modus eignet sich gut für die Übernahme in den neuen. Sie wird in der Funktion *handleHover* der Datei *WebVRJS.java* realisiert. Die Übernahme ist notwendig, da die Informationsanzeige der 2D-Visualisierung an das DOM angefügt wird, somit nicht im, sondern auf dem *Canvas* liegt und damit nicht in der virtuellen Umgebung dargestellt wird.

Aufgrund der Tatsache, dass die *Oculus Rift* nur einen kleinen Bereich erfassen und dieser damit genutzt werden kann, beschränken wir die Implementierung der Controller auf die der *HTC VIVE*. Für die Umsetzung der *HTC VIVE Controller* kann die API *THREE.ViveController.js*[3], wie in Abschnitt 3.2 erwähnt und in Listing 3.1 dargestellt, ver-

[1] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/WebVR.js

[2] https://threejs.org

[3] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/ViveController.js

wendet werden. Die Darstellung der Controller kann aus dem bestehenden VR-Modus übernommen werden. Dies geschieht in der Funktion *initControllers* der Datei *WebVRJS.java*. Mithilfe der Klasse *THREE.Raycaster*[4] und der bestehenden Implementierung des *Raycasters (utils/raycaster.js)* der neuen Architektur, kann dem *HTC VIVE Controller* ein unsichtbarer Strahl hinzugefügt werden, der parallel zu dessen Ausrichtung in den Raum geschickt wird. Dafür werden vorher die zu betrachtende Objekte festgelegt, davon alle getroffenen registriert und jenes zurückgeliefert und manipuliert, das sich am weitesten im Vordergrund befindet. Hierdurch wird sichergestellt, dass keine Objekte ausgewählt werden, die sich hinter anderen Objekten befinden und gegebenenfalls vom Benutzer nicht gesehen werden können. Dieses Prinzip ist skizzenhaft in Abbildung 4.1 dargestellt.

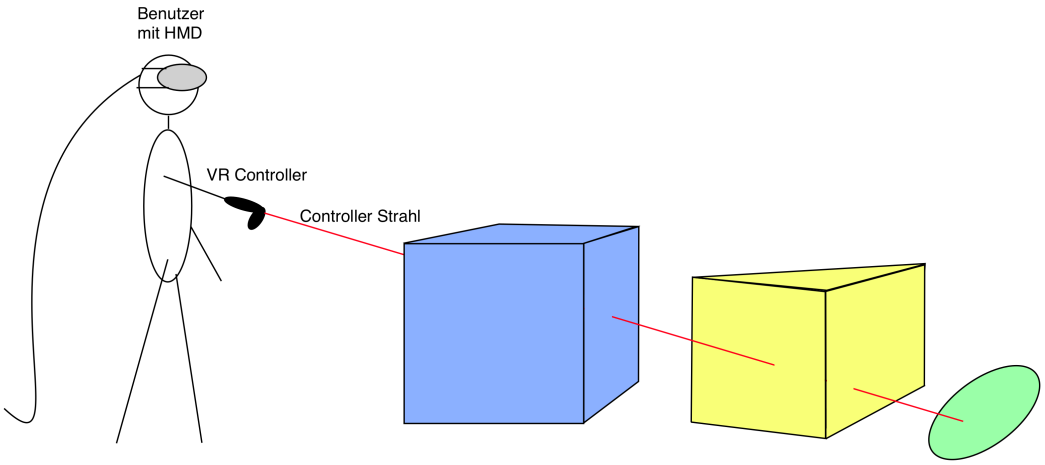

Abbildung 4.1. Registrierung von Umgebungsobjekten durch den Controllerstrahl

Die Darstellung eines Objekts im virtuellen Raum kann nur vom Raumkonzept profitieren, wenn dieses dreidimensional ist. *ExplorViz* ermöglicht bereits eine dreidimensionale Darstellung der Applikation in der Applikationsansicht, sodass die entsprechenden Programmabschnitte zur Erstellung und Gestaltung des Aussehens übernommen werden können. Diese finden sich in der Datei *components/application-rendering.js* und in den darin importierten *Ember Utils*.

[4] https://threejs.org/docs/#api/core/Raycaster

Des Weiteren haben wir in Abschnitt 3.2 dargelegt, dass sich die Verwendung eines Sternnetzwerks für die Umsetzung der kollaborativen Nutzbarkeit von *ExplorViz* eignet. Diese Netzwerkarchitektur sieht ausschließlich eine Kommunikation zwischen *Client* und *Server* vor. In Abbildung 4.2 ist eine mögliche Realisierung des Sternnetzwerks für *ExplorViz* dargestellt. Die Datenübertragung soll dabei mithilfe von TCP- und UDP-Paketen realisiert

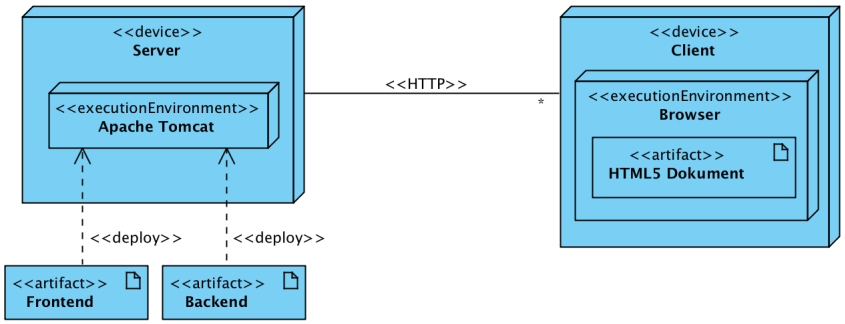

Abbildung 4.2. Die Kommunikation zwischen Clients und Server in *ExplorViz* mithilfe einer Sternarchitektur

werden. Die TCP-Pakete werden für die Übertragung der Interaktion, wie Auswählen, Rotieren, und Markieren der Umgebung verwendet, da hierbei die korrekte Übermittlung der Daten für ein positives Nutzererlebnis essentiell ist. Für die Übertragung der Visualisierungs- und Positionsdaten sollen UDP-Pakete verwendet werden, da hier die Geschwindigkeit der Datenübertragung einen größeren Einfluss auf das Nutzererlebnis hat. Die Bevorzugung der Pakete kann durch die API *SpiderMonkey* realisiert werden.

4.2. Erweiterungskonzept

Dieses Konzept beschreibt die ergänzenden Eigenschaften, die das Plugin zur Verfügung stellen muss, um das Erkunden einer Softwarelandschaft mit *ExplorViz* in virtueller Realität zu ermöglichen. Hierfür ist es erforderlich die 2D-Visualisierung der Landschaft um eine Dimension zu erweitern. Als Grundlage hierfür betrachten wir die bestehende Implementierung der aktuellen 2D-Visualisierung und erweitern diese um eine z-Komponente. *Systeme* und *Nodegroups* sollen im geschlossenen Zustand als dreidimensionale Boxen dargestellt werden, die mit dem Namen des *Systems* oder der *Nodegroup* beschriftet sind. Zur Entlastung der Grafikkarte und Steigerung der Bildrate sollen deren Inhalte dabei ausgeblendet werden. Ein geöffnetes *System* oder eine geöffnete *Nodegroup* soll als dreidimensionale Fläche dargestellt werden, auf der sich deren Inhalte befinden. Ein *Node* soll als dreidimensionale Box dargestellt werden, auf deren Oberseite sich eine farblich abgehobene Fläche befindet, welche den geschlossenen Zustand einer Applikation darstellt.

Wie in Abbildung 4.3 skizzenhaft dargestellt, soll der Benutzer einen Raum betreten, dessen Fußboden das gleiche Aussehen und die gleichen Maße wie der abgetastete reale Bereich hat. Der Benutzer soll sich nur auf dem virtuellen Fußboden bewegen, sodass der von den Basisistationen überwachte und begrenzte reale Bereich nicht verlassen wird. Auf diesem Fußboden wird die mit *ExplorViz* überwachte Software, in Form einer 3D-

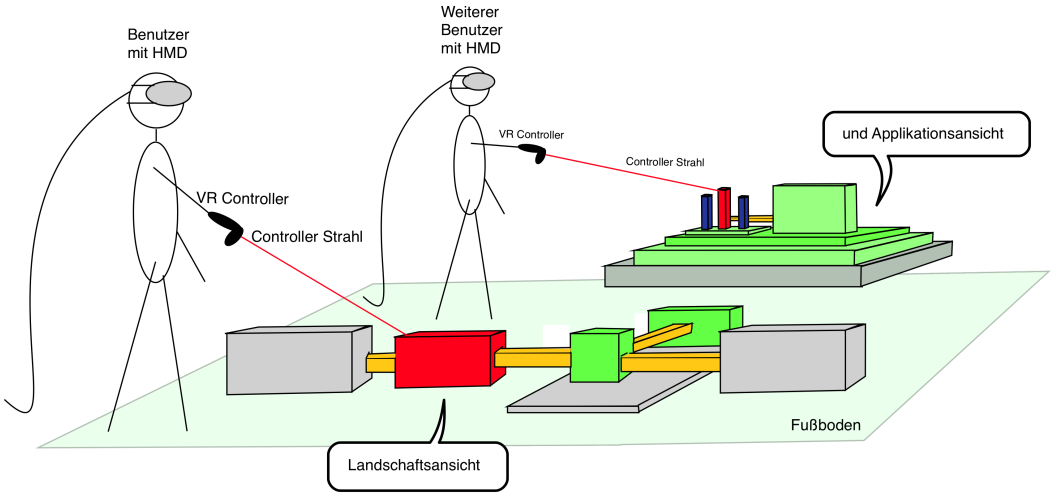

Abbildung 4.3. Visualisierung einer Software mit *ExplorViz* in virtueller Realität, die kollaborativ nutzbar ist

visualisierten Landschaft, dargestellt. Damit die Landschaft immer in den virtuellen Raum passt und somit vom Benutzer erkundet werden kann, muss diese abhängig von ihrer

Größe skaliert werden. Des Weiteren betrachtet der Benutzer die einzelnen Komponenten der Landschaft von oben und kann sich durch diese hinweg bewegen. Die Perspektive ist vergleichbar mit der eines Riesen, welcher sich durch eine normal große Stadt bewegt. Dies ermöglicht einen Überblick über die dargestellte Software und zugleich eine genauere Sicht auf die einzelnen Komponenten, indem sich der Benutzern diesen nähert. Hierzu kann der Benutzer den Kopf in Richtung der gewünschten Komponente bewegen, in die Hocke beziehungsweise Knie gehen oder sich direkt auf den Boden setzen. Analog kann sich der Benutzer ebenfalls von den Komponenten entfernen, um diese aus größerer Distanz zu betrachten. Die bisherige Landschaftsansicht stellt die Kommunikation zwischen den einzelnen Komponenten quantitativ durch die Breite der eingehenden und ausgehenden Kommunikationslinien dar. Die dreidimensionale Darstellungsweise der Landschaft soll genutzt werden, um die Anzahl der eingehenden Anfragen durch die Höhen der Komponenten zu modellieren. Komponenten mit einer hohen Anzahl eingehender Anfragen sollen dabei höher und die mit einer geringeren Anzahl niedriger dargestellt werden. Hierdurch kann schnell ein Überblick über die eingehende Kommunikation und somit über die Wichtigkeit der einzelnen Komponenten erhalten werden.

Weiterhin soll die geplante VR-Visualisierung die Landschaftsansicht mit der Applikationsansicht verschmelzen, sodass in der dreidimensionalen Landschaftsansicht eine dreidimensionale Applikation erzeugt werden kann. Das dargestellte Szenario soll mithilfe der *HTC VIVE Controller* manipuliert werden und der Benutzer durch diese interagieren können. Die Controller sollen dazu über einen Strahl verfügen, der einem Laserpointer ähnelt und damit anvisierte Komponenten rot markiert, die geometrisch verändert werden können. Der *HTC VIVE Controller* verfügt über verschiedene Knöpfe, Anschlüsse und Oberflächenstrukturen, die in Abbildung 4.4 dargestellt und mit einer Nummer versehen sind.

Für diese Arbeit sind lediglich die Knöpfe mit der Nummer *2*, *7* und *8* von Bedeutung. Der Knopf mit der Nummer *2* ist das *Trackpad* (großer runder Knopf), der mit der Nummer *7* der *Trigger* (Abzug) und jener mit der Nummer *8* ist die *Griff Taste* (seitlicher Knopf). Durch Betätigen des *Triggers* sollen die einzelnen anvisierten Komponenten geöffnet und geschlossen werden können. Weiterhin soll durch Betätigen des *Triggers* aus einer anvisierten blauen Fläche (geschlossene Applikation) eine 3D-Applikation, wie in der Applikationsansicht, erzeugt werden können. Zum Löschen einer 3D-Applikation soll eine beliebige blaue Fläche (geschlossene Applikation) anvisiert und der *Trigger* betätigt werden können.

Diese dreidimensionale Applikation soll durch Gedrückthalten des *Trackpads* rotiert, verschoben und gezommt sowie deren Pakete oder Klassen ausgewählt und detailreicher dargestellt werden können. Das Zoomen kann ebenfalls durch die Veränderung des Abstands zum Modell, mittels Gehen in der realen Welt, realisiert werden. Zudem soll die Darstellung zusätzlicher Informationen zu den einzelnen Komponenten der Landschaft oder der 3D-Applikation in einem Fenster möglich sein. Dies soll durch Betätigen der *Griff*

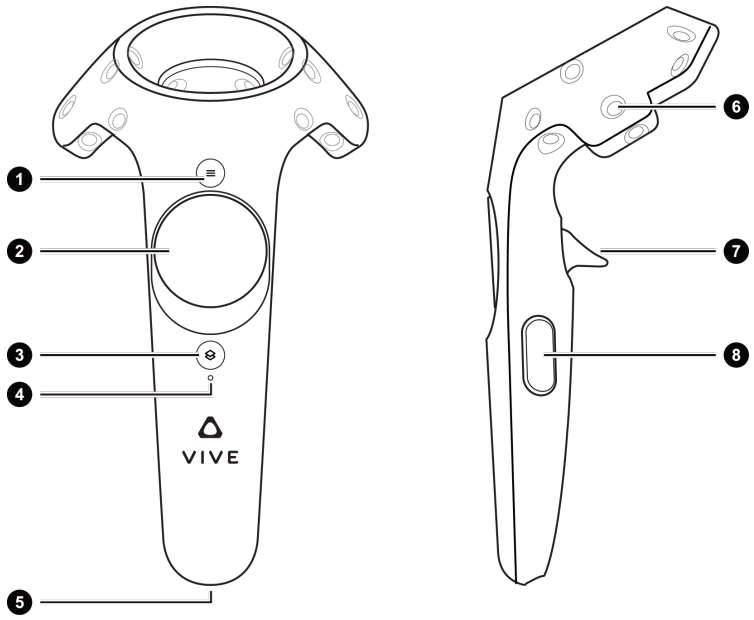

Abbildung 4.4. Überblick über den *HTC VIVE Controller* [a]

[a] https://docs.unity3d.com/Manual/OpenVRControllers.html

Taste realisiert werden.

Im Rahmen der kollaborativen Nutzbarkeit der VR-Visualisierung muss jeder Benutzer und dessen Position an das *Backend* übermittelt und dort registriert werden. Die Benutzer sollen dabei gemeinsam eine virtuelle Umgebung betreten und eine mit *ExplorViz* überwachte Software darin erkunden und manipulieren können. Für die Umsetzung des Raumkonzepts ist es erforderlich die Postion des Benutzers zu bestimmen. Hierfür können wir die Spezifizierung der HMD abstrahieren und die API *WebVR.js*[5] als Schnittstelle verwenden, um geräteunabhängig deren Position zu bestimmen. Jede Aktion eines *Clients* soll dazu führen, dass das Backend den Zustand der Umgebung aktualisiert und alle *Clients* anweist das aktualisierte Modell neu darzustellen. In der Modellierung der Avatars bietet *Three.js* eine vielversprechende Alternative zu *Unity3D*. Der Grund dafür liegt darin, dass *Three.js* eine sehr realistische, individuelle Gestaltung und Animation der Avatars ermöglicht. Der Grundbaustein eines solchen Avatars ist die Klasse *TREE.Mesh*, von welcher *SkinnedMesh*[6] erbt und die die Animation eines 3D-Objektes ermöglicht.

[5] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/WebVR.js
[6] https://threejs.org/docs/#api/objects/SkinnedMesh

Die Visualisierung einer Software mit *ExplorViz* in virtueller Realität soll durch einen zusätzliche Reiter namens *VR* in der graphische Oberfläche von *ExplorViz* zugänglich gemacht werden. Hierzu soll diese um einen zusätzlichen Reiter erweitert werden, sodass neben den Feldern *Visualisierung* und *Turoial*, wie in Abbildung 4.5 zu sehen, auch eins mit *VR* zur Verfügung steht. Dieser Reiter soll die 3D-visualisierte Kombination aus

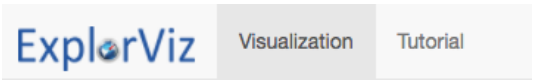

Abbildung 4.5. Darstellung der derzeitigen Navigationsleiste in *ExplorViz*

Landschafts- und Applikationsansicht sowie einen Knopf zum Betreten der virtuellen Realität bereitstellen. Nach dem Betreten der virtuellen Umgebung soll die gesamte Navigation und Interaktion vollständig durch die *VR-Hardware* erfolgen, sodass die Navigations- und Interaktionmöglichkeiten der virtuellen Realität nahtlos und bereits von Beginn an genutzt werden können.

Kapitel 5

Implementierung

Die Funktionalitäten aus dem Migrations- und Erweiterungskonzept, welche die Umsetzung der VR-Visualisierung beschreiben, werden in einer neuen Ember Komponente namens *vr-rendering* und deren importierten *Ember Utils* implementiert. Neu angelegte *Ember Utils* werden dabei im Ordner *utils/vr-rendering* abgelegt. Die neue Ember Komponente übernimmt viele Methoden der bestehenden Ember Komponenten *rendering-core.js*, *landscape-rendering.js* und *application-rendering.js* und stellt selbst keine Subklasse dieser dar. Der Grund dafür liegt darin, dass die Rendering-Schleife der Ember Komponente *rendering-core.js* angepasst werden müsste. Diese aktualisiert die Bilder der Umgebung und wird 90 Mal in der Sekunde aufgerufen, um eine Bildrate von 90 Bildern pro Sekunde zu gewährleisten. Dieser Aufruf geschieht einmal initial, danach rekursiv und kann deshalb nicht überschrieben kann.

Die Realisierung der VR-Visualisierung gliedert sich in die Abschnitte VR-Realisierung, 3D-Visualisierung und Interaktion. Abschnitt 5.1 (VR-Realisierung) umfasst die notwendigen Implementierungen für das Erschaffen und Aktualisieren der virtuellen Realiät. Grundlegend für die Darstellung und Wirkung eines Objekts in der virtuellen Realität ist dessen dreidimensionale Darstellung. Die hierfür erforderlichen Implementierungen werden in Abschnitt 5.2 (3D-Visualisierung) beschrieben. Die Manipulation von Objekten stellt eine weitere grundlegende Funktion dar, welche in Abschnitt 5.3 (Interaktion) dokumentiert ist. Abschließend werden die neu angelgten Dateien in die Struktur eines Addons gebracht, sodass diese von *ExplorViz* hinzugeladen werden können.

5.1. VR-Realisierung

Die Erzeugung der Bilder für das linke und rechte Auge, die Navigation und die Positionserkennung in der virtuellen Umgebung wird mithilfe der API *WebVR.js*[1] von *Three.js*[2] realisiert. Diese unterstützt die Verwendung der *Google Daydream*, *Oculus Rift* und *HTC VIVE*. Die folgenden Implementierungen finden in der Ember Komponente *vr-rendering.js* statt. Die genannte API stellt einen VR-Knopf zur Verfügung, der dem DOM angefügt wird und durch den der Bildschirm geteilt sowie die beiden Bilder für das linke und rechte

[1] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/WebVR.js
[2] https://threejs.org

Auge erzeugt werden. Die Voraussetzung dafür ist ein *WebVR*-fähiger Internetbrowser und ein angeschlossenes sowie betriebsbereites HMD. Erst das Benutzen dieses Knopfes aktiviert das HMD und stellt dort und im Browser die beiden Bilder dar. Diese werden durch das Bewegen des HMDs aktualisiert, sodass der Benutzer in der Lage ist durch die virtuelle Umgebung zu navigieren. Der VR-Knopf wird, wie in Listing 5.1 dargestellt, dem DOM hinzugefügt und liegt deswegen auf dem Canvas.

```
1 WEBVR.getVRDisplay(function(display) {
2    self.set('vrAvailable', true);
3    self.get('webglrenderer').vr.setDevice(display);
4          document.body.appendChild(WEBVR.getButton(display, self.get('
     webglrenderer').domElement));
5 });
```

Listing 5.1. Einfügen des Knopfes zum Betreten der virtuellen Umgebung

Im nächsten Schritt wird der VR-Modus des übernommenen *THREE.WebGLRenderers*[3] aktiviert. Dies geschieht in der Methode *initRendering()*, wie in Listing 5.2 Zeile 1 zu sehen. Des Weiteren wird in Zeile 2 der Bezug zum realen Raum hergestellt. Infolgedessen wird das Koordinatensystem der virtuellen Umgebung relativ zum realen Raum ausgerichtet, sodass dessen Koordinaten den neuen Ursprung des Koordinatensystems darstellen. Ein virtueller Fußboden hätte somit die gleiche Position wie der reale Fußboden, wenn er im Koordinatenursprung hinzugefügt werden würde.

```
1 this.get('webglrenderer').vr.enabled = true;
2 this.get('webglrenderer').vr.standing = true;
```

Listing 5.2. Umstellung des WebGLRenderers auf virtuelle Reaität

Damit die VR-Visualisierung flüssig dargestellt wird, müssen die Bilder regelmäßig aktualisiert werden. Im Selbstversuch hat sich gezeigt, dass eine Bildrate von 80-90 Bilder pro Sekunde empfehlenswert ist, um ruckelfrei durch die virtuelle Umgebung navigieren zu können. Die Bildrate wird von *Three.js* automatisch auf 90 Bilder pro Sekunde festgelegt. Die bestehende Rendering-Schleife in der Methode *initRendering* wird um eine Funktion namens *animate* erweitert, um den an VR angepassten *WebGL-Renderer* verwenden zu können. In Listing 5.3 ist die angepasste Rendering-Schleife dargestellt, welche die Bilder im vorgegeben Takt aktualisiert.

[3] https://threejs.org/docs/#api/renderers/WebGLRenderer

```
1 function animate() {
2    self.get('webglrenderer').animate(render);
3 }
4 function render() {
5    self.get('webglrenderer').render(self.get('scene'), self.get('camera'));
6 }
7 animate();
```

Listing 5.3. Darstellung der VR Rendering-Schleife

Sofern der VR-Knopf dem DOM hinzugefüft wurde, wird dieser in der Methode *cleanup()*, wie in Listing 5.4 zu sehen, wieder aus dem DOM entfernt.

```
1 if (this.get('vrAvailable')) {
2    document.body.removeChild(document.body.lastChild);
3 }
```

Listing 5.4. Entfernen des VR-Knopfes aus dem DOM

5.2. 3D-Visualisierung

Der Wechsel in die dreidimensionale Darstellung von *ExplorViz* geschieht durch einen zusätlichen Reiter. Mithilfe eines *Ember Instance Initializer* wird dazu die graphische Oberfläche von *ExplorViz* um einen zusätzlichen Reiter namens *VR* erweitert. Die folgenden Implementierungen finden in der Ember Komponente *vr-rendering.js* statt. Als Grundlage für den dreidimensionalen Raum erzeugen wir, wie in Listing 5.5 dargestellt, aus einem Foto des echten Fußbodens eine Textur (Zeile 1) und bilden diese auf das Material einer Fläche ab (Zeile 3). Des Weiteren passen wir dort die Maße dieser Fläche an die Größe des realen, für Fortbewegungen zur Verfügung stehenden Bereichs an (Zeile 2). Dabei erwartet die Klasse *THREE.BoxGeometry(width, height, depth)* die Parameter Breite, Höhe und Tiefe der realen Fläche. Die Breite und Tiefe des realen Fußbodens können wir dabei grob in Metern übergeben, sodass unser Raum etwa 5m breit und 3m tief ist. Diese Parameter müssen bei einer Größenveränderung der überwachten realen Fläche angepasst werden.

```
1 var floorTexture = new THREE.TextureLoader().load('images/materials/floor.jpg');
2 var floorGeometry = new THREE.BoxGeometry(5, 0, 3);
3 var floorMaterial = new THREE.MeshBasicMaterial({ map: floorTexture });
4 var floorMesh = new THREE.Mesh(floorGeometry, floorMaterial);
```

Listing 5.5. Abbilden der Fußbodentextur

Die Erzeugung der dreidimensionalen Landschaft wird in der Methode *populateScene()* bewerkstelligt. In dieser werden die Boxen, deren Beschriftung und Anordnung festgelegt. Die Berechnung der Anordnung von Objekten werden wir im Folgenden als *Layouting* bezeichnen. Zur Erzielung des dreidimensionale Effekts erweitern wir die bisherigen Flächen der Landschaftsansicht um eine weitere Dimension. Hierzu wird die Methode *createBox(model)* implementiert, welche eine Box aus den übergebenen Daten erzeugt. Diese Daten enthalten unter anderem die Farbe, Maße und den Namen der zu erstellenden Box. Mithilfe dieser Box wird in der 3D-Visualisierung ein geschlossenes *System*, eine geschlossene *Nodegroup* oder ein *Node* repäsentiert. Des Weiteren werden in der Methode *createPlane(model)* dreidimensionale Flächen aus den übergebenen Daten erzeugt, die den geöffneten Zustand eines *Systems* oder einer *Nodegroup* sowie den geschlossenen Zustand einer *Applikation* in der 3D-Visualiserung repäsentieren. In Abbildung 5.1 ist ein beispielhaftes Aussehen einer solchen Landschaft abgebildet. Dieses enthält geschlossenen Systeme

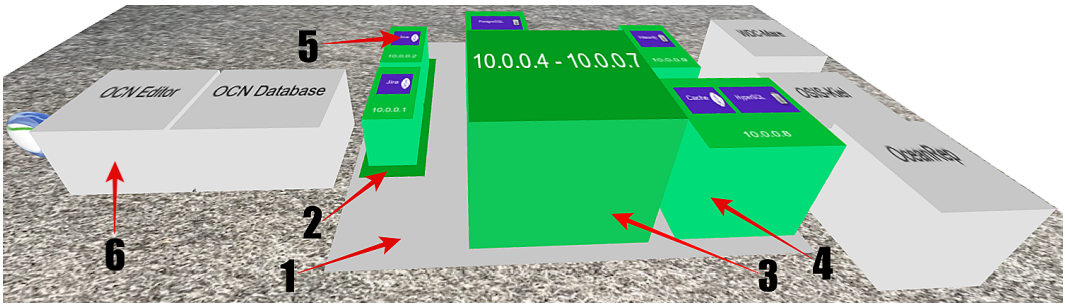

Abbildung 5.1. Darstellung der verschiedenen geometrischen Komponenten der Landschaft

(**6**), ein geöffnetes *System* (**1**), eine geöffnete *Nodegroup* (**2**) und eine geschlossene *Nodegroup* (**3**). Weiterhin sind dort *Nodes* (**4**) und geschlossene Applikationen (**5**) dargestellt.

Damit die dreidimensionale Landschaft auf dem Boden liegt und von oben betrachtet werden kann, wird diese um 90 Grad auf der X-Achse rotiert. Des Weiteren wird die dreidimensionale Landschaft durch die Methode *centerVREnvironment(vrEnvironment, floor)* mittig auf dem Fußboden ausgerichtet und direkt auf diesem platziert. Die Rotation der dreidimensionalen Landschaft um die X-Achse bewirkt das Vertauschen von Höhe und Tiefe der Komponenten. Die Tiefe (Höhe aus Benutzersicht) der Komponenten wird an die Anzahl der eingehenden Anfragen angepasst. Die Methode *assignDepth(requests)* bekommt die Anzahl der gesamten Anfragen an eine Komponente übergeben und liefert eine dafür festgelegte Tiefe. Durch dieses Verfahren wird jeder Komponente eine spezifische Tiefe zugewiesen. In Abbildung 5.1 ist dies gut an den drei *Systemen* am rechten Rand der Landschaft zu erkennen.

Die dreidimensionale Applikation wird wie in der bestehenden Ember Komponente *application-rendering.js* erzeugt. Mithilfe der Methode *add3DApplicationToLandscape(application, position, rotation)* wird die Applikation der gegebenen Rotation und Position angepasst, dieser eine graue Grundplatte angeheftet und der dreidimensionalen Landschaft hinzugefügt. Die Grundplatte wird mithilfe des *Ember Utils application-rendering/foundation-builder* bei jedem Neuzeichnen der 3D-Applikation gelöscht und neu erzeugt. Das Neuzeichnen der 3D-Applikation wird bewerkstelligt, indem erst die Methode *removeApp3D()* und dann die Methode *add3DApplicationToLandscape(application, position, rotation)* aufgerufen wird. Die 3D-Applikation wird dabei erst aus der Szene entfernt und daraufhin neu hinzugefügt. Das Aufrufen der Methode *removeApp3D()* entfernt die dreidimensionale Applikation aus der dreidimensionalen Landschaft, ohne dass diese dabei neuberechnet oder neugezeichnet werden muss. Das Kapseln der dreidimensionalen Landschaft und der dreidimensionalen Applikation führt dazu, dass diese gesondert behandelt und damit einzeln neuberechnet und neugezeichnet werden können. Dies hat den Vorteil, dass wesentlich weniger Rechenleistung benötigt wird und der Quellcode übersichtlicher ist. Die Beschriftung der Boxen Applikation wird mithilfe der Klasse *THREE.TextBufferGeometry* in dem *Ember Util /application/labeler.js* realisiert. Dafür wird eine Instanz dieser Klasse als Geometrie verwendet und mithilfe dieser ein neues Objekt vom Typ *THREE.Mesh* erzeugt. Der entsprechenden Box wird dieses Objekt als *Child* hinzugefügt und dieses an der Position der Box ausgerichtet.

Dieses Verfahren haben wir ebenfalls zur Beschriftung der Landschaftsboxen verwendet. Im Verlauf der Implementierung hat sich jedoch gezeigt, dass diese Realisierung nicht für die Landschaftsboxen in virtueller Realität geeignet ist, da sie zu viel Rechenleistung verbraucht und dies zu Einbrüchen in der Bildrate sowie Übelkeit beim Benutzer geführt hat. Aus diesem Grund werden die Beschriftungen in die Textur der Boxen integriert. Dafür wird dem *Ember Util vr-rendering/labeler.js* zusätzlich zur Box, dem Text und der Textfarbe ebenfalls die Boxfarbe übergeben. Dort wird für jede Box ein eigenes *Canvas* erzeugt und dieses dem DOM angehängt. Wie in Listing 5.6 zu sehen, wird der zweidimensionale Kontext des *Canvas* mit einem Rechteck in Boxfarbe (Zeile 2-3) sowie dem Text befüllt (Zeile 4 und 6) und entsprechend formatiert (Zeile 5). Die Methode *ctx.fillText(text, x,y,max)* befüllt den Kontext an der übergebenen x- und y-Koordinate mit Text und skaliert diesen sobald die übergebene maximale Textlänge überschritten wird.

```
1 var ctx = canvas.getContext('2d');
2     ctx.fillStyle = boxColor;
3     ctx.fillRect(0, 0, canvas.width, canvas.height);
4     ctx.fillStyle = textColor;
5     ctx.textAlign = "center";
6     ctx.fillText(text, x,y,max);
```

Listing 5.6. Füllen des Canvas mit Boxfarbe und Text

In Listing 5.7 wird die Verwendung eines *Canvas* als Boxtextur mithilfe von Quellcode dargestellt. Zunächst wird aus diesem *Canvas* eine Textur erzeugt (Zeile 1), diese auf ein Material abgebildet (Zeile 2) und für die obere Fläche der Box verwendet (Zeile 9). Damit die aus dem *Canvas* erstellte Textur korrekt dargestellt wird, muss diese aktualisiert werden (Zeile 3). Die restlichen Seiten der Box werden mit dem bisherigen Material bekleidet (Zeile 5-8, 10). Diese Technik wurde bereits im bestehenden VR-Modus von *ExplorViz legacy* zum Darstellen von Informationsflächen genutzt und wurde an die Verwendung von dreidimensionalen Boxen angepasst.

```
1  let texture = new THREE.Texture(canvas);
2  let canvasMaterial = new THREE.MeshBasicMaterial({map: texture});
3  texture.needsUpdate = true;
4
5  var materials = [oldMaterial, // Right side
6                   oldMaterial, // Left side
7                   oldMaterial, // Back
8                   oldMaterial, // Front
9                   canvasMaterial , // Top
10                  oldMaterial  // Buttom
11                 ];
12 texture.dispose();
13 canvasMaterial.dispose();
```

Listing 5.7. Verwendung eines Canvas als Boxtextur

Es ist von großer Wichtigkeit die bestehenden Materialien, wie im unteren Teil von Listing 5.7 zu sehen, mit der Methode *dispose()* korrekt zu entfernen, da sonst belegter Speicher der Grafikkarte möglicherweise nicht richtig freigegeben wird. Mit der Methode *saveTextForLabeling(textToShow, parent, color, boxColor)* wird zu jeder Box ein Eintrag in einem Feld generiert, der die Box, deren Beschriftungen, Textfarben und Boxfarben enthält. Infolgedessen wird mithilfe der Methode *drawTextLabels(font, configuration)* zu jedem Eintrag einmalig ein *Canvas* erstellt, dieses der Größe der Box angepasst und der darzustellenden Text, dessen Größe sowie Position ermittelt. Daraufhin wird die obere Seite der Box, wie in Listing 5.7 dargestellt, mit dem *Canvas* bekleidet.

Bei der Farbänderung einer Box muss zwischen Landschafts- und Applikationsbox unterschieden werden. Das Material der Landschaftsboxen besteht durch die Integration der Beschriftung aus einem Feld, welches das Material jeder Boxseite enthält. Das Anpassen einer solchen Box wird von der Methode *redrawLabel(entity, textColor, name, color)* im *Ember Util vr-rendering/labeler.js* bewerkstelligt. Der bestehende Kontext des zugeordneten *Canvas* wird zunächst mit der übergebenen Boxfarbe überschrieben. Weiterhin bekommt diese Methode den abzubildenden Text direkt als Parameter übergeben und schreibt diesen,

in der übergebenen Textfarbe, in den zuvor gefärbten Kontext. Die Applikationsboxen bestehen aus einem einzigen Material, dessen Farbe direkt in einem Attribut hinterlegt ist. Die Farbänderung wird hier durch direktes Abrufen und Ersetzen der Farbe des Materials realisiert.

Im Weiteren Verlauf der Implementierung hat sich gezeigt, dass die bestehende Realisierung der Kommunikationslinien zu Einbrüchen in der Bildrate und damit zu einem Ruckeln in der virtuellen Umgebung geführt hat. Der Grund dafür könnte darin liegen, dass zu viele Kommunikationspunkte berechnet werden, die Darstellung der Kommunikationslinien oder deren *Layouting* zu viel Rechenleistung der Grafikkarte in Anspruch nimmt. Aus diesem Grund werden die Kommunikationslinien vorerst ausgeblendet. Die Berechnung dieser Linien geschieht in der Methode *addCommunicationLineDrawing(tiles, communicationMeshes)* der Ember Komponente *vr-rendering.js*.

5.3. Interaktion

Die Interaktion mit der virtuellen Umgebung erfolgt vollständig durch die Controller der *HTC VIVE*. Die Erzeugung und Aktualisierung der Controller wird in der Ember Komponente *vr-rendering.js* bewerkstelligt. Für die Realisierung der Controller wird die API *THREE.ViveController.js*[4] verwendet, deren Anwendung bereits in Abschnitt 4.1 beschrieben wurde. Damit die Position, Richtung und Neigung der Controller stets aktuell ist, müssen diese in der Rendering-Schleife aktualisiert werden. Dazu wird diese um die Anweisungen aus Listing 5.8 ergänzt.

```
1 self.get('controller1').update();
2 self.get('controller2').update();
```

Listing 5.8. Aktualisierung der Controller

Damit die Controller in der virtuellen Umgebung dargestellt werden, müssen die entsprechenden Texturen und Informationen zum Aussehen geladen werden. Diese werden dem Controller hinzugefügt, sodass dieser nun das gleiche Aussehen wie in der realen Welt hat. Hierzu haben wir den bestehenden Ansatz aus Abschnitt 4.1 zum Laden des Aussehens nutzen können. Zur Realisierung des Laserstrahls wird dem Controller eine Linie hinzugefügt, welche an die Ausrichtung des Controllers angepasst ist. In Listing 5.9 ist die Erzeugung eines grünen (Zeile 9 - 12) und schwarzen Strahls (Zeile 5 - 8) dargestellt, die den Controllern zugeordnet werden (Zeile 14 - 15). Die Controller können somit durch die Farbe des Strahls von einander unterschieden werden. Der Controller mit dem schwarzen Strahl wird der linken Hand und der andere der rechten Hand des Benutzers zugeordnet. In Abschnitt 5.3.2 werden den Controllern Informationstexte angefügt, welche links vom

[4] https://github.com/mrdoob/three.js/blob/5e85ec8c4ed8470432539cddbd0b3bee9eebeeab/examples/js/vr/ViveController.js

rechten Controller und rechts vom linken Controller dargestellt werden, um sich zentriert im Blickfeld des Benutzers zu befinden. Die Zuordnung zur linken und rechten Hand ist notwendig, damit der Benutzer die Informationstexte gut seinem Blickwinkel anpassen kann.

```
this.set('geometry', new THREE.Geometry());
this.get('geometry').vertices.push(new THREE.Vector3(0, 0, 0));
this.get('geometry').vertices.push(new THREE.Vector3(0, 0, -1));

let line1 = new THREE.Line(this.get('geometry'));
line1.scale.z = 5;
line1.material.color = new THREE.Color('rgb(0,0,0)');
line1.material.opacity = 0.25;
let line2 = new THREE.Line(this.get('geometry'));
line2.scale.z = 5;
line2.material.color = new THREE.Color('rgb(0,204,51)');
line2.material.opacity = 0.25;

this.get('controller1').add(line1);
this.get('controller2').add(line2);
```

Listing 5.9. Erzeugung des Controllerstrahls

Die folgenden Unterkapitel beschreiben die automatische Markierung der Komponenten der Landschaft durch den Controller der *HTC VIVE* sowie dessen Knopfbelegung und damit verbundenen Funktionen. Diese Realisierungen werden in dem *Ember Util vr-rendering/interaction* umgesetzt. Des Weiteren ist die Ursprungsfarbe jeder Box in einer globalen Farbliste des *Ember Utils* hinterlegt.

5.3.1. Markierung von Objekten

Das Markieren einer anvisierten Box wird von der Methode *checkIntersection(controller)* durch eine rote Färbung realisiert. Dabei werden nur Boxen markiert die geometrisch manipulierbar sind. Darunter Fallen jene Boxen, deren Position, Rotation oder Form änderbar sein soll. Die Methode bekommt einen Controller als Parameter übergeben, dessen Strahl auf den Zusammenstoß mit einer Box geprüft werden soll. Dazu wird dem *Ember Util vr-rendering/raycaster.js* die aktuelle Position und Ausrichtung des Controllers sowie eine Auswahl von Objekten übergeben, die für einen Zusammenstoß in Frage kommen. Beispielsweise soll der Fußboden derzeit nicht getroffen werden können und ist deshalb nicht in dieser Auswahl enthalten. Das *Ember Util* liefert daraufhin das getroffene Objekt zurück, das sich am nähesten am Controller befindet. Weiterhin ist jeder Controller eindeutig durch seine *ID* identifizierbar. Sobald ein Controller eine Box markiert, soll diese nicht

erneut vom zweiten Controller markiert werden können. Zusätzlich muss zwischen den Boxen der Landschaft und der Applikation unterschieden werden, da die Beschriftung der Landschaftsboxen in die Textur integriert und die der Applikationsboxen diesen als *Child* angehängt wird. Hierfür reservieren wir je Controller einen Platz in zwei Feldern, dessen Index der *ID* des Controllers entspricht und in den die zuletzt markierte Box geschrieben wird. Hierbei enthalten die Plätze des einen Feldes die Landschaftsboxen und die des anderen die Applikationsboxen. Durch die Abfrage aus Listing 5.10 wird die Methode umgehend verlassen, sobald das getroffene Objekt identisch mit dem zuletzt markierten Landschaftsbox ist.

```
1 if(intersectedViewObj && this.get('highlightedEntities')[id] &&
2    this.get('highlightedEntities')[id].id === intersectedViewObj.object.id){
3    return;
4 }
```

Listing 5.10. Ausschließen einer Mehrfachmarkierung von Landschaftsboxen

Sofern die Methode nicht verlassen wurde und eine bereits markierte Landschafts- oder Applikationsbox existiert, wird deren Boxfarbe in den Ursprungszustand zurückgesetzt, unabhängig davon ob ein neues Objekt getroffen wurde oder nicht. In Listing 5.11 ist zu sehen wie dies bei Landschaftsboxen bewerkstelligt wird. Zunächst wird die Box klassifiziert (Zeile 3 - 10), die Box- sowie Textfarbe aus der Farbliste herausgesucht und zusammen mit der Box sowie deren Namen dem *Ember Util vr-rendering/labeler.js* übergeben (Zeile 12 - 13). Dort wird, wie in Abschnitt 5.2 beschrieben, das *Canvas* entsprechend angepasst und als Textur verwendet.

```
1  if(this.get('highlightedEntities')[id] ...){
2     let entity = this.get('highlightedEntities')[id];
3     let index = "text"+entity.type;
4     let name;
5     if(entity.type === "nodegroup"){
6        index = "textnode";
7     }
8     else if(entity.type === "application"){
9        index = "textapp";
10    }
11    name = entity.userData.model.get('name');
12    this.get('labeler').redrawLabel(entity, this.get('colorList')[index],
13                      name,this.get('colorList')[entity.type]);
14 }
```

Listing 5.11. Wiederherstellen der Ursprungsfarbe einer Landschaftsobjekts

Das Zurücksetzen der Farbe einer Applikationsbox gestaltet sich einfacher, da hier kein *Canvas* angepasst werden muss, sondern direkt auf das Material der Box und somit auf die Farbe zugegriffen werden kann. Wie in Listing 5.12 zu sehen, wird geprüft ob eine markierte Box der Applikation existiert und deren Farbe mit der entsprechenden Farbe aus der Farbliste überschrieben.

```
1 if(this.get('highlightedEntitiesApp')[id] ...){
2    this.get('highlightedEntitiesApp')[id].material.color =
3          new THREE.Color(this.get('colorListApp')[
4             this.get('highlightedEntitiesApp')[id].userData.model.get('color')]);
5 }
```

Listing 5.12. Wiederherstellen der Ursprungsfarbe einer Applikationsbox

Nach dem Zurücksetzen der Farbe einer Landschafts- oder Applikationsbox wird der entsprechende Eintrag im Feld gelöscht, da nun keine Box mehr markiert ist. Das rote Färben einer Box gestaltet sich ähnlich wie das Zurücksetzen in die Ursprungsfarbe, nur dass hier die Farbe Rot übergeben und der entsprechende Feldplatz der *ID* des Controllers nicht gelöscht, sondern mit der markierten Box beschrieben wird. Dies funktioniert problemlos, da *JavaScript* mit Referenzen arbeitet und Objekte durch eine Zuweisung identisch sind, sofern sie nicht mit der Methode *clone()* erstellt wurden. In Abbildung 5.2 ist der Unterschied eines markierten und eines normalen *Systems* dargestellt.

Abbildung 5.2. Darstellung eines normalfarbigen *Systems* links und eines rot markierten rechts

5.3.2. Belegung der Controllerknöpfe

Die Controller werden durch die *THREE.ViveController.js* API realisiert, wodurch das Betätigen eines Controllerknopfes ein entsprechendes Event auslöst. Damit auf dieses Event reagiert werden kann, wird jedem Controller ein *Event Listener* hinzugefügt, der eine Funktion ausführt sobald das Event ausgelöst wird. Listing 5.13 zeigt dies beispielhaft für die Registrierung des Events *'triggerdown'*.

```
1 self.get('controller1').addEventListener('triggerdown',
      registerControllerTriggerDown);
2 self.get('controller2').addEventListener('triggerdown',
      registerControllerTriggerDown);
3
4 function registerControllerTriggerDown(event){
5    self.onControllerTriggerDown(event);
6 }
```

Listing 5.13. Registrierung des Events *'triggerdown'*

Die dargestellte Methode *onControllerTriggerDown(event)* aus Zeile 5 implementiert die gewünschte Reaktion auf das Event. Insgesamt stellt die API die Events *'thumbpaddown'*, *'thumbpadup'*, *'triggerdown'*, *'triggerup'*, *'gripsdown'*, *'gripsup'*, *'menudown'*, *'menuup'* zur Verfügung.

Der Trigger

Das Betätigen des Triggers (Abzug, Abbildung 4.4 Nummer 7) wird dazu verwendet, anvisierte Systeme und Nodegroups der Landschaft sowie Pakete der 3D-Applikation zu öffnen und wieder zu schließen. Des Weiteren werden dadurch aus anvisierten, geschlossenen Appliaktionen (blaue Flächen auf der Oberseite der Nodes) dreidimensionale Applikationen erzeugt oder diese gelöscht. Die Methode *onControllerTriggerDown(event)* implementiert diese Funktionalitäten. Zu Beginn wird mithilfe des *Raycasters* ermittelt, ob ein Objekt getroffen wurde. Ist das der Fall, wird dieses der Applikaiton oder der Landschaft zugeordnet und das Attribut *opened* getoggelt. Mithilfe der Unterscheidung wird entweder das Event *redrawApp*, *redrawScene*, *showApplication* oder *removeApplication* ausgelöst. In der Ember Komponente *vr-rendering* wird auf diese Events gehört und mit einer internen Funktion reagiert.

Für das Öffnen oder Schließen einer Landschaftsbox wird auf das Event *redrawScene* gewartet und die Landschaft mithilfe der Funktion *populateScene()* komplett aus der Szene entfernt und neu hinzugefügt. Damit aus einer geschlossenen Applikation eine dreidimensionale erzeugt werden kann, müssen die erforderlichen Daten vom *Backend* bereitgestellt werden. Ist das nicht der Fall, dienen diese Flächen nur dazu eine bestehende 3D-Applikation zu löschen. Dafür wird auf das Event *removeApplication* gewartet und darauf mit der Methode *removeApp3D()* reagiert, welche die 3D-Applikation aus der Szene entfernt. Sofern die erforderlichen Daten der Applikation bereitstehen, wird auf das Event *showApplication* gewartet. Unter der Voraussetzung, dass noch keine 3D-Applikation existiert, fügt die interne Funktion, die das Event behandelt, der Szene eine 3D-Applikation hinzu. Für das Öffnen oder Schließen einer Applikationsbox wird auf das Event *redrawApp* gewartet und zunächst die Objektdaten, die aktuelle Position und

Rotation der 3D-Applikation gesichert. Im Anschluss wird diese durch die Methode *removeApp3D()* aus der Szene entfernet und die gesicherten Daten als Parameter an die Methode *add3DApplicationToLandscape(app3DModel, appPosition, appRotation)* übergeben, welche die aktualisierte 3D-Applikation der Szene hinzufügt.

Das Trackpad

Das Gedrückthalten des *Trackpads* (großer runder Knopf, Abbildung 4.4 Nummer 2) wird dazu verwendet eine anvisierte 3D-Applikation an den Controller zu binden und somit die Bewegungen des Controllers darauf anzuwenden. Das Rotieren der 3D-Applikation um die x-, y- oder z-Achse wird durch das Drehen des Controllers um eine der entsprechenden Achsen bewirkt. Das Verschieben der 3D-Applikation auf der x-, y- oder z-Achse wird durch das Verschieben des Controllers auf einer der entsprechenden Achsen bewirkt. Alternativ zum Annähern an die 3D-Applikation oder zum davon Entfernen, wird das Heran- und Herauszommen durch Herangeziehen oder Zurückschieben des Controllers ermöglicht. Die Methode *onControllerThumbpadDown(event)* implementiert diese Funktionalitäten. Zu Beginn wird mithilfe des *Raycasters* ermittelt, ob ein Objekt getroffen wurde und dieses Bestandteil einer 3D-Applikation ist. Daraufhin wird die Transformationsmatrix der 3D-Applikation relativ zu der Transformationsmatrix des Controllers gesetzt. Dies geschieht indem die Transformationsmatrix der 3D-Applikation mit der Inversen Transformationsmatrix des Controllers multipliziert und in die Vektoren *position*, *quaternion* und *scale* aufgeteilt wird. Im Anschluss wird die angepasste 3D-Applikation dem Controller hinzugefügt.

Das Loslassen des *Trackpads* führt dazu, dass die Bewegungen des Controllers keinen Einfluss mehr auf die 3D-Applikation haben. Dazu wird die im Controller hinterlegte 3D-Applikation von diesem entfernt und der Szene wieder hinzugefügt wird. Dies wird in der Methode *onThumbpadControllerUp(event)* realisiert. Dafür wird die obige Transformation der 3D-Applikation, durch Multiplizieren mit der Transformation des Controllers, wieder relativ zum Ursprung ausgerichtet.

Die Griff Taste

Das Betätigen der Griff Taste (seitlicher Knopf, Abbildung 4.4 Nummer 8) wird dazu verwendet Informationen zu anvisierten Objekten anzuzeigen oder zu löschen. Im Verlauf hat sich herausgestellt, dass die Lesbarkeit des Textfeldes stark abhängig vom Blickwinkel des Benutzers ist. Aus diesem Grund wird das Textfeld, ähnlich wie die 3D-Applikation bei Gedrückthalten des *Trackpads*, an den Controller gebunden und direkt links neben dem rechten beziehungsweise rechts neben dem linken Controller positioniert. Abbildung 5.3 zeigt ein beispielhaft angebrachtes Textfeld am rechten Controller. Der Ausrichtung des Controllers kann somit verändert werden, um das Textfeld dem Blickwinkel anzupassen. Die Methode *onControllerGripsDown(event, rightHand)* implementiert diese Funktionalitäten.

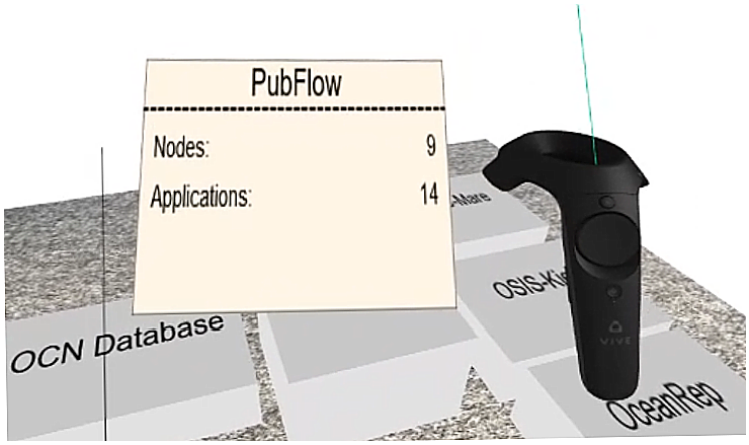

Abbildung 5.3. Anheften eines Textfeldes an die linke Seite des rechten Controllers

Zu Beginn wird mithilfe des *Raycasters* ermittelt, ob ein Objekt getroffen wurde. Sofern das der Fall ist, wird dieses dem *Ember Util vr-rendering/hover-handler.js* übergeben. Dort werden die Informationen zu diesem Objekt in ein Array eingespeist und dieses zurückgegeben. Ähnlich wie in Abschnitt 5.2 wird der Kontext eines Canvas mit den Informationen gefüllt, eine Fläche damit texturiert und dem Controller hinzugefügt. Dieses Textfeld wird vom Controller entfernt, wenn kein Objekt von diesem getroffen und die *Griff Taste* betätigt wurde.

Kapitel 6

Evaluierung

In diesem Kapitel wird die Nutzbarkeit des entwickelten VR-Modus untersucht. Im Rahmen einer Studie werden dazu ausgewählte Versuchspersonen *ExplorViz* mithilfe virtueller Realität verwenden.

6.1. Forschungsfrage

Zur Herausstellung der Nutzbarkeit von *ExplorViz* in virtueller Realität betrachten wir die Forschungsfrage, *ob der VR-Modus von den Versuchspersonen angenommen wird* und *dieser eine praktische Alternative zum Erkunden am Bildschirm darstellt*. Die VR-Visualisierung wird zur Beantwortung der Forschungsfrage nicht direkt mit der 2D-Visualisierung verglichen, sondern im Rahmen einer Studie auf die folgenden Hypothesen geprüft:

- H_1: Die zusätzliche Dimension der 3D-Visualisierung bietet einen Vorteil gegenüber der 2D-Visualisierung (Räumlichkeit)
- H_2: Die Verwendung des VR-Modus beeinträchtigt den Benutzer nicht (Nutzererlebnis)
- H_3: Die Navigation im VR-Modus wird vom Benutzer gut angenommen (Navigation)
- H_4: Die Interaktion durch die Controller wird vom Benutzer gut angenommen (Interaktion)

6.2. Methodik

Zur Verifizierung der obigen Hypothesen und zur Beantwortung der Forschungsfrage, führen wir ein Experiment mit Versuchspersonen durch. Dabei erheben wir relevante Daten, die der Verifizierung der obigen Hypothesen dienen. Für die Erhebung der Daten verwenden wir einen Fragebogen und teilen diesen in drei Kategorien auf. Einen allgemeinen Bogen, einen speziell für den VR-Modus und einen für den Studienleiter. Alle Bögen enthalten einem Ankreuzteil mit aufgeführten Behauptungen, die mit *stimme völlig zu*, *stimme eher zu*, *neutral*, *stimme eher nicht zu* oder *stimme gar nicht zu* bewertet werden können. Im Anschluss an diese Studie findet eine weitere zum *Brain-Computer-Interface* statt. Aus diesem Grund enthält der allgemeine Bogen Daten, die nicht für diese Arbeit,

sondern die entsprechende zum *Brain-Computer-Interface* relevant sind. Deshalb wird sich die Auswertung des allgemeinen Bogens auf wenige ausgewählte Punkte beschränken, die in Tabelle 6.1 dargestellt sind.

Tabelle 6.1. Auswahl der Behauptungen des allgemeinen Bogens

ID	Behauptung
D_1	Ich bin anfällig für Motion Sickness (Reisekrankheit, Seekrankheit, etc.)
D_2	Ich habe eine Sehschwäche
D_3	Ich trage eine Brille
D_4	Ich habe bereits Erfahrung mit virtueller Realität gemacht

Der VR-Bogen besteht aus einem Akreuz- und einem Freitextteil. Die Behauptungen des Ankreuzteils werden in die Kategorien *Nutzererlebnis* ($A_1 - A_7$), *Navigation* ($B_1 - B_5$) und *Interaktion* ($C_1 - C_9$) eingeteilt. Die Behauptungen zum *Nutzererlebnis* beziehen sich auf das Empfinden des Benutzers beim Erkunden der virtuellen Umgebung und sind in Tabelle 6.2 abgebildet. Die Behauptungen zur *Navigation* sollen herausstellen,

Tabelle 6.2. Behauptungen im VR-Bogen zur Bewertung des Nutzererlebnisses

ID	Behauptung
A_1	Das Erkunden der virtuellen Umgebung war angenehm
A_2	Die Darstellung der virtuellen Umgebung war angenehm
A_3	Die VR-Brille war komfortabel zu tragen
A_4	Die Rückmeldung der virtuellen Umgebung war direkt und flüssig
A_5	Ich war frei von Schwindel und Übelkeit
A_6	Ich hätte noch weitere Zeit in der virtuellen Umgebung bleiben können
A_7	Ich wüde den VR-Modus wiederverwenden

wie intuitiv und praktisch die Navigationsmöglichkeiten in der virtuellen Umgebung empfunden wurden und sind in Tabelle 6.3 dargestellt. Die Behauptungen zur *Interaktion*

Tabelle 6.3. Behauptungen im VR-Bogen zur Bewertung der Navigation

ID	Behauptung
B_1	Das Umschauen im virtuellen Raum durch Drehen des Kopfes oder Drehen des Körpers war intuitiv
B_2	Die Annäherung an Objekte durch Daraufzugehen oder durch Annähern des Kopfes war intuitiv
B_3	Die Gewinnung von Abstand zu Objekten durch Wegbewegen war intuitiv
B_4	Die Fortbewegung im virtuellen Raum durch reales Gehen war praktisch
B_5	Die Körper- und Kopfbewegungen zur Navigation (umschauen, annähern, distanzieren) waren leicht zu lernen

sind in Tabelle 6.4 abgebildet und sollen herausstellen, wie intuitiv und praktisch die Interaktionsmöglichkeiten in der virtuellen Umgebung empfunden wurden. Weiterhin

Tabelle 6.4. Behauptungen im VR-Bogen zur Bewertung der Interaktion

ID	Behauptung
C_1	Die Interaktion mit der virtuellen Umgebung durch die Controller war intuitiv
C_2	Die Bedienung der Controller war leicht zu lernen
C_3	Das automatische Markieren von Objekten durch den Strahl des Controllers war praktisch
C_4	Durch die Markierung habe ich schnell verstanden mit welchen Objekten interagiert werden kann
C_5	Das Öffnen und Schließen von Boxen durch den Abzug war praktisch
C_6	Das Rotieren, Verschieben und Zoomen eines Objektes durch Gedrückthalten des großen runden Knopfes war praktisch
C_7	Beim Rotieren, Verschieben oder Zoomen eines Objektes hatte ich das Gefühl, dieses in der Hand zu halten
C_8	Die Darstellung von Informationen in einem Fenster neben dem Controller war praktisch
C_9	Ich konnte dieses Fenster ähnlich wie eine Zeitung in die Hand nehmen und meinem Blickwinkel anpassen

wird den Bewertungseinheiten des Bogens des Studienleiters und des VR-Bogens ein fester, natürlicher Wert von Null bis Vier zugeordnet. Hierdurch ist eine einfache Berechnung der durchschnittlichen, minimalen und maximalen Bewertung möglich. Der Wert Vier repräsentiert dabei die höchste (*stimme völlig zu*) und Null (*stimme gar nicht zu*) die geringste Bewertung.

Der Freitextteil des Bogens gliedert sich in die Bereiche *Anmerkungen* und *Verbesserungsvorschläge*. Der Bogen für den Studienleiter besteht ebenfalls aus einen Ankreuzteil sowie drei Freitextfeldern für *allgemeine Bemerkungen*, *Besonderheiten* und den *Verlauf* des Experiments. Die Behauptungen des Ankreuzteils sind in Tabelle 6.5 abgebildet.

Tabelle 6.5. Behauptungen im Bogen des Studienleiters

ID	Behauptung
S_1	Die VR-Brille war einfach anzubringen
S_2	Der Proband hat das höchste System gefunden

6.3. Aufbau und Ablauf des Experiments

Die Versuchspersonen für das Experiment wurden bewusst aus keiner im Vorfeld definierten Gruppe ausgewählt, da der Kreis der potentiellen Anwender von *ExplorViz* sehr groß ist.

Im Verlauf des Experiments werden dem Probanden Geometrie- und keine Verständnisaufgaben gestellt, da die aufgestellten Hypothesen qualitativen Charakter haben. Des Weiteren werden die Hypothesen H_3 und H_4 näher auf Intuitiviät, Praktik und Lernabrkeit untersucht. Zu Begin wird dem Probanden mithilfe eines kurzen Einleitungstextes vermittelt, was in der virtuellen Umgebung dargestellt ist und wie dies zu verstehen ist. Hierdurch wird der noch fehlende Praxisbezug hergestellt. Zudem wird er darüber aufgeklärt, dass alle erhobenen Daten anonym behandelt und lediglich Bildschirmaufnahmen gemacht werden. Im Anschluss füllt der Proband den allgemeinen Bogen aus. Der weitere Verlauf des Versuchs gliedert sich die Trainings- und Hauptphase. In beiden Phasen trägt der Benutzer das HMD und bekommt akustisch Anweisungen übermittelt. In Abschnitt 6.3.1 und Abschnitt 6.3.2 werden die beiden Phasen genauer beschrieben. Alle Versuchspersonen befinden sich im selben Raum, verwenden dieselbe Ausrüstung und bekommen die gleichen Anweisungen vorgelesen. Weiterhin liefert das *Backend* die darzustellen Daten, die momentan immer gleich sind. Diese Daten beschreiben derzeit die *Demo-Landschaft*[1] von *ExplorViz*. Hiermit ist sichergestellt, dass jede Versuchsperson die gleiche Landschaft erkundet. Jede Versuchsperson füllt nach dem Erkunden der Landschaft im virtuellen Raum den VR-Bogen aus. Währenddessen wird von uns der Bogen für den Studienleiter ausgefüllt.

6.3.1. Trainingsphase

In der Trainingsphase wird die Versuchsperson mit der Navigation in der virtuellen Umgebung, den Controllern und deren Funktion vertraut gemacht. Vor dem Anlegen des HMDs wird hierzu die Bezeichnung der Knöpfe und deren Position vermittelt. Daraufhin wird der Versuchsperson das HMD angelegt und dieses justiert bis die Versuchsperson gut sehen kann. Nun werden die Controller aktiviert und entsprechend der Farbe der Laserstrahlen in die linke beziehungsweise rechte Hand gegeben.

In der virtuellen Umgebung angekommen sieht sich die Versuchsperson etwas um und bewegt sich durch den abgesteckten Raum. Des Weiteren geht sie in die Knie oder Hocke, um einen deutlicheren Eindruck der Höhen, Breiten und Tiefen der Umgebung zu bekommen. Infolgedessen visiert die Versuchsperson eine beliebige Box an, um sich, durch Drücken der Griff Taste, Informationen dazu anzeigen zu lassen. Sie lernt das Textfeld, durch Manipulieren des Controllers, ihrem Blickwinkel anzupassen sowie dieses zu löschen, indem nichts anvisiert und die *Griff Taste* erneut betätigt wird. Daraufhin öffnet die Versuchsperson das System mit dem Namen *Pubflow* und visiert die blaue Fläche mit dem Namen *Neo4J* an. Die Versuchsperson erzeugt daraus eine Applikation, indem sie den Trigger zieht. Weiterhin visiert sie eine belibiege Box dieser Applikation an und hält das *Trackpad* gedrückt. Die Versuchsperson hat die Applikation dadurch an den Controller gebunden und kann diese, durch Bewegen des Controllers, manipulieren. Sie verschiebt

[1] http://samoa.informatik.uni-kiel.de:8181/login.html;JSESSIONID=d4f3f132-59d5-44e1-b8c3-c54001a0ea90

und rotiert die Applikation, zoomt diese heran und heraus, indem sie den Controller auf der entsprechenden Achse dreht oder zu sich heranzieht beziehungsweise von sich wegschiebt. Als nächstes lernt die Versuchsperson die Applikation zu löschen, indem sie eine beliebige blaue Fläche anvisiert und den *Trigger* betätigt. Abschließend wird das geöffnete System, durch Anvisieren der zugrundeliegenden grauen Fläche und Betätigen des *Triggers*, wieder geschlossen und die Landschaft damit in den Anfangszustand zurückversetzt. Nachdem die Versuchsperson die Grundlagen der Interaktion gelernt hat, wird nun der Rundkurs in der Hauptphase gestartet.

6.3.2. Hauptphase

In der Hauptphase wendet die Versuchsperson das gelernte Wissen auf die Landschaft an. Zu Beginn bewegt sich die Versuchsperson vor das System mit dem Namen *WDC-MARE* und öffnet dieses. Als nächstes lässt sie sich Informationen zu einer der Applikationen des Systems anzeigen und diese wieder verschwinden. Daraufhin schließt die Versuchsperson das System wieder und sucht das höchste System. Sie öffnet dieses und erzeugt aus der blauen Fläche mit dem Namen *Neo4J* eine Applikation. Die Versuchsperson verschiebt die Applikation nach links, zieht sie zu sich heran und rotiert sie, damit die Beschriftungen der Boxen lesbar werden. Weiterhin öffnet sie eine beliebige Box der Applikation und lässt sich Informationen zu deren Inhalt anzeigen. Im Anschluss drückt sie die Applikation von sich weg und verschiebt sie nach oben rechts, sodass sie sich außerhalb des Blickfelds befindet. Danach löscht die Versuchsperson die Applikation und lässt sich Informationen zu der großen dunkelgrünen Box anzeigen. Abschließend öffnet sie die etwas kleinere dunkelgrüne Box und bringt die Landschaft wieder in den Ursprungszustand, indem sie das geöffnete System schließt.

6.4. Ergebnisse

Die Zeitspanne der Trainings- und Hauptphase hat von Teilnehmer zu Teilnehmer variiert, da diese unterschiedlich schnell waren. Aus diesem Grund ist diese Zeitspanne für die qualitative Bewertung der VR-Visualisierung nicht relevant und wird deshalb vernachlässigt. Den Bewertungseinheiten unseres Bogens (Bogen des Studienleiters) und des VR-Bogens wird ein fester, natürlicher Wert von Null bis Vier zugeordnet. Im Anschluss wird daraus die durchschnittlich vergebene Punktezahl berechnet. Der Wert Vier repräsentiert dabei die höchste (*stimme völlig zu*) und Null (*stimme gar nicht zu*) die geringste Bewertung. Die Auswertung unserer Angaben zum Verlauf des Experiments (Bogen des Studienleiters) ist in Tabelle 6.6 dargestellt.

Die Ergebnisse der Auswertung des allgemeinen Bogens sind in Abschnitt 6.4.1 beschrieben. Die Angaben der Versuchspersonen in den VR-Bögen werden in die Kategorien *Nutzererlebnis*, *Navigation* und *Interaktion* eingeteilt und in Abschnitt 6.4.2, Abschnitt 6.4.3 und Abschnitt 6.4.4 dargestellt.

Tabelle 6.6. Auswertung der Angaben des Studienleiters

ID	Behauptung	Punktzahl (0 - 4)		
		∅	Min	Max
S_1	Die VR-Brille war einfach anzubringen	3.92	3	4
S_2	Der Proband hat das höchste System gefunden	4	4	4

6.4.1. Versuchspersonen

Insgesamt haben zwölf Personen an der Studie teilgenommen, rund die Hälfte davon war uns bereits bekannt. Die Auswertung des allgemeinen Bogens hat ergeben, dass drei Viertel der Teilnehmer, wie im linken Diagramm der Abbildung 6.1 dargestellt, überhaupt nicht oder eher nicht anfällig für Motion Sickness, Reisekrankheit oder Seekrankheit war (D_1). Lediglich eine Versuchsperson war eher und die restlichen mittelmäßig anfällig

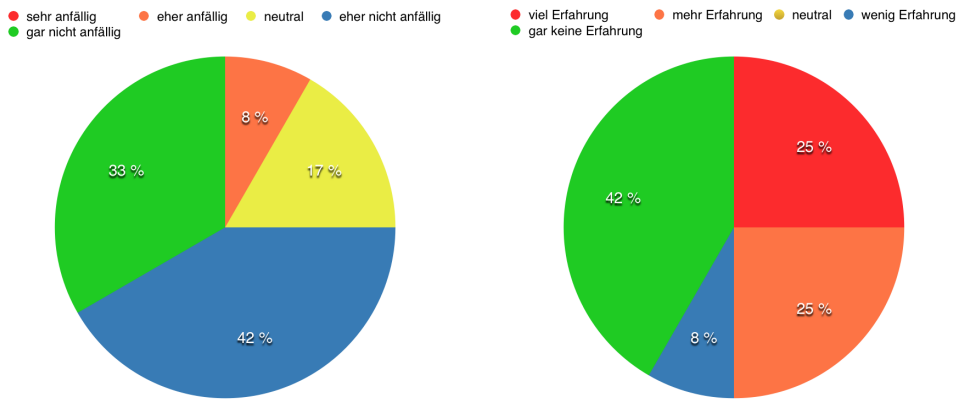

Abbildung 6.1. Auswertung des allgemeinen Bogens: Das linke Diagramm stellt die Anfälligkeit der Versuchspersonen für Motion Sickness (Reisekrankheit, Seekrankheit) dar. Das rechte Diagramm spiegelt die bereits vorhanden Erfahrung mit virtueller Realität wieder

dafür. Aus dem rechten Diagramm der Abbildung 6.1 ist ersichtlich, dass die Hälfte der Versuchspersonen bereits etwas mehr oder viel Erfahrung mit virtueller Realität hatte (D_4). Die andere Hälfte hingegen hatte vorher hauptsächlich überhaupt keine Erfahrung damit gemacht. In Abbildung 6.2 ist links die Sehstärke der Versuchspersonen dargestellt (D_2). Hieraus ist ersichtlich, dass *41 %* der Versuchspersonen *normal* bis *sehr gut* sehen können. Die verbleibenden *59 %* können *schlechter* oder *schlecht* sehen. Hierzu zeigt das rechte Diagramm der Abbildung 6.2, dass *33%* keine, *34 % selten* bis *häufig* und *33% immer* eine Brille tragen (D_3).

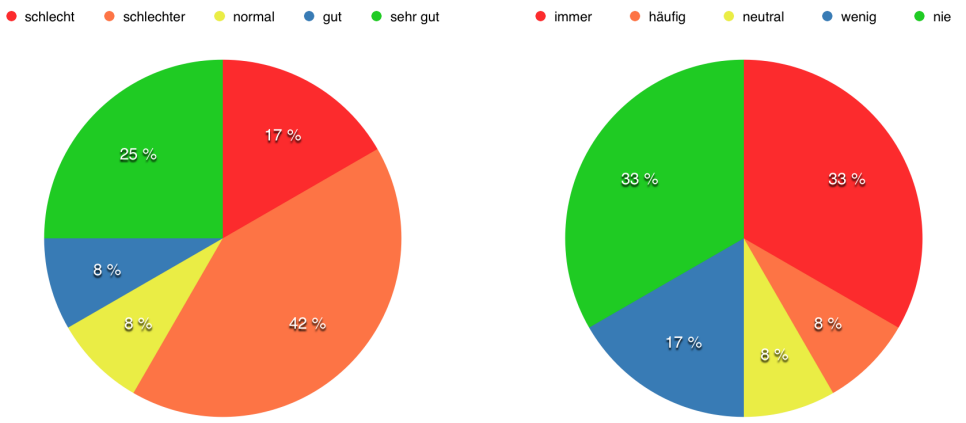

Abbildung 6.2. Auswertung des allgemeinen Bogens: Das linke Diagramm stellt die Sehstärke der Versuchspersonen dar. Das rechte Diagramm veranschaulicht wie oft die Versuchspersonen eine Brille tragen

6.4.2. Nutzererlebnis

Das Nutzererlebnis wird maßgeblich durch den Tragekomfort des HMDs, die Darstellung der virtuellen Umgebung und deren Auswirken auf das Wohlbefinden des Nutzer geprägt. Tabelle 6.7 zeigt die Auswertung der Angaben zum Nutzerempfinden. Dieses wurde durchschnittlich mit *3.56* von *4* Punkten bewertet. Anzumerken ist, dass die Behauptung A_5 minimal mit *1* von *4* Punkten bewertet wurde.

Tabelle 6.7. Auswertung der Angaben zum Nutzererlebnis

ID	**Behauptung**	**Punktzahl (0 - 4)**		
		∅	Min	Max
A_1	Das Erkunden der virtuellen Umgebung war angenehm	3.84	3	4
A_2	Die Darstellung der virtuellen Umgebung war angenehm	3.34	3	4
A_3	Die VR-Brille war komfortabel zu tragen	3	2	4
A_4	Die Rückmeldung der virtuellen Umgebung war direkt und flüssig	3.58	2	4
A_5	Ich war frei von Schwindel und Übelkeit	3.67	1	4
A_6	Ich hätte noch weitere Zeit in der virtuellen Umgebung bleiben können	3.75	3	4
A_7	Ich wüde den VR-Modus wiederverwenden	3.75	3	4
Gesamt		3.56	2.42	4

6.4.3. Navigation

Die Navigation wird durch das HMD und damit durch Körper- und Kopfbewegungen realisiert. Das Umschauen und Fortbewegen in der virtuellen Umgebung ist somit maßgeblich für das Empfinden der Navigation. In Tabelle 6.8 ist die Auswertung der Angaben zur Navigation dargestellt. Diese wurde durchschnittlich mit *3.98* von *4* Punkten bewertet. B_4 ist die einzige Behauptung, die nicht mit voller Punktzahl bewertet wurde.

Tabelle 6.8. Auswertung der Angaben zur Navigation

ID	Behauptung	Punktzahl (0 - 4)		
		∅	Min	Max
B_1	Das Umschauen im virtuellen Raum durch Drehen des Kopfes oder Drehen des Körpers war intuitiv	4	4	4
B_2	Die Annäherung an Objekte durch Daraufzugehen oder durch Annähern des Kopfes war intuitiv	4	4	4
B_3	Die Gewinnung von Abstand zu Objekten durch Wegbewegen war intuitiv	4	4	4
B_4	Die Fortbewegung im virtuellen Raum durch reales Gehen war praktisch	3.92	3	4
B_5	Die Körper- und Kopfbewegungen zur Navigation (umschauen, annähern, distanzieren) waren leicht zu lernen	4	4	4
Gesamt		3.98	3.8	4

6.4.4. Interaktion

Die Interaktion wird mithilfe der Controller und der damit verbundenen Körperbewegungen verwirklicht. Für das Empfinden der Interaktion ist die Praktik der Controller und die daraus resultierende Lernbarkeit maßgeblich. Tabelle 6.9 stellt die Auswertung der Angaben zur Interaktion dar. Die Interaktion wurde durchschnittlich mit *3.37* von *4* Punkten bewertet. Weiterhin wurden die Behauptungen C_1, C_2, C_6, C_7 und C_8 minimal mit *1* von *4* Punkten bewertet. Der Durchschnitt der minimalen Bewertungen liegt deswegen bei *2.11* von *4* Punkten.

Tabelle 6.9. Auswertung der Angaben zur Interaktion

ID	Behauptung	Punktzahl (0 - 4)		
		∅	Min	Max
C_1	Die Interaktion mit der virtuellen Umgebung durch die Controller war intuitiv	2.92	1	4
C_2	Die Bedienung der Controller war leicht zu lernen	2.83	1	4
C_3	Das automatische Markieren von Objekten durch den Strahl des Controllers war praktisch	4	4	4
C_4	Durch die Markierung habe ich schnell verstanden mit welchen Objekten interagiert werden kann	4	4	4
C_5	Das Öffnen und Schließen von Boxen durch den Abzug war praktisch	3.5	3	4
C_6	Das Rotieren, Verschieben und Zoomen eines Objektes durch Gedrückthalten des großen runden Knopfes war praktisch	3.33	1	4
C_7	Beim Rotieren, Verschieben oder Zoomen eines Objektes hatte ich das Gefühl, dieses in der Hand zu halten	2.58	1	4
C_8	Die Darstellung von Informationen in einem Fenster neben dem Controller war praktisch	3.42	1	4
C_9	Ich konnte dieses Fenster ähnlich wie eine Zeitung in die Hand nehmen und meinem Blickwinkel anpassen	3.75	3	4
Gesamt		3.37	2.11	4

6.5. Diskussion der Ergebnisse

In diesem Abschnitt werden die in Abschnitt 6.4 vorgestellten Ergebnisse analysiert und bewertet. Hierfür betrachten wir zuerst die Auswertung der eigenen Angaben zum Experiment und analysieren dazu die Einträge aus Tabelle 6.6. Das Anlegen des HMDs (S_1) war bei allen Versuchspersonen einfach, lediglich in einem Fall saß es nicht sofort ideal und musste nachjustiert werden. Zudem hat jede Versuchsperson auf anhieb das höchste System gefunden (S_2) , sodass die Möglichkeit zur Höhenanpassung der Komponenten, an deren eingehenden Anfragen und somit an deren Wichtigkeit, einen Vorteil gegenüber der 2D-Visualisierung darstellt. Die Hypothese H_1 aus Abschnitt 6.1 gilt damit als bestätigt. Die Bewertungen zu den Kategorien *Nutzererlebnis*, *Navigation* und *Interaktion* geschieht in den entsprechenden Folgekapiteln.

6.5.1. Nutzererlebnis

Zur Analyse der Navigation betrachten wir im Folgenden die einzelnen Einträge der Tabelle 6.7. Das Erkunden der virtuellen Umgebung (A_1) wurde durchschnittlich mit *3.84* von *4* Punkten bewertet und ist damit etwas schlechter als *angenehm* für jeden Benutzer gewesen. Die Darstellung der virtuellen Umgebung (A_2) wurde durchschnittlich mit *3.34* von *4*

Punkten bewertet und ist damit als etwas besser als *eher angenehm* empfunden worden. Die begrenzte Auflösung der HMD oder die leicht unscharfen Beschriftungen könnten die Qualität der Darstllung beeinträchtigt haben. Der Tragekomfort des HMDs (A_3) wurde durchschnittlich mit *3* von *4* Punkten bewertet und entspricht damit der Bewertung *eher komfortabel*. Die geringste Bewertung lag bei *2* von *4* Punkten. Der Grund dafür könnte darin liegen, dass das HMD bei manchen Versuchspersonen leicht auf die Nase gedrückt oder das Kabel beim Bewegen im Raum gestört hat. Anzumerken ist, dass das Kabel nur Teilnehmern als störend aufgefallen ist, die bisher noch keine Erfahrung mit virtueller Realität gemacht haben und diese sich gegebenenfalls erst daran gewöhnen müssen. Die direkte und flüssige Rückmeldung der virtuellen Umgebung (A_4) wurde durchschnittlich mit *3.58* von *4* Punkten beweret und liegt damit zwischen *flüssig und direkt* und *eher flüssig und direkt*. Das Minimum liegt hier bei *2* von *4* Punkten und könnte daraus resultieren, dass der Internetbrowser zeitweise langsamer war oder die Sensoren Schwierigkeiten beim Verfolgen des HMDs hatten. Weiterhin kann es dazu kommen sein, dass die Rückmeldung, durch minimale Ladezeiten beim Öffnen und Schließen der Komponenten, leicht verzögert war. Die Abwesenheit von Schwindel und Übelkeit (A_5) wurde durchschnittlich mit *3.67* von *4* Punkten bewertet und ist damit etwas schlechter als *frei von Schwindel und Übelkeit*. Eine Versuchsperson hat diesen Punkt mit *eher nicht frei von Schwindel und Übelkeit* bewertet. Die Ursache dafür liegt darin, dass diese *eher anfällig* für *Motion Sickness* war. Die Angaben, ob die Versuchsperson noch weiter in der virtuellen Umgebung hätte verbleiben können (A_6) und diese wiederverwenden (A_7) würde, wurden beide mit *3.75* von *4* Punkten und damit etwas schlechter als *stimme völlig zu* bewertet.

Insgesamt wurde das gesamte Nutzerempfinden durchschnittlich mit *3.56* von *4* Punkten bewertet und liegt damit zwischen *gut* und *sehr gut*. Folglich haben die in Abschnitt 6.4.1 dargestellten Unterschiede der Versuchspersonen, im Hinblick auf Sehschärfe oder Tragen einer Brille, keinen Einfluss auf das *Nutzerempfinden*. Fehlende Vorkenntnisse in virtueller Realität haben in unserem Fall dazu geführt, dass ein Viertel der Teilnehmer das Kabel als störend empfand. Somit beeinflussen dieses und die Anfälligkeit für Motion Sickness das *Nutzerempfinden* leicht. Zusammenfassend gilt die Hypothese H_2 aus Abschnitt 6.1 damit als bestätigt.

6.5.2. Navigation

Zur Analyse der Navigation betrachten wir im Folgenden die einzelnen Einträge der Tabelle 6.8. und gliedern diese in Intuitivität (B_1 - B_3), Praktik (B_4) und Lernbarkeit (B_5). Das Umschauen im virtuellen Raum durch Drehen des Kopfes oder Drehen des Körpers (B_1) wurde durchschnittlich mit *4* von *4* Punkten bewertet und ist damit *intuitiv*. Die Annäherung an Objekte durch Daraufzugehen oder durch Annähern des Kopfes (B_2) wurde durchschnittlich mit *4* von *4* Punkten bewertet und ist somit ebenfalls *intuitiv*. Zudem wurde die Gewinnung von Abstand zu Objekten durch Wegbewegen (B_3) mit *4* von *4* Punkten bewertet und wird damit gleichermßen als *intuitiv* eingestuft. Die Fortbewegung

im virtuellen Raum durch reales Gehen (B_4) wurde durchschnittlich mit *3.92* von *4* Punkten bewertet und ist damit leicht schlechter als *praktisch* und viel besser als *eher praktisch*. Die Lernbarkeit von Körper- und Kopfbewegungen zur Navigation (umschauen, annähern, distanzieren) (B_5) wurde mit *4* von *4* Punkten und damit als *leicht* bewertet.

Insgesamt wurde die Navigation durchschnittlich mit *3.98* von *4* Punkten und resultierend fast mit *sehr gut* bewertet. Demnach haben die in Abschnitt 6.4.1 dargestellten Unterschiede der Versuchspersonen, im Hinblick auf Sehschärfe, Tragen einer Brille, Vorkenntnissen in virtueller Realität oder Anfälligkeit für Motion Sickness, keinen Einfluss auf die *Navigation*. Die Hypothese H_3 aus Abschnitt 6.1 gilt damit als bestätigt.

6.5.3. Interaktion

Zur Analyse der Interaktion betrachten wir im Folgenden die einzelnen Einträge der Tabelle 6.9 und gliedern diese ebenfalls in Intuitivität (C_1, C_7, C_9), Praktik (C_3, C_5, C_6, C_8) und Lernbarkeit (C_2, C_4). Die Interaktion mit der virtuellen Umgebung durch die Controller (C_1) wurde durchschnittlich mit *2.92* von *4* Punkten bewertet und ist damit etwas schlechter als *eher intuitiv*. Die Bedienung der Controller (C_2) wurde durchschnittlich mit *2.83* von *4* Punkten bewertet und wird damit ebenfalls etwas schlechter als *eher leicht zu lernen* eingestuft. Die geringste Bewertung beider Angaben war *1* von *4*, sodass diese damit als *eher nicht intuitiv* und *eher nicht leicht zu lernen* klassifiziert wurden. Die Sichtung der Videoaufnahmen und Auswertung des Bogens des Studienleiters hat gezeigt, dass das Löschen der Applikation bei einem Drittel der Versuchspersonen zu Verwirrung geführt und diese aus dem Konzept gebracht hat. Die darauffolgende Neueinweisung des Teilnehmers könnte zur geringen Bewertung der beiden Punkte geführt haben. Das automatische Markieren von Objekten durch den Controllerstrahl (C_3) wurde durchschnittlich mit voller Punktzahl bewertet und wird damit als *praktisch* eingestuft. Die Markierung (C_4) selbst wurde ebenfalls durchschnittlich mit *4* von *4* Punkten bewertet und eignet sich *ideal*, um zu vermitteln mit welchen Objekten interagiert werden kann. Das Öffnen und Schließen der Boxen durch den Abzug (C_5) wurde durschnittlich mit *3.5* von *4* Punkten bewertet und liegt damit zwischen *praktisch* und *eher praktisch*. Das Rotieren, Verschieben und Zoomen eines Objektes durch das Trackpad (C_6) wurde durchschnittlich mit *3.33* von *4* bewertet und ist damit etwas besser als *eher praktisch*. Die niedrigste Bewertung hierfür war *1* von *4* Punkten und damit *eher nicht praktisch*. Möglicherweise wäre die Verwendung des *Triggers* an dieser Stelle sinnvoller gewesen, da dessen Betätigung mehr einer Greifbewegung ähnelt. Das Gefühl dieses dabei in der Hand zu halten (C_7) wurde durchschnittlich mit *2.58* und minimal mit *1* von *4* Punkten bewertet. Durchschnittlich wird das Gefühl damit zwischen *neutral* und *gut* sowie minimal mit *mäßig* eingestuft. Wahrscheinlich ist das Objekt zu weit von der Hand des Benutzers entfernt, um jedem das Gefühl zu vermitteln, dieses in der Hand zu halten. Zur Verbesserung könnte hier die Textur des Controllers, ähnlich wie in *Steam VR*, mit der des gebundenen Objekts ersetzt werden. Die Darstellung von Informationen in einem Fenster neben dem Controller (C_8) wurde durchschnittlich mit *3.42*

und damit etwas besser als *eher praktisch* bewertet. Die Handhabung dieses Fensters (C_9) wurde durchschnittlich mit *3.75* von *4* Punkten bewertet. Das Fenster ähnelt damit *gut* bis *sehr gut* einer Zeitung, die in die Hand genommen und dem Blickwinkel angepasst werden kann.

Insgesamt wurde die Interaktion durchschnittlich mit *3.37* von *4* Punkten und damit etwas besser als *gut* bewertet. Die Hypothese H_4 aus Abschnitt 6.1 kann damit als bestätigt angesehen werden.

6.6. Verbesserungsvorschläge

Alle zwölf Probanden hatten die Möglichkeit Anmerkungen und Verbesserungsvorschläge auf zwei extra Feldern des Ankreuzbogens zu hinterlassen. Abbildung 6.3 bildet die Äußerungen relativ zur deren Häufigkeit ab.

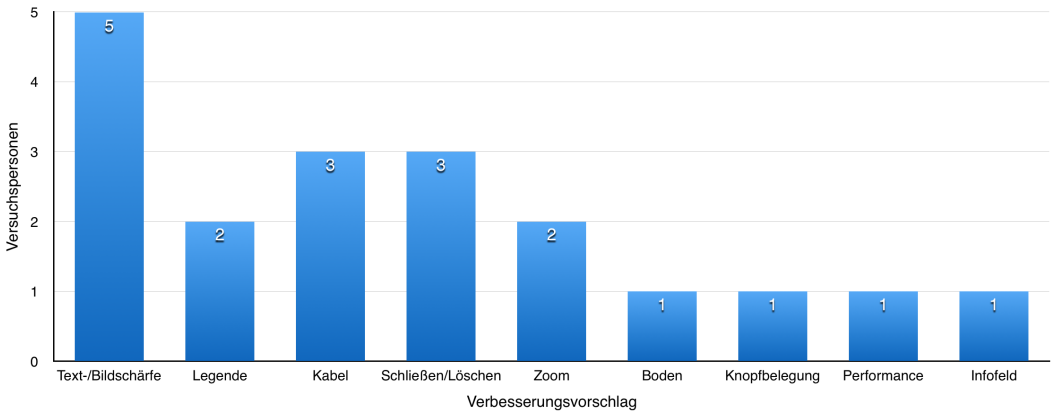

Abbildung 6.3. Aufkommnen der einzelnen Verbesserungsvorschläge

Ein Viertel der Probanden würde eine höhere Textschärfe begrüßen, da die Beschriftungen trotz Annäherung nicht gut zu lesen waren. Der Grund dafür liegt darin, dass das neue Verfahren zur Erzeugung der Beschriftungen enorm viel Rechenleistung spart, diese aber optisch weniger scharf erscheinen lässt. Zur Bewältigung dieses Problems könnte das Befüllen des Canvas, welches wahrscheinlich Optimierungsspielraum bietet, oder ein alternativer Ansatz zur Realisierung der Beschriftungen betrachtet werden. Des Weiteren haben zwei von zwölf Personen die allgemeine Darstellungsschärfe bemängelt. Hierfür ist anzumerken, dass die *HTC VIVE* eine deutlich bessere Auflösung als die Vorgängerversionen hat, sich aber noch nicht am Rande des möglichen Leistungspektrums befindet, sodass dies durch eine leistungsstärkere Nachfolgeversion verbessert werden könnte.

Ein Viertel der Versuchspersonen empfand das Kabel beim Bewegen durch den Raum

als störend. Eine geäußerte Idee zur Bewältigung dieses Problems besteht darin das Kabel an der Decke zu befestigen, sodass dieses nicht mehr direkt im Weg liegt. Alternativ könnte hier eine kabellose Nachfolgeversion der *HTC VIVE* Abhilfe schaffen. Des Weiteren hätten sich zwei Teilnehmer eine virtuelle Legende mit der Controllerbelegung gewünscht, sodass Funktionalitäten hätten nachgeschlagen werden können. Ein weiterer Teilnehmer konnte sich die Knopfbelegung schlecht merken und hätte sich ein anderes System zum Schließen und Löschen der Objekte gewünscht. Möglicherweise würde hier die vorgeschlagene Legende bereits Abhilfe schaffen. Ergänzend hätten sich zwei Teilnehmer zum Löschen der Applikation ein großes rotes X direkt darüber gewünscht.

Eine Versuchsperson empfand den Zoom als zu langsam und eine weitere als unpraktisch. Die Idee, den Controller durch die daran gebundene Applikation zu ersetzen, könnte dieses Problem lösen. Die Applikation befindet sich dadurch viel näher am Benutzer und somit reagieren der Zoom sowie das Verschieben empfindlicher. Weiterhin empfand eine Versuchsperson die Fußbodentextur als unangenehm. Eine weitere Person findet das Vertauschen der Funktionen des *Trackpads* und des *Triggers* sinnvoll, da durch das Ziehen des *Triggers* eher ein Greifgefühl aufkommt als beim Drücken des *Trackpads*. Eine Versuchsperson hätte ein Informationsfeld bevorzugt, das nicht dem Controller, sondern der Landschaft angefügt wird und somit nicht beweglich ist. Zudem hat sich ein Teilnehmer eine Leistungssteigerung gewüscht, da er die Ladezeit als ungewohnt empfand.

6.7. Gefährdung der Validität

Die Folgenden Abschnitte erläutern mögliche Faktoren, welche die Validität der Studie bedrohen könnten.

6.7.1. Anzahl der Versuchspersonen

An der Studie haben zwölf Personen teilgenommen, weshalb diese nur bedingt aussagekräftig ist. Die gewonnen Erkenntnisse sollten somit als Richtungsweiser beziehungsweise Trend interpretiert und durch eine größere Studie bestätigt werden.

6.7.2. Gestaltung des Ankreuzteils

Der Ankreuzteil des allgemeinen Bogens und des VR-Bogens ist durchgehend positiv formuliert, damit der Proband bei der Wahl der Kategorie nicht umdenken muss und durcheinander kommt. Hierdurch wird die Fehlerquote hinsichtlich Falschangaben minimiert. Es ist jedoch möglich, dass der Proband dadurch beeinflusst wurde und dieser somit positiver bewertet hat. Für den Bogen des Studienleiters ist dies weniger wichtig, da dieser lediglich zwei Behauptungen enthält.

6.8. Zusammenfassung

Die Hypothese zur Räumlichkeit (H_1) konnte zu *100%* bestätigt werden, da jeder Teilnehmer das höchste System gefunden hat und somit zusätzliche Informationen dargestellt werden können. In Abbildung 6.4 ist ein Überblick der durchschnittlichen Bewertung des Nutzerempfindens, der Navigation und Interaktion dargestellt. Dabei wurde die verwendete Punktskala relativiert, sodass prozentuale Aussagen besser nachvollziehbar sind. Die

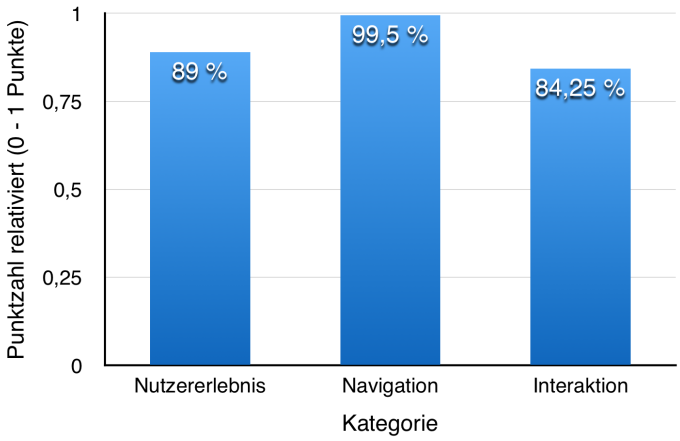

Abbildung 6.4. Durchschnittliche Bewertung der einzelnen Kategorien

Hypothese zum Nutzererlebnis (H_2) durch eine erreichte Punktzahl von *89%* bestätigt werden. Des Weiteren konnte ebenfalls die Hypothese zur Navigation (H_3) durch eine erreichte Punktzahl von *99,5%* und die Hypothese zur Interaktion (H_4) durch eine erreichte Punktzahl von *84,25%* bestätigt werden. Zur Beantwortung des ersten Teils der Forschungsfrage fassen wir zusammen, dass die VR-Visualisierung den Nutzer nicht beeinträchtigt, dieser die Navigation sowie Interaktion und damit den gesamten VR-Modus gut annimmt.

Abbildung 6.4 zeigt, dass die Navigation besonders gut angenommen wurde und sehr intuitiv ist. Das Nutzererlebnis und die Interaktion sind noch nicht optimal gestaltet und bieten noch Verbesserungsspielraum. Die zusätzliche Dimension einer 3D-Visualisierung bietet viele Möglichkeiten zusätzliche Informationen darzustellen. Dies kann beispielsweise durch die Anpassung der Höhe, Tiefe oder Breite der einzelnen Komponenten modelliert werden. Die Auswertung unseres Bogens in Tabelle 6.6 hat gezeigt, dass jeder Teilnehmer sofort in der Lage war die Höhenunterschiede der einzelnen Komponenten wahrzunehmen und das höchste System zu finden (S_2). Der Benutzer kann somit die Vorteile der 3D-Visualisierung ideal mit der Navigation im virtuellen Raum kombinieren. Folglich stellt die Kombination aus 3D-Visualisierung, intuitiver Navigation sowie recht praktischer Interaktion eine intuitive und praktische Alternative zum Erkunden am Bildschirm dar.

Kapitel 7

Fazit und Ausblick

Dieses Kapitel beinhaltet eine Zusammenfassung der Arbeit, der Studienergebnisse und einen Ausblick auf mögliche Verbesserungen und Ergänzungen der VR-Visualisierung.

7.1. Fazit

In dieser Bachelorarbeit wurde ein Plugin entwickelt, das *ExplorViz* um eine Visualisierung in virtueller Realität erweitert und dadurch die Landschafts- und Applikationsansicht vereint. Als Grundlage dafür wurde die Landschaftsansicht um eine Dimension erweitert, dreidimensional dargestellt und mit der Applikationsansicht kombiniert. Infolgedessen ist die Erzeugung einer dreidimensionalen Applikation ohne das Verlassen der Landschaftsansicht möglich. Resultierend kann eine überwachte Software vollständig in virtueller Realität erkundet werden. Dies erlaubt von Beginn an die Nutzung der Navigations- und Interaktionsmöglichkeiten der virtuellen Realität. Mithilfe einer angepassten Fußbodengeometrie und -textur, wird dem Benutzer zudem ein Bezug zum realen Raum vermittelt. Im Rahmen der Entwicklung haben wir zunächst verwendbare Anѕatze betrachtet und diese bei der Erstellung des Entwurfs miteinfließen lassen. Weiterhin haben wir die dort festgelegten Funktionalitäten implementiert und durch eine Studie evaluiert.

In der Studie konnten wir nachweisen, dass sich die Verwendung der VR-Visualisierung weitestgehend positiv auf das Wohlbefinden des Benutzers auswirkt. Des Weiteren haben wir gezeigt, dass die Navigation in der virtuellen Umgebung sehr intuitiv ist und gut angenommen wird. Wir konnten weiterhin darlegen, dass die Interaktion durch die Controller überwiegend praktisch empfunden und ebenfalls gut angenommen wird. Die zusätzliche Dimension der 3D-Visualisierung konnte genutzt werden, um ergänzende Informationen durch die Höhe der Komponenten darzustellen. Mithilfe der intuitiven Navigation im virtuellen Raum, konnte der Benutzer diese Informationen schnell zur Kenntnis nehmen. Zusammenfassend stellt das Erkunden einer Software mit *ExplorViz* in virtueller Realität somit eine praktische Alternative zum Erkunden am Bildschirm dar.

7.2. Ausblick

Während der Entwicklung des Plugins sind neue Ideen und Lösungsansätze entstanden, welche das Plugin verbessern könnten. Zusammen mit den Verbesserungsvorschlägen aus der Evaluierung werden wir diese im Folgenden vorstellen und erläutern.

Die aktuelle Realisierung der Kommunikationslinien führt zu Einbrüchen in der Bildrate, weshalb diese momentan ausgeblendet werden. Die Berechnung dieser Linien muss von Grund auf neu gestaltet und dabei effizienter berechnet werden. Mögliche Ansätze könnten auf die quantitative Reduzierung der Kommunikationslinien, auf eine effizientere Berechnung der Kommunikationspunkte oder auf eine Optimierung im *Layouting* der Kommunikationslinien abzielen.

Die Erzeugung einer 3D-Applikation aus einer geschlossenen Applikation ist nur möglich, wenn die entsprechenden Daten vom *Backend* bereitgestellt werden. Dies gilt ebenso für die Erzeugung der darzustellende Landschaft. Sofern keine Daten vorliegen, wird derzeit, abhängig vom verwendeten Internetbrowser, eine entsprechende Fehlermeldung über dem *Canvas* oder in einem kleinen Fenster vor dem *Canvas* eingeblendet. Wie in Abschnitt 4.1 bereits erwähnt, sind diese Meldungen nicht in der virtuellen Umgebung sichtbar, da sie sich nicht direkt im *Canvas* befinden. Als mögliche Realisierung wäre eine Darstellung der Fehlermeldungen, ähnlich zu den Informationstexten neben den Controllern, denkbar.

Ein Teleporter könnte das Plugin um eine zusätzliche Fortbewegungsmethode erweitern und damit die Gebrauchstauglichkeit der VR-Visualisierung stark verbessern. Besonders vorteilhaft wäre der Teleporter beim Erkunden sehr großer Softwarelandschaften, da die virtuelle Fortbewegung momentan vom realen abgesteckten Bereich beschränkt wird. Wir haben bisher auf die Umsetzung der *Oculus Rift* Controller verzichtet, da das Raumkonzept mit dieser nur beschränkt nutzbar ist. Der Teleporter könnte insbesondere die Nutzbarkeit der jetzigen virtuellen Fläche mit der *Oculus Rift* ermöglichen. Infolgedessen könnte dies eine Implementierung der *Oculus Rift Controller* interessant machen.

Die kollaborative Nutzbarkeit von *ExplorViz* in einer virtuellen Umgebung bietet nach wie vor großes Potenzial und könnte ein gemeinsames, standortunabhängiges Erkunden von Software ermöglichen. In Abschnitt 4.2 ist ein möglicher Ansatz zur Realisierung dieser kollaborativen Nutzbarkeit beschrieben und könnte in zukünftigen Arbeiten aufgegriffen werden.

Die Verbesserungsvorschlälge der Studienteilnehmer aus Abschnitt 6.6 könnten die VR-Visualisierung zusätzlich verbessern. Ein Verbesserungsvorschlag zum Löschen beziehungsweise Entfernen der 3D-Applikation aus der Szene sieht ein großes rotes *X* darüber vor, das anvisiert und betätigt werden kann. Möglicherweise könnte eine Fläche mit einer entsprechenden Textur versehen und ähnlich wie die Grundplatte der 3D-Applikation

hinzugefügt werden. Alternativ könnten statt der Fläche auch zwei rote Rechtecke gekreuzt und als Gruppe der 3D-Applikation hinzugefügt werden. Damit der Benutzer beim Betrachten der 3D-Applikation das Gefühl hat, diese wirklich in der Hand zu halten, könnte die Controllertextur durch die 3D-Applikation ersetzt werden. Hierdurch hätte diese fast dieselbe Position wie Hand des Benutzers. Abhängig von der Größe der 3D-Applikation müsste diese dabei skaliert werden. Des Weiteren könnte das Halten der 3D-Applikation in der Hand mithilfe des *Triggers* und gleichzeitige Berührung durch den Controller erfolgen. Es ist vorstellbar, dass die Boxen der 3D-Applikation leicht heller oder umrandet dargestellt werden, sobald der Controller eine definierte Distanz unterschreitet.

Ein weiterer Verbesserungsvorschlag bezieht sich auf die Beschriftungen, da deren Schärfe stark abhängig vom Blickwinkel ist. Ein möglicher Ansatz könnte in der Optimierung der Befüllung des *Canvas* bestehen. Alternativ könnte eine Umsetzung mit der Klasse *THREE.CSS3DObject* möglich sein. Diese ergänzt den Typ *THREE.Object3D* um ein zusätzliches Attribut für DOM-Elemente. Hierdurch können HTML-Dokumente direkt im *Canvas* dargestellt werden. Eine Abwägung zwischen Mehrwert und Performance sollte vorher jedoch angestellt werden. Die weitere Analyse des Feedbacks hat ergeben, dass eine Legende mit den grundlegenden Funktionalitäten der Controller von Vorteil gewesen wäre. Die Klasse *THREE.CSS3DObject* könnte sich ebenfalls gut für die Realisierung dieser Legende eigenen. Alternativ könnte die Umsetzung durch die Befüllung eines *Canvas* erfolgen.

Abschließend sollten die in dieser Arbeit gewonnen Erkenntnisse durch eine Studie größeren Umfangs verifiziert werden.

Literaturverzeichnis

[Alimadadi et al. 2016] S. Alimadadi, A. Mesbah und K. Pattabiraman. Understanding Asynchronous Interactions in Full-Stack JavaScript. In: *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. Mai 2016, Seiten 1169–1180. (Siehe Seite 11)

[Bozgeyikli et al. 2016] E. Bozgeyikli, A. Raij, S. Katkoori und R. Dubey. Point & Teleport Locomotion Technique for Virtual Reality. In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. CHI PLAY '16. Austin, Texas, USA: ACM, 2016, Seiten 205–216. URL: http://doi.acm.org/10.1145/2967934.2968105. (Siehe Seite 1)

[Brill 2009] M. Brill. Virtuelle Realität. 1. Springer Berlin Heidelberg, 2009. (Siehe Seiten 1 und 7)

[Crespo et al. 2015] R. Crespo, R. García und S. Quiroz. Virtual reality simulator for robotics learning. In: *2015 International Conference on Interactive Collaborative and Blended Learning (ICBL)*. 2015, Seiten 61–65. (Siehe Seite 7)

[Dempsey 2016] P. Dempsey. The teardown: HTC Vive VR headset. *Engineering Technology* 11.7-8 (Aug. 2016), Seiten 80–81. (Siehe Seite 8)

[Desai et al. 2014] P. R. Desai, P. N. Desai, K. D. Ajmera und K. Mehta. A Review Paper on Oculus Rift-A Virtual Reality Headset. *CoRR* abs/1408.1173 (2014). URL: http://arxiv.org/abs/1408.1173. (Siehe Seiten 7 und 20)

[Dey et al. 2016] U. Dey, P. K. Jana und C. S. Kumar. Modeling and Kinematic Analysis of Industrial Robots in WebGL Interface. In: *2016 IEEE Eighth International Conference on Technology for Education (T4E)*. 2016, Seiten 256–257. (Siehe Seite 12)

[Elliott et al. 2015] A. Elliott, B. Peiris und C. Parnin. Virtual Reality in Software Engineering: Affordances, Applications, and Challenges. In: *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. ICSE '15. Florence, Italy: IEEE Press, 2015, Seiten 547–550. URL: http://dl.acm.org/citation.cfm?id=2819009.2819098. (Siehe Seite 2)

[Fittkau et al. 2015a] F. Fittkau, A. Krause und W. Hasselbring. Exploring Software Cities in Virtual Reality. In: *IEEE 3rd Working Conference on Software Visualization (VISSOFT 2015)*. IEEE, 2015, Seiten 130–134. URL: http://eprints.uni-kiel.de/29388/. (Siehe Seiten 3, 4 und 20)

[Fittkau et al. 2015b] F. Fittkau, S. Roth und W. Hasselbring. ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes. In: *23rd European Conference on Information Systems (ECIS 2015)*. 2015. URL: http://eprints.uni-kiel.de/28067/. (Siehe Seiten 2, 8 und 19)

[Fittkau et al. 2017] F. Fittkau, A. Krause und W. Hasselbring. Software landscape and application visualization for system comprehension with ExplorViz. *Information and Software Technology* 87 (2017), Seiten 259 –277. URL: http://www.sciencedirect.com/science/article/pii/S0950584916301185. (Siehe Seiten 2, 8 und 19)

[Gallaba et al. 2015] K. Gallaba, A. Mesbah und I. Beschastnikh. Don't Call Us, We'll Call You: Characterizing Callbacks in Javascript. In: *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Okt. 2015, Seiten 1–10. (Siehe Seite 10)

[Huang et al. 2015] Y. Huang, L. Churches und B. Reilly. A Case Study on Virtual Reality American Football Training. In: *Proceedings of the 2015 Virtual Reality International Conference*. VRIC '15. Laval, France: ACM, 2015, 6:1–6:5. URL: http://doi.acm.org/10.1145/2806173.2806178. (Siehe Seiten 18, 19)

[Krause 2015] A. Krause. Erkundung von Softwarestädten mithilfe der virtuellen Realität. Bachelor thesis. Kiel University, 2015. URL: http://eprints.uni-kiel.de/29837/. (Siehe Seiten 8, 9)

[Kyriakou et al. 2015] K. I. D. Kyriakou, I. K. Chaniotis und N. D. Tselikas. The GPM meta-transcompiler: Harmonizing JavaScript-oriented Web development with the upcoming ECMAScript 6 "Harmony" specification. In: *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. Jan. 2015, Seiten 176–181. (Siehe Seiten 10–12)

[Larson et al. 2014] D. Larson, J. Liu und Y. Zuo. Analyzing the Vulnerabilities in GWT Code and Applications. In: *2014 Second International Symposium on Computing and Networking*. Dez. 2014, Seiten 525–530. (Siehe Seiten 10, 11)

[Li et al. 2014] B. Li, R. Zhang und S. Kuhl. Minication Affects Action-based Distance Judgments in Oculus Rift HMDs. In: *Proceedings of the ACM Symposium on Applied Perception*. SAP '14. Vancouver, British Columbia, Canada: ACM, 2014, Seiten 91–94. URL: http://doi.acm.org/10.1145/2628257.2628273. (Siehe Seite 20)

[Li et al. 2017] C. Li, W. Liang, C. Quigley, Y. Zhao und L. F. Yu. Earthquake Safety Training through Virtual Drills. *IEEE Transactions on Visualization and Computer Graphics* 23.4 (2017), Seiten 1275–1284. (Siehe Seiten 1, 15 und 20)

[Li et al. 2016] P. Li, X. Yu und J. Wang. Progressive compression and transmission of 3D model with WebGL. In: *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. 2016, Seiten 170–173. (Siehe Seite 10)

[Pérez Fernández und Alonso 2015] R. Pérez Fernández und V. Alonso. Virtual Reality in a Shipbuilding Environment. *Adv. Eng. Softw.* 81.C (März 2015), Seiten 30–40. URL: http://dx.doi.org/10.1016/j.advengsoft.2014.11.001. (Siehe Seiten 1 und 7)

[Potter et al. 2013] L. E. Potter, J. Araullo und L. Carter. The Leap Motion Controller: A View on Sign Language. In: *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*. OzCHI '13. Adelaide, Australia: ACM, 2013, Seiten 175–178. URL: http://doi.acm.org/10.1145/2541016.2541072. (Siehe Seite 20)

[Pradel et al. 2015] M. Pradel, P. Schuh und K. Sen. TypeDevil: Dynamic Type Inconsistency Analysis for JavaScript. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Band 1. Mai 2015, Seiten 314–324. (Siehe Seite 10)

[Roth et al. 2016] D. Roth, J. L. Lugrin, D. Galakhov, A. Hofmann, G. Bente, M. E. Latoschik und A. Fuhrmann. Avatar realism and social interaction quality in virtual reality. In: *2016 IEEE Virtual Reality (VR)*. 2016, Seiten 277–278. (Siehe Seiten 16, 18 und 20)

[Schepper et al. 2015] T. D. Schepper, B. Braem und S. Latre. A virtual reality-based multi-player game using fine-grained localization. In: *2015 Global Information Infrastructure and Networking Symposium (GIIS)*. 2015, Seiten 1–6. (Siehe Seiten 15–17, 20–22)

[Sharma et al. 2014] S. Sharma, S. Jerripothula, S. Mackey und O. Soumare. Immersive virtual reality environment of a subway evacuation on a cloud for disaster preparedness and response training. In: *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI)*. 2014, Seiten 1–6. (Siehe Seiten 1, 18–21)

[Zhang und sho Chen 2005] B. Zhang und Y. sho Chen. Enhancing UML Conceptual Modeling through the Use of Virtual Reality. In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. 2005, 11b–11b. (Siehe Seite 2)

[Zhou et al. 2016] R. Zhou, M. Huang, S. Tan, L. Zhang, D. Chen, J. Wu, T. Yue, X. Cao und Z. Ma. Modeling the impact of spatial resolutions on perceptual quality of immersive image/video. In: *2016 International Conference on 3D Imaging (IC3D)*. Dez. 2016, Seiten 1–6. (Siehe Seite 8)

[Zirkelbach 2017] C. Zirkelbach. Juggling with Data: On the Lack of Database Monitoring in Long-Living Software Systems. In: *4th Collaborative Workshop on Evolution and Maintenance of Long-Living Software Systems (EMLS)*. Softwaretechnik-Trends 2. 2017, Seiten 62–65. URL: http://eprints.uni-kiel.de/37440/. (Siehe Seite 2)

[Zirkelbach et al. 2015] C. Zirkelbach, W. Hasselbring und L. Carr. Combining Kieker with Gephi for Performance Analysis and Interactive Trace Visualization. In: *Symposium on Software Performance 2015: Joint Developer and Community Meeting of Descartes/Kieker/Palladio*. Band 35. 3. Softwaretechnik-Trends, 2015, Seiten 26–28. URL: http://eprints.uni-kiel.de/30101/. (Siehe Seite 2)

Sehr geehrter Proband/sehr geehrte Probandin,

wir bedanken uns im Vorfeld der Studie für Ihre Teilnahme und möchten Ihnen ein paar allgemeine Informationen geben.
Diese Studie ist anonym und es werden keine persönlichen Informationen nach außen geliefert. Uns ist weiterhin wichtig, dass Sie zur Kenntnis nehmen, dass wir Sie bitten unser Produkt zu bewerten. Aus diesem Grund haben wir uns im Vorfeld gegen Laufzeit-Aufnahmen von Ihnen (z.B. Bild und Ton) entschieden. Wir zeichnen lediglich den Bildschirm auf, der keine Verbindung zu Ihrer Person ermöglicht. Wir werden Sie bitten bestimmte Aufgaben zu lösen und uns am Ende einen Fragebogen zu beantworten.

Im Laufe dieser Studie werden Sie mit zwei Technologien in Berührung kommen: Der VR-Brille und dem Brain-Computer-Interface.
Bei der VR-Brille handelt es sich um eine Brille, die dem Nutzer einen 3-dimensionalen Raum aufzeigt, in dem der Nutzer interagieren, sich wie gewohnt umschauen und bewegen kann. Der Fußboden dieses Raums entspricht dem realen Raum, der für Bewegungen zur Verfügung steht und sollte deshalb nicht verlassen werden. In dieser Studie wird Ihnen in diesem Raum eine Darstellungsform einer Software (oder auch Computerprogramm) gezeigt. Wir werden diese als Landschaft bezeichnen. Es ist kein Vorwissen bzgl. eines Programms gefordert. Wir werden lediglich geometrische Fragen stellen. Die dargestellte Landschaft besteht aus Systemen, die als graue Boxen dargestellt sind. Diese Boxen enthalten Teilsysteme, die durch Öffnen dieser Boxen dargestellt werden. Eine geöffnete Box wird als Fläche dargestellt, auf der sich die Teilsysteme befinden. Die Teilsysteme werden als grüne Boxen oder blaue Flächen dargestellt. Aus den blauen Flächen können besondere Teilsysteme erzeugt werden. Diese werden als Applikationen bezeichnet und bestehen ebenfalls aus Boxen und Flächen. Die Interaktion mit den Boxen und Flächen erfolgt durch zwei Controller. Diese verfügen im virtuellen Raum über einen Strahl, der wie ein Laserpointer funktioniert. Mit dem Strahl anvisierte Boxen und Flächen werden rot markiert, wenn mit ihnen interagiert werden kann. Zu jedem anvisierten Objekt der Landschaft oder Applikation kann ein Informationstext angezeigt werden.
Das Brain-Computer-Interface (BCI) ist ein neuro-technisches Stirnband, welches die Spannung auf der Kopfoberfläche misst und darauf reagiert. Dadurch ist das Gerät begrenzt in der Lage bestimmte Gedanken wieder zu erkennen und darauf zu reagieren. Nach einer kurzen Trainingsphase sollen Sie das BCI nutzen, um ein paar simple Aufgaben zu lösen.

66

Allgemeine Angaben zur Person

ID: ___
Studiengang: _____, Fachsemester _____
Lerntyp: _____
Lernen durch Schreiben, Lesen etc.
Abschluss : _____
Bei bereits abgeschlossenem Studium

	stimme völlig zu	stimme eher zu	neutral	stimme eher nicht zu	stimme gar nicht zu
Ich fühle mich fit und gesund	☐	☐	☐	☐	☐
Ich bin anfällig für Motion Sickness (Reisekrankheit, Seekrankheit, etc.)	☐	☐	☐	☐	☐
Ich habe eine Sehschwäche	☐	☐	☐	☐	☐
Ich trage eine Brille	☐	☐	☐	☐	☐
Ich lerne schnell	☐	☐	☐	☐	☐
Ich werde leicht nervös	☐	☐	☐	☐	☐
Ich bin aufgeregt	☐	☐	☐	☐	☐
Ich kann mich gut konzentrieren	☐	☐	☐	☐	☐
Ich stehe unter dem Einfluss von Koffein	☐	☐	☐	☐	☐
Ich stehe unter dem Einfluss von Nikotin	☐	☐	☐	☐	☐
Ich stehe unter dem Einfluss von Medikamenten	☐	☐	☐	☐	☐
Ich habe bereits Erfahrung mit virtueller Realität gemacht	☐	☐	☐	☐	☐
Ich habe bereits Erfahrung mit Brain-Computer-Interfaces gemacht	☐	☐	☐	☐	☐

Notizen zum Ablauf (VR)

ID: ___

	stimme völlig zu	stimme eher zu	neutral	stimme eher nicht zu	stimme gar nicht zu
Die VR-Brille war einfach anzubringen	☐	☐	☐	☐	☐
Der Proband hat das höchste System gefunden	☐	☐	☐	☐	☐

Allgemeine Bemerkungen:

Verlauf:

Besonderheiten:

Virtuelle Realität (VR)

	stimme völlig zu	stimme eher zu	neutral	stimme eher nicht zu	stimme gar nicht zu
Das Erkunden der virtuellen Umgebung war angenehm	☐	☐	☐	☐	☐
Die Darstellung der virtuellen Umgebung war angenehm	☐	☐	☐	☐	☐
Die VR-Brille war komfortabel zu tragen	☐	☐	☐	☐	☐
Das Umschauen im virtuellen Raum durch Drehen des Kopfes oder Drehen des Körpers war intuitiv	☐	☐	☐	☐	☐
Die Annäherung an Objekte durch Daraufzugehen oder durch Annähern des Kopfes war intuitiv	☐	☐	☐	☐	☐
Die Gewinnung von Abstand zu Objekten durch Wegbewegen war intuitiv	☐	☐	☐	☐	☐
Die Fortbewegung im virtuellen Raum durch reales Gehen war praktisch	☐	☐	☐	☐	☐
Die Körper- und Kopfbewegungen zur Navigation (umschauen, annähern, distanzieren) waren leicht zu lernen	☐	☐	☐	☐	☐
Die Interaktion mit der virtuellen Umgebung durch die Controller war intuitiv	☐	☐	☐	☐	☐
Die Bedienung der Controller war leicht zu lernen	☐	☐	☐	☐	☐
Das automatische Markieren von Objekten durch den Strahl des Controllers war praktisch	☐	☐	☐	☐	☐
Durch die Markierung habe ich schnell verstanden mit welchen Objekten interagiert werden kann	☐	☐	☐	☐	☐
Das Öffnen und Schließen von Boxen durch den Abzug war praktisch	☐	☐	☐	☐	☐
Das Rotieren, Verschieben und Zoomen eines Objektes durch Gedrückthalten des großen runden Knopfes war praktisch	☐	☐	☐	☐	☐
Beim Rotieren, Verschieben oder Zoomen eines Objektes hatte ich das Gefühl, dieses in der Hand zu halten	☐	☐	☐	☐	☐
Die Darstellung von Informationen in einem Fenster neben dem Controller war praktisch	☐	☐	☐	☐	☐
Ich konnte dieses Fenster ähnlich wie eine Zeitung in die Hand nehmen und meinem Blickwinkel anpassen	☐	☐	☐	☐	☐
Die Rückmeldung der virtuellen Umgebung war direkt und flüssig	☐	☐	☐	☐	☐
Ich war frei von Schwindel und Übelkeit	☐	☐	☐	☐	☐
Ich hätte noch weitere Zeit in der virtuellen Umgebung verbringen können	☐	☐	☐	☐	☐
Ich würde den VR-Modus wiederverwenden	☐	☐	☐	☐	☐

Anmerkungen:

Verbesserungsvorschläge:

Rohdaten: Ankreuzteil Allgemeiner Bogen (2 = stimme völlig zu, 1 = stimme eher zu, 0 = neutral, -1 = stimme eher nicht zu, -2 = stimme gar nicht zu)

	Probanden											
Behauptung	**ID 3**	**ID 6**	**ID 8**	**ID 94**	**ID 37**	**ID 11**	**ID 7**	**ID 29**	**ID 134**	**ID 99**	**ID 54**	**ID 81**
Ich bin anfällig für Motion Sickness (Reisekrankheit, Seekrankheit, etc.)	-2	-1	-2	-1	-2	0	-1	0	-1	-1	-2	1
Ich habe eine Sehschwäche	1	2	-2	2	0	1	1	-1	1	1	-2	-2
Ich trage eine Brille	2	2	-1	2	0	-2	1	-2	2	-1	-2	-2
Ich habe bereits Erfahrung mit virtueller Realität gemacht	-1	2	2	-2	2	1	-2	1	1	-2	-2	-2

Rohdaten: Ankreuzteil Bogen des Studienleiters (4 = stimme völlig zu, 3 = stimme eher zu, 2 = neutral, 1 = stimme eher nicht zu, 0 = stimme gar nicht zu)

		Probanden											
ID	**Behauptung**	**ID 3**	**ID 6**	**ID 8**	**ID 94**	**ID 37**	**ID 11**	**ID 7**	**ID 29**	**ID 134**	**ID 99**	**ID 54**	**ID 81**
S1	**Die VR-Brille war einfach anzubringen1**	3	4	4	4	4	4	4	4	4	4	4	4
S2	**Der Proband hat das höchste System gefunden2**	4	4	4	4	4	4	4	4	4	4	4	4

Rohdaten: Freitextteil Bogen des Studienleiter (Allgemeine Bemerkungen, Verlauf und Besonderheiten)

	Probanden											
Freitextteil	**ID 3**	**ID 6**	**ID 8**	**ID 94**	**ID 37**	**ID 11**	**ID 7**	**ID 29**	**ID 134**	**ID 99**	**ID 54**	**ID 81**
Allgemeine Bemerkungen						Löschen der Applikation durch ein X		Benutzer würde einheitlicheres Schließen bevorzugen				
Verlauf			Löschen der Applikation war nicht intuitiv			Löschen der Applikation war nicht intuitiv	Das Löschen der Applikation hat zu Verwirrung geführt	Das Löschen der Applikation hat zu Verwirrung geführt			Das Löschen der Applikation hat zu Verwirrung geführt	Das Löschen der Applikation hat zu Verwirrung geführt
Besonderheiten												

Rohdaten: Ankreuzteil VR-Bogen (4 = stimme völlig zu, 3 = stimme eher zu, 2 = neutral, 1 = stimme eher nicht zu, 0 = stimme gar nicht zu)

		Probanden											
ID	**Behauptung**	**ID 3**	**ID 6**	**ID 8**	**ID 94**	**ID 37**	**ID 11**	**ID 7**	**ID 29**	**ID 134**	**ID 99**	**ID 54**	**ID 81**
A1	**Das Erkunden der virtuellen Umgebung war angenehm**	3	4	4	4	4	4	4	3	4	4	4	4
A2	**Die Darstellung der virtuellen Umgebung war angenehm**	3	4	4	3	3	3	4	3	4	3	3	3
A3	**Die VR-Brille war komfortabel zu tragen**	3	2	4	4	2	3	3	2	4	3	4	2
B1	**Das Umschauen im virtuellen Raum durch Drehen des Kopfes oder Drehen des Körpers war intuitiv**	4	4	4	4	4	4	4	4	4	4	4	4
B2	**Die Annäherung an Objekte durch Daraufzugehen oder durch Annähern des Kopfes war intuitiv**	4	4	4	4	4	4	4	4	4	4	4	4
B3	**Die Gewinnung von Abstand zu Objekten durch Wegbewegen war intuitiv**	4	4	4	4	4	4	4	4	4	4	4	4
B4	**Die Fortbewegung im virtuellen Raum durch reales Gehen war praktisch**	4	4	4	4	3	4	4	4	4	4	4	4
B5	**Die Körper- und Kopfbewegungen zur Navigation (umschauen, annähern, distanzieren) waren leicht zu lernen**	4	4	4	4	4	4	4	4	4	4	4	4
C1	**Die Interaktion mit der virtuellen Umgebung durch die Controller war intuitiv**	3	4	4	3	4	2	3	2	3	3	1	3
C2	**Die Bedienung der Controller war leicht zu lernen**	3	3	4	3	3	3	2	1	4	3	2	3
C3	**Das automatische Markieren von Objekten durch den Strahl des Controllers war praktisch**	4	4	4	4	4	4	4	4	4	4	4	4
C4	**Durch die Markierung habe ich schnell verstanden mit welchen Objekten interagiert werden kann**	4	4	4	4	4	4	4	4	4	4	4	4
C5	**Das Öffnen und Schließen von Boxen durch den Abzug war praktisch**	4	4	4	3	4	3	4	3	4	3	3	3
C6	**Das Rotieren, Verschieben und Zoomen eines Objektes durch Gedrückthalten des großen runden Knopfes war praktisch**	4	4	4	2	4	4	4	3	4	3	1	3
C7	**Beim Rotieren, Verschieben oder Zoomen eines Objektes hatte ich das Gefühl, dieses in der Hand zu halten**	3	3	4	1	4	3	4	1	1	2	3	2
C8	**Die Darstellung von Informationen in einem Fenster neben dem Controller war praktisch**	4	4	3	4	4	1	4	3	4	4	2	4
C9	**Ich konnte dieses Fenster ähnlich wie eine Zeitung in die Hand nehmen und meinem Blickwinkel anpassen**	4	4	4	4	4	3	4	3	4	4	3	4
A4	**Die Rückmeldung der virtuellen Umgebung war direkt und flüssig**	4	2	4	4	4	3	3	4	4	3	4	4
A5	**Ich war frei von Schwindel und Übelkeit**	4	3	4	4	4	4	4	4	4	4	4	1
A6	**Ich hätte noch weitere Zeit in der virtuellen Umgebung verbringen können**	4	4	4	4	4	4	4	3	3	4	3	4
A7	**Ich würde den VR-Modus wiederverwenden**	4	4	4	4	4	3	4	3	3	4	4	4

Rohdaten: Freitextteil VR-Bogen (Anmerkungen und Verbesserungsvorschläge)

	Probanden											
Freitextteil	**ID 3**	**ID 6**	**ID 8**	**ID 94**	**ID 37**	**ID 11**	**ID 7**	**ID 29**	**ID 134**	**ID 99**	**ID 54**	**ID 81**
Anmerkung	das Bild hätte ggf. noch etwas schärfer sein können, um angenehmer zu sein, sonst top	Die Ladezeiten beim Öffnen/ Schließen waren ungewohnt, man wurde aus der Bahn gerissen. Zumindest kurz	Sehr gut!	Das Zoomen durch wiederholtes Strecken des Armes erschien mir unpraktisch	Beschriftung der Komponenten war trotz Annähern teilweise nicht lesbar	Die Idee des Infofeldes am Controller ist interessant, jedoch hätte ich die Infos als starren Text im Blickfeld vorgezogen. Das Zoomen war zwar sehr intuitiv, jedoch langsam. evtl. ware eine Geste mit beiden Controllern, ähnlich der Zoomest eines Smartphones zusätzlich hilfreich. Text bei der niedrigen Viveauflösung ist unangenehm zu lesen.	Teilweise hat das Kabel etwas gestört mich in der virtuellen Landschaft zu bewegen	Ich hatte Probleme mir die Funktionen der Tasten zu merken, bzw vor allem wo ich hinzielen muss um Boxen wieder zu schließen		Gute kurze Einführung, um die Funktionalität zu verstehen	Es fiel mir in der VR schwer zu merken, wann ich welchen Knopf drücken muss	Sehr gut! sehr praktisch!
Verbesserungsvorschläge	Ich würde aus intuitiven Gründen das Öffnen/ Schließen mit dem Rotieren etc. von der Bedienung her tauschen.	Diese Ladezeiten umgehen, die Schriften ein bisschen größer machen. Manchmal war es schwer zu lesen für mich	Applikation durch X schließen	Vielleicht ließe sich der Zoom durch Knöpfe auf dem Controller realisieren.		ein kleines optionales Infoicon über der Tastenbelegung wäre anfangs ein sehr nettes Feature	-	Eventuell hätte mir eine Legende für den Controller geholfen, d.h. eine visuelle Darstellung welche Taste welche Funktion hat. Oder ich hätte einfach etwas mehr üben müssen.	Optimal wäre ggf. das Kabel der Brille an der Decke zu befestigen, es war gelegentlich im Weg (Gehen/ Handbewegung)	angenehmere Bodentextur, etwas neutraler	Auflösung erhöhen, besseres System zum verlassen der Elemente	Das Kabel auf dem Boden war manchmal etwas störend