|  |  |  |
| --- | --- | --- |
| C:\_Hochschule Heilbronn\6_AG MCI\4_Organisation\0_Corporate Design\1_Logo\UniTyLab_Logo.jpg |  | D:\Desktop\HHN Original (Vektor).emf |

Bachelor

Thesis

**Konzeption und Evaluation von Benutzerkonditionierung**

**in Virtual Reality**

|  |  |
| --- | --- |
| **Autor** | Robert Zlomke |
| **Studiengang** | Angewandte Informatik  Hochschule Heilbronn |
| **Matrikel-Nr.** | 193045 |
| **Abgabe** | 15.04.2021 |
| **Referent** | Prof. Dr.-Ing. Gerrit Meixner |
| **Korreferent:** | Philip Schäfer |

* 1. Danksagung
  2. Abstract

Virtuelle Umgebungen zu Fuß zu erkunden ist die realistischste und natürlichste Schnittstelle. Zeitgleich ist dies auch technisch und logistisch am anspruchsvollsten.

Allgemein wurden in bisherigen Arbeiten sich auf die Fortbewegung in übereinstimmenden oder leeren Umgebungen konzentriert. Weniger wurde darauf geachtet, wie sich eine Änderung der Nichtübereinstimmung zwischen der physischen Umgebung und ihrer virtuellen Darstellung auf die Benutzer auswirkt.

**Schlagwörter**: Wissenschaftliche Arbeit, Dokumentvorlage, Hochschule

**Inhaltsverzeichnis**

[I. Danksagung II](#_Toc77524575)

[II. Abstract III](#_Toc77524576)

[1 Einleitung 1](#_Toc77524577)

[1.1 Problemstellung 1](#_Toc77524578)

[1.2 Zielsetzung 1](#_Toc77524579)

[1.3 Vorgehensweise 1](#_Toc77524580)

[2 Stand der Technik 2](#_Toc77524581)

[2.1 Begriffserklärung: Virtual Reality 2](#_Toc77524582)

[2.1.1 Technische Charakterisierung 2](#_Toc77524583)

[2.1.2 Immersion 2](#_Toc77524584)

[2.1.3 Mensch-Computer Interaktion 3](#_Toc77524585)

[2.1.4 Präsenz 3](#_Toc77524586)

[2.2 Fortbewegungstechniken in Virtual Reality 4](#_Toc77524587)

[2.2.1 Natural Walking 5](#_Toc77524588)

[2.2.2 Redirected Walking 6](#_Toc77524589)

[2.2.2.1 Repositionierung 7](#_Toc77524590)

[2.2.2.2 Neuausrichtung 8](#_Toc77524591)

[2.2.3 Walking-In-Place 10](#_Toc77524592)

[2.2.3.1 Physikalische Schnittstellen 10](#_Toc77524593)

[2.2.3.2 Motion Tracking 11](#_Toc77524594)

[2.2.4 Abstahierte Schnittstellen des Gehens 11](#_Toc77524595)

[2.2.4.1 Joystick 12](#_Toc77524596)

[2.2.4.2 Teleport 12](#_Toc77524597)

[2.2.4.3 Point & Teleport 13](#_Toc77524598)

[2.2.4.4 Arm basierte Bewegungserfassung 14](#_Toc77524599)

[2.2.4.5 Neigungsbasierte Fortbewegung 14](#_Toc77524600)

[2.2.5 Laufbänder 16](#_Toc77524601)

[2.3 Konditionierung 17](#_Toc77524602)

[2.3.1 Grundlagen der Konditionierung 17](#_Toc77524603)

[2.3.1.1 Klassische Konditionierung 17](#_Toc77524604)

[2.3.1.2 Operante Konditonierung 18](#_Toc77524605)

[2.3.1.3 Kontextkonditionierung 18](#_Toc77524606)

[2.3.2 Konditionierung in Virtual Reality 19](#_Toc77524607)

[2.4 Unity 3D 19](#_Toc77524608)

[2.5 Oculus Quest 21](#_Toc77524609)

[2.6 HTC Vive 21](#_Toc77524610)

[2.6.1 Display 22](#_Toc77524611)

[2.6.2 Controller 22](#_Toc77524612)

[2.6.3 Tracking 23](#_Toc77524613)

[2.6.4 Vive WLAN Adapter 24](#_Toc77524614)

[3 Konzept 25](#_Toc77524615)

[4 Umsetzung 26](#_Toc77524616)

[4.1 Einbindung der Oculus Quest 26](#_Toc77524617)

[4.1.1 Schnittstelle zwischen PC und Oculus Quest 26](#_Toc77524618)

[4.1.2 Einbindung in Unity 27](#_Toc77524619)

[4.1.3 Ausführung der Anwendung 29](#_Toc77524620)

[4.2 Einbinden der HTC Vive 29](#_Toc77524621)

[4.2.1 Schnittstelle zwischen PC und SteamVR 29](#_Toc77524622)

[4.2.2 Einbindung in Unity 30](#_Toc77524623)

[4.3 Entwicklung der Szenarien 32](#_Toc77524624)

[4.3.1 Hindernisse 33](#_Toc77524625)

[4.3.2 Implementierung des Weges 34](#_Toc77524626)

[4.3.3 Implementierung der Wegerkennung 34](#_Toc77524627)

[4.3.4 Szenarien 36](#_Toc77524628)

[4.3.5 Entwicklung des Malus 38](#_Toc77524629)

[4.3.5.1 Erstellen des Canvas 38](#_Toc77524630)

[4.3.5.2 Implementierung der Animation 39](#_Toc77524631)

[4.3.5.3 Darstellung in VR 42](#_Toc77524632)

[4.3.5.4 Implementierung 42](#_Toc77524633)

[4.3.6 Positiver Verstärkung 43](#_Toc77524634)

[4.3.6.1 Erstellen der UI Elemente 43](#_Toc77524635)

[4.3.6.2 Implementierung der Animation 43](#_Toc77524636)

[4.3.7 Datenerfassung 44](#_Toc77524637)

[4.3.8 Aufgabe der Szenarien 45](#_Toc77524638)

[4.3.9 Hintergrundgeräusche 48](#_Toc77524639)

[4.3.10 Menü 48](#_Toc77524640)

[4.3.10.1 Implementierung der UI Elemente 48](#_Toc77524641)

[4.3.10.2 Funktionalität des Menüs 49](#_Toc77524642)

[4.3.10.3 Laserpointer mit der Oculus Integration 49](#_Toc77524643)

[4.3.10.4 Laserpointer mit dem SteamVR Plugin 50](#_Toc77524644)

[4.3.11 Ablaufplan 51](#_Toc77524645)

[4.4 Datenerfassung 51](#_Toc77524646)

[4.4.1 Quantitative Daten 51](#_Toc77524647)

[4.4.2 Qualitative Daten 52](#_Toc77524648)

[4.5 Evaluation und Auswertung der Ergbenisse 54](#_Toc77524649)

[4.5.1 Aufbau der Laborumgebung 54](#_Toc77524650)

[4.5.1.1 Aufbau des Weges 54](#_Toc77524651)

[4.5.1.2 Kalibrierung der HTC Vive 54](#_Toc77524652)

[4.5.2 Beschreibung des Forschungsgegenstandes 55](#_Toc77524653)

[4.5.2.1 Unabhängige Variablen 57](#_Toc77524654)

[4.5.2.2 Abhängige Variablen 57](#_Toc77524655)

[4.5.2.3 Stichprobe 57](#_Toc77524656)

[4.5.2.4 Ablauf 57](#_Toc77524657)

[4.5.3 Ergebnisse 59](#_Toc77524658)

[4.5.3.1 Fragebogen 59](#_Toc77524659)

[5 Zusammenfassung, Bewertung und Ausblick 60](#_Toc77524660)

[6 Anhang 61](#_Toc77524661)

[6.1 Literaturverzeichnis 61](#_Toc77524662)

[6.2 Eidesstattliche Erklärung 66](#_Toc77524663)

## Einleitung

### Problemstellung

Virtuelle Umgebungen zu Fuß zu erkunden ist die realistischste und natürlichste Schnittstelle. Zeitgleich ist dies auch technisch und logistisch am anspruchsvollsten.

Allgemein wurden in bisherigen Arbeiten sich auf die Fortbewegung in übereinstimmenden oder leeren Umgebungen konzentriert. Weniger wurde darauf geachtet, wie sich eine Änderung der Nichtübereinstimmung zwischen der physischen Umgebung und ihrer virtuellen Darstellung auf die Benutzer auswirkt.

### Zielsetzung

Ziel der vorliegenden Arbeit ist es…

* Konzipieren eines Szenarios, in dem das Bewegungsverhalten der Nutzer über festgelegte Reize in der VR beeinflussen zu können
* Evaluation, um die benötigten Daten zu sammeln und auszuwerten

### Vorgehensweise

In Kapitel 2 theoretische Basis geschaffen. Es werden die Grundlagen der Virtual Reality und die verschiedenen Fortbewegungstechniken erklärt. Ebenso der Themenbereich der Konditionierung, ein technischer Überblick über die Entwicklungsumgebung Unity 3D und den VR-Brillen Oculus Quest und HTC Vive

In Kapitel 4 wird die Umsetzung des Projekts in einzelnen Schritten erläutert. Dies fängt mit der Einbindung der genutzten VR-Brillen an, darauf folgt die Entwicklung der Szenarien, der Datenerfassung und zuletzt der Evaluation. Die Ergebnisse fließen zuletzt in die Zusammenfassung, Bewertung und Ausblick der Studie ein.

## Stand der Technik

### Begriffserklärung: Virtual Reality

In der Welt der Wissenschaft herrscht noch keine einheitliche Definition für den Begriff „Virtual Reality“, oder im Nachfolgenden auch VR genannt. In Science-Fiction Filmen wird die Virtual Reality, im Weiteren auch VR genannt, als eine digitale Simulation beschrieben, welche für den Menschen nicht mehr von der Realität zu unterscheiden ist. In praktischen Bezug dient die Technologie in vielen Industriebereichen als Werkzeug und findet ebenso im Home Entertainment Bereich immer weiter Einzug. Die Technologie des Virtual Reality ist noch ein etwas junges Gebiet in der Wissenschaft, welche sich jedoch durch die sehr rasante Entwicklung und Verbesserung der dazu benötigten Hardware weiterentwickelt [1].

#### Technische Charakterisierung

Die technologische Charakterisierung beinhaltet grundlegende Unterschiede zur normalen Computergraphik (siehe Abb. 1: Unterschiede von 3D-Computergraphik und Virtual Reality). Der Hauptunterschied ist das beim Benutzer am Kopf angebrachte Display, was im Englischen meistens als Head-Mounted-Display bekannt ist. Durch diese zwei, vor den Augen angebrachten, Display werden 3D Anwendungen wie z.B. eine virtuelle Welt angezeigt, die sich den Umgebungen im normal bekannten Spiele Bereich ähnelt. Neben der Anzeige werden zusätzlich noch Sinne wie Audio oder Haptik mit einbezogen, was auch eine typische Charakterisierung von VR ist. Um eine Interaktion mit der virtuellen Welt zu ermöglichen, werden Eingabegeräte wie Controller genutzt. Durch die Bestimmung der Position und Orientierung, im englischen Tracking genannt, sind natürliche Interaktionen wie z.B. Zeigen, Greifen oder Werfen möglich. Ebenso werden auch die Position und Orientierung des Nutzers bestimmt, um eine freie Bewegung physischen Raum zu ermöglichen [1].

#### Immersion

Ein weiteres wichtiges Merkmal ist die Erzeugung der Immersion, was als komplette Eintauchen in die virtuelle Welt verstanden wird. In der Literatur wird bei der Definition dieses Begriffs in physikalischer Immersion und mentaler Immersion unterteilt. Nach Slater gibt es für die physikalische Immersion 4 entscheidende Merkmale:

* Isolation von der Realität durch verschiedene, digital erstellte Sinneseindrücke
* Einbeziehung möglichst vieler Sinne
* Vollständige Umschließung der Ausgabegeräte um den Nutzer
* Darstellung der virtuellen 3D Umgebung von hoher Qualität, wie z.B. eine gute Auflösung des Bildes mit einer hohen Bildwiederholungsrate



Abb. 1: Unterschiede von 3D-Computergraphik und Virtual Reality

#### Mensch-Computer Interaktion

Unterschiede sind aber ebenso im Bereich der Mensch-Computer-Interaktionen zu finden. Die Interaktion mit der virtuellen Umgebung soll so interaktiv und natürlich wie möglich sein. Dies trifft durch die besonderen Schnittstellen in Virtual Reality zu, da der Mensch schon aus Erfahrungen im Alltag weiß, wie der Körper für bestimmte Interaktionen einzusetzen ist. Der Lernaufwand und der mentale Aufwand lassen sich so deutlich reduzieren [1].

#### Präsenz

Neben der Immersion ist die Präsenz auch ein wichtiges mentale Merkmal von Virtual Reality. In der Forschung und Weiterentwicklung der Technik wird der Bereich immer noch als wichtig angesehen. Das Ziel der Präsenz ist es, mit Hilfe der einem immersiven VR-Systems, ein Gefühl beim Benutzer zu erzeugen, sich vollkommen der virtuellen Welt zu befinden und die physische Umgebung nicht mehr wahr zu nehmen. Die Reaktionen in der virtuellen Welt sollen den Reaktionen in der realen Welt ähneln oder sogar gleichermaßen übereinstimmend sein.

Die Merkmale zur Erreichung der Präsenz setzt sich aus 3 Faktoren zusammen

* Ortsillusion: Blickwinkel und Blickrichtung abhängige Darstellen bei Kopfdrehung
* Plausibilitätsillusion: Glaubwürdigkeit der Umgebung
* Involviertheit: Interesse des Nutzers an der virtuellen Umgebung

### Fortbewegungstechniken in Virtual Reality

Die Technologie der Virtual Reality ermöglicht es den Benutzer, komplett in eine virtuelle Umgebung einzutauchen und mit dieser zu interagieren. Ein großer wichtiger Teil ist die Fortbewegung in der virtuellen Welt und ist einer der wichtigsten Komponente der Erkundung. In den letzten Jahren hat sich die Technologie der VR immer weiter verbessert und der Einzug des Gebrauchs in privaten Haushalten hat begonnen. Der Grund ist die kommerzielle Veröffentlichung von VR Hardware wie die Playstation VR, Oculus Go, Quest 1 und 2. Diese ermöglichen eine immersive virtuelle Erfahrung ohne die Notwendigkeit eines großen Labors oder eines aufwendigen Tracking Systems. Die Suche nach der besten und effizientesten Methode der Fortbewegung bzw. Lokomotion stellt sich dadurch jedoch neuen großen Herausforderungen und ist einer der wichtigsten Fragen, da sie direkten Einfluss auf Faktoren wie Spaß, Präsenz oder auch Probleme wie Bewegungskrankheit oder Ermüdung hat. Aus diesem Grund ist dieser Bereich in den letzten Jahren die Grundlagen vieler durchgeführten Studien [2].

Chernie et al veröffentlichten ein selbst durchgeführtes Literatur Review über die verschiedenen Techniken von Lokomotion in Virtual Reality vor. Auf dieser Grundlage erstellen sie eine Taxonomie, um die vielen unterschiedlichen und aktuell verfügbaren Techniken der Fortbewegung Kategorisieren zu können. Die drei grundlegende Unterscheidung findet auf der obersten Ebene zwischen Methoden statt, ob der Fokus auf den Körper des Benutzers besteht, eine externe Peripherie genutzt wird, oder beides. Unter jeder dieser drei Kategorien gibt es zwei weitere Ebenen, die weitere Unterkategorien darstellen [2].



Abb. 2: Eine Taxonomie der Fortbewegungstechniken

In den nachfolgenden Unterkapiteln werden die unterschiedlichen Methoden der Fortbewegung in virtuellen Umgebungen und dessen Stand der Technik genauer beschrieben.

#### Natural Walking

Das sogenannte Natural Walking handelt es sich um die natürlichste Art der Fortbewegung in virtuellen Umgebungen. Hier bewegt sich der Nutzer physisch durch das Gehen fort, wobei ein Schritt in der realen Welt ein Schritt in der VR von gleicher Länge bedeutet. Grundvoraussetzung für die Umsetzung ist, dass der physische Raum mindestens genauso groß, wie der virtuelle Raum ist [3].

Es ist jedoch mit verschiedenen Techniken ebenso möglich, wenn der physische Raum kleiner als die virtuelle Umgebung ist. Auf diese Thematik wird im nachfolgenden Kapitel 3.2.2 Redirected Walking genauer eingegangen. Laut Usoh et al ist das natürliche Gehen in virtuellen Umgebungen natürlich, leicht und unkompliziert. Dies kann den Gründen zu Schulden liegen, das diese Schnittstelle im Vergleich mit Walking-in-Place und Fliegen das größte Gefühl der Präsenz erzeugt und am wenigsten körperliches Unbehagen auslöst [4].

Ruddle et al führten eine Studie anhand einer Navigationsaufgabe zu, wo Ziele gesucht werden mussten. Diese hat gezeigt, dass natürliches Gehen im Vergleich zu nicht Körper basierter Fortbewegung durch die höhere und feinmotorischere Beweglichkeit der Personen zu weniger Kollisionen mit Hindernissen führt [5].

Ebenso zeigten Ruddle et al mit einer Navigationsaufgabe im Vergleich zur Fortbewegung mit einem Joystick, einem Laufband und echtem Gehen, das eine natürliche Schnittstelle des Gehens weniger Trainingszeit benötigt und die Aufgaben ohne Training schneller absolviert wurde, und dies unabhängig von der Schwierigkeit der virtuellen Umgebung [6].

Whitton et al benutzen drei Metriken um die Leistung von Bewegungsschnittstellen zu Bewerten. Diese wurden genutzt, um zu bewerten, wie die Bewegungspfade waren, die Qualität der Aufgabenleistung und die Zeit bis zu Kollisionen mit Hindernissen. Die Ergebnisse zeigten, dass die Bahnkurven bei der Schnittstelle des realen Gehens mit der Fortbewegung in der realen Welt überwiegend übereinstimmen [7].

All diese Studien stimmen als Ergebnis überein, das die Schnittstelle des Natural Walking die natürlichste Schnittstelle ist, die größte Präsenz beim Benutzer erzeugt und die Erreichung des Ziels bei zielsuchenden Aufgaben am schnellsten und einfachsten erreicht wird.

#### Redirected Walking

Wenn die virtuelle Umgebung erheblich größer ist als der physisch verfügbare Raum, stößt das natürliche Gehen schnell an seine Grenzen. Um das Problem lösen zu können, wurden in den letzten einige Techniken zum Lösen dieses Problems entwickelt und vorgeschlagen. Diese werden unter dem Begriff Redirected Walking kategorisieren. Die Route beim Fortbewegen des Benutzers in der virtuellen Umgebung wird manipuliert, ohne dass dieser dies im besten Falle bemerkt. Jedoch gibt es auch Techniken, die eine nicht subtile Lösung bieten. So hat er Benutzer die Möglichkeit, sich durch natürliches Gehen in großen virtuellen Umgebungen fortzubewegen, auch wenn der physische Raum erheblich kleiner ist [8] [3].

Suma et al. [9] stellten eine Taxonomie von Umleitungstechniken vor (siehe Abb. 3: Eine Taxonomie für Umleitungstechniken). Die Arbeit basiert und erweitert eine frühere Arbeit von Steinicke et al [10]. Sie ermöglicht diese vielfältigen Techniken anhand verschiedener Kriterien auszuwählen, die zum Szenario der virtuellen Umgebung passen, auf die sie angewendet werden soll. Die Kategorisierung basiert zum einen darauf, ob die Manipulation auf die Translation oder Rotation Einfluss hat. Weitere Unterteilungen finden statt, ob die Veränderung dabei diskret oder kontinuierlich ist und für den Benutzer subtil oder auffällig [3].



Abb. 3: Eine Taxonomie für Umleitungstechniken

##### Repositionierung

Eine Repositionierung kann subtil angewendet werden, sodass der Benutzer dies nicht wahrnehmen kann. Eine Studie von Williams et al. untersuchten das Erkunden virtueller Umgebungen mit unterschiedlich skalierten Übersetzungen im Vergleich zum natürlichen Gehen in der realen Welt. Bewegt sich der Benutzer in eine bestimmte Richtung, wird eine Verstärkung der Translation angewendet. Die Gehbewegung wird somit skaliert und kann eine größere Entfernung, als es im rein physischen Raum möglich wäre, zurückgelegt werden. Die Ergebnisse zeigen, dass dies eine praktikable Technik ist, um die Erkundung großer virtueller Welten zu ermöglichen. [11] [9].

Steinicke et al. führten mehrere analysierten über die menschliche Sensibilität gegenüber diesen Manipulationen durch [12] [13]. Die Ergebnisse zeigen, dass diese Technik vom Benutzer unerkannt bleibt, solange die Skalierung klein genug ist. Genauer gesagt können gelaufene Distanzen können im physischen Raum um 14% herunter- und um 26% hochskaliert werden, ohne erkannt zu werden. Dies deckt sich mit Erkenntnissen von Interrante et al. [14] und Bruder et al. [15], dass bei Benutzer das Phänomen auftritt, dass Distanzen in einer virtuellen Umgebung, durch Faktoren wie die Auflösung und Latenz, unterschätzt werden und Entfernungen in einer virtuellen Umgebung als komprimiert wahrnehmen [16] [9].

Bei der subtilen und diskreten Repositionierung findet die Veränderung der Position abrupt statt, um den visuellen Raum zu vergrößern. Jedoch müssen Techniken zum Einsatz kommen, dass die Verschiebung für den Benutzer unauffällig bleibt und nicht bemerkt wird. Bruder et al. Haben in Experimenten in einer Studie herausgefunden, dass ein optischer Fluss bzw. Selbstbewegungsillusionen am Rande des Blickfelds des Benutzers verwendet, werden können, um abrupte Verschiebungen der Translation zu nutzen, ohne das die Benutzer dies signifikant merken [15] [9].

Ebenso kann durch eine offene und kontinuierliche Verschiebung der virtuellen Umgebung um den Benutzer verschoben werden. Das diese Technik der Repositionierung für den Benutzer desorientierend wirken kann, können Objekte wie Aufzüge und Autos als Navigationshilfen genutzt werden. Die diskrete Neupositionierung ebenso nicht unauffällig, der Benutzer nimmt sie also bewusst wahr. Die Umsetzung dieser Technik wird durch die abrupte Teleportation an eine neue Position umgesetzt. Für den Benutzer kann es jedoch desorientierend wirken, wenn er dies nicht erwartet. Aus diesem Grund werden Portale eingesetzt, die vor allem aus Science-Fiction und Fantasy bekannt sind. Dies sind Objekte wie ein Spiegel, ein Tor oder ein magischer Wirbel aus Energie, die zwei orte miteinander verbinden. Steinicke et al. führten eine Studie durch, um in Erfahrung zu bringen welche Auswirkungen ein Portal hat, um den Benutzer einen schrittweisen Übergang in die virtuelle Umgebung zu ermöglichen. Die Benutzer bestätigten nach der Durchführung des Experiments, das ein höheres Gefühl der Präsenz wahrzunehmen war und die Fähigkeit Distanzen korrekt zu schätzen verbessert war [17] [9].

##### Neuausrichtung

Sobald der Benutzer eine Grenze der physischen Umgebung erreicht hat, z.B. durch eine Wand, wird dieser zurückgesetzt. Dabei wird der optische Fluss so manipuliert, dass die Position in der virtuellen Umgebung gleichbleibt, sich der Benutzer aber vom physischen Hindernis entfernt. Williams et al. stellten die sogenannte „2:1-Turn“ Methode vor. Sobald der Benutzer eine Grenze des physischen Raums erreicht hat, wird dieser angewiesen sich zu drehen. Dabei erfolgt die Drehung physisch um 180 Grad, während die Orientierung nach dieser Drehung gleich ist. Bei dieser Technik wird die Stärkung der virtuellen Drehung verdoppelt, d.h. die Drehung um 180 Grad im physischen Raum bedeutet eine 360 Grad Drehung in der virtuellen Umgebung. Die Rotationsverstärkung ist für den Benutzer nicht erkennbar, jedoch die Aufforderung zur Drehung. [18] [9].

Bewegt sich der Benutzer durch die virtuelle Welt, wird diese kontinuierlich um diese gedreht. So kann der Benutzer auf seinem virtuellen Pfad bleiben, bewegt sich aber ständig auf die physisch auf die Wand oder das Hindernis mit der größten Entfernung. Ist die Rotationsverzerrung klein genug, merkt der Benutzer diese nicht und nimmt sie als Eigenbewegung wahr. Hier muss ein Kompromiss gefunden werden, denn je größer die Rotationsverzerrung ausfällt, desto kleiner muss der physisch vorhandene Raum sein und desto größer kann die virtuelle Umgebung ausfallen. Jedoch besteht dann auch eine höhere Gefahr, dass der Benutzer diese dann bemerkt. Steinicke et al untersuchten in einer Studie die Erkennbarkeit von Rotationsverstärkungen von Kopf und Körper. Die Ergebnisse zeigten, dass der Benutzer, der sich in einer virtuellen Umgebung bewegt, körperlich um 49% mehr und 20% weniger gedreht werden können, ohne dies zu bemerken. Der Nachteil ist jedoch, dass dies ebenso einen großen viel größeren physischen Raum im Vergleich zu anderen Techniken voraussetzt. Razzaque et al. testeten die Anwendung der Rotationsverzerrung in einer Studie. In einer virtuellen Feuerwehrübung. In dieser mussten sie einen Raum durchqueren und dabei an mehrerer Wegpunkte vorbei. Die virtuelle Umgebung war dabei doppelt so groß als der physische Raum. Die Ergebnisse zeigen, dass der benötigte physische Raum verkleinert werden kann, wenn mit strategischen Wegpunkte gearbeitet wird. Dies stellt jedoch auch eine Einschränkung dar, da man diese Technik nur zu bestimmten Aufgabenstellungen eignet. Die nachfolgende Abbildung (siehe Abb. 4: Redirected Walking) zeigt, wie der Benutzer sich ein einer Zickzack Bewegung durch die virtuelle Umgebung fortbewegt, aber sich physischen Raum hin und her bewegt [16] [8] [9].



Abb. 4: Redirected Walking

Williams et al stellten neben der offenen kontinuierlichen Neuausrichtung (siehe Kapitel 2.2.2.2) auch Technik zur offenen diskreten Neuausrichtung zum Zurücksetzen des Benutzers vor. Bei der „Freeze-and-Turn“ Methode wird dem Benutzer sobald die Grenze des physischen Bereichs erreicht wird signalisiert, dass eine Drehung notwendig ist. Das gesamte Display und somit auch die Ausrichtung wird eingefroren und wieder fortgesetzt, sobald der Benutzer die 180 Grad Drehung weg vom physischen Hindernis vollendet wurde. Nun kann sich der Benutzer wieder frei in der virtuellen Umgebung bewegen. Da die Immersion und Präsenz durch die Unterbrechung erheblich gestört werden kann, ist diese Technik für die meisten Szenarien nicht geeignet und dient lediglich als Notfall Lösung [18] [9].

Für eine subtile und diskrete Neuorientierung wurde die Technik "Change Blindness Redirection" von Suma et vorgeschlagen. Diese sagt, das Änderungen außerhalb des Gesichtsfeldes eines Benutzers nicht wahrgenommen werden kann. Die Grundlage dafür basiert auf die Veränderungsblindheit bei Menschen. Der erste Eindruck einer Umgebung wird von der veränderten Umgebung überschrieben, ohne dies zu merken. Für die Umsetzung der diskreten Neuorientierung werden die Position und Ausrichtung von architektonischen Merkamel, vor allem Türen, verändert. Dies geschieht, sobald der Benutzer in eine andere Richtung schaut. In nachfolgender Abbildung (siehe

Abb. 5: Diskreter Szenenwechsel) wird ein Beispiel eines Szenenwechsels veranschaulicht. Sobald der Benutzer wegschaut, wird die Türe um 90 Grad gedreht. Somit verlässt der Benutzer den Raum in eine andere Richtig als beim Betreten des Raums. Es ist keine Unterbrechung für die Neuorientierung notwendig, was das Gefühl der Präsenz beibehalten kann. Die Ergebnisse der Studie zeigt, das nur einer der insgesamt 77 Testpersonen die Neuorientierung in der virtuellen Umgebung erkannt hat. Jedoch ist der Einsatz nur in Szenen geeignet und einsetzbar, sich dich in Innenräumen befinden und nicht für große und freie Umgebungen [19] [9] [20].



Abb. 5: Diskreter Szenenwechsel

#### Walking-In-Place

Walking-in-Place ist eine Kategorie von einer Bewegungsschnittstelle, bei der das natürliche Gehen imitiert wird. Anstatt dem natürlichen Gehen, wie in Kapitel 2.2.1 und 2.2.2 erläutert wird, wird ein stationärer Gang an Ort und Stelle genutzt, ohne sich tatsächlich physisch fortzubewegen. Eine oft genutzte und traditionelle Geste für die Eingabe ist eine Schrittgeste, die das Treppen steigen ähnelt. Dies wird in der virtuellen Umgebung mit zusätzlicher Verschiebung der Blickrichtung in eine Gehbewegung umgesetzt, um diese erkunden zu können [3] [2] [21].

Allgemein können die Techniken zur Umsetzung von Walking-in-Place in zwei Kategorien unterteilt werden. Zum einen wird die Manipulation einer physischen Schnittstelle, wie z.B. das Wii Balance Board, für die Schritterkennung genutzt. Zum anderen gibt es Techniken, die eine Bewegungserfassung bzw. Motion Tracking zum Erfassen von Position oder Geschwindigkeit von Körperteilen verwendet [21].

##### Physikalische Schnittstellen

Bei den physikalischen Schnittstellen basiert die Technik zum größten Teil darauf, eine indirekte Abbildung zwischen der Schrittgeste und der Blickrichtung zu erzeugen. Zur Erkennung der Schritte werden die Kräfte gemessen, die erzeugt werden, wenn die Füße beim Auftreten den Boden berühren. Als Beispiel dient das Wii Balance Board, das mit mehreren Drucksensoren ausgestattet ist. Der entwickelte Wii Algorithmus erkennt dabei, wie schnell der Benutzer mit seinem Gewicht Druck auf die Ecken des Boards ausübt. Dies wird als Gehbewegung in der virtuellen Umgebung in Richtung des Blickwinkels übersetzt [21].

Die WIP-Techniken, die sich auf physikalische Schnittstellen stützen, verwenden fast ausnahmslos eine indirekte Abbildung zwischen der Schrittgeste und der Verschiebung des Blickpunkts. Viele von ihnen nutzen die Kräfte, die beim Kontakt eines Fußes mit dem Boden auftreten, um zu erkennen, dass ein Schritt gemacht wurde. In einer Studie stellen Williams et al die Fortbewegung auf dem Wii Balance Board, mit dem Joystick und dem natürlichen Gehen gegenüber. Die Ergebnisse zeigten, dass die Fortbewegung mit dem Joystick zu höheren Drehfehlern führt. Die räumliche Ausrichtung zeigte in Bezug auf Latenz und Drehfehler war bei der Fortbewegung mit Wii Balance Board und dem natürlichen Gehen Ähnlichkeiten. Dies macht das Wii Balance Board zu einer kostengünstigen Alternative [21] [22].

##### Motion Tracking

Eine der frühesten Techniken für Walking-In-Place ist die „Virtual Treadmill“, was von Slater el al. vorgestellt wurde. Ein neuronales Netzwerk analysiert dabei das Head-Mounted-Displays des Benutzers und kann feststellen, ob der Benutzer an Ort und Stelle geht oder nicht. Wird ein Gehverhalten festgestellt, wird eine Fortbewegung in der virtuellen Umgebung in Blickrichtung des Benutzers umgesetzt. Die Studie deutet darauf hin, dass die Präsenz durch diese Technik verbessert werden kann [21] [23].

Von Zielinski et al wurde Shadow Walking vorgestellt, einer Technik zur Fortbewegung mit Hilfe von Unterflurprojektion. Durch die Nutzung von Schattenverfolgungsfunktionen ermöglicht das Gehen an Ort und Stelle und Sidestep-Gesten für die virtuelle Fortbewegung. Im Gegensatz zu früheren Implementierungen müssen beim Shadow Walking keine Anbauteile getragen werden. Dadurch ist es kostengünstig und einfach zu implementieren und erfordert keine Benutzerkalibrierung. Allerdings ist es auf Unterflur-Projektionssysteme, wie z.B. eine sechseckige CAVE, beschränkt. [24]

Darüber hinaus wurden auch neue Varianten von Methoden entwickelt, welche unter zusätzlicher Verwendung verschiedener Gesteneingaben arbeiten. Nilsson et al. [21] stellten das Tapping-In-Place vor. Dabei heben die Benutzer abwechselnd jede Ferse vom Boden ab, während die Zehen in Kontakt mit dem Boden bleiben. In einem Experiment war es die Aufgabe von Probanden, eine einfach Zielaufgabe durchzuführen und von einem Punkt zu einem Punkt über einen sichtbaren Pfad zu gelangen. Die Ergebnisse zeigten, das die Probanden die Fortbewegung mit der Tapping Methode im Vergleich zu anderen Walking-in-Place Methoden am natürlichsten und weniger anstrengend fanden. [21]

In den letzten Jahren wurden viele weitere Studien durchgeführt, die zum Ergebnis kamen, das Walkin-in-Place die meisten Techniken wie Arm Swinging, Point & Teleport, Joystick, lehnungsbasierte Techniken usw. übertrifft. Jedoch wurde auch aufgezeigt, das Walking-in-Place weniger realistischer ist als das natürliche Gehen. Darüber hinaus kann die stehende Position nicht immer geeignet sein, und die Erkundung großer virtueller Umgebungen mehr Zeit und körperliche Anstrengung benötigen. Ebenso kann die Immersion durch die Schwierigkeit verschiedener Gesten reduziert werden [3] [2] [25].

#### Abstahierte Schnittstellen des Gehens

„Virtuelles Fliegen“ ist eine weit verbreitete Technik, die vor allem von VR Spielen auf Spielekonsolen oder am PC, Verwendung findet. Ebenso ist die „Point & Teleport“ eine weit verbreitete Methode, wenn sich die Fortbewegung überwiegend auf das natürliche Gehen konzentriert, der physische Raum jedoch zu wenig Platz bietet [3]

##### Joystick

Es bestehen zwei Möglichkeiten, sich mit Hilfe eines Joysticks bzw. Controller in einer virtuellen Umgebung fortzubewegen. Oft wird der Joystick nur zur Steuerung der Translation verwendet, die die Rotation des HDM die Richtung vorgibt. Es kommt aber auch oft die Möglichkeit zum Einsatz, mit Hilfe des Joysticks die Translation und auch Rotation zu steuern. Dies ist vor allem im Bereich Gaming bekannt, um sich durch die großen komplexen Spielwelten zu bewegen. Buttussi et al untersuchten die Auswirkungen von Joystick-, Teleport- und Leaning-Techniken für die Fortbewegung an Ort und Stelle. Die Ergebnisse zeigten, dass die Fortbewegung mit dem Joystick im Vergleich zum Teleportieren signifikant langsamer ist, weniger Benutzerfreundlich ist und bei den Benutzern mehr Übelkeit verursachen kann. Jedoch waren keine Unterschiede der Präsenz festzustellen [26].

##### Teleport

Teleportation ist eine der häufigsten, nicht natürlichen genutzten Techniken zur Fortbewegung, die vor allem im Gaming Bereich bekannt ist. Der Einsatz bietet sich vor allem dann an, wenn der physische Bereich deutlich kleiner ist als die virtuelle Umgebung. Durch den Einzug von VR in den privaten Haushalten, ist dies meist der Fall. Ebenso stieg die Popularität durch ein kostenloses Plugin von Steam für Unity 3D und der Unreal Engine, mit dem diese Fortbewegungstechnik einfach verwendet werden kann. Im Vergleich zu Point & Teleport (siehe Kapitel 2.2.4.3), wird die Fortbewegung in den meisten Fällen mit einem Joystick bzw. Controller gesteuert. Das gewünschte Ziel wird durch Zeigen mit dem Controller mit Hilfe eines sichtbaren Markers ausgewählt und bestätigt und so die Position in der Umgebung geändert. Dabei fehlt jegliche Art von Selbstbewegungshinweisen. Ziele können frei wählbar sein oder feste vordefinierte Teleportpunkte in der Umgebung, die zur Auswahl stehen [3] [2] [26] [27] [28].



Abb. 6: Teleport mit Hilfe des Controllers [26]

Habgood et al. verglichen in einer Studie Point & Teleport, kontinuierliches Gehen mit einem Controller und einer neuen Variante des Points & Teleport, bei der die Knotenpunkte zur Fortbewegung vordefiniert sind. Die Benutzer hatten die Aufgabe, durch ein virtuelles Haus zu navigieren. Die Ergebnisse zeigten, dass bei der Navigation mit dem Point & Teleport die Kollisionen verringert waren und weniger Bewegungskrankheit bei den Benutzern auftrat [29].

Buttussi et al. führten einen Vergleich zwischen der Fortbewegung mit Hilfe eines Joysticks, Teleport und Leaning durch. Benutzer mussten in einer offenen virtuellen Umgebung in einer vorgegebenen Reihenfolge verschiedene Fässer, die verteilt aufgestellt wurden, erreichen. Dadurch kamen Sie zu dem Schluss, dass die Fortbewegung mit Point & Teleport schneller war, weniger Übelkeit bei den Benutzern auftrat und die Schnittstelle im Vergleich benutzerfreundlicher ist. Sie kann jedoch auch desorientierender wirken und somit Auswirkungen auf Präsenz und Immersion haben [26].

##### Point & Teleport

Im Vergleich zu der Teleport Methode (siehe 2.2.4.2), wo die Fortbewegung mit Hilfe eines Controllers funktioniert, kann bei der Point & Teleport Technik der Arm des Benutzers zur Steuerung der Bewegung genutzt. Der Arm wird verwendet, um auf den Ort zu zeigen, zu dem er sich bewegen mag. Hier können die Ziele wie bei der Teleportation mit dem Controller ebenso frei wählbar oder als Zielort fest vordefiniert sein. Das Ziel wird dann meist durch einen Tastendruck oder durch das längere Zeigen auf das Ziel ausgeführt und durch ein visuelles oder auditives Feedback bestätigt wird [2] [28]



Abb. 7: Point & Teleport Technik basierend auf die Armbewegungen [2]

Bozgeyikli et al. Verglichen die Techniken Point & Teleport, Walking-in-Place und die klassische Steuerung mit dem Joystick miteinander. Nach 2 Sekunden zeigen auf einen Zielort wird die Teleportation ausgelöst und gesenkte Arme des Benutzers macht die Technik inaktiv. Die Aufgabe der Benutzer war, 10 Ziele nacheinander abzulaufen, ohne Kollisionen mit Hindernissen zu verursachen. Die Auswertungen zeigten, dass Point & Teleport die beiden anderen Technik übertraf, da die Zielpunkte schneller erreicht werden konnten und weniger Kollisionen mit den Objekten verursacht wurden. Jedoch war die Fortbewegung mit dem Joystick einfacher zu verstehen, weniger anstrengend, erzeugte weniger Ermüdungserscheinungen und die Benutzer hatten eine bessere Kontrolle [2] [30]

##### Arm basierte Bewegungserfassung

Eine weitere Möglichkeit zur Fortbewegung besteht darin, die Arme in Form von Schwingen zu nutzen. Dadurch kann sich der Benutzer dann in die Richtung bewegen, in der er schaut. Für die technische Umsetzung ist die Verfolgung der Position und Rotation der Hand durch Tracking Techniken [2].

Wilson et al. stellten einen Vergleich zwischen dem natürlichen, physischen Gehen, Walking-in-Place und der Arm Swinging auf. Das Tracking der Arme wurde mit Hilfe eines Myo Armbandes vorgenommen, welches mit einem Gyroskop, eines Beschleunigungssensors und einem Magnetometer ausgestattet ist. Die Probanden mussten als Aufgabe sich die Standorte verschiedener Objekte, welche sich auf Säulen befangen, merken. Die Objekte waren kreisförmig um den Probanden angeordnet und bekam die Aufgabe, sich in einer vorgegebenen Reihenfolge den Objekten zuzuwenden. Das Experiment zeigte, das die physische Fortbewegung die anderen beiden Methoden zur Fortbewegung in Bereich Drehfehler übertraf. Walking-in-Place war in diesem Bereich jedoch besser als die Arm Swinging Methode. Ebenso gab es Hinweise darauf, dass Arm Swinging möglicherweise mehr Training bei den Probanden erforderte [31].

##### Neigungsbasierte Fortbewegung

Bei der neigungsbasierten Fortbewegehen wird die Neigung des Körperschwerpunktes in eine Richtung, wobei der ganze Körper oder nur ein Teil davon genutzt wird, in Bewegung umgesetzt wird. Die Neigung wird dabei meist durch ein Tracking System oder Geräte wie dem Wii Balance Board oder dem speziell entwickelten NaviChair umgesetzt [32].

**Kopfbasierte Fortbewegung**

Zielasko et al. Führten eine Studie durch, um verschiedene Fortbewegungsmethoden zu entwickeln und evaluieren, die im Sitzen am Tisch nutzbar sind. Eine Methode davon war das sogenannte „Shake your Head“, welches das Tracken des Kopfes mit Hilfe des HDM nutzt. Um sich Vorwärts oder Rückwärts zu bewegen, musste der Benutzer mi dem Kopf nach oben oder unten schauen. Ebenso war eine virtuelle Drehung anhand der Nickachse möglich. Die Aufgabe der Studie war, den kürzesten Weg zwischen zwei Eckpunkten in einer virtuellen Umgebung zu finden. Die Ergebnisse zeigten, die Fortbewegung mit dem Joystick die Fortbewegung über den Kopf übertraf. Das zeigte sich über die längere Zeit, die zur Bewältigung der Aufgabe gebraucht wurde. Ebenso äußersten sich die Probanden über weniger Präzision und eine erhöhrte Fehleranfälligkeit [2] [32].

**Rumpfbasierte Bewegungserfassung**

Neben der kopfbasierten Fortbewegung kann die Bewegungserfassung auch rumpfbasiert stattfinden. Die Fortbewegung in der virtuellen Umgebung wird hierbei durch die physische Neigung des Rumpfes im Stehen oder Sitzen umgesetzt. Die Rotation geschieht durch die Drehung des Rumpfes. Harris et al verglichen in zwei Experimenten die rumbasierte Fortbewegung Wii-Lean auf dem Wii Balance Board mit der Navigation mit einem Joystick und Walking-in-Place. Die Ergebnisse zeigten, dass die räumliche Repräsentation im Vergleich zum Joystick mit der Leaning Methode genauer war. Ebenso war die Orientierung, die gebrauchte Zeit und die körperliche Anstrengung um Vergleich zu Walking-in-Place besser, jedoch auf Kosten von Drehfehler und Latenz [2] [33].



Abb. 8: Wii-Leaning Technik auf dem Wii Balance Board [33]

Neben den rumpfbasierten Techniken im Stehen, ist die Navigation ebenso im Sitzen möglich. Kitson et al. verglichen die Navigation mit Hilfe eines Joysticks mit den im Sitzen lehnungsbasierten Techniken wie dem NaviChair, NuvMan, Head-directed und Swivel Chair (siehe Abb. 9: Sitzende, lehnungsbasierte Lokomotionstechniken [34]). Die Aufgabe der Probanden war, verschiedene Objekte in einer virtuellen Stadt zu suchen. Die quantitativen und qualitativen Ergebnisse zeigen, das die Vorhersage der Literatur in den Vorteilen wie Illusion der Eigenbewegung, räumliche Wahrnehmung und Orientierung, Spaß sowie Immersion und Präsenz nicht bestätigt werden können. Die Fortbewegung mit dem Joystick zeigte eine höhere Genauigkeit, Kontrolle und Benutzerfreundlichkeit, was die Vorhersage bestätigt [34].



Abb. 9: Sitzende, lehnungsbasierte Lokomotionstechniken [34]

#### Laufbänder

Laufbänder sollen das Gehen in jede Richtung simulieren, gerade wenn die virtuellen Umgebungen größer sind als der physische Raum. Für die Umsetzung sind verschiedene Konstruktionsschema möglich. Kommerzielle Produkte wie Kat Walk und Cyberith Virtualizer besitzen eine sehr reibungsarme Oberfläche, auf die der Benutzer steht und an der Hüfte fixiert ist, um die Fortbewegung zu kontrollieren und den Benutzer sicher an Ort und Stelle zu halten. Die Vorwärtsbewegung ist darauf einfach zu bewältigen, Bewegungen zur Seite jedoch nicht. Ebenso ist die Distanzmessung ungenau, da nur die Schritte gezählt werden und nicht die Entfernung. StriderVR ist ebenso ein kommerzielles Gerät und besteht aus einer Schicht aus Stahlkugeln. Omnideck ist eine Große Plattfirn aus Rollen um das Zentrum herum, wo der Benutzer automatisch wieder hin gezogen wird bei Inaktivität. Um eine omnidirektionale Fortbewegung zu ermöglichen, werden Interialmesseinheiten genutzt wie z.B. bei Infinadeck, Hex-Core-Prototype Virtuix Omni (siehe Abb. 10: Omnidirektionales Laufband Infinadeck (links) und HCP (rechts)). Diesen Laufbändern ermöglichen die Fortbewegung in 360 Grad, während der Benutzer in der Mitte gehalten wird [3] [35].



Abb. 10: Omnidirektionales Laufband Infinadeck (links) und HCP (rechts) [35]

In einer Studie verglichen Warren et al. die Benutzererfahrung zwischen einem Gaming Controller und dem omnidirektionale Laufband Virtuix Omni. Die Aufgabe der Probanden war es, mehrere Fortbewegungsaufgaben zu bewältigen. Die Ergebnisse zeigen, dass der Controller eine bessere Benutzererfahrung bietet. Die Gründe dafür sind, dass die Fortbewegung mit dem Virtuix Omni zwar natürlicher und auch unterhaltsamer ist, es aber den Gebrauch von einem unbequemen Gurt fordert und die Schwierigkeit beim Gehen und Drehen signifikant höher ist. Ebenso treten schnell Ermüdungserscheinungen auf. Dies zeigt, das semi-natürliche Fortbewegung, wie bei den Laufbändern, im Vergleich zu gut entwickelten Techniken wie dem Controller zu einer schlechteren Benutzererfahrung führen kann [36].

### Konditionierung

#### Grundlagen der Konditionierung

##### Klassische Konditionierung

Die Pawlowsche Konditionierung ist ein wertvolles Labor Modell. Diese beschreibt den Erwerb, Ausprägung, die Abstrahierung und den Verlust von bedrohungsbezogenen Verhalten. Die klassische Konditionierung nach Pawlow besagt, dass ein zuvor neutraler Reiz (CS, konditionierter Stimulus) mit einem unkonditionierten affektiv bedeutsamen Reiz (US, unkonditionierter Stimulus) assoziiert wird. Der neutrale Reiz CS kann bei z.B. ein schmerzhafter Reiz, ein Ton oder Bild sein. Der neutrale Reiz wird so zu einem konditionierten Reiz, da dieser die emotionalen Eigenschaften des zuvor ausgelösten unkonditionierten Reiz erhält. Bei dem Experiment von Pawlow wurden bei Hunden Futter als unkonditioniertem Stimulus US verwendet, was bei den Tieren Speichelfluss als unkonditionierter Reaktion auslösten. Dabei wurde ein Glockenton bei der Vergabe des Futters als konditioniertem Stimulus CS eingesetzt. Danach zeigten die Hunde allein beim Hören der Glocke eine konditionierte Reaktion (CR), die der dem Speichelfluss als unkonditionierte Reaktion (UR) ähnlich war [37]. Bei diesem Zustand löst der konditionierte Reiz allein die Reaktion aus. Dies kann vor allem auch eine Angst bzw. eine Abwehrreaktion wie Erstarren oder erhöhte Erregung sein. Mowrer stellte zusätzlich fest, dass die klassische Konditionierung durch die erworbene Erfahrung von Angst ein Vermeidungsverhalten hervorruft [38] [39].

##### Operante Konditonierung

Die operante Konditionierung ist ein lernpsychologisches Verfahren, was Skinner zu Grunde liegt. Kontextreize können dabei die Wahrscheinlichkeit des Auftretens einer Verhaltensweise, durch Verstärkung oder Bestrafung, verstärken oder vermindern. Dies wird weiter unterteilt in eine positive bzw. negative Verstärkung und eine positive bzw. negative Bestrafung [40].

Zusätzlich sorgt die operante Konditionierung für die Aufrechterhaltung der Angst. Aversive Assoziationen, die in der Evolution stattfanden, sind Grund einiger Phobien, wie die Angst vor Spinnen, oder Zwangsstörungen. Menschen weisen dabei eine unterschiedliche starke Konditionierbarkeit auf, was zur Folge hat das manche Menschen unter Angststörungen leiden und anderer wiederum nicht. Die operante Konditionierung ist ein Lernparadigma, was Skinner et al. zu Grunde liegt. Diese beschreibt die Veränderung des Antwortverhaltens durch eine angewendete positive oder negative Verstärkung [41] [39].

##### Kontextkonditionierung

Tiermodelle haben dabei das Wissen über die neuronalen stattfindenden Prozesse bei einer Angstkonditionierung erweitert. Diese Versuche deuten darauf hin, dass länger anhaltende Ängste, wie die posttraumatische Belastungsstörung, durch Kontextkonditionierung verursacht werden und der konditionierte Kontext dabei eine anhaltende Angstreaktion auslöst. Lau et al. kamen zudem zur Erkenntnis, dass eine große Furcht gegenüber dem konditionierten Stimulus ein Vermeidungsverhalten in einem bestimmten Kontext vorhersagt. Der Kontext besteht meist aus Details in der Umgebung wie Dinge zum Sehen, Gerüche oder Geräusche. Bei der Kontextkonditionierung wird ein unkonditionierter Reiz US (nicht durch einen konditionierten Reiz CS) ausgelöst, während die Person sich in einer bestimmten Umgebung befindet. Wenn es keinen eindeutigen bedrohungsauslösenden Reiz in der Umgebung gibt bzw. kein Hinweis auf die Vorhersage eines unkonditionierten Stimulus, entsteht ein Zustand der freischwebenden Angst bei Menschen. Der Kontext interagiert dabei auch mit der operanten Konditionierung, da ein konditionierter Stimulus einen unkonditionierten Stimulus ein einem Kontext vorhersagt, zu einem späteren Zeitpunkt in einem anderen Kontext dann ohne einen unkonditionierten Stimulus präsentiert werden. Die Kontextkonditionierung ist bei fehlenden unkonditionierten Stimulus besonders effektiv. Grillon et al. stellten fest, dass ein Kontext größere Angst und dadurch ein größeres Vermeidungsverhalten hervorruft, wenn ein unvorhersehbarer unkonditionierter Stimulus US in diesem Kontext auftrat im Vergleich zu einem vorhersehbaren unkonditionierten Stimulus. Ob eine Assoziation zwischen einem konditionierten Stimulus und dem Kontext vorliegt, wird Beim Testen in einem zweiten Kontext, dem Extinktionskontext geprüft, nachdem im ersten Kontext, dem Akquisitionskontext, die Kontextkonditionierung ausgelöst wurde. Bis heute wurden wenige Versuche beim Menschen und vermehrt bei Versuchstieren getestet, da der Laborversuchsraum als Kontext oft nicht das Ziel ist [42] [39] [38].

#### Konditionierung in Virtual Reality

Die Technologie im Bereich der Virtual Reality hat sich in den letzten Jahren signifikant verbessert. Zuvor war VR nicht weit verbreitet, da die Hardware sehr kostspielig, aufwendig zu installieren war und einen großen physischen Raum benötigte. Dies änderte sich vor allem durch die Veröffentlichung kommerzieller Geräte, die ab dem Zeitpunkt vor allem für den privaten Gebrauch konzipiert wurden. Dadurch ist der Einsatz in vielen Forschungslaboren möglich und so zunehmend nützlich für die Forschung im Bereich der Konditionierung beim Menschen. Der Vorteil von VR liegt darin, dass die Anwendung ein starkes Gefühl der Präsenz erzeugt. Dabei hat der Benutzer das vollkommende Gefühl vollständig in der virtuellen Welt zu sein und nicht mehr in der Realität. Der Begriff Immersion beschreibt dabei die Reduzierung der sensorischen Reize von der realen Umgebung und die einstimmende Bewegungsparallaxe bei der Kopfbewegung durch das Head Mounted Display. Aus diesen Gründen ist Virtual Reality eine gute Methodik, um den Einfluss und Stärke der Konditionierung beim Menschen zu untersuchen, ohne dabei den Verlust der experimentellen Kontrolle zu verlieren [39].

Glotzbach et al konnten frühere Studien bestätigen und zeigten mit Hilfe von VR, das durch Einsatz gezielter Konditionierung ein Vermeidungsverhalten bei einem Kontext auslöst, welches mit einer unvorhersehbaren Bedrohung assoziiert wird. In einem Experiment mussten Probanden zwei Kontexte betreten. In einem Angstkontext wurden leichte Elektroschocks verabreicht, in einem zweiten Kontext, dem Sicherheitskontext jedoch nicht. Danach wurde ein dritter, neutraler Kontext hinzugefügt und die Probanden mussten zwei von drei Kontexten erneut besuchen. Die Ergebnisse zeigten, dass die Probanden den Angstkontext gezielt vermieden [38].

### Unity 3D

Unity 3D ist eine Laufzeit- und für Spiele und andere 3D und anderer interaktive 3D-Grafik-Anwendungen. Ebenso das meistgenutzte Tool im Bereich VR-Entwicklung. In Szenen werden die 3D Umgebungen erstellt. Diese besteht zum größten Teil aus Objekten, Materialien bzw. Texturen, Animationen, Audio und Lichtquellen. So entsteht eine virtuelle Welt [43].



Abb. 11: Unity Editor Übersicht

Das Interface des Unity Editors besteht aus verschiedenen Bereichen (siehe Abb. 11: Unity Editor Übersicht), die im Folgenden genauer erläutert werden:

1. Die **Toolbar** im oberen Bereich des Editors beinhaltet die wichtigsten Funktionen. Auf der linken Seite befinden sich Werkzeuge für die Bearbeitung der Szenenansicht und der Objekte innerhalb der Szene. In der Mitte befinden sich Schaltflächen zum Abspielen, Pausieren und Stoppen der Simulation. Am Rand der rechten Seite befindet sich der Zugriff auf Unity Collaborate, den Unity Cloud Services und dem Unity Account. Ebenso befindet sich jeweils den Menüpinkt für die Sichtbarkeit der Ebenen und dem Editor Layout Menü [44].
2. Im **Hierarchy** Fenster im linken mittleren Bereich befindet sich eine hierarchische Darstellung jedes GameObjects, die sich in der aktuellen Szene befinden, und zeigt dabei die Struktur auf, wie die verschiedenen GameObjects miteinander in Verbindung stehen Die **Toolbar** im oberen Bereich des Editors beinhaltet die wichtigsten Funktionen. Auf der linken Seite befinden sich Werkzeuge für die Bearbeitung der Szenenansicht und der Objekte innerhalb der Szene. In der Mitte befinden sich Schaltflächen zum Abspielen, Pausieren und Stoppen der Simulation. Am Rand der rechten Seite befindet sich der Zugriff auf Unity Collaborate, den Unity Cloud Services und dem Unity Account. Ebenso befindet sich jeweils den Menüpinkt für die Sichtbarkeit der Ebenen und dem Editor Layout Menü [44].
3. Die **Game** Ansicht in der Mitte des Editors zeigt eine Vorschau, wie das fertig gerenderte Spiel durch die aktive Kamera der Szene aussehen wird. Beginnt die Simulation durch das Betätigen der Play Schaltfläche, wird die gerenderte Simulation des Spiels hier angezeigt [44].
4. Die **Scene** Ansicht bietet eine visuelle Navigation durch die Szene und deren Bearbeitung [44].
5. Das **Inspector** Fenster auf der rechten Seite zeigt eine Übersicht und Bearbeitungsmöglichkeiten aller Eigenschaften des ausgewählten GameObjects [44].
6. Das **Project** Fenster in der unteren Hälfte des Editors zeigt die Bibliothek aller importierten Assets im Projekt, die genutzt werden können. Ebenso können hier eigene Objekte wie Skripte, Prefabs usw. erstellt und geordnet abgelegt werden [44].
7. Die **Statusleiste** am unteren Rand zeigt Benachrichtigungen von den Unity Prozessen an [44].

### Oculus Quest

Bei der Oculus Quest handelt es sich um eine mobile VR Brille. Sie kann also genutzt werden, ohne zusätzlich einen leistungsstarken Computer oder Konsole zu benötigen. Ebenso werden keine externen Sensoren oder Kameras für das Tracking gebraucht. Die Auflösung des Displays ist 1.600x1.440 Pixel pro Auge mit einer Wiederholrate von 72Hz. Das Tracking passiert über die Brille selbst. Auf dieser befinden sich an der Vorderseite 4 Kameras, die mit Weitwinkel arbeiten. Mit 6DoF, in Englisch Six Degrees of Freedom, ist die Nutzung aller sechs Freiheitsgrade möglich. Somit stehen viele Steuermöglichkeiten zur Verfügung und eine Bewegung im virtuellen Raum gegeben. Dafür elementar ist die Bedienung mit den zwei dazugehörigen Controllern (siehe Abb. 12: Oculus Quest mit Controller). [45]



Abb. 12: Oculus Quest mit Controller [46]

### HTC Vive

Die erste HTC Vive wurde in der Zusammenarbeit mit Valve (SteamVR) und HTC im April 2016 für den kommerziellen Verkauf mit der Consumer Edition freigegeben. Neben dem Spiele Bereich findet die VR-Brille auch Einsatz in den Bereichen wie Unterhaltung, Bildung, Gesundheitswesen oder auch im Automobilbau [47].

Im Jahr 2019 wurde die HTC Vive Pro Eye veröffentlicht, welche eine verbesserte Auflösung bot. Ebenso befinden sich an der Vorderseite der Brille zwei nach vorne gerichteten Kameras, welche die Umgebungserkennung verbessern [48].

Neben dem Headset, werden zwei Controller und mindestens zwei Basisstationen zum Tracking benötigt. Auf diese Komponenten wird im Folgenden genauer eingegangen.



Abb. 13: HTC Vive Pro Komponenten [49]

#### Display

Beim Display handelt es sich um zwei OLED Displays mit der Größe von 3.5 Zoll Bilddiagonale mit einer Auflösung von 1440 x 1600 Pixel pro Auge. Die Bildwiederholungsrate beträgt 90Hz bei einem Gesichtsfeld von 110 Grad [50].

#### Controller

Über die beiden Controller wird eine Interaktion mit der virtuellen Umgebung ermöglicht. Die Controller besitzen verschiedene Tasten und Schnittstellen, auf die mit Hilfe der Abb. 14: HTC Vive Controller genauer eingegangen wird [51] .

1. Mit der **Menütaste** kann jeder Zeit auf das Menü zugegriffen werden [51].
2. Über das **Trackpad** können Eingaben nach oben, unten, links und rechts getätigt werden [53].
3. Das Einschalten, Ausschalten oder Koppeln des Controllers kann über die **System Taste** umgesetzt werden [51].
4. Die **Statuslampe** zeigt die verschiedenen Zustände des Controllers an. Die Farbe Grün bedeutet, das sich der Controller aktuell im normalen Modus befindet, roten Blinken wenn der Akku fast leer ist, blaues Blinken beim Koppeln des Controllers mir dem Headset. Das dauerhafte blaue Leuchten signalisiert das erfolgreiche Verbinden mit dem Headset und Orange ist beim Auflade Vorgang aktiv [51].
5. Der **Mico-USB-Anschluss** kann der Akku des Controllers mit Hilfe eines USB-Kabels aufgeladen werden [51].
6. Über die **Verfolgungssensoren** wird die Position und getätigten Eingaben durch die angebrachten Basisstationen verfolgt [51].
7. Die **Trigger Taste** wird vor allem für die Interaktion mit der virtuellen Umgebung benutzt und kann mit dem Zeigefinger betätigt werden [51].
8. Die **Griff Taste** wird vor allem zum Greifen von Objekten genutzt und kann mit dem Mittel- oder Ringfinger ausgelöst werden [51].



Abb. 14: HTC Vive Controller

#### Tracking

Die SteamVR Basisstationen 2.0 sind für das Tracking im physischen Raum zuständig. Die Basisstationen werden so außerhalb des Spielbereichs angebracht, das die Vorderseiten mittig in den Spielbereich zeigen. So werden Signale an die VR Brille und den dazugehörigen Controller geleitet und die Positionen bestimmt [52].

Die Basisstationen basieren auf der von Valve und HTC entwickelten SteamVR Tracking 2.0 Ortungstechnologie. Der physische Raum wird durch die Basisstationen mit Laserstrahlen von bis zu 5m und Synchronimpulsen mit einer Aktualisierungsrate von 1000 Hz abgescannt. Damit lässt sich die Position und Rotation jedes Sensors millimetergenau im Raum orten. Ebenso erlaubt die Technoligie den Einsatz von mehr als 2 Basisstationen unterstützt. Damit ist ein Spielbereich mit einer Größe von 10m x 10m bei dem Einsatz von 4 Basisstationen möglich [52] [53].

#### Vive WLAN Adapter

Durch die Zusammenarbeit mit Intel® WiGig, bietet Vive einen kabellosen WLAN Adapter an. Damit wird eine ungehinderte Bewegung im physischen Raum ermöglicht, da ein Kabel für die Schnittstelle zwischen der HTC Vive und einem PC damit nicht mehr benötigt wird. Der Adapter überstützt die SteamVR Basisstationen 2.0 [54].

Um den WLAN Adapter nutzen zu können, werden mehrere grundlegende Hardwarekomponenten benötigt (siehe Abb. 15: Vive WLAN Adapter Komponenten). Zum einen der WLAN Adapter mit der Halterung für die HTC VIVE, die PCIe WiGig Karte und die WLAN Linkbox als Schnittstelle zu dem PC und die HTC QC 3.0 Powerbank zur Stromversorgung des Adapters [55].



Abb. 15: Vive WLAN Adapter Komponenten [55]

## Konzept

## Umsetzung

Dieses Kapitel ist in 5 Unterkapiteln unterteilt. In den ersten beiden Unterkapiteln wird die Einbindung der VR-Brillen Oculus Quest und HTC Vive beschrieben. Im nächsten Unterkapitel geht es in der Entwicklung der Szenarien. Dieses Kapitel enthält neun weitere Unterkapitel in denen die Hindernisse, Implementierung der Wegerkennung, Szenarien, Entwicklung des Malus, Positive Verstärkung, Datenerfassung, Aufgabe der Szenarien, Implementierung des Menüs und die Datenerfassung behandelt werden. In den letzten beiden Kapiteln wird auf die Datenerfassung der quantitativen und qualitativen Daten und auf das letztes Thema Evaluation genauer eingegangen.

Für die Umsetzung wurde die Oculus Quest für die Entwicklung und die HTC Vive für die Datenerhebung eingesetzt. Zusätzlich wurde für die Entwicklung der Software die Entwicklungsumgebung Unity 3D mit der Version 20194.15 und Visual Studio 2019 verwendet.

### Einbindung der Oculus Quest

Um selbst entwickelte grafische Anwendung bzw. Spiele in Unity 3D ausführen oder testen zu können, muss eine aktive Verbindung von der Oculus Quest zum Computer und der Entwicklungsumgebung hergestellt werden. Bei der Entwicklungsumgebung handelt es sich wie zuvor erwähnt um Unity 3D in der Version 20194.15.

#### Schnittstelle zwischen PC und Oculus Quest

Um die Verbindung zum PC erfolgreich herstellen zu können, wurde die Software Oculus Link benötigt. Diese stellt die Schnittstelle zum PC her und stellt Oculus Rift Inhalte zur Verfügung. Diese wird direkt von Oculus online angeboten. Erforderlich ist ebenso das Aktivieren der Oculus Quest Brille und der Oculus Link Software mit einem aktiven Facebook Account, um die beiden Komponenten erfolgreich miteinander koppeln zu können (siehe Abb. 16: Gekoppelte Quest in Oculus Link). So kann die Brille von Unity 3D erkannt werden, und die Oculus Quest im nächsten Schritt in die Entwicklungsumgebung eingerichtet werden.



Abb. 16: Gekoppelte Quest in Oculus Link

#### Einbindung in Unity

In Unity 3D ist es nötig, den Oculus Support einzurichten. Dazu gab es zwei Möglichkeiten, welche ihre Vor- und Nachteile bieten. Das veraltete XR-Plugin Tool ist schon seit einiger Zeit standardmäßig integriert und bietet dazu die Möglichkeit, OpenVR zu nutzen. Dies wird benötigt, wenn weitere VR-Brillen in der Entwicklungsumgebung genutzt werden sollen. Ein Beispiel ist die im Projekt ebenso genutzte HTC Vive. Das XR-Management soll das XR-Plugin in Zukunft ablösen und eine zentrale Implementierung der meisten VR-Brillen ermögliche. Diese befindet sich jedoch erst in den Anfängen, und bietet noch keine Unterstützung, lediglich eine Betaversion für OpenVR an. Da die Entwicklung zu Hause mit der Oculus Quest durchgeführt wurde, die Evaluation jedoch mit der leistungsstärkeren HTC Vive, wurde die Umsetzung mit dem XR-Plugin umgesetzt. In den XR-Settings, die sich innerhalb der Player Einstellungen befinden, wurde Oculus als Virtual Reality SKD hinzugefügt und der Haken unter Virtual Reality Supported gesetzt (siehe Abb. 17: XR Einstellungen in Unity).



Abb. 17: XR Einstellungen in Unity

Ebenso benötigt wurde das Asset Oculus Integration, welches sich über den integrierten Asset Store finden und importieren lässt. Beinhaltend sind einige Kernfunktionen für VR, Komponenten, Prefabs und Plugins, um grundlegende Interaktionen wie das Umschauen und das Gehen in der Virtuellen Welt zu ermöglichen und erleichtern. Bei Prefabs handelt es sich um wiederverwendbare Objekte wie z.B. Formen, Lichter oder eine Kamera. Zur Nutzung werden diese per Drag & Drop in die Szene gezogen und können beliebig erweitert werden. Für die Umsetzung der Interaktion mit der Oculus Quest und der virtuellen Welt wurde das Prefab OVRPlayerController verwendet. Im Gegensatz zum ebenso Verfügbaren Prefab OVRCameraRig, die als stationäre VR Kamera dient, sind zusätzlich Komponenten für die Steuerung und ein Character Controller für die Fortbewegung, Room-Scale oder mit dem Controller, in der Virtuellen Umgebung enthalten (siehe Abb. 18: OVRPlayerController) [56].



Abb. 18: OVRPlayerController

Um einen augelösten Malus visuell sehen zu können, wurde zusätzlich eine Kamera UICamera unterhalb des Objekts CenterEyeAnchor innerhalb des OVRPlayerController Prefabs hinzugefügt.

#### Ausführung der Anwendung

Ist die Oculus Quest erfolgreich mit Oculus Link verbunden, kann die VR-Brille direkt in der Unity Entwicklungsumgebung über den Play Mode ausgeführt werden. Dies wird vor allem in der Entwicklung benötigt. Die Anwendung kann jedoch direkt auf der VR Brille ausgeführt werden, da ohne Kabel und angeschlossenen Computer ein freieres Bewegen in der virtuellen Umgebung möglich ist. Dazu wurden in den Grafikeinstellungen einige Änderungen vorgenommen und die Plattform wurde zu Android gewechselt. Ebenso wurde der Account als Developer freigeschalten und Entwicklermodus der Oculus Quest aktiviert.

### Einbinden der HTC Vive

Für die Entwicklung des Projekts wurde die Oculus Quest genutzt. Für die spätere Durchführung der Evaluation und des Durchlaufs der Szenarien mit den Probanden wurde die HTC Vive eingesetzt. Wie in Kapitel 2.6 ersichtlich, besitzt die VR-Brille eine bessere Hardware, besseres Tracking und unterstützt eine kabellose Schnittstelle, um die Anwendung in Unity 3D im Play Mode und gleichzeitig frei im Raum bewegend, laufen lassen zu können. Dazu ist im Labor das benötigte Tracking bereits angebracht, und somit das erneute Einrichten des Tracking Bereichs bzw. des Guardians der Oculus Quest nicht mehr notwendig und somit in der Umsetzung unkomplizierter.

#### Schnittstelle zwischen PC und SteamVR

Um die Schnittstelle zwischen der HTC Vive und dem PC herzustellen, wurde die Software SteamVR verwendet. Über die Plattform Steam wurde die aktuelle Version von SteamVR herunterladen und installiert (siehe Abb. 19: SteamVR auf der Steam Plattform). Hierzu ist ein Benutzerkonto zur Anmeldung bei Steam erforderlich. Somit ist die erfolgreiche Erkennung der VR Hardware am PC gewährleistet.



Abb. 19: SteamVR auf der Steam Plattform

#### Einbindung in Unity

Um SteamVR und die dazugehörigen VR Hardware in Unity nutzen zu können, wird das SteamVR Plugin Asset benötigt. Das Plugin ist eine Schnittstelle zu SteamVR und verwaltet das Anzeigen von 3D Modellen für die VR Controller und Hände, die Verarbeitung von Benutzereingaben über diese Controller und Interaktionssysteme, welche mit Prefabs (siehe 14.1.2) und Skripte implementiert werden. Das Plugin wurde über den integrierten Asset Store von Unity heruntergeladen und installiert [57].

In den XR-Settings, die sich innerhalb der Player Einstellungen befinden, wurde OpenVR als Virtual Reality SDK hinzugefügt (siehe Abb. 17: XR Einstellungen in Unity). Der Haken unter Virtual Reality Supported musste nicht mehr gesetzt werden, da dies schon in Kapitel 4.1.2 während der Einbindung der Oculus Quest in Unity umgesetzt wurde.

Für die grundlegenden Interaktionen zwischen der HTC Vive und der virtuellen Welt, wurde das Player Prefab verwendet, was innerhalb des SteamVR Plugin Asset zur Verfügung steht (siehe Abb. 20: Player Prefab). Dies bindet grundlegende Eingabeaktionen ein und ermöglicht eine physische Fortbewegung durch den virtuellen Raum.



Abb. 20: Player Prefab

Unterhalb der Komponente FollowHead befindet sich eine weitere Komponente HeadCollider. Hier befindet sich ein Collider mit dazugehörigem Rigidbody, welches die Position des HDM besitzt und sich bei Bewegung aktualisiert. An dieser Stelle wurden die entwickelten Skripte für die Implementierung des Malus und alle Funktionalitäten der Szenarien hinzugefügt und angepasst.

Für die grundlegenden Interaktionen mit dem Controller, werden die Standardeinstellungen in der Binding UI, welches sich im unter dem Menüpunkt SteamVR Input befindet, übernommen und gespeichert. Hierdurch sind spätere Aktionen wie das Teleportieren durch den virtuellen Raum oder die Interaktionen mit dem Menü zur Auswahl der Szenarien (siehe 4.3) möglich.

Da mit dem Player Prefab keine grundlegende Unterstützung zur Fortbewegung mit dem Controller zu Testzwecken möglich ist, wurde dies mit Hilfe des Teleport umgesetzt. Dazu musste das im SteamVR Assets befindliche Prefab Teleport in die Szene gezogen werden. Zusätzlich wurde eine Plane in derselben Höhe des Bodens erstellt, auf welches das Skript TeleportArea hinzugefügt wurde. So ist es möglich sich mit Hilfe des Controllers durch den virtuellen Raum zu bewegen, da während der Entwicklung zur meisten Zeit nicht genügen physischer Raum zur Verfügung stand (siehe Abb. 21: Fortbewegung durch Teleport).

Das SteamVR\_Play\_Area Skript wurde dem Player Prefab als weitere Komponente hinzugefügt. Da die Fortbewegung bei der Durchführung der Szenarien durch physisch Fortbewegung umgesetzt wird, lassen sich mit dieser Komponente die Grenzen des physischen Raums und der Startpunkt nach dem Start des Szenarios einstellen.



Abb. 21: Fortbewegung durch Teleport

Um einen augelösten Malus visuell sehen zu können, wurde wie zuvor in Kapitel 4.1.2, zusätzlich eine Kamera UICamera unterhalb des Objekts VRCamera innerhalb des Player Prefabs hinzugefügt.

### Entwicklung der Szenarien

Der zentralste Bestandteil des Projekts ist das Entwickeln der Szenarien. In diesen soll der Proband in der virtuellen Umgebung verschiedene Objekte finden, welche sich verteilt auf einem Weg befinden. Der Weg wird mit schwarzen Matten geformt. Die Form des Weges ergibt ein Rechteck, worin sich auf kleinen Seitenarmen die verschiedenen Objekte befinden (siehe Abb. 22: Überblick des Szenarios). Ebenso befindet sich am Anfang des Szenarios ein kleiner Weg, auf dem der Benutzer entlangläuft, um das Szenario zu beginnen oder zu beenden. Über diesem Weg wird auch das Menü zur Auswahl des Szenarios angezeigt.



Abb. 22: Überblick des Szenarios

#### Hindernisse

Die Schwierigkeit dabei ist das sich verschiedene Hindernisse auf dem Weg befinden. Diese Umgebung ist in allen Szenarien gleich. Zu Beginn des Weges bilden ein Karton und eine Kiste eine Gasse, durch die der Proband durchlaufen muss. Nach diesem Hindernis steht ein Tisch halb im Weg hinein, an dem man vorbeikommen muss (siehe Abb. 23: Hindernisse Kisten und Tisch).



Abb. 23: Hindernisse Kisten und Tisch

Daraufhin bilden zwei hohe Zäune eine Gasse und ein liegender Pfosten, quer liegend über dem Weg, die Hindernisse (siehe Abb. 24: Hindernisse Zaun und PfahlAbb. 24: Hindernisse ). Bei diesen Gegenständen besteht entweder die Möglichkeit diese zu Umgehen oder durch Hindurchlaufen bzw. drüber steigen zu bewältigen.



Abb. 24: Hindernisse Zaun und Pfahl

Es wurde ein Skript ObstacleMonitor erstellt, um zu überprüfen, ob und durch welche Hindernisse der Benutzer gelaufen ist. Dazu wurde bei jedem Hindernis ein Box Collider als Komponente hinzugefügt und die höhe auf 2 Meter angepasst. Ebenso wurde für jedes Hindernis ein eindeutigen Tag erstellt, um bei einer Kollision mit einem Hindernis feststellen zu können, um welches Hindernis es sich handelt.

Wie in Abb. 25 zu erkennen ist, wird im ObstacleMonitor Skript in der OnTriggerEnter Methode bei einer Kollision anhand der Tags überprüft, um welches Hindernis es sich handelt und im zentralen Skript DataRecorder zum Halten der Daten abgespeichert.



Abb. : Obstacle Monitor Skript

#### Implementierung des Weges

#### Implementierung der Wegerkennung

Um zu erkennen, ob sich der Benutzer auf dem Weg befindet oder nicht, wurden Box Collider implementiert (siehe Abb. 25: Box Collider für die Wegerkennung). Collider sind unsichtbare Objekte, die dafür zuständig sind, physische Kollisionen zu bemerken. Diese können genau dieselbe Form von Objekten haben, die Größe und Form kann jedoch auch unabhängig sein [58].

Die Box Collider wurden an leere GameObject Objekte gehängt, und an die Größe der Matten angepasst. Die Höhe der aller Collider beträgt 2m, da sich der Collider des Benutzers auf der Höhe der VR Brille befindet. Um die Kollision mit diesen Collider erkennen zu können, wurde der Tag Way erstellt und alle Collider mit diesem Tag versehen. Bei einem Tag handelt sich es um eine Referenzwort, welches man Objekte zuweisen kann. Durch diesen Tag lassen sich diese Objekte genau identifizieren und werden vor allem für Kollisionen von zwei oder mehrerer Collider verwendet [59]. Zu beachten ist hierbei, dass sich die Collider überschneiden um das kurze Auslösen eines Malus beim Übergang zwischen zwei u verhindern.



Abb. 26: Box Collider für die Wegerkennung

Um die Kollisionen erfassen zu können, wurde in der Rigidbody Komponente die IsKinematic Einstellung, und in der Komponente Sphere Collider die IsTrigger Einstellung aktiviert. Dies bewirkt, dass das Objekt HeadCollider keine Physik benutzt, sondern ein Trigger ist, welche bei einer Kollision mit anderen Objekt Collider, Trigger Funktionen ausführen kann.

Da beim OVRPlayerController der Controller die Implementierung des Colliders und des Rigidbodys übernimmt, mussten hier keine Einstellungen vorgenommen werden.



Abb. 27: HeadCollider Konfiguration

Um Trigger Funktionen beim Verlassen oder Betreten des Weges auszulösen, wurde ein Player Skript erstellt. Dies ist das zentrale Skript und wurde dem Player bzw. OVRPlayerController Objekt als Komponente hinzugefügt. Da zur Wegerkennung mehrere Collider verwendet wurden, zählt die Variable colliderActive, ob der Benutzer beim Verlassen eines Colliders sich noch auf dem Weg befindet oder nicht.

(Um zu erkennen, ob der Benutzer beim Start des Szenarios den Weg betreten hat, wird ein Objekt WayColliderStart erstellt. Dieses enthält ein Box Collider mit dem Tag WayStart (siehe Abb. 27: WayStart Collider am Anfag des Weges). )



Abb. 28: WayStart Collider am Anfag des Weges

#### Szenarien

Den Szenarien unterscheiden sich grundsätzlich im zu laufenden Weg. Im physischen Szenario stimmt der visuelle mit dem physischen Weg überein. Im visuellen Szenario unterscheidet sich der visuelle, richtige Weg vom physischen Weg (siehe Abb. 28: Visuelles Szenario). Der Standardszenario ich gleichzusetzen mit dem physischen, unterscheidet sich jedoch im fehlenden negativen Malus.



Abb. 29: Visuelles Szenario

Es werden drei Game Objekte Standard, Physical und Visual erstellt. Darunter befinden sich der dementsprechende Weg mit den dazugehörigen Collidern des Weges (siehe Abb. 29: Scenario Objekte). Um das gewünschte Szenario zu wählen, wird das dazugehörige Objekt vor dem Start der Anwendung aktiviert und die anderen deaktiviert. Dies ist vor allem für Entwicklungszwecke sinnvoll.



Abb. 30: Scenario Objekte

Das Objekt WayDummy enthält den gleichen Weg wie das Standardszenario und das physische Szenario, jedoch ohne die Collider. Dieses Objekt ist beim Start der Anwendung aktiviert und wird deaktiviert, sobald ein Szenario gewählt wurde.

Das Skript ScenarioHandler, welches wie die anderen Skripte als Komponente im OVRPlayerController bzw. im HeadCollider des Player Objekts hinzugefügt wurde, ist dazu da, um die verschiedenen Szenarien zu verwalten. Es enthält Methoden, zum Aktivieren oder Deaktivieren einzelner Szenarien, aktivieren eines zufällig gewählten Szenarios, Speichern des aktuellen Szenarios oder zum Deaktivieren des WayDummy nach Auswahl eines Szenarios. Die Hauptaufgabe des Skriptes besteht jedoch darin, zum Start ein Szenario zu aktivieren und nach Beendigung des Szenarios dieses zu deaktivieren.

Um das gewünschte Szenario auszuwählen, wurde die Variable scenario erstellt. Wie in Abb. 30: Scenario Handler Variablen zu erkennen, wird die Auswahl über die numerischen Werte zwischen 0 und 2 angegeben. Dabei stehen die Werte für folgende Szenarien:

* 0: Baseline
* 1: Szenario 1
* 2: Szenario 2

Das ausgewählte Szenario wird automatisch geladen, sobald Start über das Menü ausgewählt wird und das Szenario beginnt.



Abb. 31: Scenario Handler Variablen

#### Entwicklung des Malus

Ziel ist es, einen visuellen und akustischen Malus auszulösen, sobald der Proband vom Weg abkommt. Visuell sichtbar sein soll ein rotes, pulsierendes Blinken im kompletten Sichtfeld des Benutzers. Ebenso soll neben dem visuellen Reiz ein akustisches Piepen ertönen. Die Frequenz beide Stimuli ist synchron und wird bei steigender Dauer, genauer gesagt alle 5 Sekunden, schneller.

##### Erstellen des Canvas

Für die Umsetzung in Unity 3D wurde ein Canvas Objekt implementiert, welches automatisch eine EventSystem Komponente erstellt. Dies ist dazu da um UI Elemente wie z.B. Bilder, Buttons für ein Menü oder Text anzeigen zu lassen. Diese müssen als Unterkomponente eingefügt werden. Die Canvas können in zwei grundlegenden Render Einstellungen laufen, in Screen Space oder World Space. In Screen Space Modus wird das Canvas und deren UI Elemente direkt auf dem Bildschirm, über der Szene überlagernd dargestellt. In World Space dagegen werden die Elemente wie jedes andere 3D Objekt auch als Teil der virtuellen Umgebung platziert. Für die Umsetzung des Malus wurde die Screen Space Camera Einstellung implementiert, da es in VR sonst zu Darstellungsproblemen kommen kann. Diese ist ähnlich wie der Screen Space Modus, benutzt jedoch zusätzlich eine Kamera für die Darstellung der UI Elemente. Hier wurde die entsprechend erstellte Kamera des OVRPlayerController bzw. des Players in das Canvas Objekt eingefügt. Ebenso muss die Kamera und das Canvas Objekt derselbe Order in Layer besitzen und wurde auf den Wert 0 gesetzt (siehe Abb. 31: Render Modus der Canvas). Dies ist wichtig, da sonst Probleme mit der späteren Interaktion mit der Canvas des Menüs auftreten können, da das Canvas des Malus hinter diesem liegen muss. Wie zuvor in Kapitel 4.1.2 und 4.2.2 beschrieben, wurde zur erfolgreichen Darstellung in der virtuellen Umgebung eine jeweils weitere Kamera UICamera im Player und OVRPlayerController hinzugefügt. Diese Kamera muss nun im Objekt CanvasMalus eingebunden werden (siehe Abb. 31: Render Modus der Canvas).



Abb. 32: Render Modus der Canvas

Unter dem Canvas wurde ein Image Objekt ImageMalus erstellt und dessen Größe an das Canvas angepasst, so dass diese das komplette Sichtfeld ausfüllt. Da der ausgelöste Malus rot blinken soll, wurde die Farbe des Images auf Rot mit dem Hexadezimalen Wert C00000 konfiguriert. Der Alpha Wert wurde auf 0 gesetzt, sodass die Farbe des Images nicht sichtbar ist, solange der Malus nicht aktiv ist.

##### Erstellen der Animation

Für die Umsetzung des Malus wurde ein Animator Controller ImageController erstellt. Ein Animator verwaltet eine oder mehrere Animationen mit einem Zustandsautomat, eine Art Flussdiagramm mit Animationen und deren Zustände, Übergänge und Abfolge für z.B. Charaktere oder Objekte [60]. Über den Animator können die Animationen mit Hilfe eines Skripts aufgerufen werden.

Unter dem Objekt ImageMalus wurde eine Animator Komponente hinzugefügt, welche den ImageController einbindet. Für die Erzeugung des Impulses mit dem ImageMalus Objekt wurde ein Animation Objekt Impulse erstellt, welches in Abb. 32 zu sehen ist.

Diese nimmt den Alpha-Wert des Images und verändert diesen, um so einen optischen Impuls zu erzeugen. Ein Intervall des Impulses hat die Länge von einer Sekunde, wobei der Alpha Wert bei 0,05 Sekunden auf Wert 0,3 gesetzt wird. So ist die Farbe Rot für den Benutzer sichtbar und nimmt bis zum Ende des Intervalls ab. Im Inspektor der Animation wurde der Haken bei Loop Time gesetzt, sodass sich der Impuls fortlaufend wiederholt, bis dieser im Skript Player deaktiviert wird (siehe 4.3). Ebenso wird im genannten Skript die Geschwindigkeit der Animation im Animation Controller alle 5 Sekunden erhört. Die Standardgeschwindigkeit hat den Wert 1.



Abb. 33: Animation des Images

Neben dem visuellen Reiz des Impulses wird zusätzlich noch einen akustischen Reiz erzeugt. Dazu wurde eine AudioSource Komponente dem Image Objekt ImageMalus hinzugefügt, die dazu benutzt wird, um die Audiodatei einzubinden und per Skript abspielen zu können. Bei dem Audio handelt es sich um ein Piep Ton, welches frei im Internet zur Verwendung zur Verfügung steht. Um das Audio zeitgleich mit dem visuellen Impuls abspielen zu lassen, wurde ein Event PlayAudioSource in der Animation erstellt. Dieses Event wird zeitgleich mit dem höchsten Alpha Wert des roten Impulses aufgerufen. Dazu wurde zusätzlich ein Skript PlayAudio dem Image Objekt hinzugefügt, welches das Event implementiert, die AudioSource Komponente aufruft und das Audio abspielt (siehe Abb. 33). Wie in Abb. 32 und Abb. 33 ersichtlich, muss dabei der Name der Methode gleich dem Namen des erstellen Events sein, um das Event zu implementieren.



Abb. 34: PlayAudio Skript

Um die Animation über den Animator Controller abspielen zu können, wurde die Animation Impulse in den Animator Controller per Drag & Drop hineingezogen (siehe Abb. 34: Animator ImageController). Ein Übergang zwischen dem Objekt Any State und der Animation wurde erstellt. Durch den Any State Zustand kann die Impulse Animation ausgeführt werden, unabhängig davon welcher Zustand gerade aktiv ist. Da die Animation beim Deaktivieren durch die Rebind Methode beendet und neu gebunden wird, ist hier kein anderer Zustand wie z.B, der Idle Zustand bei Inaktivität notwendig[61].



Abb. 35: Animator ImageController

##### Implementierung

Zum Aktivieren oder Deaktivieren des Malus, wie zuvor in erläutert, wird das Skript Player genutzt (siehe Abb. 35: Malus Animator in Player Skript). Sobald der Benutzer den Weg verlässt, wird der Malus über den Animator in der Methode OnTriggerExit aufgerufen und abgespielt. Die Geschwindigkeit des Malus wird dabei durch die Koroutine TimerCoroutine alle 5 Sekunden verdoppelt. Eine Koroutine ist eine Art Funktion, die ihren Ablauf jedoch im Vergleich unterbrechen und zu einem späteren Zeitpunkt wieder aufnehmen können [62]. Betritt der Benutzer den Weg wieder, wird der Animator in der OnTriggerEnter Methode neu gebunden. Das Abspielen des Malus wird dadurch gestoppt und die Daten, wie die Geschwindigkeit des Malus werden zurückgesetzt.



Abb. 36: Malus Animator in Player Skript

#### Positiver Verstärker

Sobald der Benutzer ein Objekt erreicht und mit dem Controller greift oder am Ende des Szenarios den Startpunkt erreicht, wird dies mit einem grünen optischen Impuls und einem Audio signalisiert.

##### Erstellen der UI Elemente

Wie schon in Kapitel 4.3.5.1 beschrieben, wird der optische Reiz mit deinem Canvas umgesetzt. Unter dem CanvasMalus Objekt wurde ein Image Objekt ImageFinish erzeugt und die Größe an das Canvas angepasst. Da der Impuls im Gegensatz zum Malus einmalig grün aufleuchten soll, wurde die Farbe des Images auf Grün mit dem Hexadezimalcode 1BD900 eingestellt und der Alpha Wert auf 0 gesetzt.

##### Erstellen der Animation

Ebenso wie zuvor in Kapitel 4.3.5.2, wurde für die Implementierung eine Animation verwendet. Ein Animator Controller ImageFinishController und eine Animation FinishImpulse wurden erstellt. Die Länge des Impulses dauert eine Sekunde. Für die Erzeugung des optischen Impulses wurde der Alpha Wert nach 0,05 Sekunden auf den Wert 0,3 gesetzt. So ist die Farbe Grün für den Benutzer sichtbar und nimmt bis zum Ende des Zeitzyklus wieder gleichmäßig ab, was einen optischen Impuls erzeugt.

Neben dem visuellen Reiz des Impulses wird zusätzlich noch einen akustischen Reiz erzeugt. Dazu wurde eine AudioSource Komponente dem Image Objekt ImageFinish hinzugefügt, die dazu benutzt wird, um die Audiodatei einbinden und per Skript abspielen zu können. Bei dem Audio handelt es sich um ein Level-Up Ton, welches frei im Internet zur Verwendung zur Verfügung steht. Um das Audio zeitgleich mit dem visuellen Impuls abspielen zu lassen, wurde ein Event PlayAudioSource in der Animation erstellt. Dieses Event wird zeitgleich mit dem höchsten Alpha Wert des roten Impulses, also bei 0,05 Sekunden, aufgerufen. Dazu wurde zusätzlich, wie auch zuvor in Kapitel 4.3.5.2 beim Malus, ein Skript PlayAudio erstellt dem ImageFinish Objekt hinzugefügt. Das Skript implementiert das Event der Animation, ruft dabei die AudioSource Komponente aufruft und spielt das Audio ab. Hier ist ebenso zu beachten, dass der Name der Methode gleich dem Namen des erstellen Events sein muss, um das Event zu implementieren.

Um die Animation über den Animator Controller abspielen zu können, wurde die Animation FinishImpulse per Drag & Drop in den Animator hineingezogen (siehe Abb. 36: Animator ImageFinishController). Ein neues State Idle wurde erzeugt und ein Übergang von FinishImpulse Animation zum Idle State erstellt. Wird die Animation nicht abgespielt, ist der Idle State aktiv. Sobald die Animation abgespielt wird, wechselt der aktive Zustand von Idle zum FinishImpulse und führt die Animation einmalig aus Nach Beendigung der Animation, wird der Zustand durch den Übergang automatisch wieder zum Idle State gewechselt. So kann die Animation beliebig oft abgespielt werden.



Abb. 37: Animator ImageFinishController

#### Zentrale Datenspeicherung

Das Skript ScenarioHandler, welches wie die anderen Skripte als Komponente im OVRPlayerController bzw. im HeadCollider des Player Objekts hinzugefügt wurde.

Für die zentrale quantitativen Datenerfassung und Datenspeicherung bei der Durchführung der Szenarien, wurde das Skript DataRecorder erstellt und wie die meisten anderen Skripte unter dem OVRPlayerController bzw. im HeadCollider des Player Objekts als Komponente hinzugefügt. Die darin enthaltenen Methoden werden vom zentralen Player Skript aufgerufen und genutzt. Dieses Skript verwaltet und speichert die Daten. Startet der Benutzer ein Szenario mit Hilfe des Menüs, wird der Timer gestartet, um festzustellen wieviel Zeit für die Durchführung des Szenarios gebraucht wurde. Zeitgleich wird der Name des Szenarios vom ScenarioHandler übertragen. Ebenso wird bei der Aktivierung eines Malus aufgezeichnet, wie lange der Malus aktiviert war. Daraus ergibt sich ebenso der Datensatz, wie oft ein Malus bei dem Durchführen eines Szenarios aktiviert wurde und die akkumulierte Gesamtzeit, wo sich der Benutzer nicht auf dem Weg befand. Ebenso werden wie schon in Kapitel 4.3.1 erwähnt, abgespeichert, welche Hindernisse vom Benutzer durchlaufen wurden und welche nicht.

Die Datenaufzeichnung wird gestoppt, sobald ein Szenario vom Benutzer erfolgreich beendet wurde, und nach dem erfolgreichen Export zurückgesetzt.



Abb. : Data Recorder Parameter

Um die Daten einem Benutzer zuordnen zu können, wurde die öffentliche Variable userID erstellt, wie in Abb. 37 zu sehen ist. Hier lässt sich eine gewünschte einzigartige ID vor dem Start eines Szenarios eingeben.

#### Aufgabe der Szenarien

Aufgabe ist es, verschiedene Objekte in einer vorgegebenen Reihenfolge zu Erreichen. Dazu wurden an drei Seitenarmen des Weges Podeste platziert, auf denen sich die Objekte befinden. Bei den Objekten handelt es sich um eine Säge, ein Brecheisen und einem Hammer. Diese wurden schon zuvor in einem anderen Projekt implementiert. Unter jedes dieser Objekte wurde ein leeres GameObject erstellt, in der eine Halo Komponente hinzugefügt wurde. Diese soll einen grünen Kreis um das Objekt erzeugen, um den Benutzer ein optischer Hinweis zu geben, wenn das Objekt zum Finden ausgewählt ist (siehe Abb. 38: Objekt mit Halo Hinweis). Unter dem Halo Objekt wurde ein weiteres leeres GameObjekt erstellt, unter welchem ein Box Collider als Komponente hinzugefügt und für jedes der Objekte einem eindeutigen Tag zugewiesen wurde. Der Collider dient zur Erkennung, ob der Controller, welches der Benutzer in der Hand hält, sich am Objekt befindet.



Abb. 39: Objekt mit Halo Hinweis

Im Player Prefab wurde ein neues GameObject HandCollider unter dem Objekt RightHand erzeugt (siehe Abb. 39: HandCollider Objekt im Player Prefab). Ein Sphere Collider wurde diesem Objekt als Komponente hinzugefügt, die Größe auf 0.1 angepasst und die IsTrigger Einstellung aktiviert. Ebenso wurde ein Rigidbody dem HandCollider Objekt hinzugefügt, im die Trigger Funktionen nutzen zu können.



Abb. 40: HandCollider Objekt im Player Prefab

Um die Aktivierung der Objekte und die Erkennung der Kollision des HandColliders mit dem Collider am Objekt umzusetzen, wurde das Skript GameHandler unter dem HandCollider Objekt erstellt und hinzugefügt. Die Methoden werden vom zentralen Player Skript aufgerufen. Beim Starten eines Szenarios, wird das erste Objekt ausgewählt und aktiviert.(**Anzahl der Objekte ergänzen!**) Die Reihenfolge der Aktivierung ist dabei fest vorgelegt. Sobald ein Objekt mit dem Controller bzw. dem HandCollider, wird durch das Skript das nächste Objekt aktiviert. Wenn alle Objekte gefunden wurden, wird das Szenario beendet, indem der Benutzer den Startpunkt erreicht (siehe Abb. 40: Startpunkt im Szenario).



Abb. 41: Startpunkt im Szenario

Hierzu wurde ein GameObject StartPoint erstellt (siehe Abb. 41: Startpunkt Objekt), unter welches ein Game Object Circle für die Darstellung des grünen Kreises zuständig ist. Hierbei handelt es sich um ein Sphere Objekt bzw. eine Kugel, bei der die Y-Skalierung angepasst wurde, um einen flachen Kreis darzustellen. Dazu wurde ein Material Green erstellt, um den Kreis optisch grün zu gestalten. Ebenso wurde eine Halo Komponente hinzugefügt, um den zu erreichbaren Bereich zu verdeutlichen. Das darunter erzeugte Object ColliderStartPoint enthält ein Capsule Collider. Das Objekt ist für die Erkennung des Benutzers zuständig, wenn der Startpunkt betreten wird.



Abb. 42: Startpunkt Objekt

Wird am Ende eines Szenarios der Startpunkt betreten, wird dieser wieder deaktiviert und das Szenario beendet.

#### Hintergrundgeräusche

Da es sich bei der Umgebung um eine Art Industriehalle handelt, wurden dazu passende Hintergrundgeräusche für eine bessere Erzeugung der Präsenz hinzugefügt. Für die Implementierung wurden zwei verschiedene Audiodateien genutzt, die sich schon durch ein vorheriges Projekt im Asset Ordner Sounds befanden. Dazu wurde ein leeres Objekt AudioBackground erstellt, unter welchem die beiden Objekte AudioConstant und AudioInBetween erstellt wurden. Unter beiden Objekten wurde ein AudioSource als Komponente hinzugefügt und daran die Audiodateien gehängt. Das Audio, welches unter dem Objekt AudioConstant hinzugefügt wurde, wird dauerhaft nach dem Start der Anwendung abgepielt. Aus diesem Grund wurden die Haken bei PlayOnAwake und Loop gesetzt was dazu führt, dass das Audio direkt beim Start der Anwendung abgespielt wird und erneut von vorne beginnt.

Dagegen dürfen diese Haken bei dem Audio unter AudioInBetween nicht gesetzt sein. Zum Abspielen der Datei wurde ein Skript PlayAudioInBetween erstellt und als Komponente angehängt. Das Skript spielt das Audio mit Hilfe einer Coroutine in regelmäßigen Abständen ab. Die Zeit der Abstände wurde mit der der Variable PlayTime realisiert und den Wert auf 30 Sekunden gesetzt. Da die Laustärke der Geräusche nur leise im Hintergrund hörbar sein sollen, wurde die Volume Einstellung beider AudioScource Komponenten auf den Wert 0.3 gesetzt.

#### Menü

Um zu Beginn ein Szenario starten zu können, wurde ein Menü implementiert. Über das Menü kann über den Menüpunkt Start das Spiel mit dem im ScenarioHandler (siehe Kapitel 4.3.4) ausgewählten Szenario gestartet werden. Zur Interaktion mit dem Menü wird der rechte Controller genutzt.

##### Implementierung der UI Elemente

Für die Implementierung wurde das Canvas CanvasMenu erstellt (siehe Abb. 42: Menü zur Auswahl des Szenarios). In diesem Canvas wurde die Order in Layer Einstellung auf den Wert 1 gesetzt. Da es für den Malus schon ein Canvas, mit der Order in Layer mit dem Wert 0, existiert (siehe Kapitel 4.3.5.1), muss das Canvas des Menüs eine Schicht darüber liegen, um eine erfolgreiche Interaktion zu gewährleisten. Unter dem Canvas befindet sich ein Panel, unter dem jeder Menüpunkte als UI Element Button hinzugefügt wird. Das Panel dient als Zwischenschicht zwischen dem Canvas und den Buttons und wird hier genutzt, um ein optischer, dunkelgrauen Rahmen um das Menü zu erzeugen. Dafür wurde die Farbe der Image Komponente angepasst. Um die Interaktion mit den Buttons visuell sichtbar zu machen, wurde unter der Button Komponente die Highlighted Color auf dunkelgrau gesetzt. Dies bewirkt das optische dunklere verfärben des Buttons, wenn mit dem Laserpointer ein Button anvisiert wird, ohne dieses auszuwählen. Wenn ein Button mit der Trigger Taste ausgewählt wird, färbt sich dieser Button grün. Dazu wurde die Pressed Color konfiguriert (siehe Abb. 43: Button Objekt).



Abb. 43: Menü zur Auswahl des Szenarios

##### Funktionalität des Menüs

Die Funktionalität des Buttons wurde mit dem Skript MenuHandler implementiert und unter dem Button Objekt als Komponente hinzugefügt. Das Skript ruft dabei die Methode SetScenario() des Skriptes ScenarioHandler auf. Die Zuweisung der Methode wurde im OnClick() Bereich des Inspectors zugewiesen (siehe Abb. 43: Button Objekt). Dies zugewiesene Methoden wird ausgeführt, sobald der Button mit dem Trigger Taste des Controllers ausgewählt wird.



Abb. 44: Button Objekt

##### Laserpointer mit der Oculus Integration

Für die Interaktion der Controller mit dem Menü, musste zunächst im OVRCameraRig des OVRPlayerController das Skript OVRPhysicsRaycaster hinzugefügt werden.

Ebenso wurde das Prefab UIHelpers aus dem Oculus Integration Asset hinzugefügt. Dies beinhaltet zum einen das Objekt LaserPointer, welches dafür zuständig ist den Laser Pointer zum Anvisieren der Buttons zu visualisieren. An gleicher Stelle befindet sich zum anderen das Objekt EventSystem, was die Funktionalität des Laserpointers und des Controllers implementiert. Innerhalb der Skript Komponente OVRInputModule im Einstellungsbereich Ray Transform das Objekt RightHandAnchor des OVRPlayerControllers ausgewählt und unter Joy Pad Click Button den Primary Index Trigger. Das bewirkt, das der rechte Controller für den Laserpointer genutzt wird und die Trigger Taste des rechten Controllers zur Auswahl eines Buttons genutzt werden kann.

##### Laserpointer mit dem SteamVR Plugin

Für die Nutzung des Menüs für die HTC Vive bzw. für SteamVR, wurde ein schon vorhandene Menü dupliziert und angepasst. Grund dafür sind starke Unterschiede in der Implementierung, was das Anpassen des vorhandenen Menüs komplizierter macht.

Für die Interaktion mit dem Menü, um das Szenario auszuwählen, wurde ein Laserpointer implementiert, welches sich mit dem rechten Controller bedienen lässt. Dazu wurde ein PR\_Pointer Prefab erstellt und im Player Prefab unter dem RightHand Objekt hinzugefügt. welches die Komponenten Camera und Line Renderer besitzt. Die Komponente Camera ist deaktiviert und wird dazu verwendet, ein grafischen Raycast zu implementieren. Dadurch werden UI Elemente in einem Canvas durch Zeigen mit dem Controller bei Kollision registriert. Die Komponente Line Renderer dient als optische Hilfe für den Benutzer, um die Richtig des Raycast sehen zu können (siehe Abb. 44: LaserPointer mit Raycast).



Abb. 45: LaserPointer mit Raycast

Das Skript Pointer wurde dem PR\_Pointer Objekt als Komponente hinzugefügt, welches die Funktionalitäten des Raycasts implementiert. Durch das Skript werden die UI Elemente im Canvas des Menüs erkannt und die Länge bei Kollision mit einem Hindernis angepasst. Zusätzlich wurde das Skript VRInputModule erstellt und eingebunden, welches die Interaktion durch das Drücken der Trigger Tasten des rechten Controllers implementiert. Dadurch können der Button ausgewählt werden bzw. die Methoden des Skriptes MenuHandler, welches bei der Erstellung des Menüs in Kapitel 4.3.10.2 erstellt wurde, ausgeführt werden. Das Skript wurde der Komponente InputModule hinzugefügt, welches das Eventsystem des Player Prefabs ist.

#### Ablaufplan

* Schutzmaßnahmen durch die Covid Situation (QR-Code Scan. Hardware desinfizierend, Corona Schnelltest, Proband setzt Maske ab)
* Kurzer Überblick über die Studie
* Informationen über die VR-Brille und Controller
* Beschreibung der Aufgabe
* Einverständniserklärung
* Kurze Gewöhnungphase an die VR Umgebung
* Durchlauf der Studie
* Fragebogen

Bei der Ankunft in der Studie (siehe Abb. 1 Die Teilnehmer erhielten einen kurzen Überblick über die Studie, der Informationen zum Oculus Rift-Headset und eine allgemeine Beschreibung der Konditionierungsaufgabe enthielt. Nach Einverständniserklärung wurde den Teilnehmern die Verwendung

des Videospiel-Controllers erklärt. (Aufgabenbeschreibung)

### Datenerfassung

Für die Evaluation werde zwei verschiedene Arten von Daten vorgesehen. Quantitative Daten, die bei der Durchführung Szenarien entstehen, und qualitative Daten durch die spätere Befragung der Probanden mit Hilfe eines Online Fragebogens.

#### Quantitative Daten

Wie in Kapitel 4.3.7 erläutert sind die Daten, die während der Laufzeit der Szenarien gespeichert werden:

* ID des Probanden
* welches Szenario aktiv war
* Zeit, die der Proband braucht, um den kompletten Weg bis zum Ziel zu laufen
* die Anzahl der Malus, die aktiviert wurde
* die Dauer pro Malus bzw. wie lange der Proband gebraucht hat, um wieder auf den richtigen Weg zurückzukommen

Dazu wurde das Skript DataRecorder erstellt, um die erforderlichen Daten, wie zuvor erläutert, aufzuzeichnen und zu speichern. Die Methoden werden durch das Player Skript aufgerufen.

Das Skript CSVManager ist für den Export der aufgezeichneten Daten in eine CSV Datei zuständig. Dies ist für das Auslesen und weitere Auswertung der Daten entscheidend. Die Datei Report.csv wird lokal im Assets Ordner Report erstellt bzw. abgelegt. Für den Fall, das Daten erhoben und abgespeichert werden sollen, die CSV Datei aber noch nicht existiert, wird diese und der dazugehörige Ordner automatisch erstellt. Dazu werden standardmäßig folgende Header in gleicher Reihenfolge erstellt:

* ID
* Szenario
* Gebrauchte Zeit
* Malus aktiviert

Sobald ein Malus öfter als einmal aktiviert wurde, werden die dazugehörigen Header für die Dauer der einzelnen Malus dynamisch hinzugefügt. Ein Beispiel der CSV Datei mit gefüllten Daten zeigt sich in Abb. 45: CSV Datei mit Beispieldaten.



Abb. 46: CSV Datei mit Beispieldaten

#### Qualitative Daten

Für die Erhebung der qualitativen Daten wird ein Onlinefragebogen erstellt. Für die Umsetzung wird EvaSys verwendet. Hierbei handelt es sich um eine webbasierte Feedback Management Lösung. Onlineumfragen, klassische Umfragen in Papierform oder selbst Hybridumfragen sind möglich [62]. In diesem Projekt wird auf eine reine Online-Umfrage verwendet, da eine Papier- oder Hybridumfrage keine Vorteile bietet und das Auswerten mit mehr Aufwand zusammenhängt.

Die Fragen der Onlineumfrage wird in verschiedene Fragegruppen zusammengefasst, auf die im folgenden Abschnitt im Einzelnen genauer eingegangen wird. Als erstes geht es um die persönlichen Daten des Probanden (siehe Abb. 46: Fragebogen - Persönliche Daten). Hier wird das Geschlecht und das Alter als Angaben gemacht. Diese werden als Single Choice Fragen umgesetzt und das Alter in Gruppen unterteilt, was das empirische Auswerten unkomplizierter und übersichtlicher gestaltet.



Abb. 47: Fragebogen - Persönliche Daten

In der zweiten Fragengruppe geht es um allgemeine Angaben der Probanden (siehe Abb. 47: Fragebogen - Allgemeine Angaben). Die Erfahrungen in Bereich Gaming und Virtual Reality werden jeweils in einer Skala von 1 bis 5 abgefragt. Wobei 1 keine Erfahrung und 5 viel Erfahrung bedeutet. Die Anzahl der Ankreuzfelder werden im kompletten Fragebogen konsistent gehalten. In der dritten Frage geht es um die Art der VR-Technologien, mit welche der Proband schon in Erfahrung getreten ist. Hierbei handelt es sich um eine Multiple Choice Frage, da eine oder mehrere Antworten möglich sind.



Abb. 48: Fragebogen - Allgemeine Angaben

Die nächste Fragengruppe bezieht sich explizit auf die Durchführung der Szenarien (siehe Abb. 48: Fragebogen - Beurteilung Szenario). Für jedes der drei Szenarien werden dieselbe Art Fragen benutzt, um die Auswirkungen auf den Probanden zu ermitteln. Die Skala im Wert von 1 bis 5 ist hier ebenso, wie zuvor erläutert, die Art der Antwortmöglichkeiten. Der Wert 1 bedeutet, das der Inhalt der Frage gar nicht zutrifft und der Wert 5 dabei völlig zutrifft. Dazu gehört die Natürlichkeit des Laufens, die Projektion des Denkens und Handelns in der virtuellen Umgebung. Das Bewusstsein der physischen Umgebung ist ebenso ein Teil der Fragestellung. Die letzte Frage geht explizit auf den Einfluss des negativen Malus auf den Probanden ein.



Abb. 49: Fragebogen - Beurteilung Szenario

Die letzte Fragegruppe ergibt ein offenes und persönliches Feedback (siehe Abb. 49: Fragebogen - Persönliches Feedback). Der Frage Typ hierbei sind offene Fragen mit unbegrenzter Anzahl an Zeichen.

Hier geht es unter anderem darum, was die Probanden besonders positiv und negativ an der Durchführung der Szenarien fanden. Ein weiterer Punkt ist, was genau die Probanden am negativen Malus beeinflusst hat. Zum Abschluss können konkrete Verbesserungsvorschläge gebracht werden, was besonders relevant für die Weiterentwicklung des Projektes ist.



Abb. 50: Fragebogen - Persönliches Feedback

Nach der Datenerfassung mit Hilfe des Fragenbogens können die Daten direkt in EvaSys dargestellt und ausgewertet werden.

### Evaluation und Auswertung der Ergbenisse

#### Aufbau der Laborumgebung

##### Aufbau des Weges

Die Durchführung der Studie findet im Laborraum des UnityLab statt. Um den virtuellen Weg der Szenarien physisch nachzubilden, werden schwarze Bodenmatten in der Mitte des Raumes ausgelegt. Um die Größe der virtuellen Matten anzupassen, wird die physische Größe genau vermessen. Da die virtuellen Matten jedoch nicht maßstabgetreu sind, werden die Maße skaliert und angepasst. Ebenso werden die Abstände zu den Wänden ermittelt, um die virtuelle und physische Ausrichtung im Raum abzustimmen. Für die genaue Ausrichtung wird ein Controller an verschiedene Stellen des Weges gelegt und diese mit der virtuellen Position abgeglichen. Zum Schluss werden die Matten am Boden fixiert, um eine Veränderung der Position oder Ausrichtung zu verhindern.

##### Kalibrierung der HTC Vive

Damit die Position im physischen und virtuellen Raum übereinstimmt, wird die HTC Vive kalibriert. Dazu wird das HMD in die Mitte des Raumes gelegt und mit Hilfe der quick Calibration von SteamVR kalibriert. Um die Position abzusichern, wird die Datei chaperone\_info separat abgespeichert, um diese wieder herstellen zu können. Diese befindet sich im Config Ordner im SteamVR Installationsverzeichnis und enthält alle Daten über die kalibrierte Position des HMD und den Rand des Spielbereichs. Ebenso wird der Player Prefab in der virtuellen Umgebung in der Mitte platziert, damit die Position mit der kalibrierten Position des HMD übereinstimmt.

#### Beschreibung des Forschungsgegenstandes

Bei der Art der Forschungsmethode handelt es sich hier um eine quantitative Forschung. Diese wird mit einer Stichprobe mit Hilfe eines Experiments, einer standardisierten Befragung und einer statistischen Analyse realisiert .

* Alle Starten mit dem Baseline (ohne Matten), (Visuelle Szenario nochmals zum Schluss?)
* Danach 10 auf A, und 10 auf B und dann umgekehrt
* Pro Proband ca. eine Stunde Zeit einplanen, 25 Minuten für den eigentlichen Durchlauf
* -Ziel 20 Probanden





##### Unabhängige Variablen

Die erste unabhängige Variable ist der Malus mit den Ausprägungen Malus an und Malus aus als Within-Subjects Factor. Im physischen und visuellen Szenario ist der Malus aktiviert, im Standardszenario jedoch nicht.

Die zweite unabhängige Within-Subjects Factor Variable ist der taktile Untergrund vorhanden oder nicht vorhanden.

##### Abhängige Variablen

Die abhängigen Variablen des Experiments ist zum einen die Anzahl der ausgelösten Maluse. Die zweite Abhängige Variable ist die Dauer des jeweiligen ausgelösten Malus.

##### Stichprobe

„18 Teilnehmer (10 Männer, 8 Frauen) im Alter von 20-45 Jahren ( M = 30.06, SD = 7.55) nahm an der Studie teil. Wir haben sie gebeten, ihre Erfahrungen mit VR-Technologien und Computerspielen im Allgemeinen auf einer Skala von 1 (verwenden Sie selten VR-Technologien / spielen Sie Spiele) bis 7 (verwenden Sie sehr häufig VR-Technologien / spielen Sie Spiele) zu bewerten. Unsere Teilnehmer hatten wenig Erfahrung mit VR-Technologien ( M = 2,56, SD = 1,62) und durchschnittliche Erfahrung mit Spielen ( M = 4,72, SD = 1,90). Jede Lernsitzung dauerte ungefähr 75 Minuten. „

##### Ablauf

Zu Beginn werden durch die Covid-19 Situation verschiedene Schutzmaßnahmen durchgeführt. Zum einen wird die in der Studie genutzte Hardware desinfiziert. Ebenso wird beim Probanden ein Corona Schnelltest durchgeführt, so kann die Maske bei der Durchführung abgesetzt werden.

Darauf erfolgt ein kurzer Überblick über die Studie, einige Informationen über das HTC Vive Headset und dessen Controller. Nach einer detaillierten Beschreibung der Aufgabe und ggf. Beantworten offener Fragen erfolgt die Einverständniserklärung.

Nun bekommt der Proband die Möglichkeit, sich an das HTC Vive Headset und die virtuelle Umgebung zu gewöhnen. Fühlt sich der Proband bereit, wird der Durchlauf der Szenarien durchgeführt und nach erfolgreicher Durchführung der Fragebogen ausgefüllt.

#### Ergebnisse

##### Fragebogen

## Zusammenfassung, Bewertung und Ausblick

## Anhang

### Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | R. Dörner, W. Broll, B. Jung und P. Grimm, in *Virtual und Augmented Reality (VR/AR)*, Springer, 2019, pp. 12-20. |
| [2] | H. Cherni, N. Métayer und N. Souliman, „Literature review of locomotion techniques in virtual reality,“ 2020. |
| [3] | A. L. Simeone, I. Mavridou und W. Powell, „Altering User Movement Behaviour in Virtual Environments,“ April 2017. |
| [4] | M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater und F. P. Brooks, „Walking > Walking-in-Place > Flying, in Virtual Environments,“ 1999. |
| [5] | R. A. Ruddle und S. Lessels, „The Benefits of Using a Walking Interface,“ April 2009. |
| [6] | R. A. Ruddle, E. Volkova und H. H. Bülthoff, „Learning to Walk in Virtual Reality,“ Mai 2013. |
| [7] | M. C. Whitton, J. V. Cohn, J. Feasel, P. Zimmons, S. Razzaque, S. .. Poulton, B. McLeod und F. P. Brooks, Jr., „Comparing VE Locomotion Interfaces,“ März 2005. |
| [8] | S. Razzaque, Z. Kohn und M. C. Whitton, „Redirected Walking,“ 2001. |
| [9] | E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum und M. Bolas, „A Taxonomy for Deploying Redirection Techniques in Immersive Virtual Environments,“ 2012. |
| [10] | F. Steinicke, G. Bruder, L. Kohli, J. Jerald und K. Hinrichs, „Taxonomy and Implementation of Redirection Techniques for Ubiquitous Passive Haptic Feedback,“ 2008. |
| [11] | B. Williams, G. Narasimham, T. P. McNamara, T. H. Carr, J. J. Rieser und B. Bodenheimer, „Updating orientation in large virtual environments using scaled translational gain,“ 2006. |
| [12] | F. Steinicke, G. Bruder, J. Jerald, H. Frenz und M. Lappe, „Analyses of Human Sensitivity to Redirected Walking,“ 2008. |
| [13] | F. Steinicke, G. Bruder, T. Ropinski und K. Hinrichs, „Moving Towards Generally Applicable Redirected Walking,“ 2008. |
| [14] | V. Interrante, L. Anderson und B. Ries, „Distance Perception in Immersive Virtual Environments, Revisited,“ 2006. |
| [15] | G. Bruder, F. Steinicke und P. Wieland, „Self-Motion Illusions in Immersive Virtual Reality Environments,“ 2011. |
| [16] | F. Steinicke, G. Bruder, J. Jerald, H. Frenz und M. Lappe, „Estimation of Detection Thresholds for Redirected Walking Techniques,“ 2009. |
| [17] | F. Steinicke, G. Bruder, K. Hinrichs und A. Steed, „Gradual transitions and their effects on presence and distance estimation,“ 2009. |
| [18] | B. Williams, G. Narasimham, B. Rump und T. McNamara, „Exploring Large Virtual Environments with an HMD when Physical Space is Limited,“ 2007. |
| [19] | E. Suma, S. Clark, S. Finkelstein und Z. Wartell, „Leveraging Change Blindness for Redirection in Virtual Environments,“ 2011. |
| [20] | D. Simons, „Current Approaches to Change Blindness,“ 2000. |
| [21] | N. C. Nilsson, S. Serafin, M. H. Laursen, R. Nordahl, K. S. Pedersen und E. Sikström, „Tapping-In-Place: Increasing the Naturalness of Immersive Walking-In-Place Locomotion Through Novel Gestural Input,“ 2013. |
| [22] | B. Williams, S. Bailey, G. Narasimham, M. Li und B. Bodenheimer, „Evaluation of Walking in Place on a Wii Balance Board to Explore a Virtual Environment,“ 2011. |
| [23] | M. Slater, M. Usoh und A. Steed, „Taking Steps: The Influence of a Walking´Technique on Presence in Virtual Reality,“ 1995. |
| [24] | D. Zielinski, R. McMahan und R. Brady, „Shadow Walking: an Unencumbered Locomotion Technique for Systems with Under-floor Projection,“ 2011. |
| [25] | J. Lee, S. C. Ahn und J. Hwang, „A Walking-in-Place Method for Virtual Reality Using Position and Orientation Tracking,“ 2018. |
| [26] | F. Buttussi und L. Chittaro, „Locomotion in Place in Virtual Reality: A Comparative Evaluation of Joystick, Teleport, and Leaning,“ 2020. |
| [27] | M. P. Jacob Habgood, D. Moore, D. Wilson und S. Alapont, „Rapid, Continuous Movement Between Nodes as an Accessible Virtual Reality Locomotion Technique,“ 2018. |
| [28] | L. A. Cherep, A. Lim, J. W. Kelly und A. Ostrander, „Spatial cognitive implications of teleporting through virtual environments,“ 2020. |
| [29] | M. P. J. Habgood, D. Moore, D. WIlson und S. Alapont, „Rapid, Continuous Movement Between Nodes as an Accessible Virtual Reality Locomotion Technique,“ 2018. |
| [30] | F. Buttussi und L. Chittaro, „Locomotion in Place in Virtual Reality: A Comparative Evaluation of Joystick, Teleport, and Leaning,“ 2019. |
| [31] | P. T. Wilson, W. Kalescky, A. MacLaughlin und B. Williams, „VR Locomotion: Walking>Walking in Place>Arm Swinging,“ 2016. |
| [32] | D. Zielasko, S. Horn, S. Freitag, B. Weyers und T. Kuhlen, „Evaluation of Hands-Free HMD-Based Navigation Techniques for Immersive Data Analysis,“ 2016. |
| [33] | A. Harris, K. Nguyen, P. T. Wilson, M. Jackoski und B. Williams, „Human Joystick: Wii-Leaning to Translate in Large Virtual Environments,“ 2014. |
| [34] | A. Kitson, A. M. Hashemian, E. R. Stepanova, E. Kruijf und B. E. Riecke, „Comparing Leaning-Based Motion Cueing Interfaces for Virtual Reality Locomotion,“ 2017. |
| [35] | Z. Wang, H. Wei, K. Zhang und L. Xie, „Real Walking in Place: HEX-CORE-PROTOTYPE Omnidirectional Treadmill,“ 2020. |
| [36] | L. E. Warren und D. A. Bowman, „User Experience with Semi-Natural Locomotion Techniques in Virtual Reality: The Case of the Virtuix Omni,“ 2017. |
| [37] | J. Wolter und E. Walther, 26 10 2016. [Online]. Available: https://dorsch.hogrefe.com/stichwort/konditionierung-klassische. [Zugriff am 06 05 2021]. |
| [38] | E. Glotzbach, H. Ewald, M. Andreatta, P. Pauli und A. Mühlberger, „Contextual fear conditioning predicts subsequent avoidance behaviour in a virtual reality environment,“ 2012. |
| [39] | M. C. W. Kroes, J. E. Dunsmoor, W. E. Mackey, M. McClay und E. A, „Context conditioning in humans using commercially available immersive Virtual Reality,“ 2017. |
| [40] | G. Halbeisen und T. Glaser, 07 05 2019. [Online]. Available: https://dorsch.hogrefe.com/stichwort/konditionierung-operante#search=4b7c74baa316eec7a5887db2deea308f&offset=0. [Zugriff am 06 05 2021]. |
| [41] | D. Kumar, N. Sinha und U. Lahiri, „Virtual reality‑based balance training system augmented with operant conditioning paradigm,“ 2019. |
| [42] | C. Tröger, H. Ewald, E. Glotzbach, P. Pauli und A. Mühlberger, „Does pre-exposure inhibit fear context conditioning? A Virtual Reality Study,“ 2012. |
| [43] | [Online]. Available: https://unity.com/de. [Zugriff am 23 02 2021]. |
| [44] | „Unity’s interface,“ [Online]. Available: https://docs.unity3d.com/Manual/UsingTheEditor.html. [Zugriff am 07 06 2021]. |
| [45] | D. Ziesecke, „Alle Infos zur Oculus Quest – technische Daten, Preis und Laufzeit,“ 03 03 2019. [Online]. Available: https://vr-legion.de/news/alle-infos-zur-oculus-quest-technische-daten-preis-und-laufzeit/. [Zugriff am 23 02 2021]. |
| [46] | C. Spinger, „Oculus Quest im Test: Virtual Reality für die Masse,“ 30 04 2019. [Online]. Available: https://www.vrnerds.de/oculus-quest-im-test-virtual-reality-fuer-die-masse/. [Zugriff am 27 02 2021]. |
| [47] | „Unveiling the Vive Consumer Edition and Pre-order Information,“ [Online]. Available: https://blog.vive.com/us/2016/02/21/unveiling-the-vive-consumer-edition-and-pre-order-information/. [Zugriff am 10 06 2021]. |
| [48] | „HTC Vive Pro gegen HTC Vive Pro Eye gegen HTC Vive Pro 2,“ [Online]. Available: https://www.pocket-lint.com/de-de/ar-vr/kaufberatung/htc/143271-htc-vive-pro-vs-htc-vive-was-ist-der-unterschied. [Zugriff am 13 06 2021]. |
| [49] | [Online]. Available: https://business.vive.com/de/product/vive-pro-eye-office/. [Zugriff am 10 06 2021]. |
| [50] | [Online]. Available: https://www.vive.com/de/product/vive-pro-eye/overview/. [Zugriff am 13 06 2021]. |
| [51] | „Über die VIVE Controller (2018),“ [Online]. Available: https://www.vive.com/de/support/vive-pro/category\_howto/about-the-controllers---2018.html. [Zugriff am 13 06 2021]. |
| [52] | „Über SteamVR Basisstation 2.0,“ [Online]. Available: https://www.vive.com/de/support/vive-pro/category\_howto/about-the-base-stations.html. [Zugriff am 13 06 2021]. |
| [53] | [Online]. Available: https://partner.steamgames.com/vrlicensing. [Zugriff am 13 06 2021]. |
| [54] | [Online]. Available: https://www.vive.com/de/accessory/wireless-adapter/. [Zugriff am 15 07 2021]. |
| [55] | „VIVE WLAN Adapter (VIVE/VIVE Pro),“ [Online]. Available: https://www.vive.com/de/support/wireless-adapter/category\_howto/vive-wireless-adapter.html. [Zugriff am 15 07 2021]. |
| [56] | „Understand Oculus Integration Package Components,“ [Online]. Available: https://developer.oculus.com/documentation/unity/unity-utilities-overview/. [Zugriff am 17 05 2021]. |
| [57] | „SteamVR Unity Plugin,“ [Online]. Available: https://valvesoftware.github.io/steamvr\_unity\_plugin/. [Zugriff am 17 05 2021]. |
| [58] | „Colliders,“ [Online]. Available: https://docs.unity3d.com/Manual/CollidersOverview.html. [Zugriff am 21 05 2021]. |
| [59] | „Tags,“ [Online]. Available: https://docs.unity3d.com/Manual/Tags.html. [Zugriff am 15 07 2021]. |
| [60] | „Animation State Machines,“ [Online]. Available: https://docs.unity3d.com/Manual/AnimationStateMachines.html. [Zugriff am 19 05 2021]. |
| [61] | A. States. [Online]. Available: https://docs.unity3d.com/Manual/class-State.html. [Zugriff am 31 05 2021]. |
| [62] | [Online]. Available: https://evasys.de/evasys/. [Zugriff am 28 02 2021]. |
| [63] | S. Augsten, „Definition „Git SCM“ - Was ist Git?,“ 27 08 2019. [Online]. Available: https://www.dev-insider.de/was-ist-git-a-850847/. [Zugriff am 24 03 2021]. |
| [64] | TerryGLee, „Neues in Visual Studio 2019,“ 10 11 2020. [Online]. Available: https://docs.microsoft.com/de-de/visualstudio/ide/whats-new-visual-studio-2019?view=vs-2019. [Zugriff am 22 02 2021]. |
| [65] | B. Wagner, „Überblick über C#,“ 28 01 2021. [Online]. Available: https://docs.microsoft.com/de-de/dotnet/csharp/tour-of-csharp/. [Zugriff am 23 02 2021]. |
| [66] | O. B. Dr. Veikko Krypczyk, „Einführung in die Programmierung: Objekte, nichts als Objekte,“ 19 04 2016. [Online]. Available: https://entwickler.de/online/development/einfuehrung-programmierung-objektorientierte-programmentwicklung-197372.html. [Zugriff am 24 02 2021]. |
| [67] | J. N. Templeman, D. P. S. und L. E. Sibert, „Virtual Locomotion: Walking in Place through Virtual Env ironments,“ Dezember 1999. |
| [68] | A. L. Simeone, E. Velloso und H. Gellersen, „Substitutional Reality: Using the physical environment to design virtual reality experiences,“ Januar 2015. |
| [69] | P. Fink, P. Foo und W. H. Warren, „Obstacle avoidance during walking in real and virtual environments,“ Januar 2007. |
| [70] | S. Razzaque, D. Swapp, M. Slater, M. C. Whitton und A. Steed, „Redirected Walking in Place,“ 2002. |
| [71] | J. N. Templeman, P. S. Denbrook und L. E. Sibert, „Virtual locomotion: Walking in place through virtual environments,“ Dezember 1999. |
| [72] | D. A. Bowman, D. Koller und L. F. Hodges, „Travel in Immersive Virtual Environments: An Evaluation of Viewpoint,“ 1997. |
| [73] | E. A. Suma, S. L. Finkelstein, M. Reid, S. V. Babu, A. C. Ulinski und L. F. Hodges, „Evaluation of the Cognitive Effects of Travel Technique in Complex Real and Virtual Environments,“ 2010. |
| [74] | V. Interrante, B. Ries und L. Anderson, „Seven League Boots: A New Metaphor for Augmented Locomotion through Moderately Large Scale Immersive Virtual Environments,“ 2007. |
| [75] | L. F. Hodges, R. Kooper, T. C. Meyer, B. O. Rothbaum, D. Opdyke, J. J. Graaff, J. S. Williford und M. M. North, „Virtual Environments for Treating the Fear of Heights,“ 1995. |
| [76] | G. Bruder, F. Steinicke und K. H. Hinrichs, „Arch-Explore: A natural user interface for immersive architectural,“ 2009. |
| [77] | T. Peck, H. Fuchs und M. Whitton, „Evaluation of Reorientation Techniques and Distractors for Walking in Large Virtual Environments,“ 2008. |
| [78] | N. C. Huff, J. A. Hernandez, M. E. Fecteau, D. J. Zielinski, R. Brady und K. S. LaBar, „Revealing context-specific conditioned fear memories with full immersion virtual reality,“ 2011. |

### Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Heilbronn, 19. September 2016 **Fehler! Verweisquelle konnte nicht gefunden werden.**