Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jonathan Becker

Experimente zur Raumwahrnehmung in Virtuellen Welten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jonathan Becker

Experimente zur Raumwahrnehmung in Virtuellen Welten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultechnik und Informatik
der Hochschule fr Angewandte Wissenschaften Hamburg

Betreuender Prfer: Prof. Dr. Kai von Luck
Zweitgutachter: Dr. Susanne Draheim

Eingereicht am: 02.01.2018

Jonathan Becker

Thema der Arbeit
Experimente zur Raumwahrnehmung in Virtuellen Welten

Stichworte
Virtual Reality, Raumwahrnehmung, Landmarken, Unity, Plattform für Untersuchungen, Rotation, freie Bewegung, Cybersickness

Kurzzusammenfassung
In dieser Bachelorarbeit wird eine Plattform entwickelt, welche Untersuchungen zur Rotation und zur freien Bewegung in VR ermöglicht. Freie Bewegung bedeutet, dass der User selbst auf einer kleinen realen Fläche große virtuelle Räume erkunden kann. Die Rotation steht für die Drehung des Users um seine horizontale Achse, was ihm ermöglicht an Wänden und Decken zu laufen. Zunächst werden Faktoren zur Raumwahrnehmung aus der Literatur zusammengestellt und eine Plattform entworfen. Diese wird dann umgesetzt, mit Alpha-Testern auf ihre Nutzbarkeit getestet und hinsichtlich der definierten Anforderungen evaluiert.

Jonathan Becker

Title of the paper
Spacial Perception Experiments in Virtual Worlds

Keywords
virtual reality, spatial cognition, landmarks, Unity, platform for investigations, Rotation, free movement, cybersickness

Abstract
In this bachelor thesis, a platform is developed which enables investigations into rotation and free movement in VR. Free movement means that the user can explore large virtual spaces even if the real space is small. The rotation represents the rotation of the user around his horizontal axis, which allows him to walk on walls and ceilings. First, factors for spatial perception are compiled from literature and a platform is designed. The platform will be implemented, tested for its usability using alpha-testers, and evaluated against the defined requirements.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Das Konzept der Virtual Reality (VR) hat sich inzwischen weit verbreitet. Mit jedem Jahr werden die Welten der VR realistischer und die Anwendungen nützlicher. Dadurch hat die VR in vielen Gebieten bereits ihre Wurzeln geschlagen, z. B. in der Therapie, um Höhenangst zu überwinden. Training für Zugführer gibt es inzwischen auch virtuell, da sich so komplexe Szenarien besser darstellen lassen. In anderen Gebieten wird VR inzwischen auch zum Training eingesetzt, besonders dort wo es teuer wäre Trainingsanlagen zu bauen oder gefährliche Situationen ohne Gefahr trainiert werden sollen. Bildung bietet heutzutage der VR auch einen Platz an, in dem sie wachsen kann, so gibt es bereits Museen, die VR nutzen, um Exponate zu animieren und auszustellen. Ausstellungsstücke, die normalerweise zu groß wären, um sie auszustellen, wie vollständige Walskelette, nehmen in der VR keinen Platz weg und sind des Weiteren sogar noch transportabel. VR kann außerdem genutzt werden, um ”Fremde Orte” zu besuchen, sei es nun die Vorschau des Hotels oder Landstriche, die normaler Weise nicht besucht werden können.

Es gibt mittlerweile viele Einsatzgebiete für VR, die Forschung gehört zu den Feldern, die schon länger mit VR arbeiten, zumindest einzelne Zweige. Spezielle Setups, die in der Realität nicht möglich wären oder viel Aufbau brauchen würden, lassen sind mit der VR Technik vergleichsweise einfach erstellen. Mit der Zeit sind es immer mehr Möglichkeiten geworden, die in der VR umgesetzt werden, besonders da inzwischen die Head-Mounted Displays (HMD) von den Preisen her keine so hohe Schwelle mehr darstellen.

1.1. Motivation

Die Virtuelle Welt ist grenzenlos. Sie ist nicht an die Gesetze der Realität gebunden und erlaubt es, die Regeln der Virtuellen Welt nach belieben zu ändern, worauf diese dann direkt erlebt werden können. Am Anfang meiner Forschungsbemühungen stand der Gedanke, die Virtuelle Welt um Konzepte zu erweitern, die in der Realität und der zweidimensionalen (Bildschirm-) Welt nicht existieren. Anschließend sollten die umgesetzten Konzepte auf ihre Auswirkungen für den Menschen untersucht werden. Das Ergebnis wäre eine Sammlung von Elementen, die

die Virtual Reality erweitern und Antworten auf die Frage ob diese Elemente die virtuelle Welt bereichern oder Grund für Kopfschmerzen und Übelkeit sind.

Die Idee hatte den Namen "Vom Unbehagen der Unwirklichkeit" und umfasste eine Vielzahl von Ansätzen, wie die Erkundung der VR in einem fremden Körper, der eine andere Schrittlänge hat oder andere verzerrte Körperproportionen besitzt. Eine weitere Idee war die Menge an Informationen in der VR zu erhöhen, zum Beispiel indem ein Teil des Farbspektrums für andere Informationen genutzt wird. Eine anderer Gedanke ist die Erstellung von unmöglichen Räumen, die die Euklidische Geometrie verletzen und so z. B. Rundgänge aus drei Wegen zulassen, die jeweils im rechten Winkel zueinander stehen. Das letzte an dieser Stelle zu erwähnende Beispiel beschäftigt sich damit, dem User die Kontrolle über die Schwerkraft zu überlassen. Dadurch wäre es möglich an Decken oder Hauswänden zu gehen, wo ganz neue Schluchten und Abgründe auftauchen, wenn einen der Spaziergang entlang einer Hausfassade führt.

Von diesen Ideen ist am Ende die Manipulation der Schwerkraft geblieben, die als äquivalente Rotation des Users im virtuellem Raum abgebildet wird, wenn dieser bspw. beginnt, an einer Wand heraufzulaufen. Zu diesem Themenbereich gibt es zum aktuellem Zeitpunkt noch keine Untersuchungen. Um Untersuchungen zu diesem Thema zu ermöglichen, ist als erstes eine Umgebung notwendig, in der die Schwerkraft und Rotation kontrolliert manipulierbar sind.

Daraus ergibt sich das Ziel dieser Arbeit.

1.2. Ziel der Arbeit

Das Ziel dieser Arbeit ist es, eine Plattform zu schaffen, mit der Untersuchungen zur Raumwahrnehmung in der Virtual Reality gemacht werden können. Die Experimente, die die Plattform ermöglichen soll, beschäftigen sich mit den Auswirkungen von Rotation auf die Raumwahrnehmung. Dazu gehört auch die Ausrichtung im virtuellem Raum und die Veränderungen in der Raumwahrnehmung durch die Möglichkeit, sich frei an Wänden und Decken zu bewegen.

Für diese Umgebung sind zwei Komponenten nötig. Einerseits der grafische Teil, der sich mit dem Aussehen der Umgebung beschäftigt und für unterschiedliche Szenarien ausgetauscht wird. Andererseits ein Plattform-Teil, der sich mit der Implementierung beschäftigt, die freie Bewegung auf beliebigen Oberflächen wie Böden, Wänden und Decken erlaubt und sich in einer Vielzahl von Szenarien einsetzen lässt. Der größere Teil dieser Arbeit wird sich mit dem Design und der Implementierung der Plattform beschäftigen. Diese soll in der Lage sein in unterschiedlichen Szenarien eingesetzt zu werden. Um dieses zu ermöglichen soll die

Struktur der Plattform robust gegen Weiterentwicklung oder Änderungen an den verwendeten Technologien sein, z. B. soll der Wechsel des HMD keine Hürde darstellen.

Am Ende soll die Plattform den Prozess ”Szenarien erstellen” erleichtern, indem sie eine Abstraktion von der Technik erlaubt. Auf diesem Weg soll die Plattform als Hilfe dienen, die es erlaubt, einen größeren Fokus auf die Gestaltung der Szenarien und Tests zu legen. Die Auswirkungen der Rotation im VR sollen sich so leichter untersuchen lassen.

1.3. Aufbau der Arbeit

Die Arbeit ist in vier Kapitel geteilt, beginnend mit dieser Einleitung.

Im zweitem Kapitel Aspekte der Raumwahrnehmung geht es um Raumwahrnehmung und wie Virtual Reality sie verändert. Dabei werden einzelne Komponenten der Raumwahrnehmung vorgestellt, wie Landmarken und Mentale Karten. In diesem Kapitel wird auch der Übergang von Realität in die Virtualität vorgenommen, dazu gibt es einen Exkurs über Cybersickness. Am Ende werden einige mögliche Szenarien für Tests vorgestellt, sowie deren Bedingungen an die Plattform. An dieser Stelle wird auch die Zielsetzung formuliert.

Im drittem Kapitel Design der Plattform geht es um die Anforderungen und die Implementierung der Plattform. Im Kapitel der Anforderungen werden jene Punkte beschrieben, die notwendig sind, um eine funktionsfähige Plattform zu schaffen, die Tests in VR unterstützt. Danach geht es um das Design und die Implementierung. In diesem Abschnitt wird auch das Konzept der Plattform vorgestellt, auf die grundlegende Software eingegangen und die einzelnen Komponenten werden vorgestellt. Am Ende des Abschnittes des Designs und der Implementierung wird mithilfe der Plattform ein Testszenario gebaut, um die Anwendung der Plattform zu demonstrieren. Abgeschlossen wird das dritte Kapitel mit der Beurteilung der Plattform im Zusammenhang mit den Anforderungen.

Das vierte Kapitel Evaluation befasst sich mit der Evaluation der Plattform im Betrieb und zeigt, dass diese nutzbar ist. Dazu kommen die ersten Eindrücke, die mit Alpha-Testern im Test-Szenario gewonnen werden konnten. Abschließend geht es um die möglichen Weiterentwicklungen der Plattform und an welchen Stellen Besserungen möglich sind.

Das letzte Kapitel bildet mit einer Zusammenfassung der Arbeit und einem Ausblick in die Zukunft den Schluss.

2. Aspekte der Raumwahrnehmung

”Spatial Cognition is concerned with the acquisition, organization, utilization, and revision of knowledge about spatial environments.” [3]

Raumwahrnehmung beschäftigt sich mit der Aneignung, Organisation, Nutzung und Überarbeitung von Wissen über räumliche Gebiete.

2.1. Was ist Raumwahrnehmung?

Das Forschungsgebiet der Raumwahrnehmung ist über viele Disziplinen verteilt. Sie ist etwas, das ständig von vielen Systemen genutzt wird, sei es nun Mensch, Tier oder Elektronik. Ohne die Fähigkeit ”Räume wahrzunehmen”, sei es nun in einer Wohnung oder der Stadt, wäre es unmöglich, sich in diesen Räumen zu orientieren.

Es gibt unterschiedliche Arten wie Informationen über Räume aufgenommen werden können. Primär geschieht dieses über den visuellen Sinn, das Sehen. Doch ist dieses nicht der einzige Sinn, der Räume wahrnehmen kann. Es gibt weitere wie Haptik oder Akustik.

Durch die Vielfalt der beteiligten Komponenten, vom Sinn bis zur Information, gibt es viele Disziplinen, die sich das Gebiet der Raumwahrnehmung teilen. Ein paar davon sind: Psychologie, Kunst, Biologie, Neurowissenschaften und Informatik, wobei jede Disziplin ihre eigenen Schwerpunkte hat.

In der **Kunst** zum Beispiel gibt es eine Vielzahl von Künstlern, die sich mit räumlicher Darstellung beschäftigen, z. B. M.C. Esher mit seinen unmöglichen Konstruktionen und anderen Werken, die eine perspektivische Darstellung nutzen. Auf dem Bild 2.1 ist eine solche Konstruktion von unmöglichen Treppen zu sehen.
Es gibt ganze Stilrichtungen, die räumliche Konzepte in ihrem Kern haben, wie zum Beispiel perspektivische Malerei. Zur perspektivischen Malerei gehören auch jene Straßenmalereien, die, wenn aus dem richtigem Winkel betrachtet, wie dreidimensionale Objekte in den Raum hineinragen. [1] [9] Das folgende Bild 2.2 zeigt hierfür ein Beispiel.

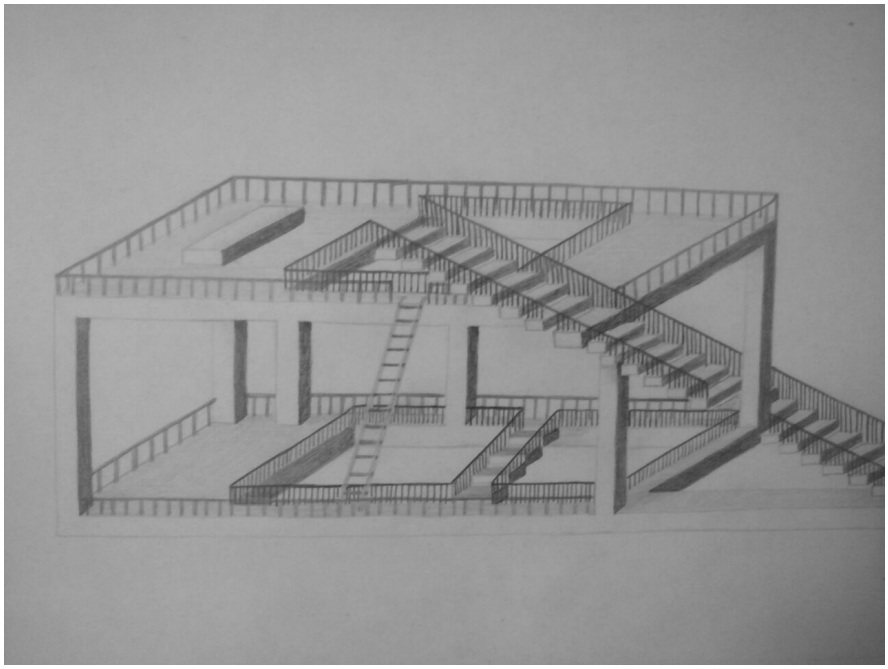


Abbildung 2.1.: Bild einer unmöglichen Treppen-Konstruktion [24]

Abbildung 2.2.: Street Art: Beneath every street. London, England. [9]

In der **Biologie** gibt es wieder ganz andere Schwerpunkte. Bienen beispielsweise, welche Informationen über die Umgebung mit Tänzen übermitteln oder Fledermäuse, die sich über akustische Signale orientieren.

In der **Informatik** ist Raumwahrnehmung wichtig für Systeme, die sich selber oder andere im Raum verorten müssen oder sich durch diesen bewegen, wie zum Beispiel autonome Roboter. Jedoch unterscheiden sich die Funktionsweisen der Raumwahrnehmung je nach Anwendung deutlich von der menschlichen Wahrnehmung. Doch haben sie alle Gemeinsamkeiten. Sie alle brauchen Orientierungspunkte, um sich zurechtzufinden und müssen diese in einen Kontext zueinander stellen. Seien es Geo-Positionen oder Bilddaten, die ausgewertet und dann mit Karten in Verbindung gebracht werden.

2.2. Landmarken

Es gibt unterschiedliche Arten die eigene Position im Raum zu verorten. Eine Möglichkeit dieses zu bewerkstelligen erfolgt über das Erinnern von Landmarken. [2, S. 249] Doch was ist eine Landmarke? Und welche Kriterien muss sie erfüllen, um als solche nützlich zu sein?

2.2.1. Definition

”Landmarke, ein weithin sichtbarer, in See- und Luftfahrtkarten eingetragener Ort bes. wichtig an der Küste als Orientierungsmerkmal für die nautische Ortung. [...]”, [26, Buch 11, S.91, Landmarke]

Landmarken waren schon immer Orientierungspunkte von großer Bedeutung, die nach Möglichkeit auch langlebig waren. Oftmals waren dieses deshalb Leuchttürme, Burgen, markante geologische Formationen oder Türme von Kirchen. Die Langlebigkeit war vor allem ein wichtiges Kriterium, da das Aktualisieren von Karten in der Vergangenheit mit deutlich mehr Aufwand verbunden war, als heute im digitalen Zeitalter.

Inzwischen wird der Begriff Landmarke als Sammelbegriff für alles genutzt, was als Orientierungspunkt bei der Wegfindung dient. Es handelt sich aber auch heutzutage meistens um auffällige Merkmale. [19, S. 248] Die Kriterien, die für die Eigenschaften einer Landmarke wichtig sind, unterscheiden sich allerdings von Fall zu Fall. In der Schifffahrt ist es weiterhin ein wichtiges Kriterium, dass Landmarken von weitem sichtbar sind, da sie vor allem der Positionsbestimmung auf großem Gebiet dienen. Beim Spazierengehen sieht dieses anders aus. Die Sichtweite ist oft deutlich beschränkter, was dazu führt, dass mehr lokale Landmarken

für die Orientierung verwendet werden. Wenn es dann aber zum Beispiel um eine Wegbeschreibung geht, ist es nicht einmal wichtig, dass die Landmarken besonders langlebig oder einprägsam sind, denn sie müssen nur einmalig ihren Zweck erfüllen. Oftmals ist es sogar möglich, während des Erklärens auf diese zu zeigen. Somit gelten für jedes Szenario eigene Regeln, wenn es um die wichtigen Eigenschaften für Landmarken geht. [2, S. 250]

2.2.2. Salience - Drei Modelle

Wenn es um die Qualität einer Landmarke geht, egal ob in der realen oder virtuellen Welt, wird dabei im Englischen von salience, zu deutsch: Auffälligkeit gesprochen.

Die salience wird oft genutzt, um die Relevanz von Landmarken zu beschreiben. Es gibt unterschiedliche Untersuchungen über die verschiedenen Merkmale und Konzepte, die einen positiven oder negativen Effekt auf die salience einer Landmarke haben können.

An dieser Stelle werden drei Modelle beschrieben, mit denen sich die salience von Landmarken charakterisieren lässt. Dieses heißt aber nicht, dass eine Landmarke exklusiv nur einer Kategorie angehört. Die einzelnen Modelle zeigen lediglich die unterschiedlichen Aspekte, die erinnert werde, je nach dem in welche Kategorie die Landmarke fällt. [19, S. 210]

Optische salience beschäftigt sich mit den optischen Eigenschaften, wie Form, Farbe und Kontrast einer Landmarke. Zum Beispiel sind große Landmarken einfacher wiederzufinden als kleine. In dem Artikel ”The Giessen virtual environment laboratory: human wayfinding and landmark salience” – [19] wurde untersucht wie Merkmale Landmarken beeinflussen. Die Untersuchung hat zum Beispiel ergeben, dass die Farbe ein weniger starkes Kriterium ist als die Form, wenn es um das Erkennen einer Landmarke geht.

Semantische salience beschäftigt sich mit dem bereits vorhandenen Wissen über die Landmarke, wie Bekanntheit oder persönliche Bedeutung. Landmarken, die einem besser bekannt sind lassen sich so einfacher im Zusammenhang mit ansonsten unabhängigen Informationen erinnern und werden auch einfacher wiedererkannt.

Strukturelle salience unterscheidet sich ein wenig von den anderen beiden Kategorien. Sie wird durch das Wissen über die eigentliche Wegplanung wichtig, wie zum Beispiel die Anzahl der Abzweigungen. Allerdings wird die Strukturelle salience nur dann verwendet, wenn es keine direkten Landmarken gibt, da ihre Eigenschaften abstrakt sind. Eine Landmarke fällt somit in dieses Konzept, wenn sie zum Beispiel an der fünften Abzweigung nach links steht und die nächste Abzweigung rechts genommen werden muss.

2.3. Kognitive Karten

(Land-)Karten sind ein schon lange bekanntes Hilfsmittel wenn es um die Orientierung geht. Oftmals sind wichtige Merkmale hervorgehoben, um einfache Orientierungspunkte zu schaffen. Solche physischen Karten lassen sich in der heutigen Zeit einfach duplizieren. Karten können auch von mehreren Personen gleichzeitig genutzt werden, was den Austausch über diese vereinfacht.

Kognitive Karten aber sind bei jeder Person unterschiedlich, denn sie bestehen aus den Erinnerungen an Räumlichkeiten. Mit den unterschiedlichen Weisen wie Personen ihre Umgebung wahrnehmen und erinnern, ist es normal, dass sich die Erinnerungen unterscheiden. Dieses wirkt sich dementsprechend auch auf die kognitiven Karten aus, die so von Person zu Person unterschiedlich sind. Damit erschwert sich der Austausch, wenn es um das Weitergeben von kognitiven Karten geht. Deshalb findet der Austausch von kognitiven Karten meistens über Landmarken statt. Wenn es um die Qualität von kognitiven Karten geht, ist auch die Quelle der Information für diese Karten wichtig. In dem Artikel "Spatial knowledge acquisition from maps and from navigation in real and virtual environments"- [17] wird das Lernen einer Umgebung auf drei Arten untersucht: mit Karte, durch Begehung oder durch eine virtuelle Desktop Umgebung. (VR war im Jahr 1999 noch nicht erschwinglich).

Die Resultate zeigen, dass die Fehler, die bei Orientierungsaufgaben gemacht werden, sich je nach Lernmedium unterscheiden. Dieses ist darauf zurückzuführen, dass je nach Art der Informationsaufnahme die Fähigkeit kognitiven Karten zu generieren unterschiedlich erfolgreich ist. Obwohl dieser Prozess von der einzelnen Person abhängig ist, gibt es doch allgemeine Trends, wenn es um die Art geht wie mit diesen räumlichen Informationen umgegangen wird. [17, Figure 3-7] In diesen Trends sind sich die reale und virtuelle Begehung in vielen Punkten ähnlich. Die virtuelle Begehung lag in der Untersuchung in den meisten Fällen gegenüber der Realen zurück. Die Abweichungen beim Zeigen auf Landmarken mit dem Lernen über die virtuellen Begehung waren am schlechtesten, siehe Abbildung 2.3. Das Lernen mit Karte verschlechtert die Entfernungs-Schätzung, siehe Abbildung 2.4. Dafür ist dass Einschätzen von Positionen über Stockwerke hinweg mit Kartenwissen genauer, sofern die Probanden beim Zeigen die gleiche Ausrichtung wie die Karte hatten, siehe Abbildung 2.5.

Berücksichtigt man in diesem Rahmen die Weiterentwicklung der Technik, werden vermutlich die Differenzen in der Wahrnehmung zwischen Virtualität und Realität kleiner werden. Ob dies Einfluss auf die bisherigen Unterschiede im Lernen von Karten hat, muss sicher weiter untersucht werden.

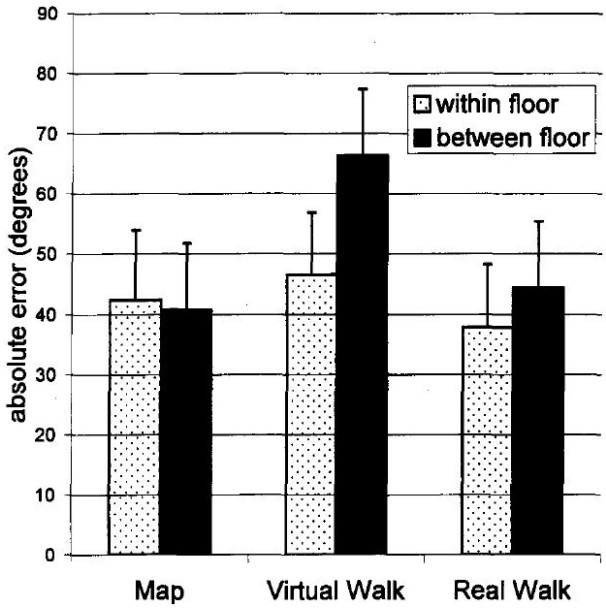


Abbildung 2.3.: Gesamte Abweichungen beim Zeigen auf Landmarken im selbem und anderen Stockwerken [17, Figure 3.]

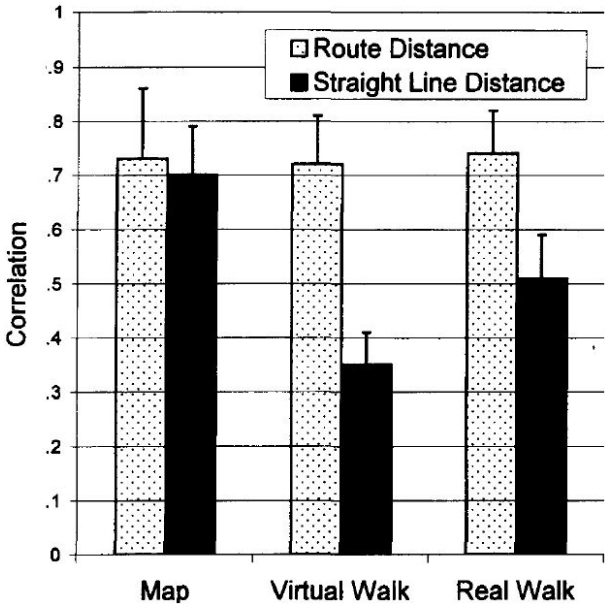


Abbildung 2.4.: Zusammenhang zwischen objektiver und geschätzter Entfernung [17, Figure 7.]

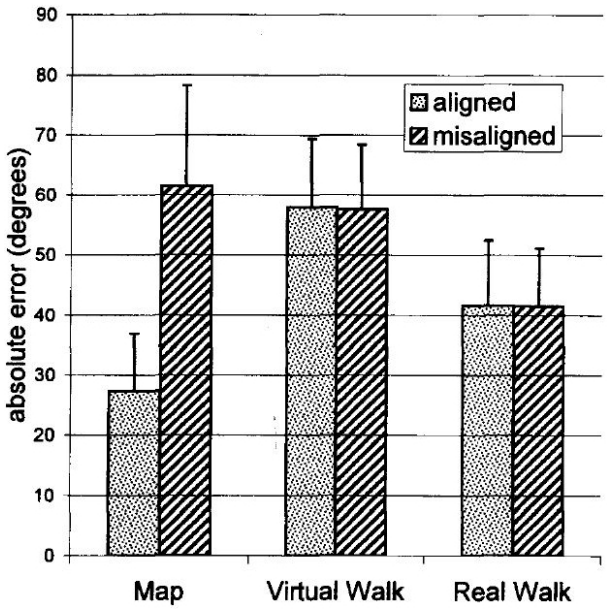


Abbildung 2.5.: Abweichungen beim Zeigen in Karten und nicht Karten Ausrichtung [17, Figure 5.]

2.4. Untersuchungen in Virtual Reality für die Reale Welt

Während in der realen Welt schon viele Untersuchungen über Raumwahrnehmung angestellt wurden, bietet die Virtual Reality ganz neue Möglichkeiten und Probleme, die in diesem Zusammenhang untersucht werden können. Die Frage, wie weit sich Ergebnisse aus der virtuellen Wellt in die reale Welt übertragen lassen, ist noch nicht geklärt. Dass nicht alle Erfahrungen auf die reale Welt übertragen werden können, ist am Beispiel eines Spieles deutlich, in dem der User ohne Hilfsmittel fliegen kann. Durch diese Diskrepanzen, die nicht nur durch die VR Technik entstehen, ist es nicht möglich, jede Art von Erkenntnis direkt in die reale Welt zu übertragen. In den meisten Fällen werden sich Annahmen übertragen lassen, wenn der virtuelle Versuchsaufbau nicht zu sehr von der Realität abweicht. Es wird aber noch Zeit vergehen bis, genaue Richtlinien für das Übertragen von Untersuchungen zwischen den beiden Welten existieren.

An anderer Stelle kann die in der virtuellen Welt existierende Freiheit aber auch genutzt werden, um Untersuchungen zu ermöglichen, die in der Realität nicht möglich sind. In ”Representation of impossible worlds in the cognitive map” - [10] handelt es sich um einen Ansatz herauszufinden, wie kognitive Karten aufgebaut sind. Um die Art der inneren Präsentation besser zu untersuchen, wurde mit unmöglichen Welten gearbeitet, da so angenommene Konzepte

gebrochen werden konnten. Zum Beispiel ein Rundgang, der aus drei Gängen besteht, bei dem die Wege aber alle im rechten Winkel zueinander stehen. Die Probanden durchliefen diesen Weg und hatten hinterher die Aufgabe die Strecke blind nachzugehen. Das Ergebnis war, dass die Winkel in denen die Gänge zueinander standen, richtig wiedergegeben wurden, auch wenn hierdurch die Topologie fehlerhaft wurde. Abbildung: 2.6.

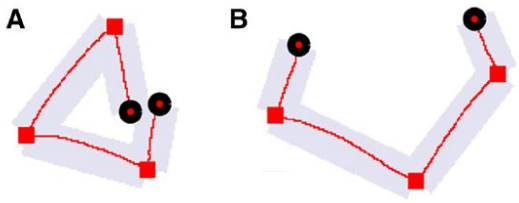


Abbildung 2.6.: Zwei Beispiele für Weg Reproduktion, (a) stammt von einem möglichem Dreieck und weist eine kontinuierliche Topologie auf während (b) von einem unmöglichem Dreieck stammt und eine nicht kontinuierliche Topologie aufweist. [10, Figure 7]

An dieser Stelle gibt es auch das Problem, dass Menschen in der VR andere Annahmen treffen als in der realen Welt. Durch den Wechsel ins Virtuelle sind ungewöhnliche und fantastische Dinge etwas, das dort als normal wahrgenommen werden kann. Die Hobbits im Herrn der Ringe bieten keine Überraschung, während sie in der realen Welt überraschen würden. In diesen virtuellen Welten stellt es nur für wenige ein Problem dar, solche Elemente zu akzeptieren. Ein weiter Punkt der noch untersucht werden muss ist, wie sich die veränderte Erwartungshaltung auf die Untersuchungen auswirkt.

2.4.1. Immersion

Wenn Untersuchungen in der virtuellen Welt gemacht werden ist es wichtig, dass diese eine hohe Immersion, einen hohen ”Wirklichkeits-Grad” besitzen. Besonders wenn Phänomene untersucht werden sollen, die auf die reale Welt übertragen werden sollen. In diesen Fällen ist es notwendig, dass die virtuelle Welt nicht von der Realen abweicht. Natürlich ist es auch möglich, Merkmale der realen Welt wegzulassen, um dann Rückschlüsse machen zu können, ob diese Merkmale von Bedeutung sind. Doch auch in solchen Fällen ist es Voraussetzung, dass die Abweichungen gering sind, da durch Abstraktion die Möglichkeit besteht, dass ganz andere Verhalten als gewünscht getestet werden. Dazu ist es hilfreich, wenn die virtuellen Umgebung als „real“ wahr genommen wird. Dieses bedeutet nicht, dass diese virtuellen Welten der Realität ähneln müssen, z. B. könnte es möglich sein an Wänden und Decken zu laufen.

Die virtuelle Welt muss aber als neue Realität akzeptiert werden, um die äußere reale Welt verblassen zu lassen. Um die Immersion zu erhöhen, ist es eine Möglichkeit viele aus der realen Welt bekannte Interaktionen in die virtuelle Welt zu übernehmen. Eine der wichtigsten Interaktionen ist in diesem Fall die Bewegung, als eine der häufigsten Interaktionen mit der Welt. Die Szenarien sollten auch in ihrer Ausstattung glaubwürdig wirken, um die Immersion zu erhöhen.

2.5. Exkurs Cybersickness

Cybersickness teilt sich einige der Probleme mit der Motion sickness und Simulator sickness. Es gibt auch auch noch weitere Bezeichnungen für andere Situationen, die ähnliche Symptome erzeugen. Die Symptome lassen sich dabei in drei grobe Kategorien einteilen, die weitestgehend überschneidungsfrei sind: Oculomotor (Augenschmerzen, Fokussierungs-Probleme, Kopfschmerzen), Nausea (Übelkeit, Schwitzen, Magenverstimmung) und Disorientation (Desorientierung, Gleichgewichts-Störung). Siehe Abbildung A.1 im Anhang auf Seite 73. Je nach Auslöser treten diese Symptome mit unterschiedlicher Stärke und Wahrscheinlichkeit auf. Bei der Cybersickness ist es vom Stärkstem zum Schwächstem: Disorientation, Nausea, Oculomotor. [16, S. 104]

Drei Theorien

"The three most prominent theories for the cause of cybersickness are poison theory, postural instability theory and sensory conflict theory."-[6, S.5]

Der Ursprung des Ganzen sind immer die selben Faktoren, die aber bei jeder Person anders ausfallen können. Die persönlichen Faktoren lauten: Alter, Geschlecht, Krankheiten und Position(Haltung). Die Faktoren der Hard- und Software lauten: Latenzen, Flackern, Kalibrierung und Ergonomie. Die Länge der in der VR verbrachten Zeit spielt ebenfalls eine Rolle, genauso wie die Art, wie diese Zeit verbracht wird. Eine ruhige Oase bietet wenig Gefahren für Cybersickness, bei einer Achterbahn sieht dieses schon anders aus. [6, S.6]

Worin sich die Theorien aber unterscheiden, sind die auslösenden Prozesse für die Symptome. Mit der Zeit sind unterschiedliche Theorien entstandenen und auch wieder verworfen worden. An dieser Stelle werden drei vorgestellt.

Die Vergiftungs Theorie (poison theorie) führt das Ganze auf einen evolutionär bedingten Mechanismus zurück, der die wahrgenommenen Konflikte für Halluzinationen hält. Der Körper

geht dann davon aus, dass etwas Giftiges gegessen wurde und versucht dieses dann über die Übelkeit wieder los zu werden. Diese Theorie hat sich für Cybersickness im Laufe der Zeit als ungenügend herausgestellt. [16, S.105-106] [6, Punkt 5]

Die Balance Instabilitäts Theorie (postural instability theory) bezieht sich darauf, dass es eines der Hauptziele für Menschen ist, stabil zu stehen. Die Aussage ist, dass die Symptome schlimmer werden je länger es nicht möglich ist, stabil zu Stehen. In der VR sind Annahmen über die richtige Position immer wieder mal falsch. Zum Beispiel ein Bild, dass an der Wand hängt, welches aber auf Grund seiner Ausrichtung vermuten lassen würde, dass die Gravitation in eine andere Richtung als angenommen zeigt, kann für Instabilität ein Auslöser sein. Siehe Abbildung 2.7.

Abbildung 2.7.: Beispiel für falsche Wahrnehmung von Gravitation [16, Figure 1]

Die Sensorische Konflikt Theorie (sensory conflict theory) besagt, dass die Symptome entstehen, wenn sich die einzelnen Wahrnehmungsaspekte widersprechen. Wenn zum Beispiel das Sehen sagt es geht vorwärts, das Gleichgewichtsorgan aber fühlt, dass grade keine Bewegung stattfindet. Dieses wäre besonders der Fall bei virtuellen Achterbahnfahrten oder wenn ein Controller benutzt wird, um Bewegung zu steuern.

Umgang mit Cybersickness

"The actual cause of cybersickness is not known and the underlying physiological responses uncertain."- [6, S.6]

Insgesamt ist es nicht im Detail klar, wie die Mechanismen funktionieren. Dennoch gibt es generelle Richtlinien, die helfen Cybersickness zu vermeiden. Gewisse Vorhersagen lassen sich

über Elemente treffen, die Cybersickness zuverlässig auslösen. Es gibt unterschiedliche Wege mit der Cybersickness umzugehen.

Nur weil jemand nun aber betroffen ist, heißt es nicht, dass er auf VR verzichten muss. Denn es gibt unterschiedliche Grade von Auswirkungen, die einen betreffen können. Dieses kann dazu führen, dass VR nutzbar bleibt, auch wenn dementsprechend die Nutzungszeit begrenzt werden muss. In vielen Fällen werden die Auswirkungen mit der Zeit schwächer, was an einer gestiegenen Toleranz liegen kann. Oder daran, dass vorkommende Effekte mit der Zeit akzeptiert werden. Wobei es natürlich gilt, dass es für alles Extremfälle gibt, was beutet dass es auch Menschen gibt, die VR überhaupt nicht nutzen können. Dafür gibt es aber auch Personen, bei denen die Probleme nur sehr schwer zu provozieren sind.

Somit lassen sich in vielen Fällen Probleme eingrenzen oder können auch in Kauf genommen werden, wie in "Cyber sick but still having fun". - [13] Dort ging es darum zu sehen, ob trotz absichtlich erzeugter Cybersickness ein Spiel noch Spaß machen kann, was schließlich auch das Ergebnis war. Letzten Endes muss jeder selber entscheiden wie viel Cybersickness er in Kauf nimmt.

Dennoch gibt es gewisse Kriterien die nachweislich Cybersickness auslösen und einige Konzepte, wie sich Cybersickness reduzieren lässt. In "Review on Cybersickness in Applications and Visual Displays" – [16, S.116] wird ein Überblick über Studien, die sich mit einigen der Parametern der Cybersickness beschäftigen, gegeben. Die Fortbewegungs-Geschwindigkeit ist einer der angesprochenen Parameter. Je schneller und mehr Bewegung vorhanden ist, desto höher ist die Gefahr für Cybersickness. Festgestellt wurde auch, dass die Art der Geschwindigkeitskontrolle einen Einfluss hat, so ist ein direktes Mapping von Neigung des Contoller auf Geschwindigkeit besser als wenn Schub gegeben wird, der erst zunimmt und dann wieder abnimmt. Die Nähe zur Realität kann ein weiteres Problem werden, wenn in der Szene die Hinweise für das "Oben" nicht denen der Realität entsprechen. In diesem Fall ist die Auswirkung geringer, wenn die Szene weiter von der Realität entfernt ist, da die Einflüsse so weniger stark bewertet werden. Ebenso ist die Größe des Sichtfeldes ein wichtiger Parameter. Je weniger Bild gesehen wird, desto geringer die Gefahr für Cybersickness. Eine zu starke Einschränkung stört aber auch das Erlebnis, da die Wahrnehmung der Welt eingeschränkt wird.

2.6. Bewegung in Virtual Reality

Wenn es darum geht, sich in virtuellen Welten zu bewegen, VR oder Bildschirm, gibt es eine Vielzahl von Möglichkeiten. Doch sind viele der Techniken im Roomscale VR nicht praktikabel in der Anwendung und nicht verträglich für den User.

Kabelanbindungen sind in den meisten Situationen störend und zu kurz, um diese mit einem Controller zu nutzen, somit lässt sich diese Gruppe nur in eingeschränkten Situationen nutzen. Kabellose Controller sind heutzutage kein Problem mehr aber auch nicht immer eine Lösung. Denn sie haben mit ihren Formen und Funktionen Vor- und Nachteile, abgesehen von dem Problem wie die Steuerung in Bewegung übersetzt wird. Durch die Eigenschaften der VR entstehen so Probleme mit den bekannten Bewegungsmodellen, doch gibt es auch gleichzeitig einen Vorteil, denn VR erlaubt auch die einfachste Art, sich zu bewegen. Mithilfe von Fortbewegung, wie sie auch in der Realität gehandhabt wird, z. B. mit gehen oder kriechen. Doch ist der Reale Raum meistens deutlich kleiner als der Virtuelle Raum und setzt diesen Fortbewegungsarten Grenzen.

Im Folgenden werden einige der Möglichkeiten, das Problem der zu kleinen Fläche zu umgehen, vorgestellt und deren Auswirkungen auf die Fortbewegung aufgelistet.

2.6.1. Techniken - Überblick

Bei den vorhandenen Techniken gibt es verschiedene Ansätze, die reale Fläche zu erweitern. Einer ist der Einsatz einer Omnidirectional Treadmill, einem Laufband mit zwei Freiheitsgraden (2DoF), das sich in der Ebene in alle Richtungen bewegen kann. Ein weiterer Ansatz ist es, die Virtuelle Welt so zu manipulieren, dass der Nutzer die zur Verfügung stehende Fläche nicht verlässt (Redirection) oder indem der User aktiv mit dem System zusammenarbeitet, um Redirection zu aktivieren.

Des Weiteren gibt es auch Ansätze für spezielle Lösungen. In dem Artikel ”Interactive System Based on the False Perception of Acceleration and Verticality” - [15], wurde damit gespielt, dass die Geschwindigkeits-Wahrnehmung über eine Kippung der Sitzfläche manipuliert wurde. Da über die Kippung des Sitzes das Gleichgewichtsorgan stimuliert wird, ist die Wahrnehmung der Beschleunigung so verstärkt. Die Einsatzmöglichkeiten von solchen Ansätzen sind beschränkt, erlauben aber dennoch Einblicke in einzelne Mechaniken der Wahrnehmung.

2.6.2. Omnidirectional Treadmills

Omnidirectional Treadmills erweitern die verfügbare Fläche, indem sie den User in der Mitte halten. Durch die benötigten zwei Freiheitsgrade werden diese Laufbänder sehr komplex. In "The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds" - [5] von 1997 wird eine solches vorgestellt. Die verfügbare Fläche betrug 1,27 m Kantenlänge. Das Laufband hatte eine Größe von 2,2 m mal 2 m und wog 544 kg. Dieses Laufband bestand aus mehreren tausend einzelnen Rollen. Das Bild des Laufbandes von 1997: 2.8.

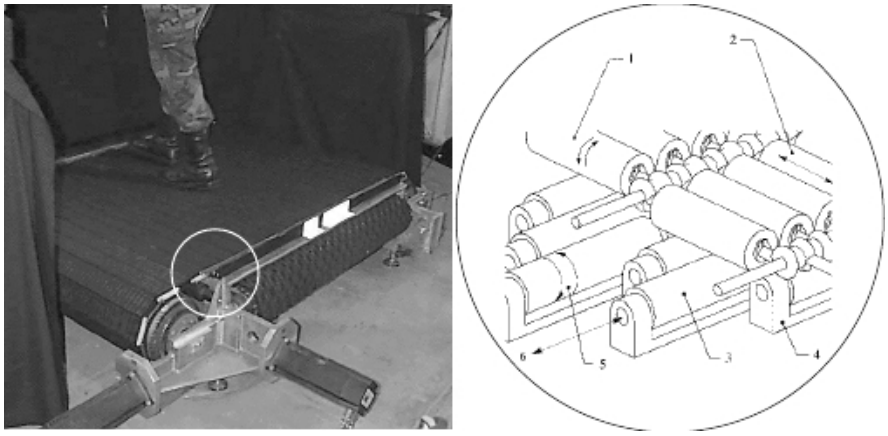


Abbildung 2.8.: Omnidirectional Treadmill von 1997 [5, Figure 4]

In einem neueren Artikel von 2011 "CyberWalk: Enabling Unconstrained Omnidirectional Walking Through Virtual Environments" - [20] geht es um ein neueres Modell, das eine nutzbare Fläche von 4 m Kantenlänge bietet. Das Gewicht dieses Laufbandes beträgt allerdings 12000 kg, wobei die bewegte Masse davon 7500 kg wiegt. Bei diesem handelte es sich um ein Laufband, das aus 25 einzelnen je 0,5 m breiten Laufbändern besteht 2.9.

Ein Problem besteht im Beschleunigen und Abbremsen auf diesen Laufbändern, da dieser Vorgang zu Instabilität führt, wenn der Untergrund den selben Vorgang durchläuft um die Bewegung auszugleichen. Ein weiteres Problem sind die mechanischen Kräfte, die nötig sind, um das mehrere tausende Kilogramm schwere Laufband zu bewegen. Es müssen Vorsichtsmaßnahmen getroffen werden, damit es unmöglich ist mit den Händen an den Boden zu kommen und damit die Mechanik im Falle eines Sturzes sofort abschaltet, um Verletzungen zu vermeiden. Damit sind diese, bis die Entwicklung fortschreitet, noch kein Gut, das außerhalb von speziellen Forschungsanlagen nutzbar ist.

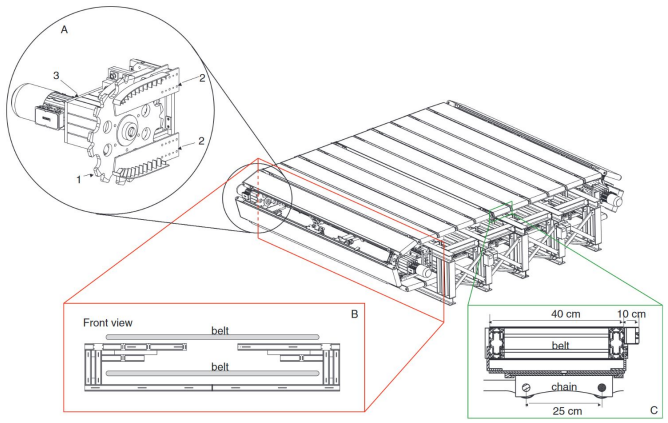


Abbildung 2.9.: Omnidirectional Treadmill von 2011 [20, Figure 1]

2.6.3. Redirection

Wenn es nicht möglich ist den realen Raum zu manipulieren, um unendliche Weiten zu erzeugen, dann fällt die Wahl auf den Virtuellen, wobei hier natürlich aufgepasst werden muss, dass dessen Manipulation nicht auffällt, um keine Verwirrung zu erzeugen oder Immersion zu brechen. Dieses hat dazu geführt, dass sich unterschiedliche Ansätze entwickelt haben, um diesem Problem beizukommen.
Manche Techniken manipulieren ganz direkt den Raum im Unbemerkten, während andere Systeme die Mithilfe des Nutzers benötigen um zu funktionieren.

Wenn es darum geht den User auf einer Kreisbahn zu lenken ist es wichtig, dass der Radius groß genug ist, damit der dadurch entstehenden Fehler beim virtuellem geraden Gehen nicht wahrgenommen wird. Für die minimale Größe des Radius gibt es unterschiedliche Werte. In "Revisiting Detection Thresholds for Redirected Walking: Combining Translation and Curvature Gains" - [8, S.113 - Abstract] wird als Ergebnis ein Radius von 11.6 Meter angegeben. Es wird auch noch ein weiterer Kurvenradius für ein anderes Experiment mit 6,4 Metern angegeben, wobei dort angenommen wird, dass die Probanden auf Grund des Testaufbaus mit der Zeit toleranter geworden sind. Ein anderes früheres Experiment kam auf einen Radius von 22 m. - [21, S.25 – 4.4.3]

Doch selbst Flächen von einem Radius von 6,4 m stehen nicht jedem zur Verfügung. "Safe-&-round: Bringing Redirected Walking to Small Virtual Reality Laboratories" - [12] zeigt einen Weg wie Redirection dennoch funktionieren kann. Dieses bedeutet allerdings, dass der User das Ganze aktiv steuert, indem der vorhandene Bereich aufgeteilt wird. Im inneren ist die Zone

in der normale Bewegung möglich ist und außen ein schmaler Ring, der für die Redirection genutzt wird, die solange aufrechterhaltenen wird, bis der User wieder in die Mitte zurücktritt.

Andere Systeme nutzen die normalen Drehungen des Kopfes, um innerhalb des Toleranzbereiches Diskrepanzen zu erzeugen, die dazu führen, dass der vorhandene Bereich nicht verlassen wird. Viele Systeme versuchen dabei kontinuierlich die Ausrichtung anzupassen, was aber auch eine gewisse Mindestgröße der Fläche voraussetzt, damit ein solches Prinzip funktionieren kann. [27] [22]

Ein anders Konzept arbeitet wiederum mit distractors (etwa: Ablenkern), Objekte, die die Aufmerksamkeit des Users auf sich ziehen, um ihn abzulenken und Kopfbewegungen zu provozieren, damit im Hintergrund die Welt neu ausgerichtet werden kann. In ”Towards Imperceptible Redirected Walking: Integrating a Distractor into the Immersive Experience” - [4] wurde ein virtueller Drache genutzt, der den User ablenkte und ihn dazu anregte, den Kopf zu drehen, um die Ausrichtung nachzubessern, wenn er dem Rand der begehbaren Fläche zu nahe kam.

Des Weiteren gibt es noch viele Ideen und Möglichkeiten, die noch erkundet werden, wie die Möglichkeit die Virtuelle Welt während des Blinzelns zu drehen, um die verursachte Bewegung zu verschleiern, wobei diese natürlich entsprechend klein sein müssen. [11]

Redirection Treadmills

Es gibt auch Ansätze, die versuchen mit einem herkömmlichem Laufband auszukommen, hierbei ist es die Idee mithilfe von Redirection dafür zu sorgen, dass der User weiterhin geradeaus geht, damit er in der Laufrichtung des Laufbandes bleibt. Dies Verfahren hat den Vorteil, dass die Kosten für das System deutlich niedriger sind, da keine speziellen Laufbänder entwickelt werden müssen. Der Nachteil hierdurch ist, dass die Breite des Laufbandes im Zusammenhang mit der Geschwindigkeit die maximalen Kurven bestimmt und Seitwärtschritte nur stark begrenzt möglich sind.

2.6.4. Bewegung mit Controller

Der Nutzen von Controllern ist aber nicht vollkommen ausgeschlossen. Besonders jene, die speziell für ein Szenario gebaut sind, bringen dort einen Vorteil. Dafür lassen sie sich außerhalb davon meist nicht nutzen. Die direkteste Methode, die Bewegung wie in den meisten Spielen zu steuern, ist für die VR allerdings nicht besonders geeignet, da Cybersickness durch das Beschleunigen und Bremsen in den meisten Fällen verstärkt wird. Es gibt allerdings auch weitere Bewegungskonzepte, die andere Mechanismen nutzen um Bewegung zu realisieren.

Eines davon nutzt Teleportation, durch den direkten Platzwechsel soll Cybersickness vermieden werden. Andere Verfahren schränken die Bewegungsfreiheiten und oder das Sichtfeld ein. Vollständig wird das Problem aber nicht behoben, besonders da nicht jeder Mensch mit jedem Bewegungskonzept kompatibel ist.

2.7. Wegfindung in Virtual Reality

Wegfindung funktioniert auch in der VR, doch kommen mit den verschiedenen Methoden und ihrem Aufbau auch Abweichungen zur Realität zum Vorschein. Einige von diesen stellen Probleme dar und andere lassen sich gezielt nutzen, um Wegfindung zu manipulieren.

2.7.1. Schwierigkeiten in der VR: Rotationen und Abstände

Wird ein Controller in der VR nicht nur für Bewegung sondern auch noch für Rotation genutzt, kann dieses zu sogenanntem ”Nonturner” Verhalten führen. Dieses bedeutet, dass die eigene Rotation nicht richtig mit den zurückgelegten Kurven geändert wird, Abbildung 2.10. In dem Beispiel wird an Punkt x1 die eigene Rotation nicht aktualisiert, was dazu führt, dass der Ursprung in der falschen Richtung vermutet wird. Nonturner glauben nämlich weiterhin, dass sie noch immer in die ursprüngliche Richtung sehen. In dem Artikel ”Moving through virtual reality without moving?” - [18, S.293], wurde festgestellt, dass eine realistischere Umgebung die Fähigkeit, die eigene Rotation zu aktualisieren, verbessert. Dieses deutet drauf hin, dass Immersion für Rotationen wichtig ist.

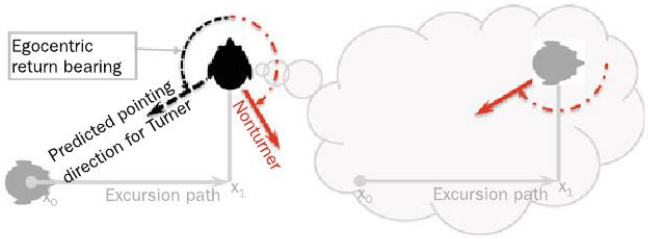


Abbildung 2.10.: Darstellung des Verhaltens von Turner und Nonturner. Nonturner haben Probleme ihre Rotation richtig zu erneuern, so gehen sie von einer falschen Drehung aus. [18, Figure 1]

Ein weiteres Problem ist das Einschätzen von Distanzen, die in VR für gewöhnlich unterschätzt werden. In ”A Full-body Avatar Improves Egocentric Distance Judgments in an Immersive Virtual Environment” - [14] wurde untersucht, wie das Anzeigen des eigenen Avatars in VR

die Fähigkeit Distanzen einzuschätzen beeinflusst. Das Ergebnis war, dass die Fehler, die bei den Distanzen gemacht wurden, kleiner wurden, wenn der Avatar mit eingeblendet war.

2.7.2. Unnatürliche Welten

Die Fähigkeit seine Wege in der VR zu finden unterliegt den bereits genannten Schwierigkeiten. Doch wie wirkt es sich auf die Wahrnehmung und Wegfindung in VR aus, wenn die Räume nicht mehr euklidisch sind? Wenn ein Rundgang weniger als 360° hat oder die Winkel eines Dreiecks alle 90° haben? Also die virtuellen Räume nicht mehr den normalen Gesetzen räumlicher Vorstellung unterliegen? Zu solchen Räumen mit topologischen Unmöglichkeiten gibt der Artikel "Representation of impossible worlds in the cognitive map" - [10] eine Erklärung. Wenn nun lediglich einzelne Eigenschaften von Winkeln und Entfernungen erinnert werden, sorgt dieses für eine geringere Chance, dass eine solche Unmöglichkeit auffällt oder die Wegfindung beeinflusst wird. Wie es aber mit komplexeren Szenarien aussieht, ist noch nicht untersucht.

Eine weitere Untersuchung ist in dem Artikel "ARES: An Application of Impossible Spaces for Natural Locomotion in VR" - [7] zu den Auswirkungen von topologisch unmöglichen Räumen geschehen. Dort wurde ein kleines Spiel gebaut, das mit einer recht kleinen Fläche auskommt, da sich Bereiche überlagern aber durch Gänge verbunden sind. Es ist nicht möglich zwei Räume gleichzeitig von dem Verbindungsgang aus zu sehen, so können die Räume ausgetauscht werden, um die Fläche mehrfach zu nutzen. Das Ergebnis war, dass diese unmöglichen Konstellationen nicht negativ aufgefallen waren, wenn sie denn überhaupt bemerkt wurden. Die Orientierung in dem Spiel wurde durch diesen Mechanismus nicht beeinträchtigt. Das verwendete Szenario ist in Abbildung 2.11 zu sehen.

2.7.3. Unsichtbare Änderungen - change blindness

Wenn Räume sich in unbeobachteten Momenten verändern und diese Änderungen nicht erkannt werden, wird von change blindness gesprochen. Solche Änderungen können genutzt werden, um Räume zu erweitern. In "Exploiting change blindness to expand walkable space in a virtual environment." - [23] wird dieses Phänomens genutzt, um einen langen Gang mit vielen anliegenden Zimmern zu schaffen, die alle der Reihe nach aufgesucht werden können.

Die Grafik 2.12 zeigt einen der Wechsel, die im Hintergrund vorgenommen werden können, ohne dass diese auffallen. Die auf der Abbildung befindliche Szene stammt aus dem vorher genantem Szenario. Die Abbildung eine Ausschnittes aus dem Szenario befindet sich im Anhang A.2 auf Seite 74. Die zweite Abbildung zeigt den Wiederkehrenden Ablauf der in dem Experiment stattgefunden hat. Der User betritt einen Raum und geht auf den Bildschirm zu.

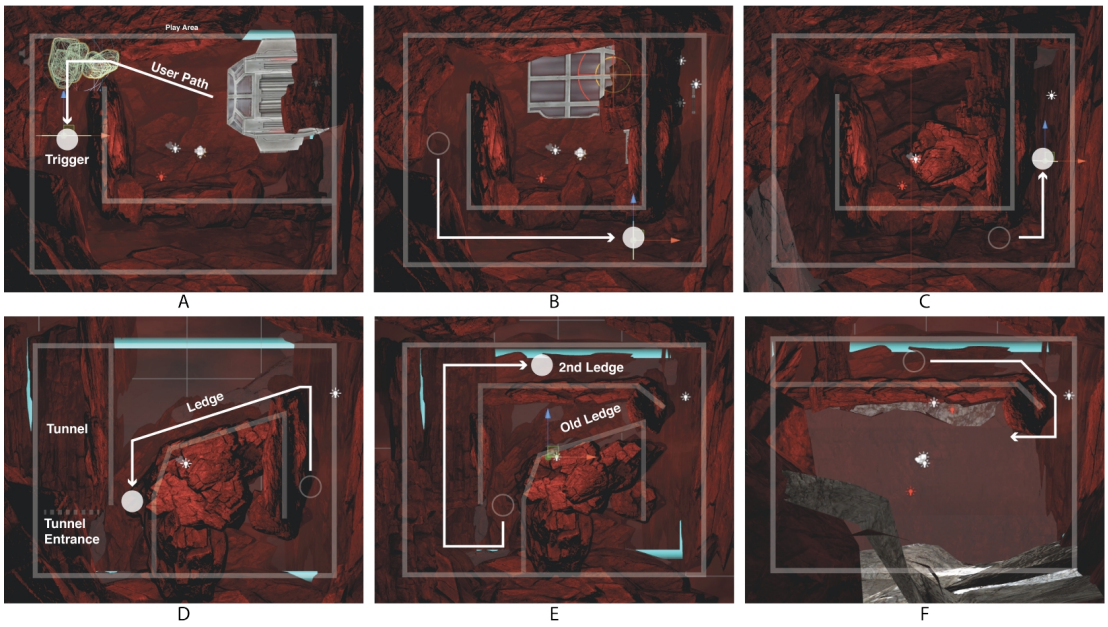


Abbildung 2.11.: Darstellung von übereinanderliegender Architektur. Die weißen Kreise stellen Auslöser da, die das aktuelle Szenario mit dem Nächstem austauschen. Bilder A, D, und F sind drei unterschiedliche Räume. B, C, und E sind schmale Korridore und Vorsprünge zischen den Räumen. [7, Figure 1]

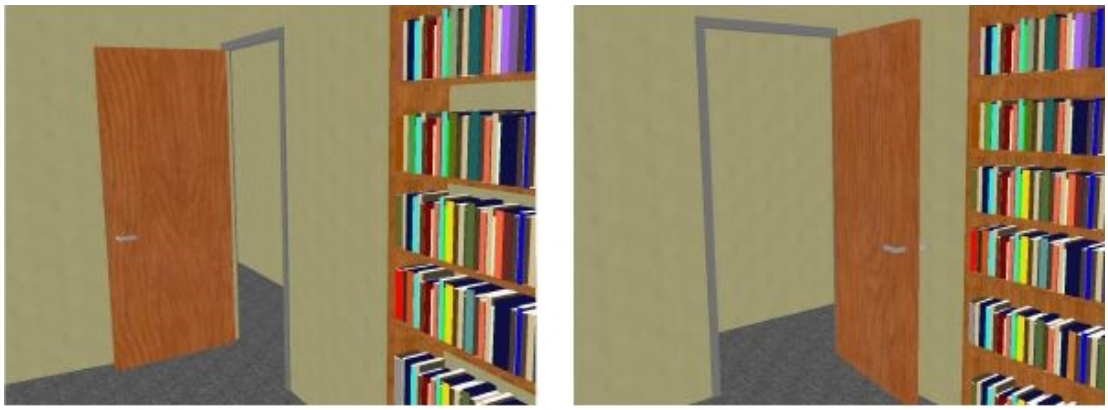


Abbildung 2.12.: In der Abbildung ist einer der Wechsel abgebildet die in den meisten Fällen unerkannt bleiben.[23, Figure 1]

Der User aktiviert den Bildschirm und augenblicklich wird der Raum um 90 Grad gedreht. Der User verlässt den Raum und der alte Raum wird durch einen neuen ersetzt. Der User betritt

einen Raum und ... Damit wird das ganze eine Endlosschleife, die es Schafft, den User innerhalb der beschränkten Fläche zu halten, in dem Experiment waren die Räume zum Glück auf 12 beschränkt.

Diese Technik erzeugt letztendlich auch unmögliche Räume, nur dass sie diese auf eine andere Art und Weise erzeugt. Auch diese Technik hat in dem Testszenario kein Unwohlsein verursacht. Wenn es um die Betrachtung im Rahmen von Desorientierung geht, hat dieses Verfahren wahrscheinlich stärkere Auswirkungen, als die unnatürlichen Welten, die stabile Karten erzeugen. Wie weit die unsichtbaren Änderungen sich nun auf die Wegfindung auswirken ist noch nicht untersucht.

2.8. Untersuchungsmethoden in Virtual Reality

Um qualitative Untersuchungen in VR zu machen, sind entsprechende Methoden notwendig. An dieser Stelle sollen ein paar der Methoden angesprochen werden, die im Zusammenhang mit dieser Arbeit relevant sind. Sie sind jedoch nicht allein maßgebend für die Anforderungen, denn es sollen beliebige Ansätze unterstützt werden.

Fehlermessung beim Zeigen auf Landmarken

Wenn es um Orientierung geht, ist das Zeigen auf Landmarken eine Methode Abweichungen festzustellen, die bei der Orientierung stattgefunden haben. Mithilfe von Controllern lässt sich dieses in der VR automatisieren, was den Vorgang einfacher macht und die Fehler in der Messung reduziert. Alternativ reicht auch schon das HMD aus, wobei dann in die entsprechende Richtung geblickt werden muss.

Fehlermessung in der Wegfindung

Eine weitere Möglichkeit ist es die Fehler zu messen, die in der Wegfindung stattfinden. Zum Beispiel in einem Irrgarten, der am Anfang auf irgendeine Weise gelernt wird und hinterher abgelaufen werden soll. In diesem Fall könnten unterschiedliche Landmarken vorhanden sein und es könnte gemessen werden, wie viele Fehler in der Wegfindung durch diesen Irrgarten stattfinden.

Messung von Cybersickness

Cybersickness wird in vielen Untersuchungen über Fragebögen festgestellt, da sie eine recht einfache Lösung sind, die Wirkungen auf Probanden aufzunehmen. Es kann auch sinnvoll sein,

mehrere Fragebögen in einem Durchgang ausfüllen zu lassen, um Ausreißer zu vermeiden, denen bereits vorher schlecht war. Cybersickness lässt sich ebenfalls mit Hilfe von Sensoren feststellen, indem die Körperparameter überwacht werden, wobei diese Art von Untersuchung eine höhere Hürde für die Probanden darstellt.

2.9. Analyse

Dieses Kapitel beginnt mit einer Zusammenfassung des Wissens aus der Literatur. Darauf werden einzelne Szenarien beschrieben, mit denen Untersuchungen betrieben werden können. Abschließend geht es um das Ziel der Arbeit: Eine Plattform zu entwickeln, die bei Untersuchungen zur Raumwahrnehmung unterstützt.

2.9.1. Revision der Literatur

In vergangenen Arbeiten wurden einzelne Fragen bereits geklärt. Was ist die Raumwahrnehmung? Wie und wann helfen Landmarken 2.2? Das mentale Karten sich von Person zu Person unterscheiden, aber auch Gemeinsamkeiten haben 2.3 und dass die Quelle der Informationsgewinnung einen großen Einfluss auf kognitive Karten hat. Da es das Ziel ist sich mit Rotationen in der VR zu beschäftigen, ist auch das Thema der Cybersickness wichtig 2.5. In dem Kapitel wurden drei unterschiedliche Theorien vorgestellt, von denen zwei für diese Arbeit von Interesse sind. Die Balance Instabilitäts Theorie und die Sensorischer Konflikt Theorie wären beide etwas, dass bei dieser Art von Untersuchung Plausibel ist.

Danach ein Abstecher zu unterschiedlichen Bewegungskonzepten, die in der VR möglich sind 2.6.1, um die begrenzte reale Fläche zu erweitern. Omnidirectional Treadmills, die bereits existieren, aber noch zu kolossal sind und andere Methoden, wie Teleportation und Redirection. Danach geht es um einige Probleme, die in der VR existieren, wie zum Beispiel ”Noturner” Verhalten, dass durch eine realistischere Umgebung gemildert werden kann. Ein ähnliches Problem ist, dass Entfernungen in der VR generell unterschätzt werden. Insgesamt ist das bewegen in unendlichen Welten noch in seinen Möglichkeiten eingeschränkt. Auch wenn für einige Fragen bereits Erklärungen und Ergebnisse vorliegen, gibt es noch viele Unterschiede zwischen den Welten, die noch nicht untersucht und erklärt sind.

Das Interesse dieser Arbeit besteht darin, wie sich Unterschiede durch Rotationen auf die Raumwahrnehmung auswirken. Insgesamt gibt es viel einzelne Fragestellungen die es zu beantworten gilt. Wird die Orientierung in der VR schneller verloren? Wie wirken sich die Rotationen auf die Orientierung aus? Welche Effekte treten durch Rotation verstärkt auf,

wie z. B. Cybersickness? Es gibt viele Punkte in denen sich die reale von der virtuelle Welt unterscheidet. Um diese Unterschiede zwischen den Welten zu untersuchen ist es notwendig, dass die entsprechenden Phänomene in der VR erlebt werden können.

2.9.2. Szenarien

An dieser Stelle werden einige Szenarien beschrieben die helfen können, einzelne Charakteristika zu untersuchen. Dabei handelt es sich hier nur um einen kleinen Ausschnitt der möglichen Szenarien, die untersucht werden können. Der Fokus liegt dabei mehr auf den Komponenten, die für die VR benötigt werden, als den Methoden, die für die Ergebnisse genutzt werden.

Erkundung von Räumen über die Decke

In diesem Szenario geht es darum, dass ein Raum erkundet wird, in dem der User kopfüber an der Decke steht. Die Frage ist, wie gut die Positionen von Landmarken im Raum eingeschätzt werden können, wenn diese danach vom Boden aus geortet werden. Hierfür ist freie Bewegung in der Ebene nötig, um das Raumgefühl zu verstärken und einen besseren Einblick in die Anordnung der Landmarken zu gewinnen. Ein entsprechendes Szenario mit Landmarken, das in der Virtual Reality erkundet werden kann, ist ebenfalls nötig. Zur Auswertung könnte das Verfahren aus dem Abschnitt ”Fehlermessung beim Zeigen auf Landmarken” genutzt werden.

Erkundung von Dreidimensionalen Wegen

Für dieses Szenario, das in seiner einfachen Version zur Eingewöhnung in die VR und kippende Räume dient. Die ersten Erfahrungen können mit diesem Vorgehen in Räumen mit ungewohnten Reaktionen gemacht werden. Auch Wegbeschreibungen für dreidimensionale Wege können in diesem Szenario untersucht werden. Dazu muss es im Raum möglich sein zwischen den einzelnen Flächen zu wechseln, z. B. zwischen Boden, Wand und Decke. Um das zu Realisieren muss es möglich sein an bestimmten Orten (Trigger) die eigene Rotation zu ändern. Dafür ist die freie Bewegung notwendig sowie ein passendes Szenario.

Auswirkung von Rotation, kontinuierlich oder manuell ausgelöst

Eine weitere Untersuchung kann über die Auswirkung von manueller Rotation, die über Trigger ausgelöst wird, im Vergleich zu kontinuierlicher Rotation, die sich dauerhaft dem Untergrund anpasst, gemacht werden. Dafür muss es möglich sein zwischen den beiden Arten hin und her zu schalten. Um die Variablen für dieses Szenario einzugrenzen, könnten bestimmte

Rotationspunkte auf Verträglichkeit geprüft werden. Ist es angenehmer, wenn in einem Bogen die Rotation in grober schrittweise oder kontinuierlich angepasst wird.

Verlust von Orientierung

Woran lässt sich der Verlust von Orientierung erkennen? Wie kann die Orientierung aufrecht erhalten, bzw. wiederhergestellt werden? Für einen solchen Test ist vor allem die grafische Seite interessant. Des Weiteren sollte es möglich sein, sich frei bewegen zu können.

Ebenso wäre ein Raum möglich, in dem sich unterschiedliche Landmarken befinden und das Ziel wäre es, nach einer Lernphase, zu erkennen auf welcher Wand man sich gerade befindet, nachdem diese zufällig geändert wurde. An dieser Stelle wäre es ausreichend eine Unterscheidung zwischen Boden, Wand und Decke zu machen, um erste Ergebnisse zu bekommen. Unterschiedliche Einrichtungen wären z. B. ein typisches Wohnzimmer mit Stühlen, Tisch, Fernseher, Bilder, Lampe, usw., eines das auf alle Flächen verteilt ist und ein Raum der nur mit abstrakten Formen gefüllt ist die keine typischen oben- und unten Merkmale haben.

2.9.3. Zielsetzung

Um die vorgestellten Untersuchungen auf einheitlicher Basis zu ermöglichen, muss noch weiter gegangen werden. Es werden Stellgrößen benötigt, die den Mittelpunkt einer Untersuchung bilden, während der Rest stabil bleibt. Das erfordert, dass die Szenarien entsprechend Vorbereitet sind, um die einzelnen Charakteristika zu untersuchen. Damit diese und weitere Szenarien überhaupt einheitlich umgesetzt werden können ist eine Plattform notwendig.

Daraus ergibt sich auch das Ziel dieser Arbeit, das Entwickeln einer Plattform. Diese Plattform soll den Prozess ”Szenarien für die Untersuchungen erstellen” vereinfachen und ermöglichen. Ein weiterer Vorteil ist das Szenarien so einheitlich und besser vergleichbar werden.

Damit die Plattform als Grundlage für die Untersuchungen von Rotationen in der VR genutzt werden kann, muss die Plattform einen Mehrwert bei der Entwicklung von Szenarien haben. Dazu muss sie dafür sorgen, dass die Logik nicht jedes Mal neu implementiert werden muss und wenn Änderungen benötigt werden, diese einfach durchzuführen sind. Zu den Kriterien in denen die Plattform helfen soll sind folgende:

1. Freie Bewegung in der VR ermöglichen
2. Die Änderung der Rotation ist kontrolliert möglich
3. Die Plattform ist nicht an ein bestimmtes HMD gebunden

4. Die Plattform ist einfach einzusetzen
5. Die Plattform lässt sich erweitern
5. Die Plattform ist weitestgehend nicht Hardware gebunden
6. Die vorhandenen Komponenten lassen sich Parametrisieren
7. Untersuchungsabläufe lassen sich kontrollieren und steuern

Damit die Funktionsweise der Plattform auch an einem konkretem Beispiel gezeigt werden kann, soll ein entsprechendes Szenario im Rahmen dieser Arbeit entstehen. Dieses Szenario soll die Basisfunktionen der Plattform und die Lauffähigkeit in einer produktiven Umgebung zeigen. Dazu muss das Szenario sowohl von freier Bewegung, als auch von Rotation Gebrauch machen. Damit die Rotation aber wahrnehmbar wird, muss es sich um ein Szenario handeln, in dem ausreichend Landmarken vorhanden sind. Ebenso soll die Rotation nicht zufällig ausgelöst werden, weshalb auch entsprechende Handlungspunkte vorhanden sein müssen. Da dieses Szenario als Einführung dient wäre es auch sinnvoll, wenn es einen abgeschlossenen Handlungsstrang besitzt. Damit sollten auch die Stellgrößen nicht zu groß sein, um erste Eindrücke mit dem Szenario gewinnen zu können.

3. Design der Plattform

Um Untersuchungen in VR zu machen, müssen die Anforderungen verschiedener Szenarien an die Plattform erfüllt werden. Dieses Kapitel beschäftigt sich mit den Anforderungen, dem Design und der Implementation. Als erstes geht es um die Anforderungen, die an die Plattform gestellt werden. Darauf folgt die Umsetzung, in der die Implementation und Schnittstellen beschrieben werden. Am Ende dieses Teilabschnittes wir ein Szenario mit der beschriebenen Plattform erstellt. Abgeschlossen wird dies Kapitel von einer Beurteilung der Umsetzung der Plattform aufgrund der Anforderungen.

3.1. Anforderungen

In diesem Kapitel wird es um die Anforderungen gehen, die zu erfüllen sind, um eine Plattform zu erschaffen, die es erlaubt unterschiedliche Tests durchzuführen. Dazu werden die Anforderungen beschrieben ohne tiefer auf spezifische Grundlagen einzugehen, so dass nach Möglichkeit keine Abhängigkeiten zu Software oder Hardware entstehen.

3.1.1. Allgemeines

In diesem Part werden die allgemeinen Anforderungen aufgelistet, die nicht in eine der größeren Kategorien fallen. Viele dieser Anforderungen beruhen darauf, dass die Plattform für eine ganze Reihe von verschiedenen Untersuchungen einsetzbar sein soll. Dazu muss die Plattform den Entwicklungsprozess von Szenarien zur Untersuchung von Raumwahrnehmung vereinfachen.

1. Frei einsetzbar
   Als ein wichtiges Merkmal gilt, dass die Plattform auch in andere Projekte eingebunden werden kann und sich so weiterverwenden lässt. Dieses macht es wichtig, dass die Plattform angepasst werden kann, um unterschiedlichen Szenarien gerecht zu werden. Um diese Bedingung zu erfüllen, ist es notwendig, dass mit der Plattform Szenarien einfach gebaut werden können. Dafür muss die Nutzung möglichst wenig zusätzlichen Aufwand bei ihrer Einbindung erzeugen. Es müssen entsprechende Stellgrößen in der Plattform vorhanden sein, die die entsprechenden Einstellungen erlauben.

2. Modularität der Plattform
   Ein allgemeines Kriterium für die entstehende Plattform ist, dass ihre Komponenten sich erneuern, austauschen und abschalten lassen. Ebenso muss die Plattform anpassbar sein, weshalb die Struktur der Plattform modular sein soll. Dieses hat den Vorteil, dass für Untersuchungen nicht-benötigte Komponenten ausgeschaltet werden können. Dieses erhört auch die Langlebigkeit der Plattform.

3. Ersetzbarkeit des HMD
   Das HMD, das in die Plattform eingebunden ist, sollte sich nach Möglichkeit ohne Eingriffe in den Code austauschen lassen. Dieses ist besonders aufgrund der Geschwindigkeit, mit der sich die HMDs weiterentwickeln, notwendig.

   Da an dieser Stelle eine Abstraktion des HMD vorgesehen ist, soll auch ein First Person Contoller über diese Stelle eingebunden werden, damit die Plattform sich auch ohne HMD betreiben lässt. Der First Person Contoller schränkt den Nutzen zwar ein, hilft aber beim Testen sowohl von Szenarien und als auch der Plattform. Des Weiteren wird es so möglich einen Teil der Eigenschaften auf Rechnern zu nutzen, die nicht VR fähig sind.

4. Nutzung in der VR
   Damit die Nutzung in der VR angenehm ist, müssen die Konzepte der Steuerung für den Benutzer einfach zu bedienen sein. Dazu sollten die grundlegenden Steuerkonzepte schnell zu verstehen sein. Des Weiteren sollte auch ein Hilfsmittel, das die Stabilität erhöht und Cybersickness reduziert, vorhanden sein.

5. Kontrollierbarkeit von Szenarien
   Damit sich die erstellen Szenarien kontrollieren und einrichten lassen, ist es wichtig, dass entsprechende Schnittstellen vorhanden sind. Diese sollten eine einfache Parametrisierung von Szenarien erlauben aber auch das Ändern von Einstellungen im laufenden Prozess. Die entsprechenden Schnittstellen sollen es einfacher machen, die richtigen Parameter zu testen und die Kontrolle zu behalten.

3.1.2. Hardware und Software

Es ist notwendig, dass alle Berechnungen, die während des Experiments anfallen, in Echtzeit erledigt werden können. Hieraus ergeben sich gewisse Anforderungen an die Hardware und Software. Diese werden durch die Notwendigkeit, ein Head Mounted Display(HMD) zu

unterstützen, noch weiter verschärft. Es gelten Echtzeitanforderungen mit Latenzen von maximal 50ms, hohe Frameraten und oft müssen auch 3 Bilder per Frame berechnet werden. Um diese Bedingungen zu erfüllen, ist eine entsprechende Software-Landschaft nötig, in der sich entwickeln lässt. Die Plattform muss für diese Software-Landschaft erstellt werden, doch auch die Szenarien müssen sich darin Erzeugen lassen. Die Software soll nach Möglichkeit mit unterschiedlicher Hardware kompatibel sein. Dieses ist erforderlich um den Einsatz an verschieden Orten unter verschiedenen Bedingungen möglich zu machen. Das ganze benötigt dann noch einen entsprechenden Rechner der das HMD betrieben kann.

3.1.3. Bewegung

Eine Welt will erkundet werden, was zu der Notwendigkeit führt sich in dieser zu bewegen. In VR muss dabei in den meisten Fällen zwischen zwei Arten von Bewegungen unterschieden werden. **Direkter Bewegung**, die sich innerhalb der Real verfügbaren Fläche abspielt und **Indirekter Bewegung**, die über diese Fläche hinaus geht. Direkte Bewegung wird in diesem Fall direkt von dem HMD aus abgegriffen und übertragen. Bei der indirekten Bewegung, die über die vorhandenen Grenzen hinausgeht, ist es notwendig ein Hilfsmittel zu nutzen um diese zu realisieren. Oftmals wird versucht, die indirekte Bewegung in eine direkte Bewegung umzulenken, sei es durch Treadmills die den User wieder in die Mitte zurück schieben oder durch Redirection um ihn wieder zurück-zu-Lenken. Andere Verfahren setzen auf Controller, um Bewegung zu Simulieren oder um den User zu Teleportieren.

In diesem Fall scheiden die oben genannten Bewegungsarten für die indirekte Bewegung aus. Allerdings sollen diese nachträglich zu implementieren sein, ohne tiefere Eingriffe vornehmen zu müssen. Der Aufwand hängt allerdings weiterhin von der gewünschten Bewegungsart ab. In diesem Fall soll auf eine einfache Bewegungsart zurückgegriffen werden, die sich auch mit Walking in Place (auf der Stelle Gehen) vertragen würde. Das Modell funktioniert folgendermaßen: Wenn der User sich an den Rand der vorhandenen Fläche begibt und somit eine gewisse Distanz zum Rand unterschreitet, beginnt er in die Richtung in die er sich vom Uhrsprung entfernt hat voranzugleiten. Dieses hört wieder auf, wenn der Abstand zum Rand wieder größer wird. Dieses hat den Vorteil, dass keine Controller benötigt werden und bei einer Gleitgeschwindigkeit, die der Geh-Geschwindigkeit ähnelt, die Bewegung sich recht nah am normalen Gehen orientiert.

Käfig

VR Systeme bieten in der Regel ihre eigenen Guardian Systems an, meistens in der Form von Gitternetzen, die auftauchen, wenn der User sich zu nah an den Rand der vorhandenen Spielfläche begibt und ihm so die Grenzen der Realen Welt anzeigt. Da diese Begrenzung normalerweise echte Wände darstellt, ist diese für die Bewegungsteuerung nicht sonderlich geeignet. Erstens da diese sich störend auf die Immersion ausüben, zweitens da aie im Zusammenhang mit der Bewegung zu spät warnen und zum dritten diese auch vom Tracking System abhängig sind.

Deshalb soll an dieser Stelle ein weiteres System entstehen, welches mit der Bewegung über Annäherung besser zusammenarbeitet, um die Grenze zu beschreiben. Wünschenswert ist es, dass diese Markierung in ihrer Größe editierbar ist, um die reale Fläche künstlich zu verkleinern und anzupassen, damit Kollisionen mit den Wänden vermieden werden. Ein weiterer Vorteil ist es, dass ein dauerhaft eingeblendetes System beim Halten des Gleichgewichts unterstützt und Cybersickness mindern kann.

Dieser soll es auch ermöglichen, dass die Mitte der nutzbaren Fläche verschoben werden kann. Dieses erlaubt das verfügbare Gebiet in Kombination mit einer angepassten Größe besser zu nutzen falls Gebiet Verschiebungen nötig sind, bei denen es nicht möglich ist das Tracking System neu zu kalibrieren, wenn dieses sich denn überhaupt wie gewünscht einrichten lässt.

Automatische Bewegung blockieren

Ein vorhandenes Problem ist, dass der User beim Starten nicht unbedingt in der Mitte der verfügbaren Fläche steht. Der Bewegungsmechanismus könnte so unbeabsichtigt Aktiviert werden, was nicht erwünscht ist. Je nach Startpunkt kann es passieren, dass auf dem Weg zur Mitte Hindernisse durchschritten werden müssen. Die Kollisionserkennung würde in diesem Fall anspringen und dieses Verhindern, was in diesem Fall ein Problem darstellen würde. Die einfachste Methode, dieses Problem zu beseitigen, ist es Bewegungsalgorithmen zu blockieren bis der Nutzer zum erstem mal die Mitte betritt. Dieses würde auch Probleme mit der Gravitation lösen. Sollte der Startpunkt in einem Szenario auf einer Plattform liegen die nicht die gesamte Tracking Fläche ausfüllt, könnte es dazu kommen das der User in einem Bereich startet, der keinen Boden hat. In dem Fall würde der User fallen, wenn die Physik bereits aktive ist. Doch mit dieser Hilfsmethodik ist auch dieses Problem gelöst.

3.1.4. Kollision

Wenn es um Kollisionen geht ist es wichtig, dass sich diese mit VR und Rotationen vertragen. Dieses bedeutet, dass gewisse Anforderungen an die Algorithmen zu stellen sind, denn direkte und indirekte Bewegung müssen unterschiedlich behandelt werden. Bei der indirekten Bewegung sind die normalen Maßnahmen ausreichend, um Kollisionsdurchdringung zu verhindern. Bei der direkten Bewegung ist es etwas komplizierter. Da in diesem System auch Gravitation verwendet wird, wäre es von Nachteil, wenn Begrenzungen wie Geländer durchdrungen werden können. Eine solide wirkende Wand verliert schnell ihre Immersion, wenn sie einfach durchschritten werden kann. Da es sich in diesem Fall der direkten Bewegung um reale Bewegung handelt, lässt diese sich nicht einfach kürzen, sondern ihr muss aktiv entgegen gearbeitet werden. Dieses bedeutet zwar, dass die Welt bei einer versuchten Durchdringung zu fliehen versucht. Dafür kann das Phänomen aber auch genutzt werden, um die virtuelle Begrenzung bis zur realen Begrenzung zu verschieben. Dieses wäre normalerweise nicht möglich, wenn diese Bereiche nicht weit genug am Rand liegen, um sie mit indirekter Bewegung zu Verschieben. Damit ist es somit ein Hilfsmittel für Bereiche, in denen keine indirekte Bewegung möglich ist.

3.1.5. Anpassung von Rotation um die waagerechte Achse

Um die gewünschten Tests durchführen zu können ist der wichtigste Punkt die Möglichkeit der Rotation, um auf anderen Flächen laufen zu können. Dazu muss es möglich sein diese anzupassen. Um an dieser Stelle Tests zu vereinfachen soll eine Komponente entstehen, die es erlaubt Rotationen zu berechnen, durchzuführen und zu automatisieren. Um das Automatisieren zu erleichtern soll es möglich sein Trigger zu haben, die bei Berührung bestimmte Rotationen auslösen. Natürlich soll es weiterhin möglich sein Rotation auch über andere Wege auszulösen.

3.2. Design und Implementierung der Untersuchungsplattform

In diesem Kapitel wird es um das Design und die Implementierung gehen. Dabei wird das System als Ganzes und seine einzelnen Komponenten vorgestellt.

3.2.1. Basisstruktur der Plattform

Ein Ziel der Plattform ist es eine hohe Modularität zu erreichen, um Änderungen besser vornehmen zu können. Dieses spricht für ein modulartiges System mit einzelnen Komponenten, die eine möglichst geringe Koppelung aufweisen. Es lässt sich allerdings auch so nicht vermeiden, dass ein gewisser Kern vorhanden bleibt, der sich nur mit großem Aufwand austauschen lässt,

da er eine höhere Komplexität besitzt. Doch zumindest die äußeren Komponenten sollten nach Möglichkeit selbständig sein. Weiterhin bleiben natürlich auch Abhängigkeiten von der genutzten Software, die sich nicht austauschen lässt, und es ist möglich, dass mit der Zeit Änderungen in dieser stattfinden. Die Modularität soll auch diesem entgegenwirken.

3.2.2. Unity

Bei der Software wurde die Entscheidung für Unity getroffen. Einer der Gründe ist die Laborumgebung, in der diese Arbeit zum größten Teil erstellt wurde. In dieser ist Unity die Standard Plattform für VR Entwicklung. Ein weiterer Vorteil ist die recht weite Verbreitung. Wegen der Nutzung von Unity als Software-Plattform muss das Modell entsprechend angepasst sein. Die Plattform sollte nach Möglichkeit auch den selben Bedienkonzepten wie Unity folgen.

Bei Unity handelt es sich um eine Game Engine, was die Erstellung von Szenarien erleichtert, da dieses zum Standardumfang von Unity gehört. Besonders wird dieses durch den Umstand erleichtert, dass Unity einen großen Assed Store hat, in dem bereits viele Modelle zur Verfügung stehen. Ein Teil der verfügbaren Asseds ist kostenlos, was die Erstellung von Szenarien weiter begünstigt. Als Programmiersprache wurde C# gewählt, da es die am weitesten verbreitete Sprache ist, die Unity unterstützt.

An dieser Stelle werden die für dieses Kapitel relevanten Eigenschaften von Unity aufgelistet und später ihr Einfluss aufgezeigt und wie dies für die Entwicklung wichtig ist.

Hierarchie von Unity

Eines der wichtigsten Tools in Unity ist die Szenen Hierarchie. In ihr sind sämtliche Objekte, die in einer Szene existieren, aufgelistet und durch Parenting lassen sich Positionen von Objekten relativ zu ihrem Elternteil setzen. Siehe Abbildung: 3.1 In dem Beispiel ist ein Game-Objekt Car zu sehen, das als Elternteil fungiert und vier Wheels als Kinder hat. Wenn nun das Car verschoben wird, bleiben die Positionen der Wheels zum Elternteil die selben und werden mit verschoben.

Auch ist die Hierarchie für Zugriffe auf Objekte wichtig, da ein Objekt immer nur auf Objekte zugreifen kann, die direkt über oder unter ihm liegen. Ein wichtiger Punkt ist dieses, da Skripte an den Objekten der Hierarchie hängen und somit in ihren direkten Zugriffen beschränkt sind. Dieses bedeutet, dass sich an der Hierarchie entlang gehangelt werden müsste um auf entferntere Komponenten zugreifen zu können. Darüber lässt sich zwar auf jedes Objekt zugreifen, sollte die Hierarchie aber geändert werden, kommt es schnell zu Problemen und Änderungen an vielen Stellen. Dieses macht es sinnvoll, eine Struktur für Zugriffe zu bauen,

Abbildung 3.1.: Beispiel: Unity Hierarchie

die es erlaubt global auf die Schnittstellen der Komponenten zuzugreifen. Auf diese Art kann das Entlang-Hangeln in der Hierarchie vermieden werden.

Local Space und World Space

Wenn die Sprache von der Hierarchie und dem Parenting ist, kommt an dieser Stelle das Konzept von Local Space und World Space zum Tragen. World Space ist immer die absolute Position eines Objektes in der Szene, während Local Space immer die lokale Positionsverschiebung vom Elternteil her ausdrückt. Dieses gilt sowohl für Position als auch Rotation und muss beachtet werden, wenn es um Berechnungen geht. Bewegungen und Rotationen sollten deshalb immer im richtigem Space verarbeitet werden. Die Fehler, die bei Verwendung des falschen Spaces entstehen können, haben die Eigenschaft, dass sie nicht immer sofort auftreten. Dieser Fehler würde z. B. nicht auftreten, wenn das Objekt noch kein Elternteil hat oder wenn Local und World Space noch das selbe sind. Dieses führt später zu einer aufwändigen Fehlersuche, die durch etwas geschickte Konsistenz vermieden werden kann.

Ein Beispiel hierfür wäre eine Standuhr mit ihren Zahnrädern, die auf einem Tisch steht. Angenommen die Uhr würde rotiert, müssten die Rotationsachsen der Zahnräder auch angepasst werden. Sollten die Rotationen der Zahnräder nun aber im World Space beschrieben sein, ändern diese sich nicht automatisch mit der Bewegung der Uhr. Die Zahnräder würden sich so weiter um ihre ursprünglichen Achsen drehen, was dazu führen würde, dass sie sich nicht mehr um ihre Vertikale sondern vielleicht eher um die Horizontale drehen würden. Wird die Uhr aber nur bei gleichbleibender Rotation verschoben, würde es zu keinen Problemen kommen, da die Rotationsachsen die selben bleiben. Dieses ist der Punkt, der es schwer machen kann, den Fehler zu finden.

Der Inspector

Einer der Hauptbestandteile von Unity ist der Inspector. In ihm werden alle Objekte, die in der Hierarchie enthalten sind, verwaltet und die Komponenten, die auf ihnen liegen. Wenn diese Komponenten Variablen besitzen, die entweder public oder als im Inspector sichtbar gekennzeichnet sind, lassen sich diese dann im Inspector manipulieren. Da dieses jederzeit getan werden kann ist dieses eine bequeme Art Parameter zu ändern, wenn es darum geht diese zu testen. Über den Inspector lassen sich Variablen mit Referenzen auf Objekte und Scripte der Szene belegen. Dieses erlaubt ein schnelles Anbinden ohne die Hierarchie mit Code zu durchsuchen. In größeren Konstellation sorgt dieses aber für Unübersichtlichkeit, da diese Abhängigkeiten schwer einsehbar sind. Letztlich lassen sich über den Inspector Skripte auch abschalten.

Das Bild 3.2 ist ein Beispiel für den Inspector. Die Komponente Transform gehört zu dem Objekt selber und ist für seine Position, Rotation und den Scale zuständig. Darunter sind weitere Eigenschaften des GameObjects mit dem Namen ”PlaneCheckerboard (2)” zu sehen. Unter dem Transform liegt das Script ”CheckerboardController”, das Einstellungsoptionen bietet. Die ersten beiden Optionen sind für die Größe des Checkerboards (Schachbrettmusters) zuständig, darunter ist die Breite der einzelnen Kacheln (Tiles) einzustellen. Danach ist noch die Möglichkeit, die Tiefe des Colliders einzustellen, was für Physik-Berechnungen interessant ist. Über die letzten drei Optionen lassen sich noch zwei Materialien für die Farbe oder Textur der Kacheln referenzieren und tauschen, wenn die Checkbox aktiviert ist. Abschließend ist noch ein Knopf (RegenerateFloor) geboten, der eine Neuberechnung des Checkerboards anhand der eingestellten Optionen auslöst.

So unterstützt der Inspector eine Vielzahl von weiteren Elementen wie: color picker, Array, Enum oder Physik-Ebenen. Doch ist es auch möglich das Aussehen und Verhalten des Inspector mit Hilfe von Editor Scripten, zu ändern, wie in diesem Beispiel, wo ein Button eingefügt wurde der eine Neuberechnung auslöst.

Berechnungszyklen

Ein wichtiger Punkt in Unity ist der Umgang mit Updates. Es gibt zwei Arten von Updates, Update() welches vor jedem Frame aufgerufen wird und FiexdUpdate(), das in Batches ausgeführt wird und dessen durchschnittliche Ausführung in der Standardeinstellungen alle 20 Millisekunden stattfindet. Deshalb eignet das FiexdUpdate() besonders für Physik-Berechnungen, da keine Zeitschwankungen in der internen Engine zu erwarten sind. Der Flowchart A.3 im Anhang auf Seite 75 gibt einen genaueren Einblick in die Execution-Order der Unity Abläufe.

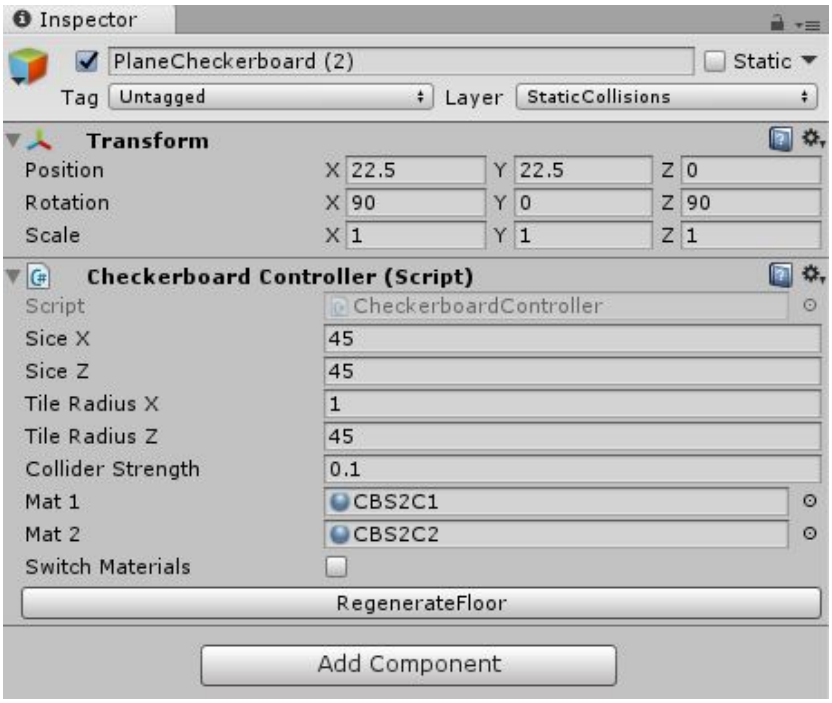


Abbildung 3.2.: In Beispiel für den Unity Inspector. Die gezeigte Komponente hat den Namen PlaneCheckerboard (2) und hat ein Script CheckerboardController

Eigenschaften von Komponenten

Wenn es darum geht eine ”Komponente” in Unity umzusetzen, gibt es mehrere Möglichkeiten wie dieses realisiert werden kann. Für die Plattform soll jede Komponente ein eigenes Script bekommen, diese werden dann über die Game-Objekte in der Hierarchie sortiert. Dieses erlaubt es, eine recht flache Hierarchie aufzubauen, da mehrere Skripte auf einem Objekt liegen dürfen. Die Übersichtlichkeit lässt sich so durch eine sinnvolle Gruppierung von Skripten fördern. Innerhalb der Game-Objekte ist es auch möglich diese Skripte in ihrer Reihenfolge zu sortieren, um eine bessere Übersicht zu erhalten. Da jedes Script in jedem Berechnungszyklus einzeln bedacht wird, entsteht auch kein Nachteil durch die Sammlungen. Eine flache Hierarchie lässt sich des Weiteren auch schneller überblicken, wenn alle verfügbaren Optionen der Skripte auf einmal angezeigt werden.

Um das System zu vereinfachen, werden die vorhandenen Komponenten an dieser Stelle grob in drei Klassen unterteilt.

Kernkomponenten sind jene Komponenten, deren Abwesenheit das ganze System zerstören würde. Sie sind grundlegend in ihrer Funktion, so dass, wenn sie ersetzt werden, ihre ganze

Funktionalität ausgetauscht werden muss. Diese Komponenten zeichnen sich durch große Schnittstellen und komplexes inneres Verhalten aus. In den meisten Fällen bauen diese direkt auf der Engine auf und abstrahieren ihre Funktion, um spezifische Prozesse zu vereinfachen.

Benötigte Komponenten: Unter diesem Begriff sind alle Komponenten zusammengefasst, die zwar unverzichtbar sind aber deren Funktionalität gering ist oder die nur wenige Anbindungen nach außen besitzen.

Freie Komponenten: Dieses sind Komponenten, die am äußeren Ende der Hierarchie sitzen. Ihr Wegbleiben hat auf die Ausführbarkeit des Systemes keinen Einfluss. Dieses sind zum Beispiel Komponenten, die später noch hinzugefügt werden, wenn einzelne Verhaltensweisen fehlen. Hierfür sind ausreichende Schnittstellen in den tieferen Komponenten nötig.

Abstraktion

Ein großes Problem und gleichzeitig ein Vorteil in Unity ist, dass jede Komponente auf jede Komponente zugreifen darf. Dies ist zwar beim Prototyping oftmals sehr hilfreich, da es das Erstellen von spezifischen Schnittstellen unnötig macht. Aber bei Projekten, die wachsen und größer werden, kann dieses schnell zu unsauberen Grenzen zwischen Komponenten führen, was Änderungen, Debugging und Tests deutlich erschwert. Aus diesem Grund ist es einerseits besonders wichtig, dass Zugriffsmöglichkeiten auf Komponenten dokumentiert und gut sichtbar sind. Andererseits, dass direkte Zugriffe auf die Transforms, die die Position beinhalten, vermieden werden.

Kommunikation der Komponenten

Um die Kommunikation der Komponenten einfach zu halten, sollten diese direkt über ihre Schnittstellen kommunizieren. Dieses erlaubt des Weiteren eine Komponente auszutauschen, da eine Neue lediglich die Vorgaben für die Schnittstelle erfüllen muss, um funktionsfähig zu sein. Die Weiterentwicklung wird somit einfacher, da der Funktionsumfang bekannt ist. Ein weiterer Vorteil dieses Vorgehens ist, dass die Kopplung zwischen den Komponenten so auf ein Minimum reduziert wird, ohne dass die Kopplungen unübersichtlich werden. Des Weiteren erlaubt dieses Konzept auch, dass Komponenten abgeschaltet werden können, wenn sie weiter außen sitzen. Dieses kommt der Untersuchung, der Qualität der Tests und der Plattform entgegen, da sich so einfacher Störfaktoren ausschalten lassen. Um die Zugriffsstrukturen zu erleichtern, ist eine Komponente hilfreich, die Referenzen auf die Kernkomponenten zur Verfügung stellt. Dieses hilft dabei die Zugriffe über die vorhandenen Schnittstellen zu führen, da diese so einfach zu erreichen sind. Diese Bündelung hat auch den Vorteil, dass Referenzen

bei einer Erneuerung von Komponenten zentral ausgetauscht werden können. Ebenso führen die gebündelten Zugriffe dazu, dass Seiteneffekte vermieden werden können, die bei direkten Zugriffen entstehen können, wenn diese an den Schnittstellen vorbei gehen.

3.2.3. Struktur

Da in Unity die Organisation von Komponenten standardmäßig über die Hierarchie von GameObjekten geregelt wird, (Unity, 3.2.2) ist es hilfreich, wenn die Plattform selber diesem Schema folgt. Auf diese Weise lässt sich eine Struktur implementieren, die einen hohen Freiheitsgrad behält. So wird die Plattform einfacher verständlich, da keine neuen Strukturen hinzukommen und sie mit dem Wissen für die Verwendung von Unity kompatibel ist.

Hierdurch ergeben sich nun aber unterschiedliche Arten von Gruppierungen. Das erhöht die Komplexität nicht, für das Verständnis ist es aber wichtig, dass klar ist über welche Gruppierung gesprochen wird. Hier muss zwischen der Gruppierungen über die Hierarchie von Unity und der Gruppierung über die Komponenten unterschieden werden, wobei die Einzelteile immer aus Skripten bestehen. In der Gruppierung über die Hierarchie ist es vor allem wichtig, dass die Sortierung für den User nützlich ist. Dieses bedeutet, dass die Scripte nach den Einstellungen, die möglich sind, gruppiert sind und wichtige Einstellungen nicht in der Hierarchie vergraben werden. Bei den Komponenten geht es um die Sicht, die für das System wichtig ist, um Verbindungen im System selber zu klären.

Durch die Eigenschaften von Unity entstehen auch für die Schnittstellen zwei unterschiedliche Arten. Die erste Art von Schnittstelle sind die Optionen, die im Inspector von Unity zu den Variablen angeboten werden. Die zweite Art ist die Klassische über Schnittstellen, die die Kommunikation zwischen den Skripten regelt. Die Unterschiede zwischen den Schnittstellen sind groß, da die erste für ausgewählte Variablen ist, auf die der User zugreift, während die zweite für die entsprechenden Funktionen ist und im Code stattfindet.

Komponenten-Modell

In diesem Modell sind die Abhängigkeiten der Komponenten von Funktionen anderer Komponenten dargestellt, siehe Abbildung 3.3, dabei kann es sich um Abhängigkeiten von Einstellungen, dynamischen Werten oder Funktionen handeln.

Movement, Gravity und Rotation sind alles **Benötigte Komponenten**. Sie werden nicht vom Kernsystem angesprochen, dennoch aber von Triggern, die ausgelöst werden können, um Parameter dieser Komponenten zu manipulieren. User, Anchor, Physik und Controller gehören

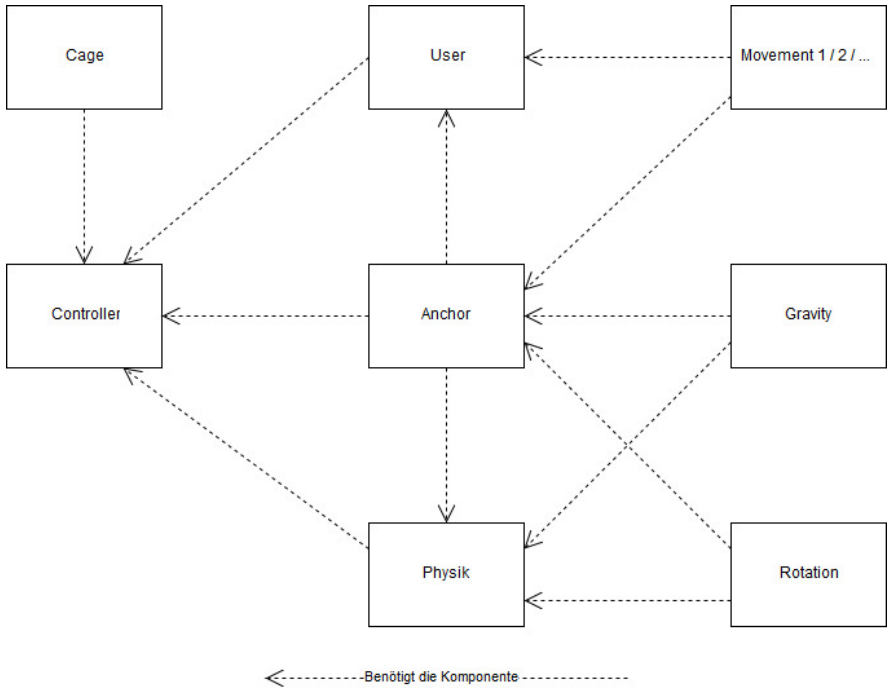


Abbildung 3.3.: Komponenten Modell der Plattform

zu den **Kernkomponenten** und stellen Abstraktionen sowie Funktionen des Grundsystems da. Der Cage ist eine **Freie Komponente**, zu der keine Abhängigkeiten bestehen.

Das Modell über die Hierarchie 3.4 ist anders aufgebaut, da sich manche Komponenten ein Game-Objekt teilen, um die Übersicht im Inspector zu erhöhen.

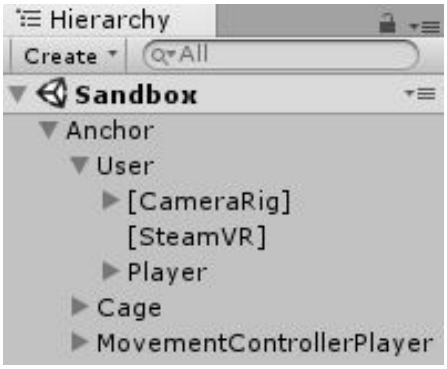


Abbildung 3.4.: Hierarchie Modell der Plattform

Auf dem Anchor sitzen die Komponenten Controller, Anchor, Movement, Gravity und Rotation, da diese Komponenten die Optionen zur Verfügung stellen, die am wichtigstem sind.

Kommunikations-Modell

Dem Diagramm 3.3 ist nicht entnehmbar, dass der Controller von allen Komponenten als Broker genutzt wird, um an die Referenzen zu den anderen Komponenten zu kommen. Dieses erhört zwar die Kopplung der Komponenten, hierdurch werden aber die Zugriffswege vereinfacht, was den Überblick über Abhängigkeiten erhöht und das Ändern von Komponenten vereinfacht. Der größte Vorteil ist aber, dass so die verfügbaren Schnittstellen nach außen gebündelt werden. Probleme lassen sich so vermeiden, die sonst beim Umbenennen und Austauschen von Komponenten entstehen können. Die Alternative, Referenzen über den Inspector zu setzen, würde ab einer gewissen Anzahl von Komponenten unübersichtlich. Des Weiteren müssen diese Referenzen immer von Hand neu gesetzt werden, wenn sie verloren gehen. Die andere Art wäre es, diese Referenzen in der Hierarchie zu suchen, was bei einer größeren Anzahl nicht wünschenswert ist, da dieses Verfahrenen einen großen Overhead besitzt.

Die Kommunikation ist größtenteils identisch mit den Abhängigkeiten der Komponenten. Die genaueren Kommunikationswege sind in den einzelnen Komponenten beschrieben.

3.2.4. Contoller

Dieses Script ist das Herzstück der Kommunikation, denn er verwaltet alle Referenzen auf Scripte, die interne und externe Schnittstellen haben. Des Weiteren hält er auch folgende Optionen zur Verfügung:

- Zwischen VR und Desktop zu wechseln
- Die Größe des verfügbaren Gebietes einzustellen
- Den Offset der realen Fläche zu regulieren
- Den Scale des Users einzustellen
- Die Option Debug-Informationen generell auszuschalten
- Die Lebensdauer von grafisch angezeigten Debug Informationen zu regeln

Sollte der Controller eine der benötigten Komponenten nicht finden, macht er in der Konsole darauf aufmerksam. Sollte die Komponente aber nicht benötigt werden funktioniert das System weiterhin.

Auch stellt der Controller die zum Debugging nötigen Funktionen bereit. Dieses vereinfacht die Umrechnung und erleichtert den Einsatz. Das Debugging besteht aus zwei Komponenten, die Erste zur Ausgabe in der Konsole und die Zweite zur grafischen Darstellung in Unity. Der Vorteil dieser Bündelung ist, dass sich das Debugging global abschalten lässt. Dieses ist notwendig, da das Debugging die Framerate reduziert. Die meisten Komponenten besitzen des Weiteren ihre eigene Option im Inspector, das Debugging lokal ein- und auszuschalten, um die angezeigten Informationen zu kontrollieren.

3.2.5. Anchor

Der Anchor steht an der obersten Stelle der Hierarchie der Plattform, für die Komponenten unter ihm ist er die Mitte des realen Tracking Bereiches. Durch diese Abstraktion sind alle Berechnungen von Kind-Objekten einfacher verständlich, da diese so im Local Space des Anchors arbeiten können. Berechnungen lassen sich so einfacher nachvollziehen, da die Richtungen mit der Ausrichtung des Anchors mit wandern. Ein weiterer Vorteil, der durch die Verschiebung der Berechnungen in den Local Space entsteht, ist, dass wenn der Anchor selber ein Kind-Objekt wird, die Berechnungen sich nicht selbst zerlegen. In dem Beispiel mit der Uhr 3.2.2 wurden die Folgen mangelnden Abstraktion bereits erklärt. Dementsprechend wird nach Möglichkeit grundsätzlich versucht im Local Space zu bleiben, um Fehler zu vermeiden und den Code sauber zu halten.

Die Komponente des Anchor ist zweigeteilt, in den **Anchor Controller**, der Funktionen zur Verfügung stellt und die Komponente **Anchor Movement**, die sich aktiv um das Sammeln und Verarbeiten von Bewegung kümmert. Die Trennung in zwei Komponenten ist aufgrund der unterschiedlichen Funktionen hilfreich, um eine passive und eine aktive Komponente zu haben.

Anchor Controller

Dieses Skript dient als Schnittstelle für die Position und Rotation des Anchors, auch kann über diese Komponente die Ausrichtung als Vector im World Space abgegriffen werden. Geboten wird auch eine zentralisierte Möglichkeit für die Umrechnung von Local Space und World Space. Dieses ist für alle möglichen Umrechnungen hinterlegt (Vector, Punkt und Richtung). Die dritte Eigenschaft, die geboten wird, ist der Zugriff auf die Physik, wobei hier zwei Arten angeboten werden. Beide nehmen einen dreidimensionalen Vector entgegen und prüfen die gewünschte Bewegung auf Durchführbarkeit mithilfe der Physik Komponente. Danach besteht die Möglichkeit, die Bewegung in ihren Dimensionen einzuschränken. Diese Werte lassen sich

im Inspector oder über die Schnittstelle setzen. Im letzten Schritt ist der Unterschied, dass die eine Methode lediglich die mögliche Bewegung zurück gibt, während die andere diese direkt ausführt. Die Zentralisierung für diese Zugriffe erhört die Übersicht und reduziert die Gefahr für Seiteneffekte, die bei Änderungen in der Hierarchie oder in Komponenten auftreten könnten. Diese Probleme haben eine besonders hohe Chance aufzutreten, sollten Zugriffe von außerhalb der eigentlichen Plattform passieren.

Anchor Movement

Diese Komponente verwaltet die Bewegung. Sie trennt direkte und indirekte Bewegung und erlaubt es, die Verarbeitung von Bewegung zu unterbrechen. Automatisch passiert dieses sobald der Abstand zur Mitte zu groß wird. Andersherum wird die Bewegung wieder freigeschaltet, wenn der User sich wieder weit genug in die Mitte begibt. Das Verhindern der Bewegungsberechnung lässt sich des Weiteren auch durch einen Overwrite blockieren, das genauso wie das normale Blockieren als Schnittstelle zur Verfügung steht. Die Distanzen für Aktivierung und Deaktivierung lassen sich im Inspector einstellen. Diese Komponente bündelt auch die Quellen für indirekte Bewegung, um nicht für jede Komponente die Physik einzeln berechnen zu müssen. Die Abstände für Deaktivierung und Aktivierung von Bewegung lassen sich über den Inspector einstellen.

3.2.6. User Komponente

Im User Komponente geht es um die Abstraktion von dem Head-Mounted Display (HMD) und die Integration des First Person Controllers. In dieser Komponente finden weitere Abstraktionen statt, eine davon ist es, dass wenn ein Offset angegeben ist, dieser entsprechend wieder herausgerechnet wird, damit er vor den anderen Komponenten versteckt wird. Des Weiteren werden hier auch Globale und Lokale Position des HMDs angeboten. Dieses ist sowohl die Kopfposition, die der Standard für HMDs ist, als auch die Fußposition, die ohne die Höhe des Users ist. Genauso ist es auch möglich, die aktuelle Höhe des HMDs abzufragen.

VR Adapter

Die Wahl zwischen VR und Desktop und Einbindung anderer VR Devices wird in der User Komponente verwaltet, da hier das HMD abstrahiert wird um nach außen eine einheitliche Schnittstelle anzubieten, selbst wenn das HMD oder seine Schnittstelle durch ein anderes Device ersetzt werden sollte. Um die Einbindung an dieser Stelle zu vereinfachen, muss das entsprechende HMD lediglich in der Hierarchie als Kind-Objekt unter das User Objekt gesetzt

Abbildung 3.5.: Inspector des User Controllers

werden und es müssen über den Inspector die Referenzen auf die entsprechenden Teile des HMD angepasst werden. In der Abbildung 3.5 sind die entsprechenden Referenzen zu sehen. In das Feld HMD Rig muss das Elternteil des HMD Rigs gesetzt werden. In HMD Display gehört die Komponente, die die aktuelle Position vom Ursprung aus bereitstellt. Meistens ist dieses auch die Komponente, die auch die Kameras enthält. Das vorletzte Feld ist nicht immer notwendig, bietet aber Platz für eine weitere Komponente, die zum Rig gehört. In diesem Fall das Objekt SteamVR, dieses erlaubt es auch eine weitere Komponente automatisch auszuschalten, sollte sie nicht benötigt werden. Als Letztes kommt die Referenz auf einen First Person Controller, der auf die selbe Weise wie das VR Rig ausgetauscht werden kann. Abgesehen davon, dass er sein eigenes Feld hat. Auf diese Weise ist das Austauschen ein trivialer Prozess, der auch ohne Änderungen im Code auskommt.

First Person Controller

Die weitere Funktion der User Komponente ist, dass der First Person Controller an dieser Stelle zur Verfügung gestellt wird. Dieses erlaubt die Steuerung über Tastatur und Maus, somit ist es für einen ersten Test nicht nötig, ein HMD betriebsbereit zu haben und die Grundfunktionen lassen sich auch auf einem anderen Rechner nutzen, was während des Entwickelns einen Vorteil schafft. Der Controller selber bietet Einstellungen für seine Parameter im Inspector 3.6, wobei diese im Normalfall nicht geändert werden müssen.

3.2.7. Bewegung

Um die Bewegung zu realisieren, sind mehrere Dinge von Nöten. Es wird eine Komponente benötigt, die sich um die physikalischen Kalkulationen kümmert, diese heißt ”Movement Controller Player” in der Hierarchie. Eine, die die gewünschten Bewegungen verwaltet und bündelt. Hierfür ist die zweiteilige Komponente des Anchors aus Movement und Controller zuständig. Für die Bewegung sind dann noch zwei weitere Komponenten nötig: Eine, die direkte

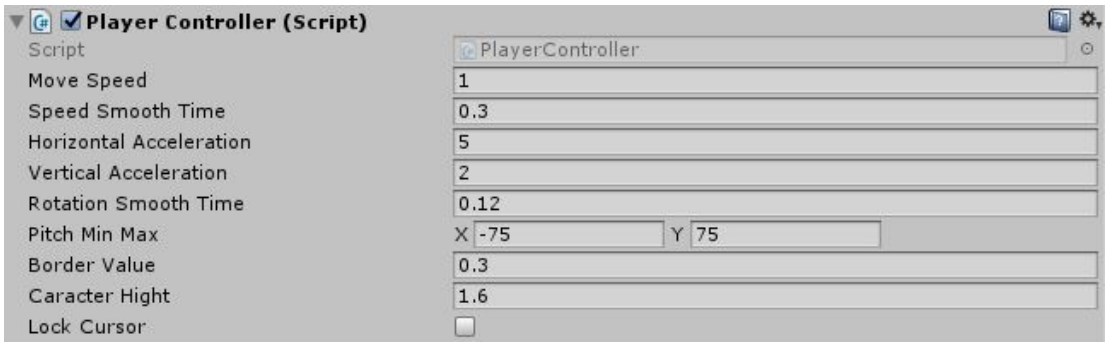


Abbildung 3.6.: Inspector des Player Controllers

Bewegungen des User verarbeitet und die andere, die für indirekte Bewegungen da ist, in diesem Fall für das Annäherungsprinzip. Für jede weitere Art von gewünschter Bewegung soll es möglich sein, eine eigene Komponente hinzuzufügen. Diese kann einfach ihr Movement an die Komponente Anchor Movement weitergeben. Dadurch ist es möglich beliebige Komponenten, die Bewegung erlauben, einzubauen, da über die Bündelung im Anchor Movement vermieden wird, dass durch mehr Movement Komponenten mehr Rechenleistung benötigt wird.

Problem

In Unity gibt es zwei Konzepte wie mit Kollisionen umgegangen werden kann. Auf der einen Seite gibt es den ”Character Controller” (CC) und auf der anderen den Rigidbody. Beide sind auf ihre eigene Art problematisch. Der CC ist von den angebotenen Funktionen an sich optimal. Er besitzt eine Move(Vektor3) Funktion, der ein dreistelliger Vektor übergeben werden kann. Der CC versucht soviel wie möglich von der Bewegung auszuführen. Dabei werden Unebenheiten wie Schrägen und Stufen automatisch berücksichtigt. Zurückgegeben wird die Bewegung, die ausgeführt wurde, was genau das ist, was an dieser Stelle gebraucht wird. Da sich seine Rotation aber nicht uneingeschränkt ändern lässt, ist der CC in diesem Fall nicht verwendbar. Der CC ist immer senkrecht ausgerichtet, was bedeutet, dass er im Zusammenhang mit Rotationen um die waagrechte Achse nicht nutzbar ist.

Die andere Möglichkeit ist der Rigidbody. Dieser hat das Problem mit der Rotation nicht, dafür aber fehlt ihm die Move(Vecktor3) Komponente, da er dafür ausgelegt ist über Kräfte oder direkt manipuliert zu werden. Ein weiterer Grund warum er nicht optimal ist, ist die Art wie Kollisionen mit dem Rigidbody verarbeitetet werden. Kollisionen für den Rigidbody werden von Unity nur in der Physik Phase (Internal pysics update) berechnet. Die Abbildung der Execution-Order von Unity A.3 ist im Anhang auf Seite 75. Dieses würde die Trennung von direkter

und indirekter Bewegung erschweren. In dieser Berechnungsphase sind nämlich manuelle Eingriffe nicht möglich und so würden beide Bewegungen in einem Schritt abgearbeitet und die Verarbeitung der Ergebnisse erschwert.

Das größte Hindernis aber ist, dass der Rigidbody direkt kontrolliert werden müsste, um exakte Bewegungen zu ermöglichen. Direkte Kontrolle bedeutet aber, dass beim Verschieben Kollisionen nicht erkannt werden und Überscheidungen von Objekten entstehen. Damit wird das auflösen der Kollisionen ein Problem, denn das ”Physik Update” versucht den Rigidbody auf kürzestem Weg aus der Kollision herauszuschieben. Dieses wird über die Normale des Objektes mit dem die Kollision besteht erledigt. Dieses führt dazu, dass wenn der Rigidbody auf einer Schrägen nach oben gegangen ist, die in der Realität zurück gelegte Strecke größer ist, als die in der Virtuellen Welt zurückgelegte Strecke. Siehe Abbildung 3.7. Dieses ist allerdings ein Problem das nicht nur auf Schrägen, sondern auf allem Unebenen Untergründen passiert. Die hierdurch entstehende Diskrepanz ist störend und führt zu disjunkter Bewegung, was zu einem Grund für Cybersickness werden kann, und sich beim Bewegen unangenehm anfühlt.

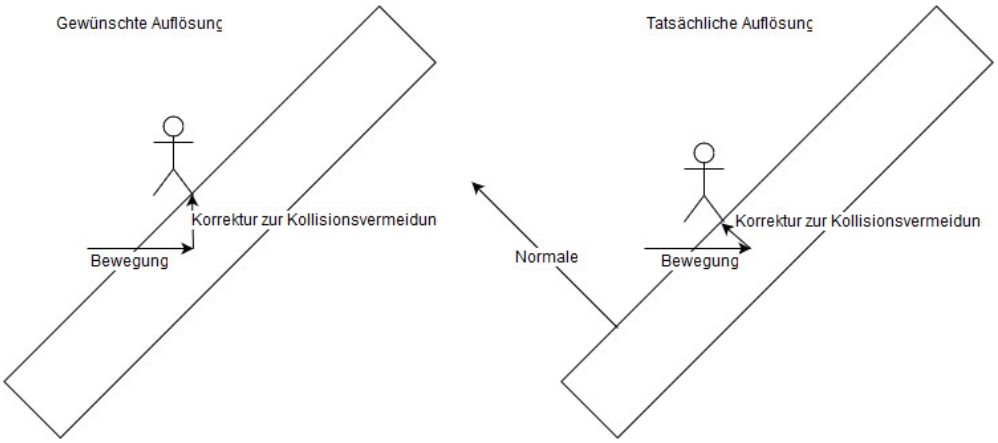


Abbildung 3.7.: Auf der Linken Seite ist die Gewünschte Kollisionsvermeidung zu sehen, der User wird um die entsprechende Höhe angehoben. Auf der Linken seite ist die Reaktion des Rigidbody zu sehen der anhand der normalen aus der Kollision herausgeschoben wird.

Es gibt zwar Systeme die selber ein Äquivalent für den CC sind. Diese haben allerdings entweder das gleiche Problem wie der Rigidbody oder sind zu komplex, unfertig, zu groß oder eine Mischung davon, weshalb sie für diesen Zweck ungeeignet sind.

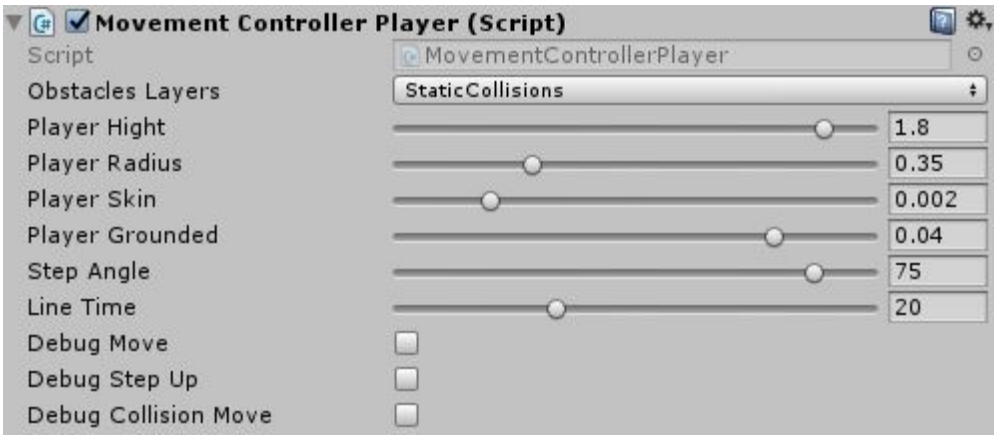


Abbildung 3.8.: Inspector der Komponente Movement Controller Player

Lösung

Da im Moment kein System existiert, dass die geforderten Anforderungen erfüllt, wird an dieser Stelle eine eigene Lösung entwickelt. Ziel ist es, diese Komponente in der Zukunft zu ersetzen, um Wartung zu vermeiden, die bei Unity Updates entstehen kann. Gut wäre eine Komponente, die direkt aus Unity kommt, da diese wahrscheinlich effizienter und weniger fehleranfällig für Updates von Unity ist.

An dieser Stelle muss nun eine Komponente entwickelt werden, die mit Kollisionen umgehen kann. Dieses System soll so einfach wie möglich sein. Kollisionen können grob in zwei Klassen ”Unüberwindbare Hindernisse” und ”Überwindbare Hindernisse” geteilt werden. Die Entscheidung kommt dadurch zu Stande, dass mit den beiden Klassen unterschiedlich umgegangen werden muss. Allerdings müssen beide Klassen auch im selben Rechenzyklus abgearbeitet werden können.

In der Abbildung 3.8 sind die Einstellungen der Komponente zu sehen, die über den Inspector vorgenommen werden können. Die erste Option erlaubt es festzulegen mit welchen Physik-Ebenen kollidiert wird. Die nächsten Optionen bestimmen die Größe des Kolliders (Player Hight und Player Radius), drauf folgt der Abstand, der von Hindernissen gehalten werden soll (Player Skin), um etwas Spielraum für weitere Überprüfungen zu haben. Player Grounded gibt den Abstand zum Boden vor. Der Step Angle ist die Beschränkung für unebenen Boden und die weiteren Optionen sind fürs Debuggen.

Kollisionen mit unüberwindbaren Hindernissen Wenn es um die Kollisionen mit Hindernissen geht, dann ist als Erstes die Unterscheidung zwischen überwindbar und unüberwind-

bar wichtig. Man unterscheidet zum Beispiel Türschwellen von Wänden dadurch, dass die Höhe, bis zu welcher der Charakter mit dem Hindernis kollidiert, berechnet. wird Im Fall von einer Kollision ist es wichtig, dass trotz Hindernis so viel Bewegung wie möglich ausgeführt wird. Es ist nicht wünschenswert, dass nur weil mit der Wand kollidiert wird die gesamte Bewegung zum Stillstand kommt. Dieses wäre ein einfacher Weg in die Cybersickness und würde in engen Gebieten nur wenig Spaß machen. Die Umsetzung ist in diesem Fall, wenn mit einem solchem Hindernis eine Kollision entsteht, dass die Normale des entsprechenden Hindernisses ermittelt und die Bewegung in den orthogonalen Teil und den Rest aufgeteilt wird. Die auszuführende Bewegung wird dann in zwei Schritten ausgeführt, als erstes der orthogonale Teil und danach der Rest. Wenn Kollisionen entstehen, werden die Vektoren jeweils gekürzt. Die übrigbleibende Bewegung kollidiert nicht mit dem Hindernis und führt möglichst viel davon aus. Beispiel 3.9. Da diese Berechnungen im FixedUpdate abgehandelt werden, ist es ausreichend den Schritt einmal pro Berechnungszyklus auszuführen, denn die Bewegungs-Vektoren sind so klein, dass der abgeschnittene Teil für die Bewegung vernachlässigbar ist.

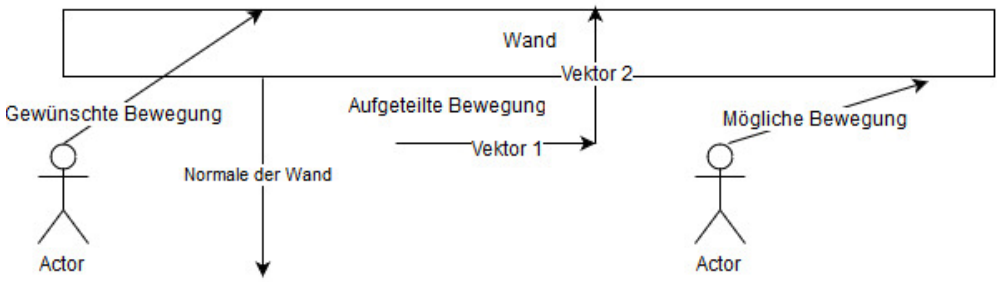


Abbildung 3.9.: In dieser Grafik wird die Berechnung bei Kollisionen mit unüberwindbaren Hindernissen dargestellt. Bei einer Kollision wird die gewünschte Bewegung in zwei Teile zerlegt. Dieses ermöglicht es, die größte gemeinsame Summe für die Bewegung zu bilden, ohne dass eine Durchdringung des Hindernisses entsteht.

Kollisionen mit Überwindbaren Hindernissen Die zweite Klasse von Hindernissen sind jene, die überwunden werden können, hierzu zählen Stufen, Rampen und unebener Boden. Besonders beim zuletzt genannten wäre verwunderlich, wenn es ein Hindernis darstellen würde. Wenn ein Hindernis überwunden werden soll, muss der User, um das zu überwindende Level, angehoben werden. Hierbei muss allerdings auf weitere Kollisionen, wie zum Beispiel eine niedrige Deckenhöhe, geprüft werden. Die maximale Höhe, die in einem Rechenschritt bewältigt werden kann, ist Abhängig vom Grad der Unebenheit und der im Rechenschritt zurückgelegten Entfernung. Wenn das Hindernis nicht zu steil ist wird die benötigte Anhebung

berechnet. Sollte das Hindernis aber zu steil oder hoch sein, wird es zu einem unüberwindbaren Hindernis, und entsprechend behandelt. Aufgrund dessen, dass in Unity Kanten von Kollidern typischerweise als rund betrachtet werden, funktioniert dieses auch mit Stufen und Schwellen.

Movement

An dieser Stelle sollen die Komponenten, die für die unterschiedlichen Arten von Bewegung verantwortlich sind, vorgestellt werden. Die Komponenten unterscheiden sich in der Tiefe ihrer Integration in der Plattform und ihren Aufgaben.

Movement By Player In dieser Komponente werden die Bewegungen des Users verarbeitet. Das bedeutet, dass die Bewegung auf Plausibilität geprüft wird. Hierbei werden auch Höhenunterschiede mit berücksichtigt die angepasst werden müssen, wenn die Position geändert wird. Diese Komponente ist auch dafür verantwortlich, dass der User nicht in Objekte hineinlaufen kann. Dazu werden Ausgleichs-Bewegungen berechnet die den Anchor verschieben, damit der User nicht in Objekte eindringen kann. Dieses vermittelt den Eindruck, dass die Welt dann vor dem User flieht, Beispiel 3.10. Das Spiel kann solange getrieben werden, bis das virtuelle Hindernis mit einer realen Grenze übereinstimmt, da in dem Fall die reale Wand den User aufhält.

Movement By Proximity Die Komponente Movement By Proximity ist für die indirekte Bewegung verantwortlich. Sie verarbeitet den Abstand des Users von der Mitte der realen Fläche aus und ermittelt so den Abstand zum Rand des begehbaren Bereiches, der im Controller eingestellt ist. Wird der eingestellte Abstand zum Rand unterschritten, wird die Bewegung in die entsprechende Richtung gestartet ohne das der User sich weiterhin in die Richtung bewegen muss. Sollte der eingestellte Abstand zum Rand wieder überschritten werden, wird diese Form der Bewegung eingestellt. Die von dieser Komponente berechneten Bewegungen werden alle logarithmisch geglättet. Dieses verhindert den abrupten Einstig in die Bewegung, ist aber schnell genug um keine Verzögerung zu bewirken und somit hilfreich um Cybersickness zu vermeiden. Um die Bewegung einzustellen gibt es eine Vielzahl von Optionen die verfügbar sind. Siehe Abbildung 3.11.

Die oberste Einstellung schaltet die Komponente ein und aus (Calculate). Danach folgt der Abstand zum Rand der unterschritten werden muss um die Bewegung zu starten (Point Of Moving). Drauf folgt die Basis-Geschwindigkeit (Start Speed) und der Geschwindigkeitsfaktor (Speed With Greater Distance), der für einen größeren Abstand zur Mitte hinzukommt. Gefolgt von der Maximalen Geschwindigkeit (Max Speed), die sich aus den beiden Werten ergeben

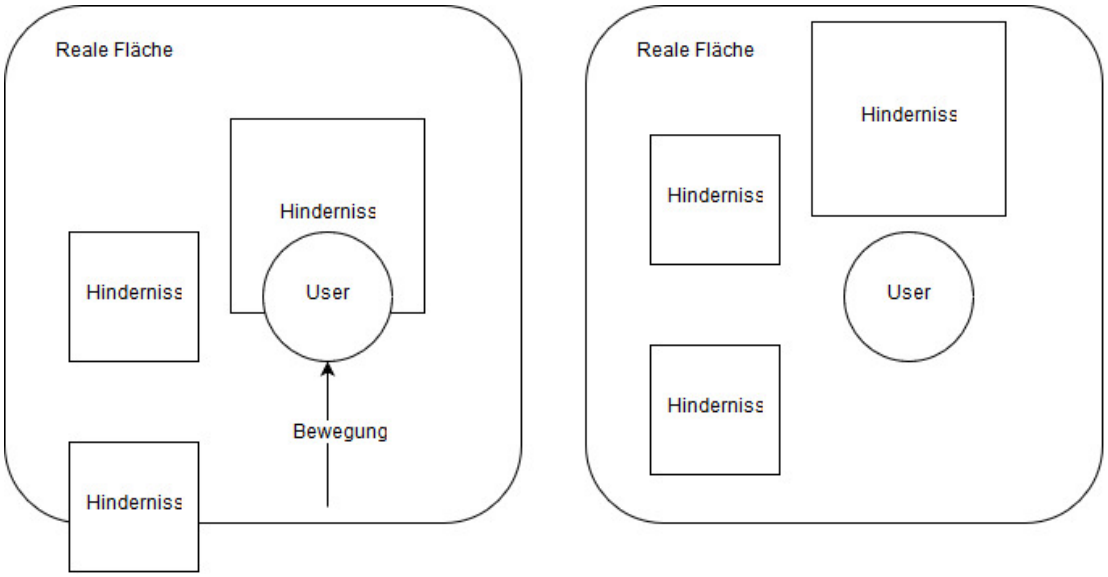


Abbildung 3.10.: In dieser Abbildung ist die Reaktion der Plattform auf eine versuchte Durchdringung von einem Hindernis zu sehen. Wird eine Durchdringung versucht, verschiebt sich die Position des Anchors und die Welt scheint vom User zu fliehen.

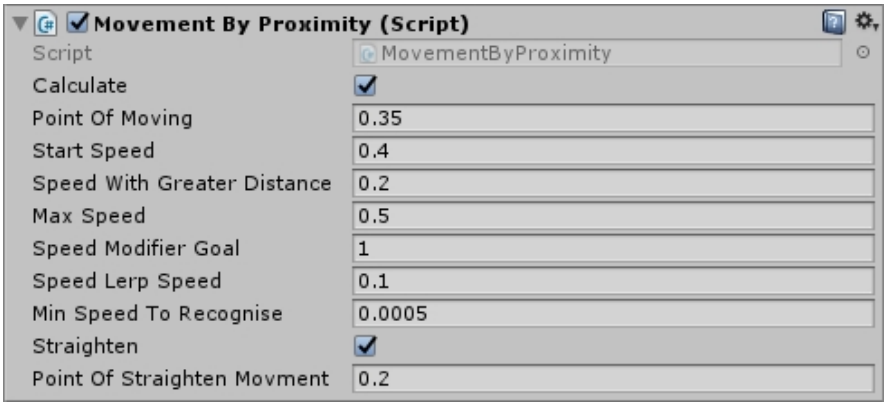


Abbildung 3.11.: Inspector von Movement By Proximity

kann. Der darauf folgende Wert (Speed Modifier Goal) ist auch von außen manipulierbar und stellt einen Faktor für das Ergebnis der Komponente da. Eine Anwendung hierfür ist wenn der Scale geändert wird, um die Geschwindigkeit proportional mit zu ändern. Damit keine Bewegungen weitergegeben werden die zu klein sind um Auswirkung zu haben, ist die nächste Option (Min Speed To Recognise) die es erlaubt alle Bewegungs-Vektoren unter einer gewissen

Länge herauszufiltern. Die Letzten beiden Optionen sollen helfen sich gerade durch den Raum zu bewegen solange die Ausrichtungen stimmen.

Cage

Dynamische Anpassung der Größe, des Scale und des Offset. Das sind die drei Punkte, die die Implementation des Cage ausmachen. Der Cage selber besteht aus Blöcken die dynamisch in die richtige Größe und Form gebracht werden, um entsprechende Werte widerzuspiegeln. Es ist eine **freie Komponenten** von der keine weiteren Komponenten abhängen. Somit ist es auch einfach diesen auszutauschen oder abzuschalten wenn eine andere Art von visueller Begrenzung gewünscht ist. In der Abbildung 3.12 ist der Käfig zu sehen.

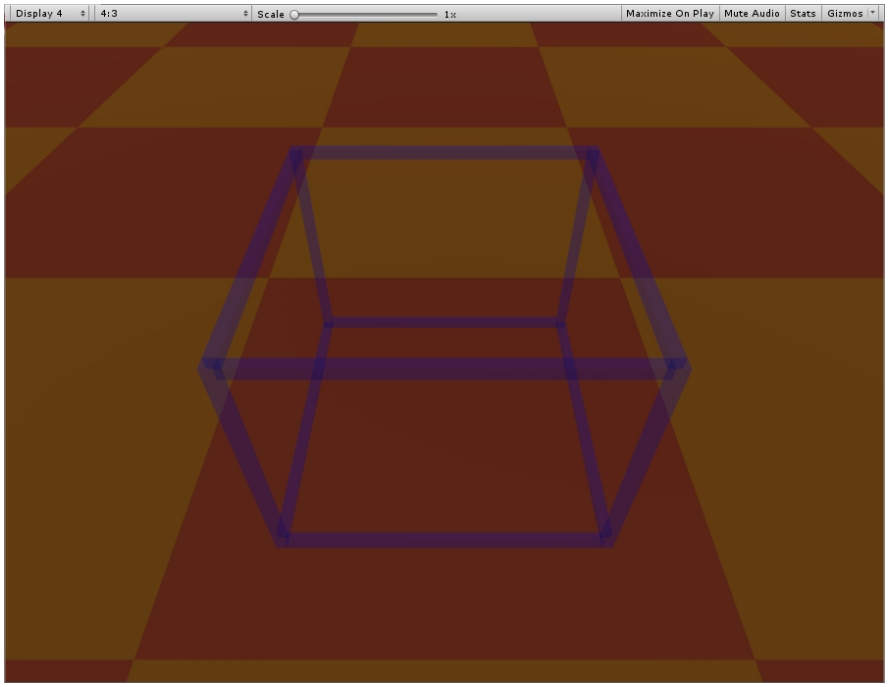


Abbildung 3.12.: Inspector von Movement By Proximity

Movement by Gravity

Gravitation ist auch eine Bewegung, sie wird genauso abgearbeitet wie andere Bewegungen, mit dem Unterschied, dass sie vertikale Bewegung verursacht. Einstellungen die sie zur Verfügung stellt, sind die Optionen sie auszuschalten, Debugging einzuschalten und die Geschwindigkeit des Fallens zu kontrollieren. Die Geschwindigkeit ist in diesem Fall eine Konstante, um Cyber-

sickness vorzubeugen. Sollte ein Boden vorhanden sein, wird für einen konstanten Abstand zum Boden gesorgt.

Movement by Rotation

Die Rotation ist eine wichtige Komponente und sollte immer die kürzeste Rotation berechnen, wenn der User sich um seine waagerechte Achse dreht. Die neue Ausrichtung wird durch einen Vektor bestimmt, der in die Richtung des neuen Oben zeigt. Für dieses wird dann die Rotation berechnet, die den Vektor des Anchors, auf kürzestem Weg, in die neue Position dreht. In dieser Komponente ist es möglich die Geschwindigkeit der Rotation einzustellen, die Drehgeschwindigkeit ist konstant und unabhängig von der Größe der Rotation. Diese lässt sich auch über den Inspector einrichten.

Es gibt zwei Arten von Anpassungen. Eine die zu einem bestimmten Vektor hin rotiert und durch einen Trigger ausgelöst wird, der auch die Rotation vorgibt. Die Andere, die den nach oben zeigenden Vektor anhand der Normalen des Untergrundes bestimmt und die Ausrichtung konstant anpasst, solange eine Kollision mit dem entsprechendem Trigger stattfindet. Die zweite Art von Anpassung hat Vorrang vor der Ersten, wenn Beide aktiv sind.

Der Vorgang ist in drei Teile gespalten. Am Anfang steht die Kollision mit einem entsprechendem Trigger, dessen Oben wird dann an die Komponente für Rotation weitergegeben. Diese prüft als erstes, ob eine Rotation überhaupt notwendig ist. Wenn eine neue Rotation berechnet werden muss, wird die benötigte Rotation aus dem neuen und dem alten Oben berechnet und als aktuelle Rotation abgelegt, dann wird noch die Update Methode aktiviert die diese Rotation durchführt. Die Funktion, die mit dem FixedUpdate aufgerufen wird, führt jetzt in jedem Schritt ein Teil der Rotation durch bis die ganze Rotation ausgeführt ist.

Ein Problem das mit diesen Rotationen verwirrend wirken kann, ist wenn die Rotation und Bewegung in gewissen Konstellationen angewendet werden, die die ursprüngliche Ausrichtung verdrehen. Dieses ist z. B. der Fall bei der kontinuierlichen Anpassung, was für den User zur Verwirrung führen kann, da die Welt eine andere Ausrichtung als zuvor hat, wenn der User an den Ausgangspunkt zurückkehrt. In den Abbildungen 3.13 und 3.14 ist zu sehen wie die ursprüngliche Ausrichtung durch Rotation über runde Oberflächen verloren geht.

3.2.8. Erstellung eines Test Szenarios

In diesem kurzen Kapitel werden die nötigen Schritte vorgestellt, um die Plattform in einem Szenario einsatzfähig zu machen. Dieses geschieht am Beispiel eines Test-Szenarios, das freie Bewegung und Rotation erlaubt.

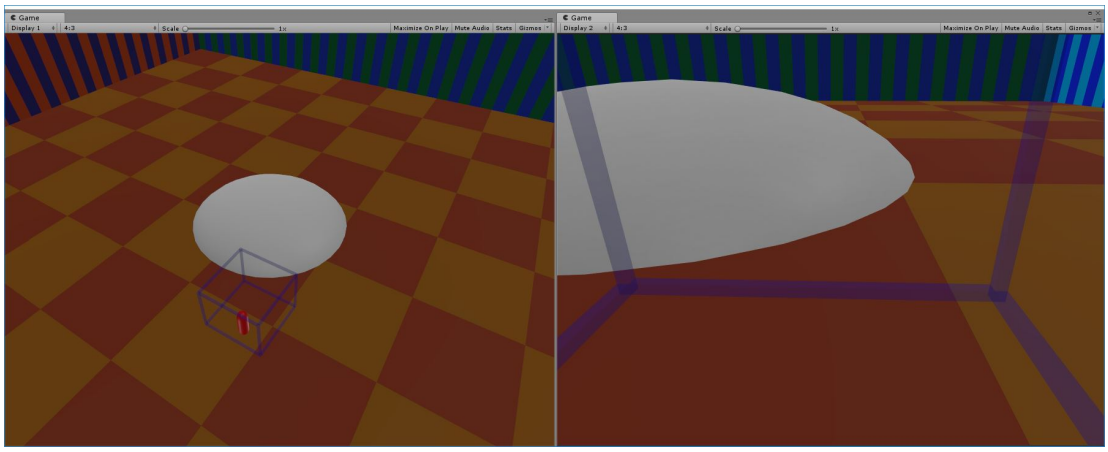


Abbildung 3.13.: Vor der Rotation über runden Oberflächen. Hier ist der Käfig noch am Schachbrettmuster ausgerichtet.

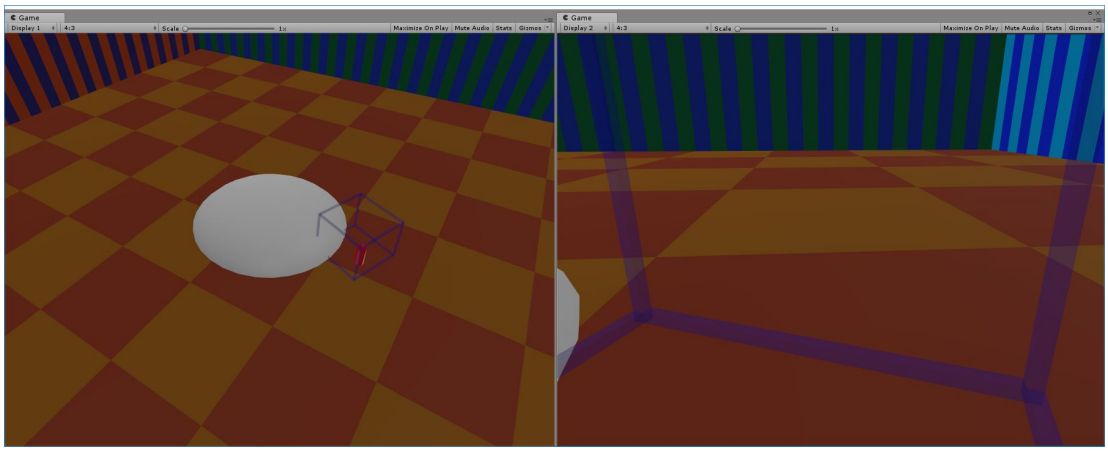


Abbildung 3.14.: Nach der Rotation über runde Oberflächen mit Kreisbewegung, ist es möglich, dass beim zurückdrehen der Rotation die ursprüngliche Ausrichtung verloren geht. Der Käfig ist nun nicht mehr an der Rotation des Schachbrettmusters orientiert.

Anforderungen an das Szenario

Als Erstes muss dazu das Szenario spezifiziert werden. Dazu zählen auch jene Phänomene, die abgebildet werden sollen, z. B. die Umgebung in der VR und welche Landmarken in der Umgebung vorhanden sind. Dazu werden die Anforderungen an dieses Szenario aufgelistet.

1. Freie Bewegung
   Freie Bewegung sollte im Szenario vorhanden sein, um den Raumeindruck zu verstärken und die Handlungsmöglichkeiten zu erweitern.

2. Die Änderung der Rotation ist kontrolliert möglich
   Damit Rotationen des Users kontrolliert möglich sind, muss es Punkte im Raum geben, an denen diese ausgelöst werden. In diesem Szenario sollen dazu Rampen dienen, die sowohl die Handlungsoptionen für Rotation verdeutlichen, als auch die Rotationen in Bahnen lenken. Des Weiteren sollen die Rampen die Transition erleichtern, sodass sich die Rotationen weniger auf das Wohlbefinden auswirken.

3. Landmarken Im Szenario
   Als Landmarken für dieses Szenario soll gemütlicher Wohnraum mit Bett, Schrank, Arbeitsplatz, fünf Lampen und Fernsehecke verwendet werden.

4. Abgeschlossener Handlungsstrang
   Der User soll einen vorgegebenen und abgeschlossenen Weg durch den Raum gehen, dazu sollen die Rampen so angerichtet sein, dass sie einen ununterbrochenen Weg durch den Raum, über Wände und Decke, erlauben.

Gestaltung des Szenario

Um das Szenario zu gestalten, werden Einrichtungs-Gegenstände benötigt. Diese stammen alle aus dem asset store von Unity. In der Grafik 3.15 ist die Auswahl der Gegenstände zu sehen. Das Environment ist zum großem Teil von Uli Meyer eingerichtet und zusammengestellt.

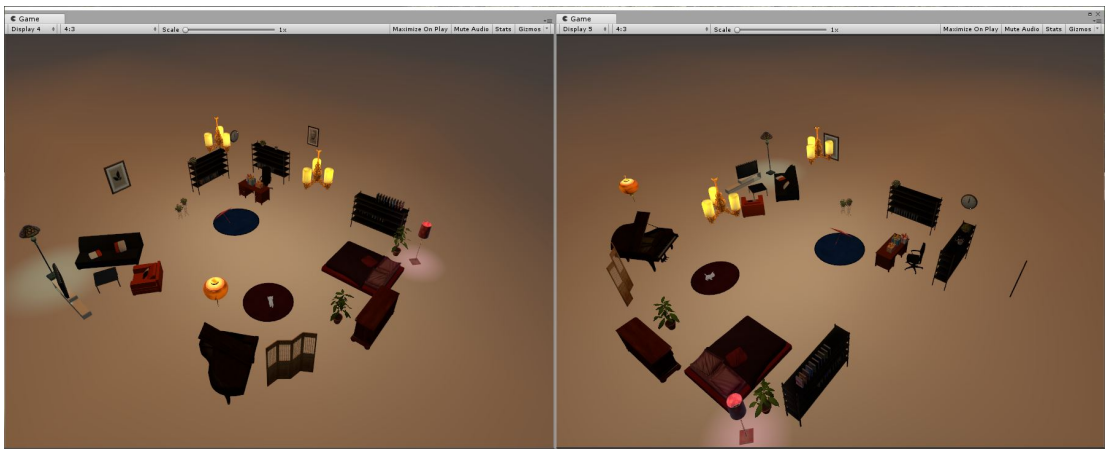


Abbildung 3.15.: Die ausgesuchten Möbelstücke für das Szenario

Danach musste die Größe des Raumes festgelegt werden. Dieser sollte nicht sich zu groß sein, damit die Laufwege nicht zu lang werden. Die Breite des virtuellen Raumes beträgt 17 Meter, die Tiefe 14 Meter und die Höhe 6 Meter. Mit dieses Abmessungen ist der Raum auch hoch genug, um zu vermeiden, dass beim Laufen an der Decke die Gegenstände am Boden im Weg sind. Damit die Begrenzungen des Szenarios auch zum Raum passen, benötigen diese entsprechende Texturen. Der Boden im Raum 3.16 erhält eine Holzdielen-Textur und die Wände eine senkrecht gesteifte Tapete mit hölzernen Fußleisten. Die Richtung der Streifen auf der Tapete sollten im Einklang mit der Gravitation stehen, also senkrecht verlaufen.

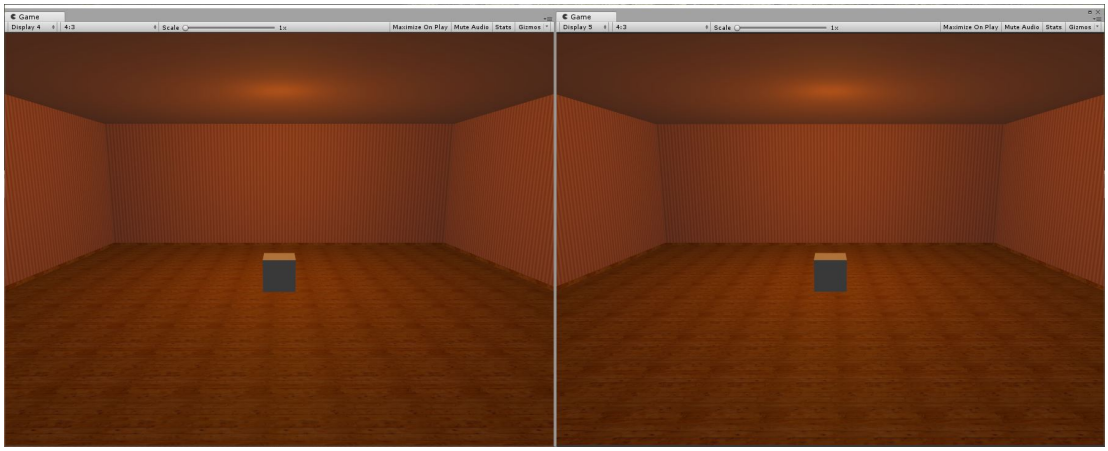


Abbildung 3.16.: Der Raum der, für das Szenario bereitsteht. In der Mitte ein Würfel mit der Kantenlänge von einem Meter zum Größenvergleich.

Als nächstes muss der Laufweg festgelegt werden. Für dieses Szenario soll er von der Mitte aus starten und dann links oder rechts bis zur Wand führen. An der Wand soll dann die automatische Rotation geändert und der Weg an der Wand fortgesetzt werden. Nach einem kurzem Stück Wand wird die Rotation zum zweiten Mal geändert und der Weg über die Decke fortgesetzt. Auf der anderen Seite werden noch zwei weitere Rotationen ausgeführt, bis der Rundweg in der Mitte abgeschlossen ist.

Da nun der Weg durch den Raum bekannt ist, kann jetzt eingerichtet werden, ohne dass der Laufweg verstellt wird. Die Einrichtung verteilt sich über den Raum 3.17, lässt aber den Weg in der Mitte frei. Damit der Effekt der Rotationen stärker wahrgenommen wird, ist eine hohe Immersion wichtig: eine animierte Katze und Zimmer Pflanzen verstärken die Immersion der Rotation.

Als letztes müssen die Rampen in den Raum 3.18 eingefügt werden. Diese sind schlicht und nicht an die Umgebung angepasst, damit sie mehr in den Vordergrund treten. Es handelt sich

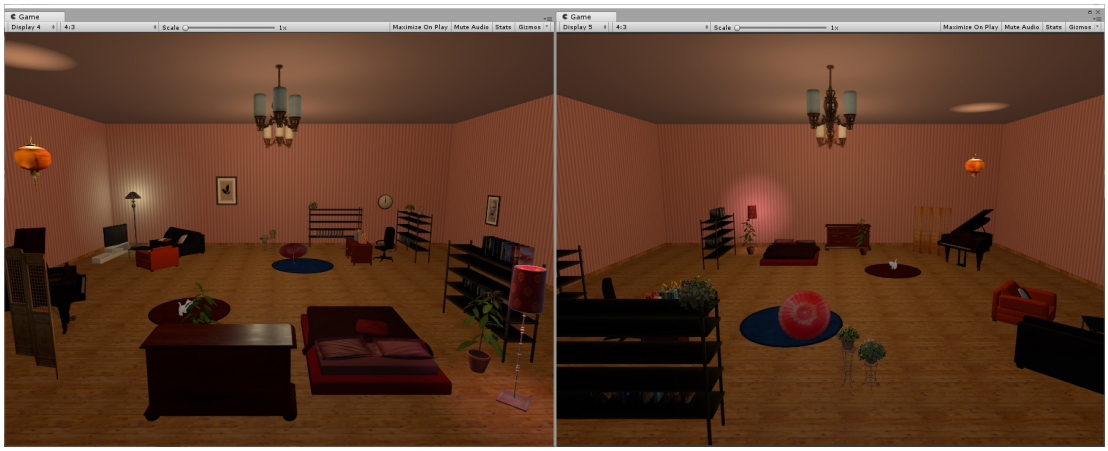


Abbildung 3.17.: Der eingerichtete Raum

bei den Rampen lediglich, um sieben zusammen gesetzte Blöcke. Einer davon ist der schräge Boden und drei jeweils das Geländer auf beiden Seiten.

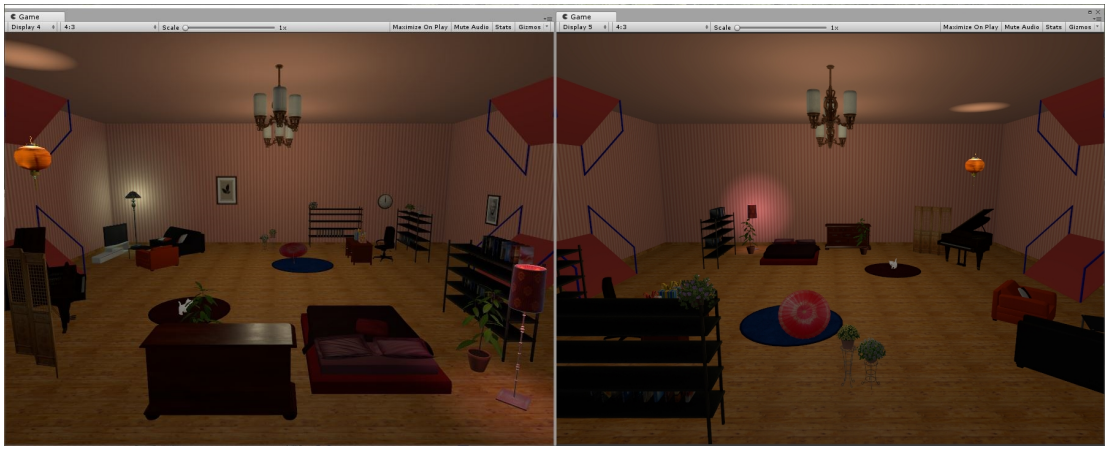


Abbildung 3.18.: Der eingerichtete Raum mit Rampen

Einbindung der Plattform

Der Raum ist nun fertig eingerichtet und so kann es an die Einbindung der Plattform gehen. Dazu müssen die benötigten Dateien in Unity eingefügt werden. Dieses kann über unterschiedliche Wege passieren, der einfachste ist es, die Ordnerstruktur hinzuzufügen. Dabei handelt es sich um ein Material für die Farbe des Käfigs und eine weitere für die Player-Kapsel. Das Prefab

(das Vorgefertigte Game-Objekt) und die dazugehörenden Skripte sollten auch hinzugefügt werden.

Da nun alle assets in der Szene vorhanden sind, geht es als nächstes darum, die Physik-Ebenen einzurichten, damit die Kollisionserkennung auch funktionieren kann. Die Grafik 3.19 zeigt das Game-Objekt MovementControllerPlayer mit der gleichnamigen Komponente. Dort müssen zwei Einstellungen erfolgen. Es muss als erstes ein Physik-Layer für den Charakter gewählt werden, dieses muss oben rechts unter dem Layer des Game-Objektes eingestellt werden, in der Grafik ist es das Layer CharController. Dabei ist zu beachten, dass die Kind-Objekte nicht mit umgestellt werden. Zweitens müssen die Layer eingestellt werden, mit denen Kollisionen erkannt werden sollen, dieses geschieht über die Variable Obstacles Layers im Inspector. Die Option beinhaltet ein Dropdown Menü, in dem alle Ebenen einzeln ausgewählt werden können, an dieser Stelle wurde das Layer mit dem Namen StaticCollisions gewählt.

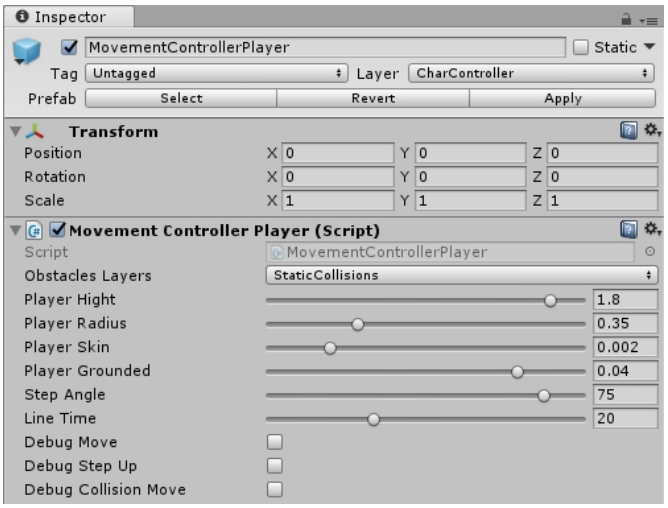


Abbildung 3.19.: Inspector des Game-Objekts MovementControllerPlayer. Großes Bild im Anhang A.7 auf Seite 77.

Nun, da die Einstellungen für die Kollisionserkennung abgeschlossen sind, kann die Szene angepasst werden. Oftmals haben Objekte einen eigenen eigenen Collider, in diesem Fall ist es ausreichend dem entsprechendem Game-Objekt das Layer, das für die Kollisionen ausgewählt wurde, zuzuordnen. Für alle Objekte, die keinen Collider haben, sollte ein eigener Collider gesetzt werden, hierfür eignet sich in den meisten Fällen der Box Collider. Dieser muss dann entsprechend auf die Wände und Gegenstände gesetzt werden, die nicht passiert werden sollen. In der Abbildung 3.20 sind die Kollider in der Szene zu sehen.

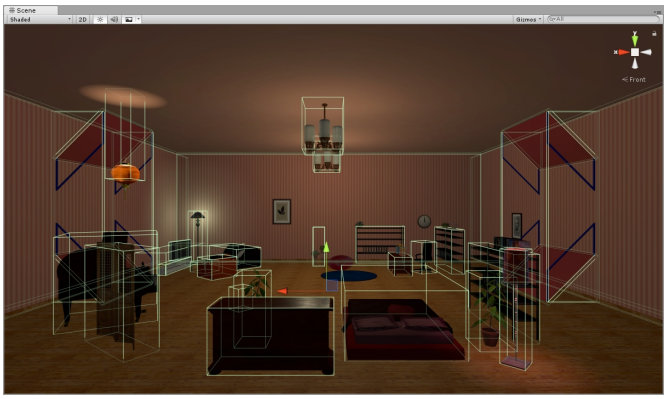


Abbildung 3.20.: Der eingerichtete Raum mit Kollidern. Großes Bild im Anhang A.8 auf Seite 78.

An dieser Stelle ist es hilfreich die beiden weiteren für die Szene benötigten Physik-Layer anzulegen. Hinzugekommen sind an dieser Stelle die Layer Gravity und Speed. Das Gravity Layer ist für, dass auslösen der Drehungen zuständig und das Speed Layer dient dazu die Bewegungs-Geschwindigkeit anzupassen. Nun sollten im PysicsManager 3.21 die Kollisionen zwischen den einzelnen Ebenen eingestellt werden. Alle angezeigten Layer bis auf StaticCollisions, CharController, Gravity und Speed werden von der Plattform nicht benötigt. Für die Einstellungen ist es wichtig, dass der CharController nur mit den Elementen kollidieren darf, die ein Hindernis darstellen sollen, in diesem Fall StaticCollisions. Für Gravity und Speed ist es wichtig, dass diese mit sich selber kollidieren dürfen um die entsprechenden Kollisionen zu erkennen.

Danach müssen die Collider für die Rotationen an die Rampen angebracht werden, es werden einfache Box Collider genommen. Dabei ist es wichtig, dass das Oben des Colliders in die Richtung zeigt, die das neue Oben sein soll. Auf dem Bild 3.22 sind die Collider für die Rotation zu sehen. Jede der Rampen hat zwei Collider, damit die Rotation in beide Richtungen möglich ist. Die Rotation zur neuen Ebene ist jeweils am Ende der Rampe. Die Collider werden dabei alle dem Gravity Layer zugeordnet. Der entsprechende Trigger sitzt unter dem Game-Objekt MovementControllerPlayer mit dem Namen GravCollider und muss in dem selbem Layer sitzen wie die Collider Boxen. Damit das Game-Objeckt aber auch auf die Kollisionen mit einem Trigger reagiert, liegt ein Skript darauf, dass die Rotation der Colliderboxen an die Komponente für Rotation weitergibt.

Als letztes werden die Collider für die Geschwindigkeits-Kontrolle eingebunden. Dieses kommen auch auf die Rampen und reduzieren die Geschwindigkeit über Annäherung, damit die

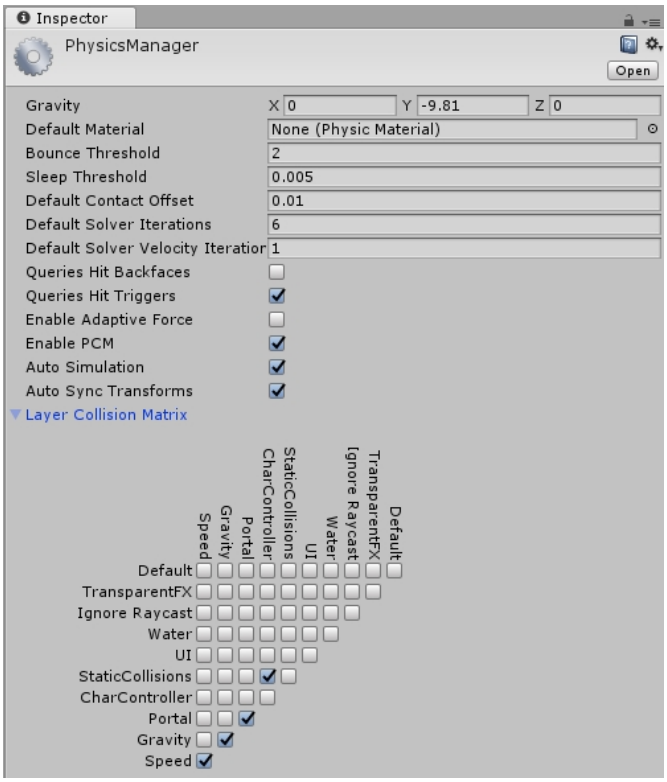


Abbildung 3.21.: Inspector des Unity Pysics Manager. Großes Bild im Anhang A.9 auf Seite 79.

Abbildung 3.22.: Bild der Rotations-Collider in Unity. Großes Bild im Anhang A.10 auf Seite 80.

zusätzliche Bewegung während der Rotation reduziert ist. Dieses soll Gleichgewichtsstörungen vermindern, indem die zusätzlich Querbewegung reduziert wird. Diese Collider bestehen ebenfalls aus Collider-Boxen und werden großflächig über die Rampe gelegt. Zusätzlich brauchen

diese aber noch ein Skript, das den entsprechenden Faktor für die Geschwindigkeit enthält und diese auch der Komponente mitteilt und wieder aufhebt, wenn der Collider verlassen wird.

Mit diesen Schritten sind alle Funktionen für die Szene gesetzt und die Szene ist einsatzbereit.

Kontrollen erstellen

Abschließend soll noch ein Beispiel für die Kontrolle des Szenarios gegeben werden. Dieses soll anhand der manuellen Aufhebung des SuspendCalculatePositionOverrite geschehen. Hierzu wird ein weiteres Skript benötigt, das an eine beliebige Komponente angehängt wird. Der Code, der für diese Funktion benötigt wird, ist lediglich wenige Zeilen lang 3.2.8.

```
1
2 void Update()
3 {
4     if (Input.GetKeyDown("s"))
5     {
6         SBC.AnchorMovement.SuspendCalculatePositionOverrite = false;
7     }
8 }
```

Dieser kurze Code besitzt die gewünschte Funktion. Der SBC ist die Statische Komponente, bei der die anderen Komponenten registriert sind. Danach wird auf AnchorMovement zugegriffen und über die Schnittstelle der Overrite ausgeschaltet. Da es sich hier um C# handelt ist die Funktion mit einem Getter/Setter über eine definierte Schnittstelle gleichzusetzen und nicht mit einer Öffentlichen Variable gleichzusetzen.

3.3. Erfüllung der Anforderungen

In diesem Kapitel wird die Plattform anhand der gestellten Anforderungen 3.1 beurteilt. Dazu werden auch die Szenarien aus Kapitel 2.9 herangezogen.

3.3.1. Allgemeine Anforderungen

Die allgemeinen Anforderungen an die Plattform sind durch die Implementation erfüllt. Im folgenden wird auf diese eingegangen.

1. Frei einsetzbar
   Das wichtigste Merkmal der Plattform, freie Einsetzbarkeit, ist durch mehrere Umstände

gesichert. An erster Stelle steht, dass als Grundlage Unity genommen wurde, das weit verbreitet ist. An zweiter Stelle ist die Modularität der Plattform, sie erlaubt nicht nur ein einfaches Erneuern, Austauschen und Abschalten von Komponenten. Sie erzwingt auch Schnittstellen zwischen den Komponenten und erleichtert somit das Parametrisieren der Komponenten. Dieses ist für fast alle Komponenten über den Inspector möglich.

Ungültige Konstellationen von Parametern können Komponenten aber auch funktionsunfähig machen, z. B. die Physik Komponente. Aus diesem Grund haben die viel genutzten Komponenten Regler, die den Umfang der verfügbaren Werte einschränken, um Fehler zu vermeiden. Sollten diese aber zu anderen Werten geändert werden müssen, ist es notwendig die Annotation der Variabelen zu ändern, da diese die Werte einschränken.

2. Modularität der Plattform
   Es ist bereits unter Punkt 1 erwähnt worden, dass der Plattform eine Modulstruktur zugrunde liegt. Im Betracht des Wandels der Technik ist sie auch optimal, um die Langlebigkeit zu optimieren.

3. Ersetzbarkeit des HMD
   Durch die Komponente ”User Komponente, 3.2.6” findet die Abstraktion des HMD statt, so dass auch das Kriterium, das HMD austauschen zu können, erfüllt ist. Durch diese Funktion wird auch der Desktop Modus funktionsfähig und kann über eine Checkbox ein- und ausgeschaltet werden. Der First Person Controller lässt sich mit ”WASD” und Maus steuern und erlaubt die Nutzung der Plattform auch ohne HMD. Dieses führt zu einer kleineren Einschränkung in der Handlungsfreiheit, da Funktionen wie Ducken in diesem Modus noch nicht implementiert sind. Dass der First-Person Controller bei der Entwicklung hilfreich ist wurde schon festgestellt, besonders da er den Betrieb der Plattform auf nicht VR Rechnern erlaubt. An dieser Stelle muss aber aufgepasst werden, denn durch das Wegbleiben einiger Funktionen kann es passieren, dass Tests unzureichend sind, um das ganze Spektrum der möglichen Bewegung abzudecken. In VR kann es dann zu Bugs kommen, die beim Testen nicht aufgefallen waren. Die Qualität dieser Fehler ist oft mit der von ”Local Space - World Space - Fehlern” vergleichbar.

4. Nutzung in der VR
   Die Nutzung in der VR ist einfach. Das Verfahren für Annäherung ist simpel genug. Auch der Käfig erledigt seine Aufgabe und reduziert die Probleme mit dem Gleichgewichtssinn während der Drehungen. Somit hilft er auch der Cybersickness entgegen zu wirken, da er ein Fixpunkt ist, der immer gleich zu der Realität steht.

5. Kontrollierbarkeit von Szenarien
   Szenarien lassen sich über die vorhandenen Optionen der Plattform justieren, dazu zählen die Begrenzungen durch die Physik. Des Weiteren gibt es die Einstellungs-Möglichkeiten über den Inspector, sowie mithilfe von weiterem Code über die definierten Schnittstellen. Dieses ist bereits im Test-Szenario geschehen, das als Beispiel dient.

3.3.2. Hardware und Software

Die Anforderungen der Software an die Hardware sollten für alle Rechner, die ”VR Ready” sind, erfüllt sein. Dieses gilt zumindest für die aktuellen Test-Szenarien, da der größte Faktor die Grafik im Szenario ist, denn die Anforderungen der Plattform sind im Vergleich zur Grafikberechnung zu vernachlässigen. Was die Framerate während der VR Nutzung zerstören kann ist das Debugging, um dieses zu vermeiden lässt es sich global abstellen, damit es nicht an einzelnen Stellen vergessen werden kann.

Durch die Wahl von Unity als Softwareumgebung ist die Kompatibilität mit einer Vielzahl von Hardware gegeben und lässt sich sogar auf Smartphones exportieren, sofern diese leistungsstark genug sind. Die Plattform lief bereits auf unterschiedlichen Rechnern.

3.3.3. Bewegung

Die Tests haben gezeigt, dass das Annäherungsverfahren für den User nutzbar ist. Die Begrenzungen der realen Fläche stellen durch die so erzeugte Bewegung kein Hindernis für größere virtuelle Räume da. Auf Grund der implementierten Schnittstelle für Akkumulation von Bewegung ist das Hinzufügen von weiteren Bewegungskonzepten nicht schwieriger, als diese selbst zu implementieren. Die Komponente aus ”Anchor Contoller” und ”Anchor Movement” erfüllt ihre Aufgabe, die einfache Implementierung zu unterstützen. In diesem Fall wirkt sich auch das Arbeiten im Local Space positiv aus, da so kein Umdenken nötig ist, wenn es um das Übertragen von Richtungen zwischen Realität und Virtualität geht.

Die Freischaltung der indirekten Bewegung nach dem Erreichen der Mitte funktioniert fehlerfrei. Zusätzlich ist die Option indirekte Bewegung über einen Overwrite auszuschalten bei einer kleineren realen Fläche sehr nützlich. Andernfalls würde die indirekte Bewegung hier zu früh ausgelöst werden und würde zu ungewollter Bewegung führen. Diese würde Irritation auslösen und den Start in das Szenario erschweren. Wenn nötig lässt sich die Plattform auch noch in einem Gebiet von 2 mal 2 Metern problemlos nutzen, wenn der Overwrite von Hand gesteuert wird.

3.3.4. Kollision

Kollisionen sind ein heikles Thema, die Algorithmen, die im Moment verwendet werden, arbeiten gut genug, um Tests zu ermöglichen ohne fürchten zu müssen, dass der User z. B. durch den Boden fällt. Die Art, wie Durchdringung durch direkte Bewegung verhindert wird, ist ungewöhnlich, führt aber zu wenig Problemen. Die Versuche, Gegenstände zu durchdringen, finden normalerweise lediglich unbeabsichtigt und selten statt. In diesen Fällen erfüllt das System genau seinen Zweck. Besonders für die Abgrenzung von virtuellen Gebieten ist dieses Verfahren hilfreich.

Auch das Bewältigen von Stufen und Unebenheiten im Boden funktioniert, dabei lässt sich auch die maximale Höhe, die überwindbar ist, ändern. Die Auflösung von Kollisionen mit virtuellen Wänden erzielt auch die gewünschten Ergebnisse und lässt den Teil der Bewegung zu, der möglich ist. Somit besteht keine Gefahr an der Wand ”kleben” zu bleiben.

Die eigene Implementierung des Charakter Controllers hat einen Makel. Die Funktionalität ist für eine Testumgebung ausreichend, dennoch ist es wünschenswert diese Komponente durch eine Alternative zu ersetzen, sobald eine vorhanden ist, die ihre Aufgaben besser erfüllt. Denn es treten vereinzelt Probleme auf, z. B. wenn das Tracking verloren geht und das HMD in der Zeit zu weit bewegt wird, bevor das Tracking wieder einsetzt. In diesen Fällen ist es möglich, dass Kollisionen nicht richtig erkannt werden, da die Position in diesen Fällen oft springt.

3.3.5. Anpassung von Rotation um die waagerechte Achse

Die Komponente für die Rotationen kann Rotationen berechnen und durchführen. Dieses funktioniert sowohl für die kontinuierliche Rotation als auch die einzelnen Berechnungen durch Trigger. Hier ein Beispiel für die Rotation auf einer Rampe mit Trigger im Anhang: A.4, A.5. Die kontinuierliche Rotation funktioniert auch, wobei diese für gekrümmte Oberflächen besser geeignet ist. Jede der beiden Rotationsarten hat somit ihren eignen Anwendungsfall, in dem sie nützlicher als die andere ist.

3.3.6. Abschließende Beurteilung

Die geschaffene Plattform erfüllt all die an sie gestellten Anforderungen von der Bewegung bis zu der Erweiterbarkeit. Es ist möglich, Tests mit der Plattform zu betreiben und diese entsprechen anzupassen. Ebenso sind Erweiterungen aufgrund der Schnittstellen einfach zu designen, um weitere Testszenarien zu entwerfen und durchzuführen, die andere Eigenschaften benötigen. Dabei ist es möglich, die Plattform unabhängig vom Szenario einzusetzen, denn

aufgrund der modularen Struktur sollten die meisten Testmethoden leicht implementierbar sein. Dieses ist auch in dem Test-Szenario deutlich geworden, da die Methoden zum Kontrollieren der Plattform sehr offen sind. Dadurch ist es möglich weitere Geräte wie z. B. Biosensoren zu nutzen, um Testergebnisse zu erhalten. So lassen sich auch Phänomene, wie Cybersickness auf unterschiedliche Arten untersuchen.

Damit ist die Plattform gut geeignet, um ihre Aufgaben zu erfüllen. Die einzelne Komponenten funktionieren wie gewünscht und erlauben es, beliebige Szenarien zu entwerfen und umzusetzen.

4. Evaluation

In ”Erste Erfahrungen mit Probanden” werden die ersten Eindrücke und Ergebnisse vorgestellt. Darauf aufbauend geht es um die Weiterentwicklung der Plattform.

4.1. Erste Erfahrungen mit Probanden

In diesem Kapitel geht es um die ersten Erfahrungen mit der Plattform, im Zusammenhang mit Alpha-Testern. Inzwischen waren über 150 Personen in dem Szenario, die Altersklassen waren dabei weit gestreut. Wichtig in diesem Zusammenhang ist auch, dass viele der Alpha-Tester noch nie eine VR Brille aufgehabt hatten. Während zu diesem Zeitpunkt noch keine Untersuchungen gelaufen sind, konnten dennoch Charakteristiken erkannt werden. Das genutzte Szenario ist jenes, das in Erstellung eines Test Szenarios gebaut wurde, Abbildung im Anhang A.6 auf Seite 77. Da es in dieser ersten Runde mit Alpha Testern keine speziellen Vorgaben für die Tests gab, sind die Rückmeldungen nicht konsistent. Einzelne Punkte konnten bereits festgestellt werden.

4.1.1. Der Ablauf

Im Zusammenhang mit der bisherigen Nutzung soll auch ein typischer Durchlauf durch das Testszenario beschrieben werden.

Am Anfang des Durchganges gibt es ein kurze Beschreibung des Szenarios, in dem erklärt wird, dass es in dem Virtuellem Raum möglich ist, an den Wänden und Decken zu gehen. Daraufhin beginnt die Prozedur das HMD aufzusetzen, dieses gestaltet sich unterschiedlich schwierig, je nach Frisur, Brille oder Erfahrung mit VR. In den meisten Fällen steht der User nicht mittig auf der realen Fläche. Die nächste Aufgabe lautet daher, in die Mitte des transparent blauen Kastens zu gehen, dort schaltet sich dann unmerklich die Bewegung frei. Dieses gibt dem User auch die Möglichkeit sich umzusehen. An dieser Stelle wird dann auf die Rampen hingewiesen, die dazu dienen, die Rotation zu ändern. Damit diese aber auch erreicht werden können, wird nun das Konzept für die Bewegung erklärt: ”Einfach in die gewünschte Richtung gehen, bis zum Rand des blauen Käfigs. Dieser beginnt dann langsam nach Vorne zu gleiten, einen Schritt

zurück und er hört wieder auf. Auf diese Art können Sie sich im Raum frei bewegen." Wenn die Distanz zur Rampe dann kleiner wird, gibt es den Hinweis stabil zu Stehen und wenn das Gleichgewicht merkwürdig wird, die Augen zu schließen. Die erste Drehung wird von fast allen gut überstanden. Gefolgt wird diese erste Rotation meistens von drei weiteren über die verbleibenden Rampen, bis der Boden wieder erreicht wird. An dieser Stelle ist die Rundgang dann zu Ende.

Natürlich gibt es viele Stellen, an denen der Ablauf von der Beschreibung abweichen kann. Oft wird die Katze entweder vor der ersten oder nach der letzten Rotation begutachtet. Das Klavier wurde ebenfalls ab und zu wahrgenommen, meistens im Zusammenhang dieses spielen zu wollen. Sehr selten kam es vor, dass Leute den ganzen Raum erkunden wollten. Die überwiegende Anzahl setzte das HMD selber nach der letzten Rotation ab. Oftmals, wenn der Zeitrahmen auf Grund einer größeren Gruppe beschränkt war, wurden auch nur zwei Rotationen erlebt. Wenn es sich um Gruppen handelte, war der Monitor, der das Geschehen aus der Perspektive des Users zeigte, besonders wichtig, damit die anderen auch etwas sehen konnten. Dieses half auch den aktuellen Status in der VR zu zeigen, damit entsprechend auf den User reagiert werden konnte.

4.1.2. Immersion

Da das Test-Szenario als erster showcase für die Rotationen dienen soll, ist es wichtig, dass sich diese auch real anfühlen. Unter diesem Gesichtspunkt ist die Immersion wichtig. Die Einrichtung in dem Szenario erfüllt ihren Zweck unter diesem Aspekt gut genug, um das Szenario wirklich wirken zu lassen. Gleichzeitig dienen die einzelnen Einrichtungsgengenstände als Landmarken, die das Gefühl vermitteln der Raum würde auf dem Kopf stehen. Als besondere Merkmale der normalen Ausrichtung der Szene gelten die hängende Lampe, die im rechtem Bild A.6 rechts hinten hängt und die Animierte weiße Katze. Die Katze war ein besonderer Anziehungspunkt in diesem Szenario, was daran gelegen haben könnte, dass sie das einzige wahrnehmbare animierte Objekt im Raum war. Manche Alpha-Tester sagten, dass die Katze das beste Objekt im Raum war, dieses könnte an der Animation gelegen haben. Insgesamt ist das Szenario überzeugend genug gewesen, um das Gefühl der Rotation zu vermitteln.

4.1.3. Rotation

Die Rotation wurde überwiegend gut angenommen, es gab lediglich zwei Alpha-Tester, die sich die Rotation nicht zugetraut haben. Viele der Alpha-Tester haben kleinere Störungen im Gleichgewichtig während der Rotation erfahren. Schwere Störungen im Gleichgewicht sind

aber sehr selten aufgetreten. Um die Auswirkungen der Rotation zu mildern, ist es in dieser Situation hilfreich, bewusst stabil zu stehen. Dieses führt dazu, dass das reale Unten deutlicher wahrgenommen wird, dadurch reduziert sich die Gefahr das Gleichgewicht zu verlieren. Eine kurze Störung im Gleichgewicht ist normal und tritt besonders beim erstem Mal auf. Mit jeder weiteren Rotation nehmen die Störungen des Gleichgewichts ab. Die Stärke der Störung hängt des Weiteren von der Blickrichtung ab: ist der Blick während der Rotation auf die Wand gerichtet, sind die Auswirkungen stark reduziert. Die Bilder im Anhang auf Seite 80 zeigen die Drehung in drei Schritten, vor der Drehung A.11, während der Drehung A.12 und nach der Drehung A.13. Da in dieser Blickrichtung keine Objekte als Landmarken dienen, werden die auftretenden Probleme reduziert. Wenn der Blick während der Drehung aber in den Raum gerichtet ist, steht der Gleichgewichtssinn unter einer höheren Herausforderung, dieses ist nicht jedem zu empfehlen, da die Auswirkungen um vieles stärker sind. Im Vergleich bleiben Rotationen mit Blick in den Raum dauerhaft die größere Herausforderung.

Die Orientierung nach den Drehungen war für die überwiegende Anzahl kein ernstes Problem, wobei es einzelne Berichte von Desorientierung gab. Es war aber in so gut wie allen Fällen keine Schwierigkeit für die Alpha-Tester den Rundgang abzuschließen.

4.1.4. Cybersickness

Die typische Zeit, die eine Person in der VR und damit dem Szenario verbringt, überschreitet die 5 Minuten Grenze nur selten. Aufgrund dessen lässt sich lediglich etwas über sofortig eintretende Cybersickness aussagen. Die Möglichkeit, sich über die Annäherung zu bewegen, war lediglich für sehr wenige ein echtes Problem, wobei die meisten dieser Personen generell mit VR ein Problem hatte. Eine recht häufige Bemerkung war, dass es sich ein wenig wie Rolltreppe anfühle, einige berichteten auch davon, dass die Bewegung etwas unangenehm gewesen sei. Abgesehen von einigen Ausnahmen wurde die Funktionsweise der Bewegung gut verstanden. Insgesamt ist das System mit seiner moderaten Geschwindigkeit erträglich, denn in diesem Fall gilt die Faustregel, dass mehr Geschwindigkeit auch mehr Cybersickness bedeutet.

Während die Rotationen nicht explosionsartig zu Cybersickness führten, nehmen diese doch einen Einfluss darauf. Auf Grund der langsamen Geschwindigkeit der Rotationen waren die auftretenden Probleme aber minimal. Auch an dieser Stelle gilt die Faustregel für die Wahl der Geschwindigkeit, wenn diese aber zu weit abgesenkt wird, stört es das Erlebnis mehr als dass es hilft.

Als letztes soll über die Rampen gesprochen werden. Diese sind einer der Faktoren, die in einigen Fällen unangenehmen aufgefallen sind. Besonders kommt an dieser Stelle zum Tragen dass die Position über den Kopf bestimmt wird, so führt nämlich eine Bewegung des Kopfes schon zu einer Bewegung auf der Rampe und damit auch zu einer Höhenänderung, was als unangenehm wahrgenommen wird. In den meisten Fällen wurde auf der Rampe nicht angehalten, weshalb diese Mechanik nicht zum Tragen kam.

4.1.5. Abschließend

Insgesamt waren die ersten Begegnungen, die mit der Plattform und den Alpha Testern stattgefunden haben, erfolgreich. Dabei sind die ersten Eindrücke positive. Es gab weder mit der Rotation noch mit der freien Bewegung generelle Probleme.

4.2. Weiterentwicklung

In diesem Kapitel geht es um die Weiterentwicklung der Plattform, dabei werden jene Erweiterungen vorgestellt, die die Funktionalität erweitern und die Nutzung vereinfachen.

4.2.1. Kollisions Erkennung

Die Komponente, die am meisten von einem Upgrade profitieren würde, ist jene, die sich mit den Kollisionen beschäftigt. Es würde ein deutlicher Mehrgewinn entstehen, wenn die aktuelle Implementierung durch eine ersetzt werden könnte, die die Qualität des Character Controllers (CC) von Unity hätte und die Fähigkeit rotiert zu werden. Auf diese Weise würde die Kollisionserkennung an Qualität gewinnen und die Wartung einfacher werden. Die Änderung der Physik Komponente wäre einfach durchzuführen, da lediglich die Methoden, die als Schnittstelle dienen, umgeschrieben werden müssen und sich diese an der Funktionalität des CC orientieren.

4.2.2. Erweitertes Tracking

Als weiteres wäre ein Tracking der Füße hilfreich, dieses könnte helfen das aktuelle Problem, das auf Rampen existiert, in den Griff zu bekommen. Hierdurch könnten unnötige Höhenschwankungen vermieden werden und auch die Bewegung über Annäherung würde profitieren, wenn sie nicht an den Kopf gebunden ist, denn auch so würde die Position des Users stabiler. Hierfür müssten dann Gewichtungen der Positionen stattfinden, um die neue Mitte zu bestimmen oder ein neuer oder angepasster Algorithmus für die Bewegung durch Annäherung

erstellt werden. Nachteil könnte sein, dass ein Teil der Einfachheit verloren geht, wenn nicht mehr eindeutig ist, wie sich die Positionsberechnung zusammensetzt.

4.2.3. Walking in Place

Da zumindest einige ein Problem mit der Bewegung über Annäherung hatten, wäre es schön eine Alternative zu haben, die ebenfalls ohne weitere Contoller auskommt. Eine Möglichkeit hierfür wäre Walking in Place. Dieses würde das, was als ungutes Gefühl in der gleitenden Bewegung wahrgenommen wurde, reduzieren. Hierfür würde aber ein Algorithmus gebraucht, der dieses zuverlässig erkennt, am besten direkt über das HMD. Es wäre auch möglich die Erkennung an weitere Sensoren zu koppeln, dieses würde aber verhindern, dass diese Option überall verfügbar ist. Aufgrund der Modularität sollte es aber zu keinen Problemen führen. Eines der Probleme ist, dass die Erkennung innerhalb der gesamten Fläche möglich sein sollte, um die Bewegungsfreiheit nicht einzuschränken. Es gibt bereits Untersuchungen, die ein Neuronales Netz für die Erkennung verwendet haben, dieses hat den Nachteil, dass es für eine genauere Erkennung auf jede Person trainiert werden muss. Eine Möglichkeit dieses Problem zu entschärfen wäre das Einschränken des Walking in Place auf einen Teil der Fläche. Der Aufenthalt in der dafür vorgesehenen Fläche könnte als Indiz genommen werden, dass das Walking in Place grade erwünscht ist und es wäre nicht so schlimm wenn die Erkennung schneller anspringt oder auch etwas zu früh.

4.2.4. Wechsel des Käfigs

Um die Immersion zu verbessern, gibt es ein Punkt der auffällt. Es ist der Käfig, dieser könnte durch ein Hilfsmittel ausgetauscht werden, das weniger Präsenz besitzt. Eine mögliche Weiterentwicklung wäre es, den Radius auf dem Boden anzuzeigen, so würde die Sicht nicht beschränkt. Wie dieses bewerkstelligt werden kann und ob es angenehmer ist, ist eine andere Frage. Alternativ wäre auch eine Variante des Käfigs interessant, die ihre Transparenz ändert, je nach Abstand, um weniger im Weg zu stehen.

4.2.5. Trigger Layer

Im Moment gibt gibt es für unterschiedliche Trigger Funktionen unterschiedliche Physik-Ebenen. Es wäre von Vorteil, wenn nur noch ein Layer benötigt würde, da diese in Unity auf 32 begrenzt sind. Dieses könnte über Skripte gelöst werden, die von dem entsprechenden Trigger angesprochen werden und dann unterschiedliche Optionen auslösen. Die Option hätte den Nachteil, dass jeder Trigger entsprechende Informationen kennen muss, damit die richtige

Option ausgelöst werden kann. Insgesamt würde es das System aber universeller und einfacher zu nutzen machen, da über diese Schnittelle beliebige Aktionen ausgelöst werden könnten.

Die Uhr

Für eine Weiterentwicklung des Szenarios wäre es hilfreich die Immersion zu erhöhen. Eine Möglichkeit dieses zu bewerkstelligen wäre es in dem Raum mehr sichtbar animierte Objekte zu haben. Ein Beispiel dafür wäre die Uhr, die bereits animiert ist und die richtige Zeit anzeigt aber nicht wahrgenommen wird, in eine größere umzuwandeln, die ein Pendel hat und akustisch wahrnehmbar ist.

5. Schluss

Im letztem Kapitel wird diese Arbeit in ihren Kapiteln zusammengefasst und ein Ausblick in die Zukunft der Plattform und der Untersuchungen gegeben.

5.1. Zusammenfassung

In dieser Arbeit wurde ein Plattform für Bewegung und Rotation in der VR entworfen, welche Entwickler von Szenarien für die VR unterstützen soll. Die Plattform ist eine Komponente, die in Unity eingefügt werden kann und dann eine ganze Reihe von Aufgaben übernimmt. Zu diesen gehören die Berechnungen für Bewegung und Rotation, diese lassen sich einfach für eine Szene einrichten.

Raumwahrnehmung

Im dem Kapitel Aspekte der Raumwahrnehmung wurden für die Arbeit wichtige Aspekte der Raumwahrnehmung aus der Literatur vorgestellt. Es wurde über die Fähigkeit sich in Räumen zu Orientieren gesprochen und dass diese Fähigkeit an vielen unterschiedlichen Parametern hängt. Einige der Parameter, die vorgestellt wurden, waren die Beschaffenheit von Landmarken, die Qualität der verfügbaren Informationen und wofür diese benötigt wurden. Dann wurden auch Informationen über die Transition in die Virtuellen Welten untersucht und es ließ sich draus schließen, dass es Unterschiede zwischen den beiden Welten gibt. Einer dieser Punkte war die Cybersickness, die aufgrund der anderen Parameter der virtuellen Welt öfter anzutreffen ist, als ihre Variante Motion sickness in der realen Welt. Danach wurden Möglichkeiten sich in der VR zu bewegen untersucht, gefolgt von den Eigenheiten, die in der Wegfindung in VR auftauchen. Abgeschlossen wurde die Literatur mit einem kleinen Überblick über Untersuchungsmethoden, die in der VR genutzt werden können.

Aus der folgenden Zusammenfassung der Literatur sind einige Beispiele für Szenarien entstanden. Diese bieten die Möglichkeit einzelne Phänomene besser zu untersuchen. Damit diese aber umsetzbar werden, ohne jedes Mal ein spezifische Lösung zu implementieren, wurde das Ziel formuliert eine Plattform zu bauen. Dieses soll die Entwicklung unterstützen, so

dass mehr Zeit in das Design der eigentlichen Szenarien gesteckt werden kann. Dazu wurden die wichtigsten Anforderungen aufgestellt, die von der Plattform zu erfüllen sind, um diese attraktiv zu machen.

Entwicklung

In diesem Teil der Arbeit ging es um die Erstellung der Plattform und des Testszenarios. Als erstes werden die Anforderungen an die Plattform spezifiziert, die für eine reibungslose Nutzung der Plattform notwendig sind. Die wichtigste in diesem Punkt war die Modulstruktur der Plattform, denn sie erleichtert viele der Anforderungen, wie die Austauschbarkeit von Komponenten und die Möglichkeit zur Erweiterung. Daraufhin wurden Spezifikationen zu den Bewegungsmöglichkeiten gemacht und spezielle Schnittstellen für einzelne Komponenten gefordert. Dieses war z. B. eine Abstraktion für das HMD und die Möglichkeit unterschiedliche Bewegungsquellen einzubinden.

Im folgendem Teil wurden die Spezifikationen in ihre entsprechenden Komponenten umgesetzt und es wurden Entscheidungen für das Design getroffen. Zwei davon waren, Unity als Engine und C# als Programmiersprache zu verwenden. Dabei sind Komponenten mit unterschiedlichen Kopplungsgraden entstanden. Die entsprechenden Grade wurden in drei Ränge aufgeteilt. Von ”Kernkomponenten”, die tief in der Plattform liegen, bis zu den ”Freien Komponenten”, die für die Funktionsweise des Systems nicht notwendig sind. So wurde die Bewegung über Annäherung umgesetzt und dafür gesorgt, dass auch Bewegung in der realen Welt keine Gegenstände durchdringen kann. Auch wurde eine Kollisionserkennung für Bewegung geschrieben, da die von Unity verfügbaren Methoden nicht für VR geeignet waren.

Danach wurde ein Testszenario gebaut, an dem die Funktionsweise der Plattform sichtbar wurde. An diesem wurde auch eine Anleitung zur Nutzung der Plattform erklärt, die den einfachen Einsatz der Plattform erlaubt.

Abschließend wurde die Plattform auf die Erfüllung der Anforderungen untersucht, wobei einige sich schon davor bei der Erstellung des Test-Szenarios als funktionsfähig erwiesen haben. Es wurden alle Anforderungen erfüllt, auch wenn es das Ziel ist, die Kollisionserkennung irgendwann durch eine geeignete Komponente von Unity zu ersetzen. Dennoch sind die gebotenen Möglichkeiten der Plattform zur Erstellung von Szenarien nützlich genug, um dienlich zu sein, da viele Aufgaben abgenommen werden.

Evaluation

In dem Kapitel der Evaluation wurde als erstes unter Beweis gestellt, dass die Plattform im Umgang mit Alpha-Testern tatsächlich funktioniert. Dabei sind auch bereits Erkenntnisse gewonnen worden: dass die Rotationen in diesem Szenario nur ein begrenztes Problem darstellen, die Bewegung durch Annäherung das Gefühl einer Rolltreppe hat und animierte Katzen Anziehungspunkte darstellen.

Die Rotation hat lediglich bei einer sehr geringen Anzahl von Personen zu Problemen geführt. Die einzigen häufigen Aufkommen sind, dass während der ersten Rotation kleine Fluktuationen im Gleichgewicht auftauchen, die aber nicht gefährlich sind. Dieses Problem wird mit den folgenden Rotationen besser und die Fluktuationen verschwinden bei den meisten vollkommen. Die Bewegung über Annäherung hat, abgesehen von leichtem Unwohlsein, das für die meisten nur am Anfang bestand, gut funktioniert. Es gab auch immer wieder vereinzelte User, die das Prinzip nutzten, um den Raum zu erkunden, überwiegend wurde aber die Katze betrachtet.

Darauf wurde unter dem Punkt, der Weiterentwicklung über die Punkte gesprochen, die noch nachgebessert werden sollten. Zum Beispiel die Physik Komponente, die sich mit der Kollisionserkennung beschäftigt. Andere Punkte die, dabei angesprochen wurden, sind der Käfig und eine Erweiterung des Trackings für eine bessere Positionsbestimmung, da die aktuelle ich auf Rampen negativ bemerkbar macht.

5.2. Ausblick

Die Plattform bietet alle Funktionen, um zum Beispiel folgende Untersuchungen zur Raumwahrnehmung durchzuführen:

- Auswirkungen des Alters der Test-Personen auf die Raumwahrnehmung in der VR
- Gibt es Nachwirkungen durch die vertikale Rotation in der VR?
- Lassen sich Treppen in der VR durch An-Der-Wand-Laufen ersetzen?
- Wie ist die Akzeptanz von vertikaler Rotation in der VR zum Beispiel im Gegensatz zum Fliegen?

Diese und weitere Untersuchungen werden mit der zunehmenden Popularität von VR immer wichtiger. Da die Verbreitung der VR mit immer neuen Möglichkeiten einhergeht, macht es

Untersuchungen umso wichtiger, da die virtuellen Welten immer realistischer sein können und andererseits fantastische Welten immersiver werden.

Die Untersuchungsmöglichkeiten beschränken sich nicht nur auf die Raumwahrnehmung im Zusammenhang mit Rotationen. Weitere Konzepte, die mit der Raumwahrnehmung zusammenhängen, sind zum Beispiel die virtuelle Körpergröße des Users. Wie hängt sie mit der Raumwahrnehmung des Einzelnen zusammen? Wird die Raumwahrnehmung verändert, wenn es möglich ist, sich unter verschiedenen Gegenständen hindurch zu ducken, während ein Gebiet erkundet wird? Wie wirkt sich Teleportation auf die Raumwahrnehmung aus?

Für diese weiteren Funktionen ist es notwendig weitere Komponenten zu erstellen, die die entsprechenden Aufgaben übernehmen. So wäre es zum Beispiel gut, Komponenten für die Messung der Abweichung beim Zeigen zu implementieren, damit diese dann auch zur Verfügung stehen, um den Prozess des Auswertens zu vereinfachen.

Ein weiterer wichtiger Punkt ist die Entwicklung der VR auf Smartphones. Noch ist die vorhandene Rechenleistung für hochauflösende VR nicht vorhanden, die Möglichkeiten werden in den nächsten Jahren aber immer größer. Damit wäre es auch ein Option, zu versuchen, die Plattform für Smartphones nutzbar zu machen.

A. Anhang

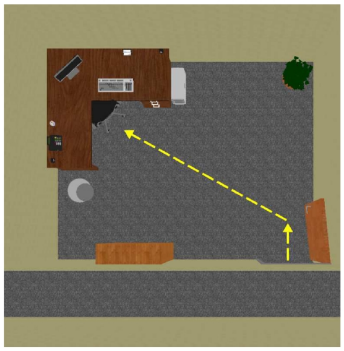
Abbildungen

SSQ symptom	Weight		
	Nausea	Oculomotor	Disorientation
General discomfort	1	1	0
Fatigue	0	1	0
Headache	0	1	0
Eyestrain	0	1	0
Difficulty focusing	0	1	1
Increased salivation	1	0	0
Sweating	1	0	0
Nausea	1	0	1
Difficulty concentrating	1	1	0
Fullness of head	0	0	1
Blurred vision	0	1	1
Dizzy (eyes open)	0	0	1
Dizzy (eyes closed)	0	0	1
Vertigo	0	0	1
Stomach awareness	1	0	0
Burping	1	0	0
Column weighting for category scores	9.54	7.58	13.92

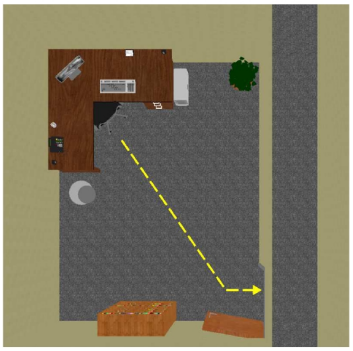
The total score is the summed symptom scores multiplied by 3.74

Abbildung A.1.: Einteilung der Symptome der Cybersickness in Nausea, Oculomotor und Disorientation [16, Tabelle 3]

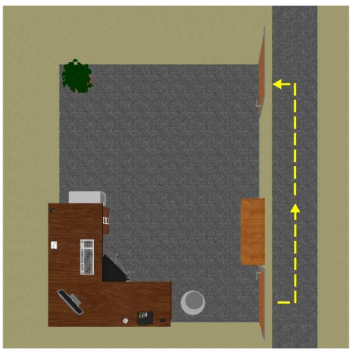
(a) **State 1**: The user approaches the doorway to the first room.



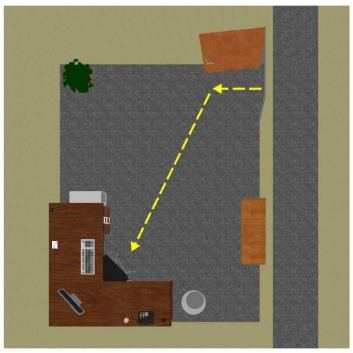
(b) **State 2**: The user enters the first room and walks towards the desk to activate the computer monitor.



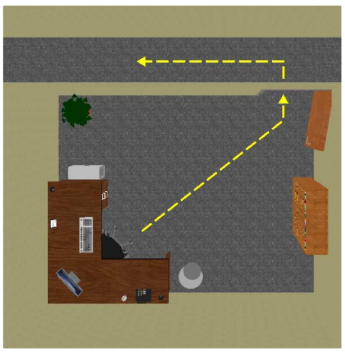
(c) **State 3**: When the user activates the monitor, the corridor and the door to exit the room are instantly rotated by 90 degrees. The user then turns around and exits the room.



(d) **State 4**: When the user enters the hallway, the second doorway is added, and the contents of the room are swapped with the next room.



(e) **State 5**: The user enters the second room and walks towards the desk to activate the computer monitor.



(f) **State 6**: When the user activates the monitor, the corridor and the door to exit the room are instantly rotated by 90 degrees. The user then turns around and exits the room.

Abbildung A.2.: In der Abbildung ist ein Teil des Szenarios aus Ëxploiting Change Blindness to Expand Walkable Space in a Virtual Environment [23, Figure 2] zu sehen. **(a)** Der User nähert sich der Tür zum ersten Raum. **(b)** Der User betritt den ersten Raum und geht auf den Tisch zu, um den Computer Bildschirm zu aktivieren. **(c)** Wenn der User den Bildschirm aktiviert, wird der Gang und die Tür zum Verlassen des Raumes augenblicklich um 90 Grad gedreht. Dann dreht der User sich und verlässt den Raum. **(d)** Wenn der User den Gang betritt, wird eine weitere Tür hinzugefügt und der Inhalt des Raumes wird mit dem des nächstem ausgetauscht. **(e)** Der User betritt den zweiten Raum und geht auf den Tisch zu, um den Computer Bildschirm zu aktivieren. **(f)** Wenn der User den Bildschirm aktiviert, wird der Gang und die Tür zum Verlassen des Raumes augenblicklich um 90 Grad gedreht. Dann dreht der User sich und verlässt den Raum.

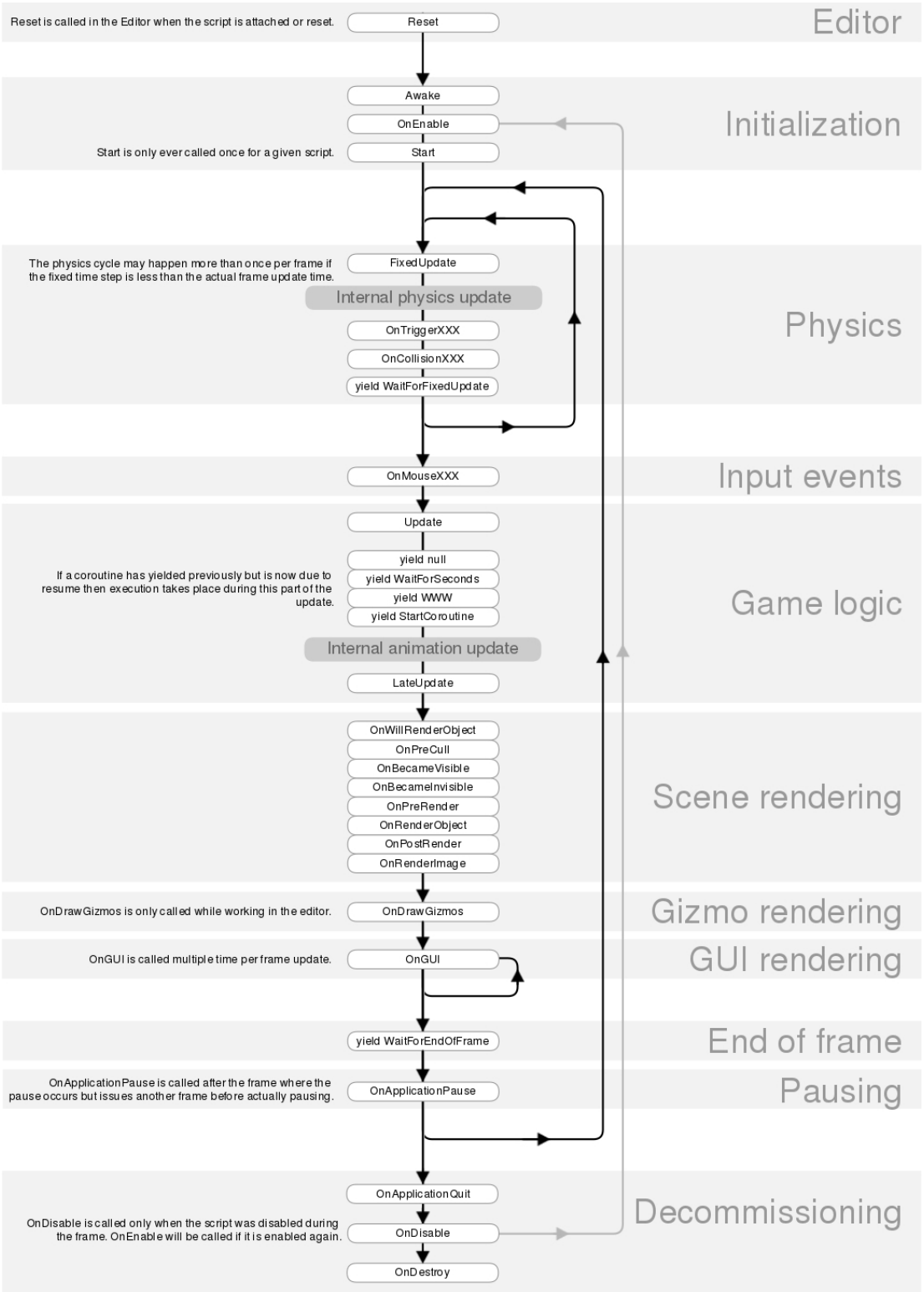


Abbildung A.3.: Die Abbildung stellt die Unity Execution Order dar [25]

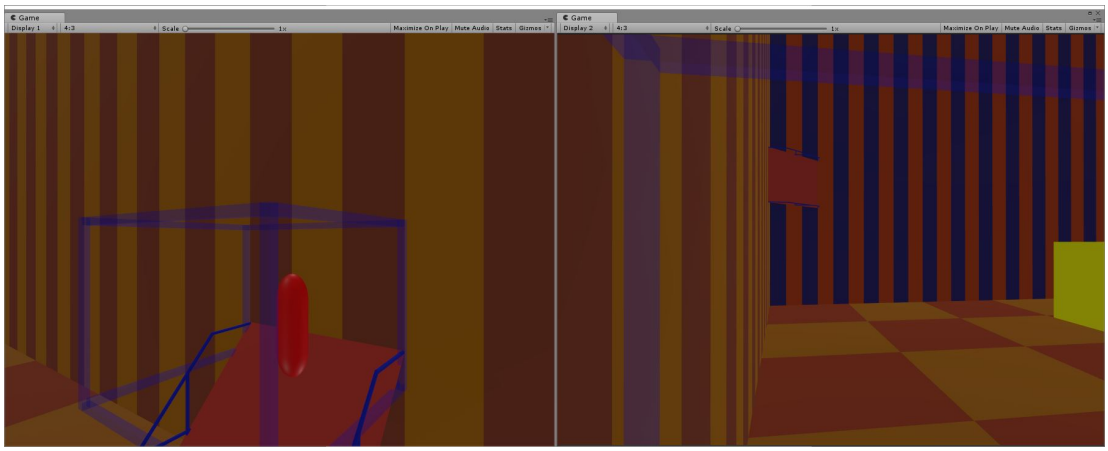


Abbildung A.4.: Vor der Rotation auf einer Rampe, linkes Bild externe Sicht, rechtes Bild Charakter-Sicht. Die Kapsel symbolisiert den Charakter.

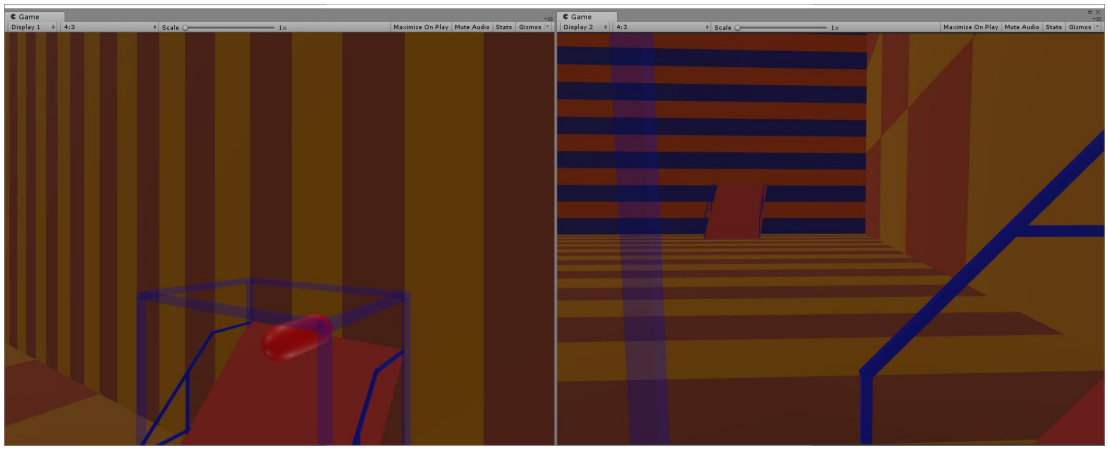


Abbildung A.5.: Nach der Rotation auf einer Rampe, linkes Bild externe Sicht, rechtes Bild Charakter-Sicht. Die Kapsel symbolisiert den Charakter.

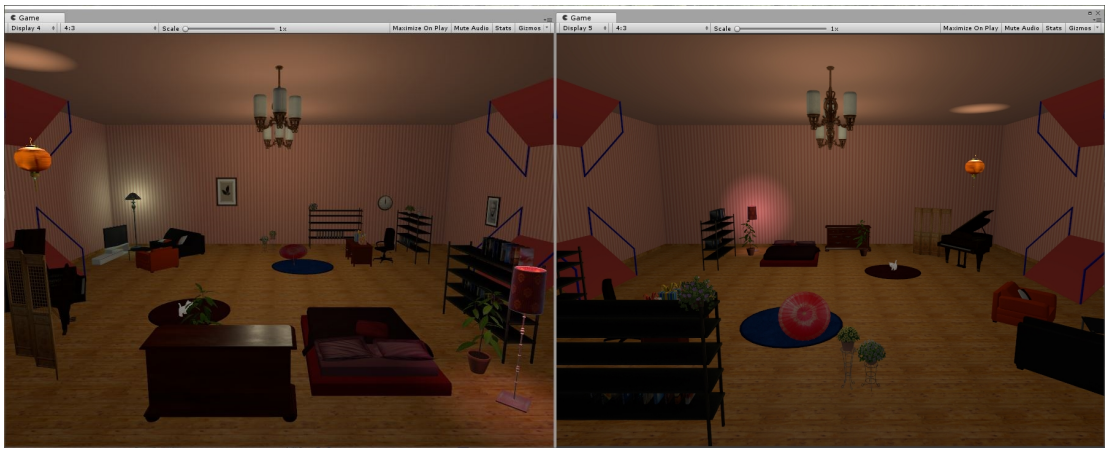


Abbildung A.6.: In den Bildern ist das Test-Szenario von beiden Seiten aus zu sehen

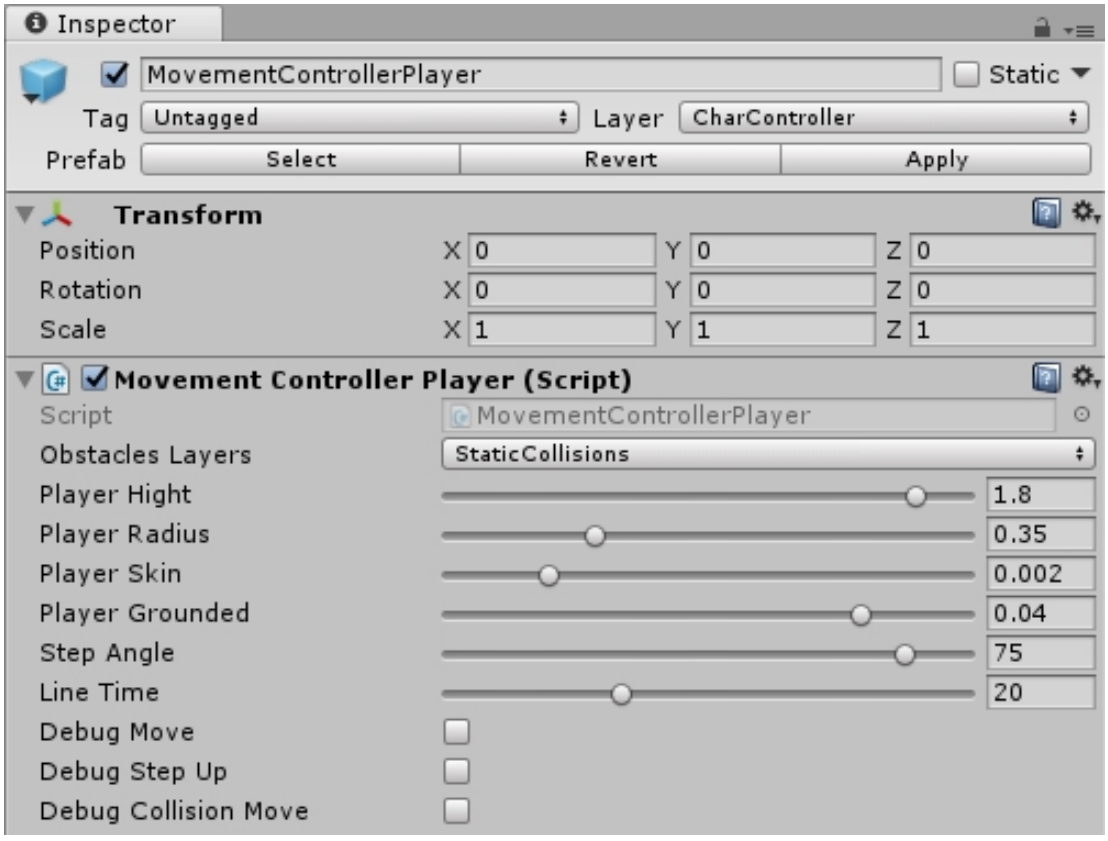


Abbildung A.7.: Großes Bild vom Inspector des Game-Objekts MovementControllerPlayer

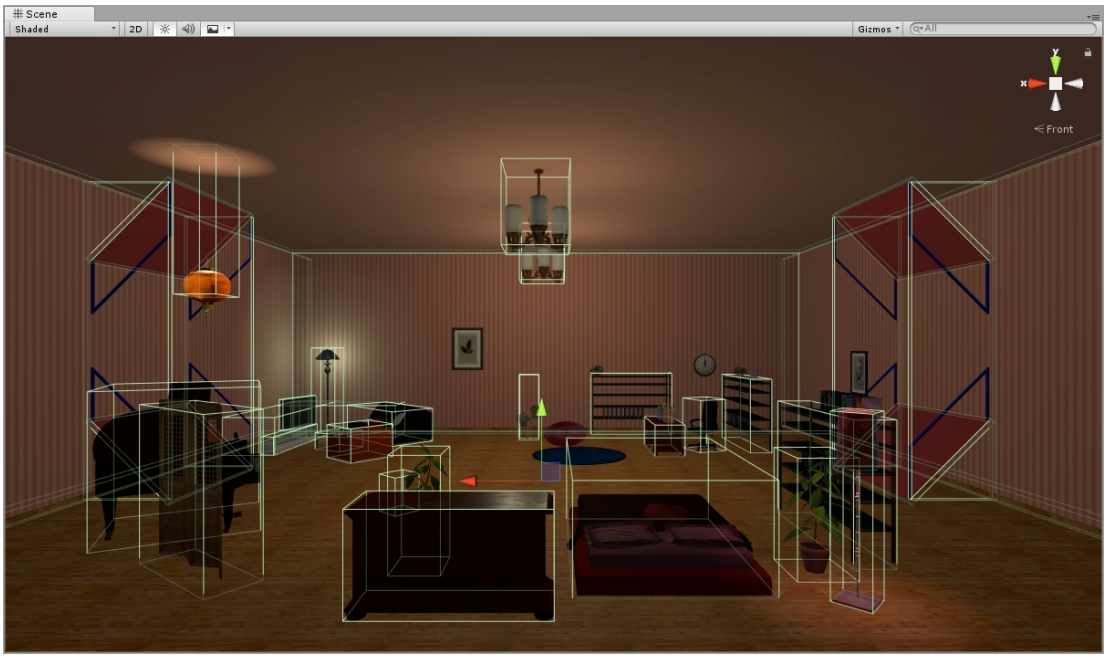


Abbildung A.8.: Großes Bild vom eingerichteten Raum mit Kollidern

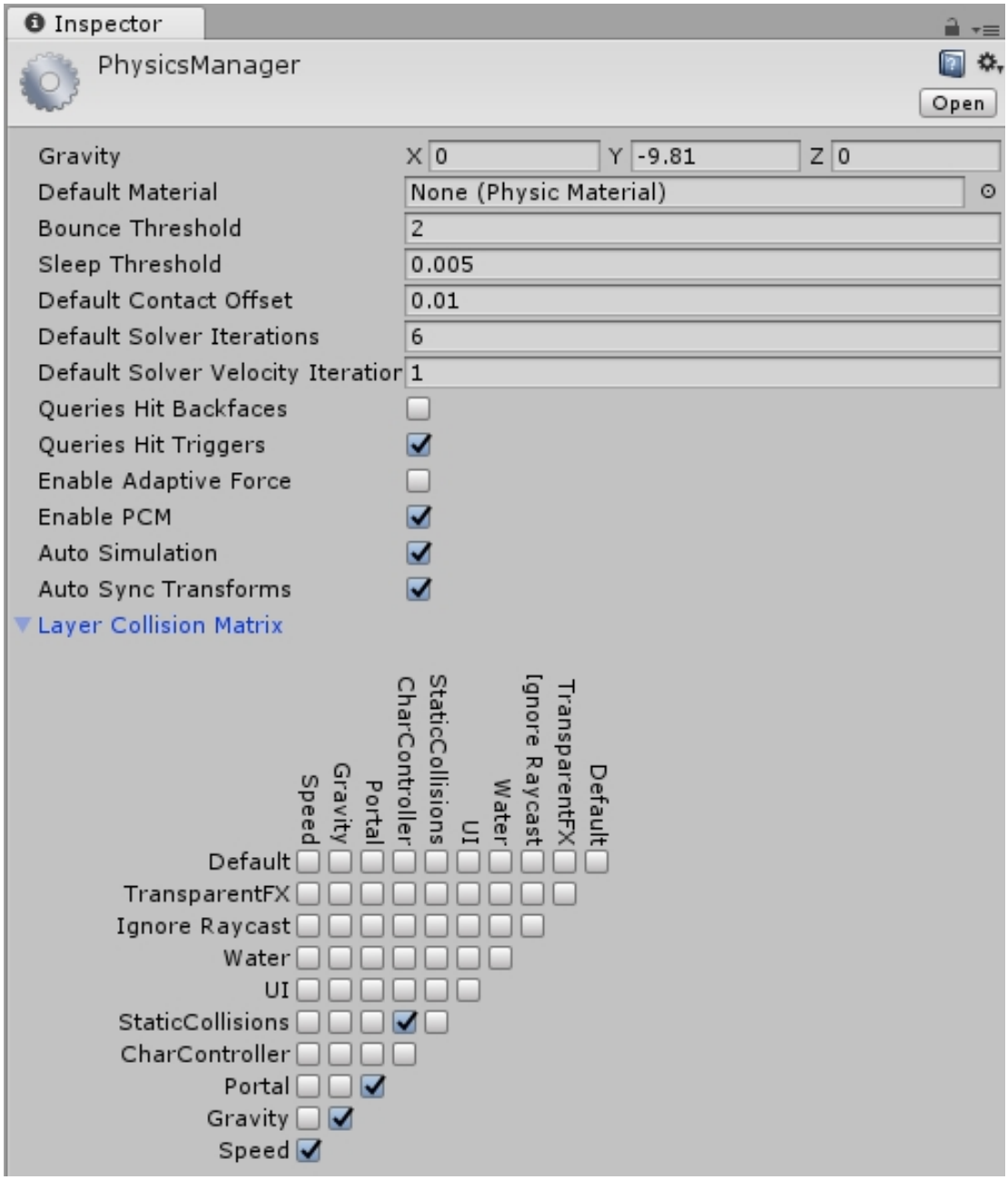


Abbildung A.9.: Großes Bild vom Inspector des Unity Physics Manager

Abbildung A.10.: Großes Bild von den Rotations-Collideren in Unity

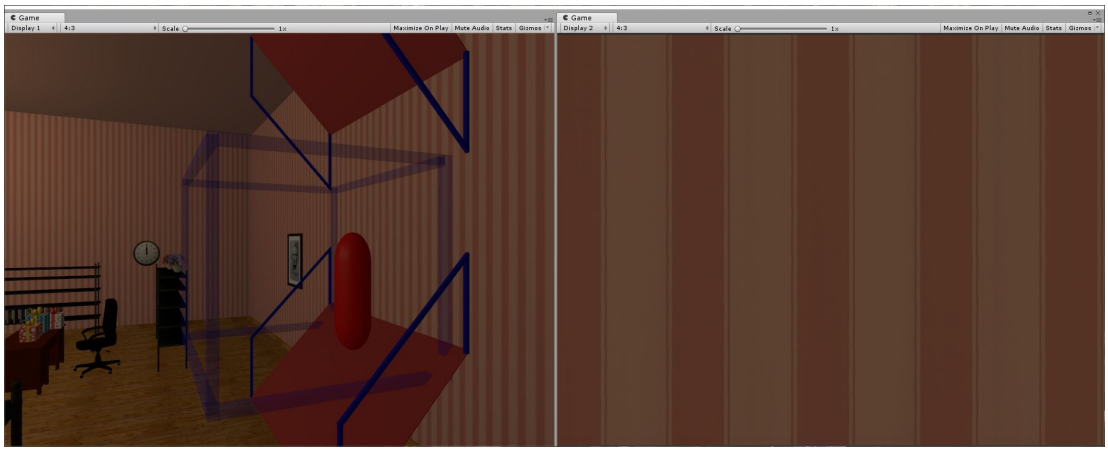


Abbildung A.11.: Vor der Rotation auf einer Rampe im Test-Szenario. Linkes Bild externe Sicht, rechtes Bild Charakter-Sicht. Die Kapsel symbolisiert den Charakter.

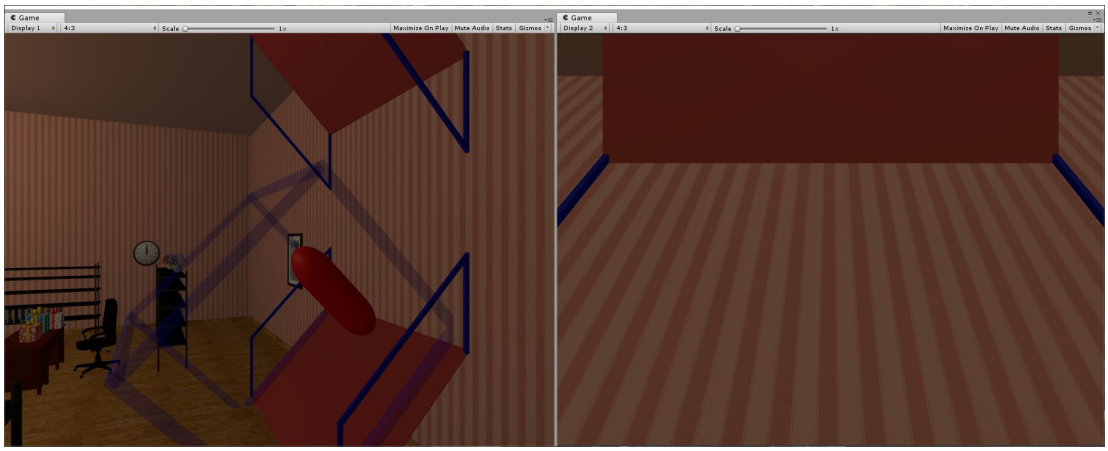


Abbildung A.12.: In der Rotation auf einer Rampe im Test-Szenario. Linkes Bild externe Sicht, rechtes Bild Charakter-Sicht. Die Kapsel symbolisiert den Charakter.

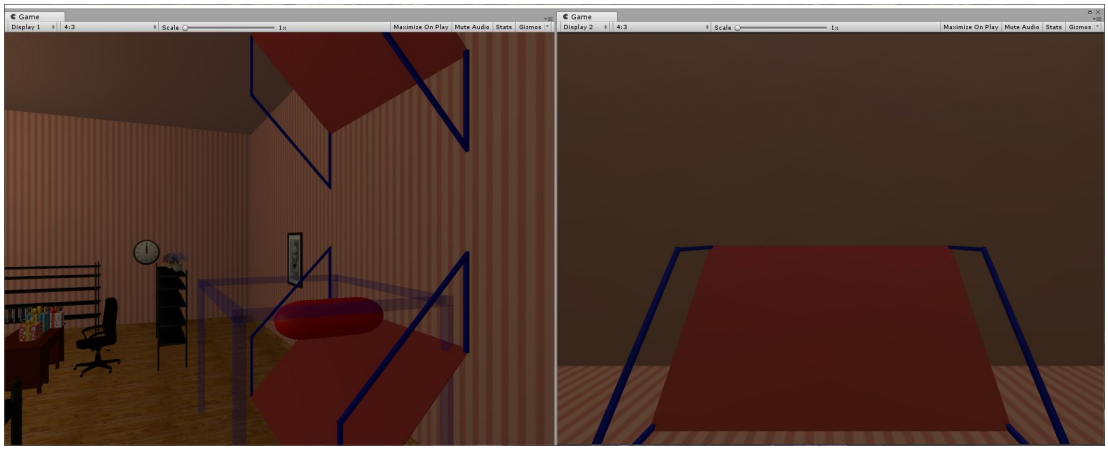


Abbildung A.13.: Nach der Rotation auf einer Rampe im Test-Szenario. Linkes Bild externe Sicht, rechtes Bild Charakter-Sicht. Die Kapsel symbolisiert den Charakter.

Literaturverzeichnis

[1] Andra ; Andra (Hrsg.): *The City as a Canvas: Three-Dimensional Street Art*. Eingesehen 2017-08-14. – URL https://pixel77.com/city-as-a-canvas-three-dimensional-street-art/

[2] Caduff, David ; Timpf, Sabine: On the assessment of landmark salience for human navigation. In: *Cognitive Processing* 9 (2008), Nr. 4, S. 249–267. – URL http://dx.doi.org/10.1007/s10339-007-0199-2. – ISSN 1612-4790

[3] Center, Bremen Spatial C. ; Center, Bremen Spatial C. (Hrsg.): *Bremen Spatial Cognition Center*. Eingesehen 2017-07-04. – URL http://bscc.spatial-cognition.de/node/12

[4] Chen, Haiwei ; Fuchs, Henry: Towards Imperceptible Redirected Walking: Integrating a Distractor into the Immersive Experience. In: *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. New York, NY, USA : ACM, 2017 (I3D '17), S. 22:1–22:2. – URL http://doi.acm.org/10.1145/3023368.3036844. – ISBN 978-1-4503-4886-7

[5] Darken, Rudolph ; R. Cockayne, William ; Carmein, David: *The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds*. 01 1997

[6] Davis, Simon ; Nesbitt, Keith ; Nalivaiko, Eugene: A Systematic Review of Cybersickness. In: *Proceedings of the 2014 Conference on Interactive Entertainment*. New York, NY, USA : ACM, 2014 (IE2014), S. 8:1–8:9. – URL http://doi.acm.org/10.1145/2677758.2677780. – ISBN 978-1-4503-2790-9

[7] Garg, Amit ; Fisher, Joshua A. ; Wang, Wesley ; Singh, Karan P.: ARES: An Application of Impossible Spaces for Natural Locomotion in VR. In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2017 (CHI EA '17), S. 218–221. – URL http://doi.acm.org/10.1145/3027063.3048416. – ISBN 978-1-4503-4656-6

[8] Grechkin, Timofey ; Thomas, Jerald ; Azmandian, Mahdi ; Bolas, Mark ; Suma, Evan: Revisiting Detection Thresholds for Redirected Walking: Combining Translation and Curvature Gains. In: *Proceedings of the ACM Symposium on Applied Perception*. New York, NY, USA : ACM, 2016 (SAP '16), S. 113–120. – URL http://doi.acm.org/10.1145/2931002.2931018. – ISBN 978-1-4503-4383-1

[9] Kelle, Peteris ; Kelle, Peteris (Hrsg.): *50 Absolutely Stunning 3D Street Art (Paintings)*. Eingesehen 2017-08-14. – URL http://www.hongkiat.com/blog/absolutely-stunning-3d-street-art-paintings

[10] Kluss, Thorsten ; Marsh, William E. ; Zetzsche, Christoph ; Schill, Kerstin: Representation of impossible worlds in the cognitive map. In: *Cognitive Processing* 16 (2015), Nr. 1, S. 271–276. – URL http://dx.doi.org/10.1007/s10339-015-0705-x. – ISSN 1612-4790

[11] Langbehn, Eike ; Bruder, Gerd ; Steinicke, Frank: Subliminal Reorientation and Repositioning in Virtual Reality During Eye Blinks. In: *Proceedings of the 2016 Symposium on Spatial User Interaction*. New York, NY, USA : ACM, 2016 (SUI '16), S. 213–213. – URL http://doi.acm.org/10.1145/2983310.2989204. – ISBN 978-1-4503-4068-7

[12] Lubos, Paul ; Bruder, Gerd ; Steinicke, Frank: Safe-&-round: Bringing Redirected Walking to Small Virtual Reality Laboratories. In: *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*. New York, NY, USA : ACM, 2014 (SUI '14), S. 154–154. – URL http://doi.acm.org/10.1145/2659766.2661219. – ISBN 978-1-4503-2820-3

[13] Mammen, Sebastian von ; Knote, Andreas ; Edenhofer, Sarah: Cyber sick but still having fun. In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, VRST 2016, Munich, Germany, 2-4 November, 2016*, URL http://doi.acm.org/10.1145/2993369.2996349, 2016, S. 325–326

[14] Mohler, Betty J. ; Bülthoff, Heinrich H. ; Thompson, William B. ; Creem-Regehr, Sarah H.: A Full-body Avatar Improves Egocentric Distance Judgments in an Immersive Virtual Environment. In: *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization*. New York, NY, USA : ACM, 2008 (APGV '08), S. 194–. – URL http://doi.acm.org/10.1145/1394281.1394323. – ISBN 978-1-59593-981-4

[15] Norimichi, Idehara ; Takahiro, Ohtsuka ; Michika, Inoue: Interactive System Based on the False Perception of Acceleration and Verticality. In: *Proceedings of the 2015 Virtual Reality International Conference*. New York, NY, USA : ACM, 2015 (VRIC '15), S. 21:1–21:4. – URL http://doi.acm.org/10.1145/2806173.2806189. – ISBN 978-1-4503-3313-9

[16] Rebenitsch, Lisa ; Owen, Charles: Review on Cybersickness in Applications and Visual Displays. In: *Virtual Reality* 20 (2016), Juni, Nr. 2, S. 101–125. – URL http://dx.doi.org/10.1007/s10055-016-0285-9. – ISSN 1359-4338

[17] Richardson, Anthony E. ; Montello, Daniel R. ; Hegarty, Mary: Spatial knowledge acquisition from maps and from navigation in real and virtual environments. In: *Memory & Cognition* 27 (1999), Nr. 4, S. 741–750. – URL http://dx.doi.org/10.3758/BF03211566. – ISSN 1532-5946

[18] Riecke, Bernhard E. ; Sigurdarson, Salvar ; Milne, Andrew P.: Moving through virtual reality without moving? In: *Cognitive Processing* 13 (2012), Nr. 1, S. 293–297. – URL http://dx.doi.org/10.1007/s10339-012-0491-7. – ISSN 1612-4790

[19] Röser, Florian ; Hamburger, Kai ; Knauff, Markus: The Giessen virtual environment laboratory: human wayfinding and landmark salience. In: *Cognitive Processing* 12 (2011), Nr. 2, S. 209–214. – URL http://dx.doi.org/10.1007/s10339-011-0390-3. – ISSN 1612-4790

[20] Souman, J. L. ; Giordano, P. R. ; Schwaiger, M. ; Frissen, I. ; Thümmel, T. ; Ulbrich, H. ; Luca, A. D. ; Bülthoff, H. H. ; Ernst, M. O.: CyberWalk: Enabling Unconstrained Omnidirectional Walking Through Virtual Environments. In: *ACM Trans. Appl. Percept.* 8 (2011), Dezember, Nr. 4, S. 25:1–25:22. – URL http://doi.acm.org/10.1145/2043603.2043607. – ISSN 1544-3558

[21] Steinicke, F. ; Bruder, G. ; Jerald, J. ; Frenz, H. ; Lappe, M.: Estimation of Detection Thresholds for Redirected Walking Techniques. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), Jan, Nr. 1, S. 17–27. – ISSN 1077-2626

[22] Suma, Evan A. ; Azmandian, Mahdi ; Grechkin, Timofey ; Phan, Thai ; Bolas, Mark: Making Small Spaces Feel Large: Infinite Walking in Virtual Reality. In: *ACM SIGGRAPH 2015 Emerging Technologies*. New York, NY, USA : ACM, 2015 (SIGGRAPH '15), S. 16:1–16:1. – URL http://doi.acm.org/10.1145/2782782.2792496. – ISBN 978-1-4503-3635-2

[23] Suma, Evan A. ; Clark, Seth ; Finkelstein, Samantha L. ; Wartell, Zachary: Exploiting change blindness to expand walkable space in a virtual environment. In: Lok, Benjamin (Hrsg.) ; Klinker, Gudrun (Hrsg.) ; Nakatsu, Ryohei (Hrsg.): *VR*, IEEE Computer Society, 2010, S. 305–306. – URL http://dblp.uni-trier.de/db/conf/vr/vr2010.html#SumaCFW10. – ISBN 978-1-4244-6258-2

[24] Thanner, Ulrike: *Unmögliche Treppen.* 1983

[25] Unity ; Unity (Hrsg.): *Unity Execution Order.* Eingesehen 2017-12-14. – URL https://docs.unity3d.com/560/Documentation/Manual/ExecutionOrder.html

[26] Wiesbaden, F.A. B.: *Brockhaus Enzyklopädie.* 1970

[27] Zhang, Ruimin ; Walker, James ; Kuhl, Scott A.: Improving Redirection with Dynamic Reorientations and Gains. In: *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception.* New York, NY, USA : ACM, 2015 (SAP '15), S. 136–136. – URL http://doi.acm.org/10.1145/2804408.2814180. – ISBN 978-1-4503-3812-7

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

Hamburg, 02.01.2018 Jonathan Becker