Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Niklas Alexander Gerwens

Virtueller Handschlag - Interaktion in einem Mehrbenutzer VR System

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Niklas Alexander Gerwens

Virtueller Handschlag - Interaktion in einem Mehrbenutzer VR System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: M.Sc. Tobias Eichler

Eingereicht am: 08. Dezember 2017

Niklas Alexander Gerwens

Thema der Arbeit
Virtueller Handschlag - Interaktion in einem Mehrbenutzer VR System

Stichworte
Virtuelle Realität, Mensch-Computer-Interaktion, Mehrbenutzer VR, interaktive 3D-Welten, Tracking, Ganzkörper-Avatar

Kurzzusammenfassung
In dieser Bachelorarbeit wurde ein Mehrbenutzer VR-System entworfen, welches einen virtuellen Handschlag ermöglicht. Dazu wurde eine interaktive 3D-Welt mit Ganzkörper-Avataren für die Nutzer erstellt und verschiedene Tracking Systeme kombiniert, um die Bewegung der Nutzer zu verfolgen. Zur Darstellung der 3D-Welt wurde ein Head-Mounted Display verwendet. Das System wurde vollständig implementiert und durch die Durchführung des virtuellen Handschlags in unterschiedlichen Szenarien evaluiert. Die Ergebnisse der Evaluation zeigen, dass Latenz und Präzision des Systems den Anforderungen eines erfolgreichen virtuellen Handschlags entsprechen.
Niklas Alexander Gerwens

Title of the paper
Virtual Handshake - Interaction in a multi-user VR System

Keywords
virtual reality, human-computer interaction, multi-user VR, interactive 3D-worlds, tracking, full body avatar

Abstract
In this bachelor thesis a multi-user VR system was designed, which enables a virtual handshake. For this an interactive 3D-world was created in which the users were represented with full body avatars. Multiple tracking systems were combined in order to track the users movement. To visualize the 3D-world a head-mounted display was used. The system was completely implemented and the virtual handshakes execution was evaluated in different scenarios. The evaluation results show that the systems latency and precision meet the requirements for a successful virtual handshake.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Konzept von Virtual Reality (VR) ist einem Menschen durch Simulation verschiedener Sinneseindrücke eine virtuelle Welt so zu vermitteln, dass dieser keinen Unterschied zur realen Welt bemerkt. Bekannt ist VR unter anderem durch ein häufiges Auftreten in der Science Fiction, welches dieser eine futuristische Note verliehen hat. Dabei ist VR auch schon seit einiger Zeit Gegenstand der Forschung. Bereits 1968 entwickelte Ivan Sutherland das „Sword of Damocles “, welches allgemein als das erste VR Head-Mounted Display gilt. Wirklich ausgebreitet hat sich das Thema VR jedoch erst in den letzten Jahren.

Grund dafür sind Fortschritte in der Kommunikations- und Informationstechnologie, vor allem leistungsfähigere Prozessoren und Grafikkarten sowie Displays mit höherer Auflösung. Diese ermöglichen kostengünstig eine Darstellung der virtuellen Welt mit einer Qualität, welche zum Beginn der Forschung noch undenkbar war. Die Qualität der Darstellung ist hierbei nur ein Faktor der VR, um den Grad der Immersion, also den Effekt inwieweit der Mensch die virtuelle Welt als real empfindet, zu manipulieren. Ein weiterer Faktor ist die Gestaltung der Interaktion mit der virtuellen Welt, da ein Mensch eine Welt grundsätzlich nicht nur wahrnimmt, sondern auch mit ihr interagiert.

Für die Interaktionsgestaltung in VR ergeben sich zwei übergeordnete Möglichkeiten. Einerseits können Interaktionen den aus Erfahrungen in der Realität entstandenen Erwartungen des Menschen entsprechen und so den Grad der Immersion erhöhen. Andererseits sind Interaktionen in der VR nicht an die Grenzen der Realität gebunden und können unter Verzicht auf eine exakte Spiegelung der realen Welt neue Funktionalitäten bereitstellen. Welche Art der Interaktion erstrebenswert ist, hängt dabei von dem Anwendungskontext und den zu erfüllenden Aufgaben ab.

Die Realisierung von interaktiven virtuellen Welten erfolgt durch VR-Systeme. Diese identifizieren die Interaktion eines Nutzers über Eingabegeräte, simulieren den Effekt der Interaktion in der virtuellen Welt und visualisieren dem Nutzer das Ergebnis über Ausgabegeräte. Aktuell

finden sich verschiedene VR-Systeme auf dem Endverbrauchermarkt, die eine Vielzahl von Interaktionen unterstützen. Dabei sind die Systeme meistens nur für einen Nutzer ausgelegt. Folglich wird ein wesentlicher Teil der Interaktion eines Menschen in der realen Welt, die Interaktion mit anderen Menschen, von diesen nicht umgesetzt. Aus diesem Zustand ergibt sich die Zielsetzung dieser Arbeit.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist die Durchführung eines virtuellen Handschlags zu ermöglichen. Ein virtueller Handschlag wird im Kontext dieser Arbeit folgend definiert:

- Zwei Personen befinden sich in der realen Welt im gleichen Raum.
- Beide sehen nur eine virtuelle Welt, in der sie sich ebenfalls im gleichen Raum aufhalten.
- Beide Personen sehen sich und die andere Person in Form einer Ganzkörperdarstellung in dieser virtuellen Welt.
- Die Bewegung der Personen im realen Raum wird im virtuellen Raum gespiegelt und die relationale Position beider im virtuellen Raum stimmt mit der im realen Raum überein.
- Die Personen vollführen simultan in der realen und in der virtuellen Welt einen Handschlag.

Für die Realisierung dieses Szenarios wird in dieser Arbeit ein VR-System entwickelt, welches die Interaktion von mehreren Benutzern wahrnehmen, simulieren sowie darstellen kann. Hierbei liegt der Fokus bei der Entwicklung des Systems auf den Interaktionen, die im Szenario auftreten. Grundsätzlich soll das System aber in der Lage sein unterschiedliche Interaktionen anzubieten und so als Basis für weitere Mehrbenutzer VR-Anwendungen zu dienen.

1.2 Aufbau der Arbeit

Diese Arbeit ist in fünf Kapitel aufgeteilt. Die Einleitung ist dabei das erste Kapitel und endet nach diesem Abschnitt.

Das Kapitel 2 beginnt mit der Vorstellung von zwei Verwandten Arbeiten. Anschließend wird das in der Einleitung bereits angesprochene Themengebiet der Interaktion in VR genauer erörtert und für das System geeignete Eingabegeräte beschrieben. Danach werden die Konzepte von Koordinatentransformation und Latenz im Kontext eines VR-Systems erklärt. Darauf folgt eine Beschreibung der Laborumgebung und dessen Kommunikationsinfrastruktur. Abgeschlossen wird das Kapitel mit einer Analyse der Anforderungen an das System.

In Kapitel 3 wird das Design des Systems erläutert. Zuerst wird eine Übersicht der Kommunikation und der Komponenten des Systems gegeben. Im Anschluss dazu werden Aufbau, Ablauf und Schnittstellen der einzelnen Komponenten thematisiert. Die Komponenten werden dabei in die Bereiche Tracking und Virtuelle Welt gruppiert.

Die Evaluation des Systems findet in Kapitel 4 statt. Bestandteile der Evaluation sind die Bewertung der in Kapitel 2 aufgestellten Anforderungen und ein Ausblick auf mögliche Erweiterungen des Systems.

Das Kapitel 5 fasst die Ergebnisse dieser Arbeit zusammen. Abschließend wird ein Ausblick auf Anwendungsgebiete von VR und Mehrbenutzer VR-Systemen gegeben.

2 Analyse

In diesem Kapitel werden die Anforderungen an das Mehrbenutzer VR-System für den virtuellen Handschlag analysiert. Dabei werden benötigte Grundlagen erörtert und die Laborumgebung beschrieben.

2.1 Verwandte Arbeiten

Um einen ersten Ausblick zu geben wie ein System aufgebaut sein könnte, welches das virtuelle Handschlag Szenario umsetzen kann, möchte ich zwei Projekte vorstellen, die ähnliche Zielsetzungen haben und beide ein VR-System für Mehrbenutzer Interaktion implementieren.

2.1.1 High Five in VR

Young u. a. (2015) beschäftigen sich in ihrer Arbeit mit der Interaktion zwischen zwei Benutzern in einer virtuellen Umgebung. Sie untersuchen die Fragestellung, ob die Form der Avatare oder die physische Nähe der Benutzer eine entscheidende Rolle spielt. Um dieses zu beantworten wird ein Mehrbenutzer VR-System entworfen und ein Versuch mit der High Five Geste als beispielhafte Interaktion durchgeführt.

Der Entwurf des Systems beinhaltet zwei Personen, die sich im gleichen realen Raum befinden und durch ein optisches Tracking System erfasst werden. Dieses besteht aus acht Kameras, die einen fünf mal fünf Meter großen Raum abdecken und die Personen anhand von sechs Markern erkennen, welche am Kopf, an der Hüfte sowie an den Händen und Füßen befestigt sind. Die für den Betrieb des Tracking System notwendige Software läuft auf einem Computer, der die gesammelten Daten an einen zweiten Computer sendet. Dort angekommen werden die Daten auf virtuelle Avatare abgebildet. Für die Zuordnung der Avatare wird eine manuelle Kalibration benötigt.

Anschließend werden beide Avatare an zwei weitere Computer weitergeleitet. Diese simulieren die virtuelle Welt und generieren den Output, der über ein Head-Mounted Display (HMD)

den Nutzern dargestellt wird. Dabei ist jeweils ein Computer mit angeschlossenem HMD für einen Nutzer zuständig. Beide Rechner arbeiten mit dem gleichen Code und unterscheiden nur welcher Avatar ihrem Nutzer gehört, setzen also die Perspektive unterschiedlich. Von den HMDs bereitgestellte Tracking Daten werden nicht genutzt.

Abbildung 2.1: High Five in der Virtuellen Realität (Young u. a. (2015))

Bei dem Versuch wird die High Five Geste wiederholt ausgeführt und dabei die Avatare zwischen einer Ganzkörperdarstellung und einer Darstellung, bei der nur die Hände und Arme visualisiert werden, ausgetauscht. Weiterhin wird die Distanz zwischen den beiden Benutzern variiert, so dass sie einmal physisches Feedback haben und einmal nur visuelles. Um Vergleichswerte zu bekommen, wird die Zeit der Durchführung und die Entfernung zwischen den Handmittelpunkten durch Kollisionserkennung in der Simulation der virtuellen Welt gemessen. Die Autoren kommen zu dem Schluss, dass die Form des Avatars eine Relevanz besitzt, die physische Distanz der Benutzer hingegen nicht.

2.1.2 MetaSpace II

MetaSpace II (MS2) ist ein weiteres Projekt, welches ein System zur Mehrbenutzer Interaktion in einer virtuellen Umgebung implementiert. Benutzer sollen sich in einer virtuellen Welt frei bewegen und miteinander oder mit bestimmten Objekten interagieren können. Um die Interaktionen natürlicher zu gestalten und den Grad der Immersion zu erhöhen wird eine Ganzkörperdarstellung verwendet und Feedback für den Nutzer bei der Interaktion mit Objekten durch passive Haptik generiert. Bei der passiven Haptik wird Feedback nicht technisch simuliert, sondern durch Gegenstände im realen Raum, welche ein Objekt in der virtuellen Welt repräsentieren, geschaffen.

Im MS2 wird zur Visualisierung von Räumen ein 3D-Scan der realen Umgebung durchgeführt und dieser durch verschiedene Bildbearbeitungsprozesse zu einer interaktiven, virtuellen Welt erweitert. Die Abbildung vom physischen auf den virtuellen Raum ermöglicht einerseits die ungebundene Bewegung der Benutzer, aber schränkt andererseits die Gestaltung der virtuellen Welt ein. Einen gewissen Ausgleich dazu bietet die freie Wahl von Texturen. Die Darstellung der fertigen Welt erfolgt dann durch HMDs.

Zur Umsetzung des Trackings werden im MS2 hauptsächlich Kinect Kameras genutzt. Diese liefern sowohl die Skelett Daten für die Darstellung der Benutzeravatare als auch die Daten der interaktiven Objekte, welche durch Mustererkennung identifiziert werden. Für die Rotation des Kopfes werden zusätzlich die Tracking Daten der HMDs benötigt. Alle Tracking Daten werden über einen Cloud Server kommuniziert.

Bei den ersten Versuchen im MS2 wurden zwei verschiedene Systemeinstellungen eingesetzt, die in Abb. 2.2 dargestellt sind. Die erste Einstellung sieht vor, dass jede Kinect Daten an einen Nutzer überträgt, welcher diese über den Cloud Server an andere Nutzer weiterleitet. Die zweite Einstellung hingegen besitzt nur eine Kinect, die ihre Daten direkt an den Cloud Server sendet (vgl. Sra und Schmandt (2015)).

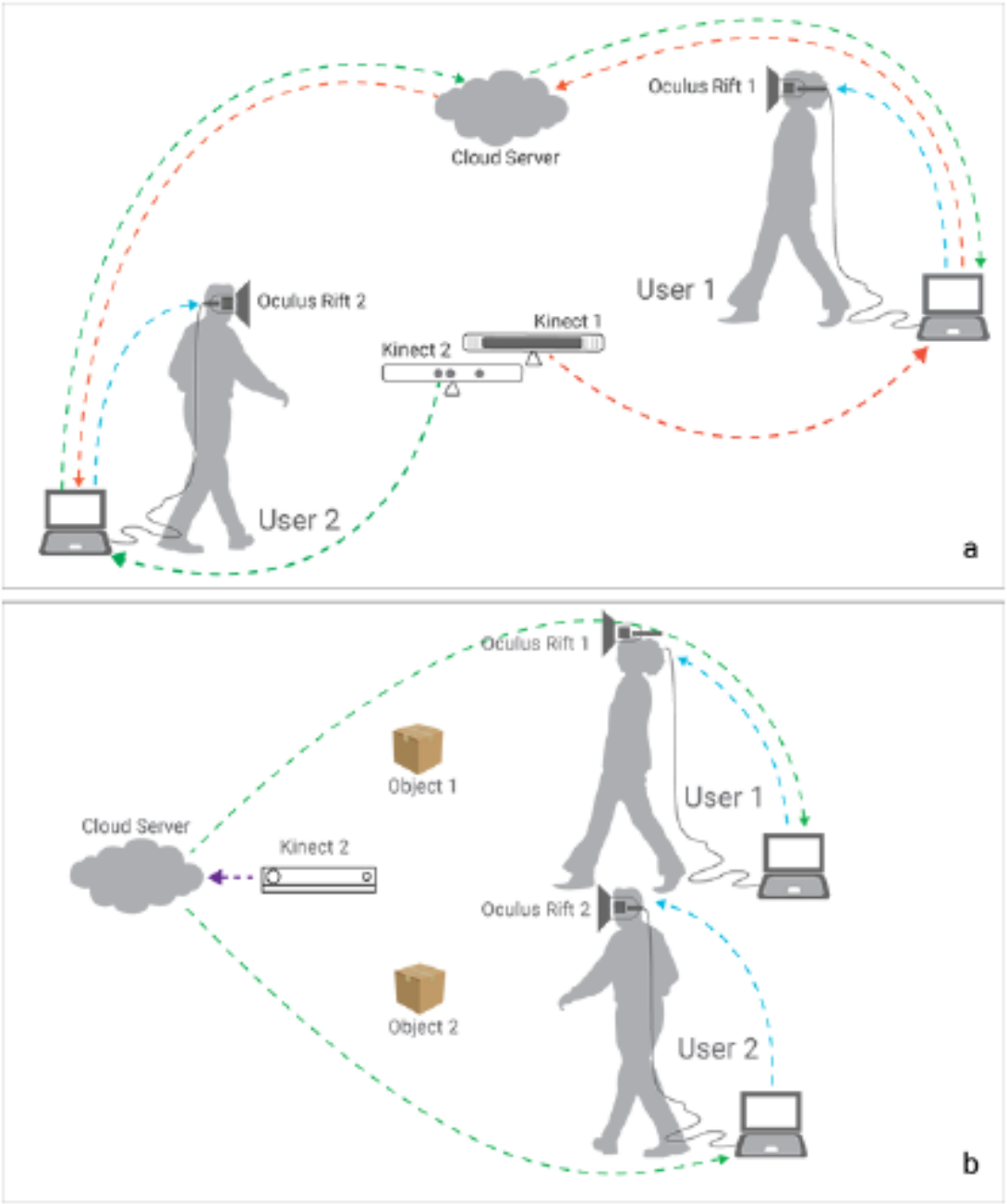

Abbildung 2.2: Systemeinstellungen im MetaSpace (Sra und Schmandt (2015))

2.2 Interaktion in Virtueller Realität

Die in Verwandte Arbeiten vorgestellten Ansätze haben beide das Ziel Interaktion in einem Mehrbenutzer VR-System zu ermöglichen, setzen dieses aber auf unterschiedliche Art und Weise um. Zur besseren Analyse beider Projekte, auch in Hinblick auf das Design des in dieser Arbeit entwickelten Systems, muss zuerst die Frage genauer erörtert werden was das Kernstück all dieser Projekte, nämlich die Interaktion in Virtueller Realität, ausmacht.

Da das Wesen der Interaktion in VR eng verknüpft ist mit dem Grundkonzept der VR, lässt sich eine Antwort auf diese Frage aus einem Definitionsansatz der VR ableiten. Dieser sieht VR als innovative Form der Mensch-Maschine Interaktion mit dem Ziel Mensch-Maschine-Schnittstellen zu erschaffen, die im Vergleich zu traditionellen Benutzungsschnittstellen ein besonders natürliches oder intuitives Interagieren mit der dreidimensional simulierten Umgebung ermöglichen.

Das wesentliche Merkmal der Interaktion in VR ist folglich die Natürlichkeit mit der ein Mensch mit der virtuellen Umgebung interagiert. Bevor weiter ausgeführt wird wie genau eine solche Interaktion aussieht, sollte erstmal ein Grundverständnis geschaffen werden wie eine virtuelle Umgebung überhaupt aufgebaut ist.

Zur Realisierung einer virtuellen Umgebung werden eine Virtuelle Welt und ein VR-System benötigt. Ein VR-System ist ein Computersystem, welches drei Aufgaben erfüllt, die Erfassung von Informationen und Interaktionen des Nutzers über Eingabegeräte, die Erzeugung von Reizen für den Nutzer über Ausgabegeräte sowie die Simulation der Virtuellen Welt (vgl. Dörner u. a. (2014)).

2.2.1 Virtuelle Welten

Unter dem Begriff „Virtuelle Welten“ versteht man die Inhalte von VR-Systemen. Virtuelle Welten setzen sich grundlegend aus 3D-Objekten und abstrakten, unsichtbaren Objekten zusammen. Dabei besitzen die 3D-Objekte ein dynamisches Verhalten und können auf Nutzereingaben reagieren, während die abstrakten Objekte z. B. Licht- und Klangquellen, virtuelle Kameras sowie Ersatzkörper für effiziente Kollisionsprüfungen, die Simulation der Darstellung der virtuellen Welt unterstützen.

Besonders wichtig bei Virtuellen Welten sind die Prinzipien der Echtzeitfähigkeit und der Interaktivität. Echtzeitfähigkeit bedeutet, dass die virtuellen Welten möglichst ohne Verzöge-

rungen dargestellt und aktualisiert werden. Interaktivität sagt aus, dass der Nutzer sich in der virtuellen Welt bewegen und das Verhalten der 3D-Objekte in der Virtuellen Welt beeinflussen kann (vgl. Dörner u. a. (2014)). Diese Prinzipien haben großen Einfluss auf das Erlebnis des Nutzers und sind eine Notwendigkeit für optimierte Interaktionen in Virtuellen Welten.

2.2.2 Interaktionen in Virtuellen Welten

Wenn ein menschlicher Nutzer mit einer Virtuellen Umgebung interagieren will, werden Informationen zwischen einem Menschen und einem Computer, auf dem das VR-System läuft, ausgetauscht. Diese Art von Interaktion wird Mensch-Computer-Interaktion (MCI, engl. Human-Computer Interaction, HCI) genannt.

Bestandteile des MCI sind das Design, die Evaluierung und die Realisierung interaktiver computerbasierter Systeme und darüber hinausgehender Phänomene. Ein Fokus liegt hierbei darauf die Schnittstellen anhand von Erkenntnissen der Informatik, aber auch anderer Gebiete wie der Psychologie und Kognitionswissenschaft, benutzergerecht zu gestalten. Um dieses zu gewährleisten wird bei der MCI viel Wert auf das Prinzip der Usability (deut. Gebrauchstauglichkeit) gelegt, welches nach ISO 9241 folgend definiert wurde:

> „Usability ist das Ausmaß, in dem ein Produkt durch bestimmte Nutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ (ISO92411.11 (2017)).

Der User soll durch das technische System bei der Erfüllung seiner Ziele so gut wie möglich unterstützt werden. Im Zuge dieser Anforderungen haben sich in den letzten Jahrzehnten hauptsächlich Graphische Benutzungsschnittstellen z. B. WIMP (Windows, Icons, Menus, Pointing)-Schnittstellen als ein Paradigma der MCI durchgesetzt. Diese wurden ursprünglich jedoch für 2D-Systeme und Aufgaben innerhalb dieser z. B. Dokumentenverwaltung entwickelt und sind bei der Interaktion mit 3D-Inhalten sehr ineffizient.

So kann die Verschiebung eines 3D-Objektes in der VR auf natürliche Weise durch Greifen und Verschieben realisiert werden, während bei typischen WIMP-Schnittstellen mehr Arbeitsschritte, Verschieben in der xy-Ebene, danach Verschiebung auf der z-Achse, anfallen. Durch die natürliche Weise der Interaktion in VR ist auch der Lernaufwand der Nutzer geringer, da diese ihr Vorwissen über Interaktionen mit physischen Objekten aus der realen Welt in die Virtuelle Welt übertragen können.

Weiterhin benötigen natürliche Interaktionen z. B. Gesten oder Sprachsteuerung keine direkten Eingabegeräte wie Maus und Tastatur. Dies erhöht den Freiheitsgrad des Nutzers.

VR bietet im Kontrast zu dieser natürlichen Weise der Interaktion mit 3D-Objekten auch eine übernatürliche Weise. So könnte der Nutzer sich Teleportieren oder weit entfernte Objekte manipulieren. Dieses bietet mehr Möglichkeiten und neue Funktionalitäten. Welche Art, natürlich oder übernatürlich, erstrebenswert ist, hängt dabei von dem Anwendungskontext ab.

Die grundlegenden Interaktionen in Virtuellen Welten lassen sich in die Aufgabengebiete Selektion, Manipulation, Navigation und Systemsteuerung unterteilen. Bei der Selektion bestimmt der Nutzer einen Punkt, eine Fläche oder ein Volumen in der Virtuellen Welt oder wählt eine für sein Ziel relevante Teilmenge dieser Welt aus und kann die Eigenschaften dieser Objekte im Anschluss durch Manipulation verändern. Als Eigenschaften gelten hierbei die charakterisierenden Objektparameter des Objekts z. B. dessen Ort, Orientierung im Raum, Größe, Form, Geschwindigkeit oder Erscheinung. Da Selektion und Manipulation in einem Zusammenhang stehen, sollten diese bei dem Entwurf einer VR-Interaktion nicht separat behandelt, sondern aufeinander abgestimmt, werden.

Eine Voraussetzung für die Manipulation ist die Navigation, da der Nutzer sonst auf natürliche Weise nicht in der Lage wäre die 3D-Objekte der Virtuellen Umgebung zu erreichen. Das Aufgabengebiet der Navigation wird grundsätzlich in die Teilbereiche Wegfindung und Bewegungskontrolle aufgeteilt. Ziel der Wegfindung ist es eine kognitive Karte der Virtuellen Welt zu entwickeln. Die Bewegungskontrolle hingegen erlaubt dem Nutzer den Virtuellen Kameraausschnitt zu verschieben oder drehen, also sich durch die Virtuelle Welt zu bewegen. Interaktionen mit dem VR-System selbst um z. B. eine neue Szene zu laden, gehören zur Systemsteuerung (vgl. Dörner u. a. (2014)).

2.3 Tracking

Bei dem virtuellen Handschlag Szenario interagieren Nutzer durch ihre Bewegung mit dem VR-System, da diese auf die Avatare der Nutzer in der virtuellen Welt übertragen wird. Folglich werden Eingabegeräte benötigt, welche die Position der Nutzer im realen Raum bestimmen und diese der virtuellen Welt über eine Schnittstelle zur Verfügung stellen. Das Verfahren der Positionsbestimmung wird folgend Tracking genannt.

Hierbei gelten verschiedene Ansprüche an die für das Tracking genutzten Eingabegeräte und ihre Schnittstellen. So ist für die Ganzkörperavatare ein Tracking des gesamten Körpers der Nutzer erforderlich. Dabei müssen Position und Rotation des Kopfes erfasst werden, um den Sichtbereich der Nutzer innerhalb der virtuellen Welt anzupassen. Außerdem sollte für die Durchführung des Handschlags ein gewisses Maß an Präzision bei der Position der Hände beider Nutzer erreicht werden. Um all diese Anforderungen zu erfüllen, werden mehrere Eingabegeräte und Services benötigt, die im Folgenden vorgestellt werden.

2.3.1 Kinect

Für das Ganzkörpertracking im System bietet sich die Kinect an. Die Kinect ist eine von Microsoft entwickelte Kamera, die Farb-, Tiefen- und Infrarotbilder erzeugt. Außerdem kann die Kinect mehrere Personen anhand ihrer Skelette verfolgen (vgl. Microsoft1 (2017)). Ein Skelett besteht hierbei aus 25 Joints, welche Gelenke und Knochen einer Person repräsentieren (vgl. Abb. 2.3).

Diese Skelettverfolgung bildet eine gute Grundlage für die Simulation der Bewegung von Personen in einer virtuellen Welt und ist deswegen sehr verbreitet im VR-Kontext. Die Kinect bestimmt jedoch keine Rotation der Joints und kann deswegen den Sichtbereich der Nutzer nicht korrekt darstellen. Auch erschwert die fehlende Fingerunterscheidung die Durchführung eines Handschlags. Hierfür werden andere Eingabegeräte benötigt, welche im weiteren Verlauf des Tracking Kapitels erläutert werden.

Weiterhin beziehen sich die Positionen der Skelettjoints auf das Koordinatensystem der Kinect, welches die Kinect selbst als Ursprung hat. Für das Koordinatensystem des virtuellen Raumes ist jedoch ein Ursprung auf Bodenhöhe in der Raummitte natürlicher und effektiver bei der Einbindung anderer Tracking Daten sowie bei der Erstellung der virtuellen Welt. Ein solches Koordinatensystem findet sich bei dem Advanced-Realtime-Tracking (ART).

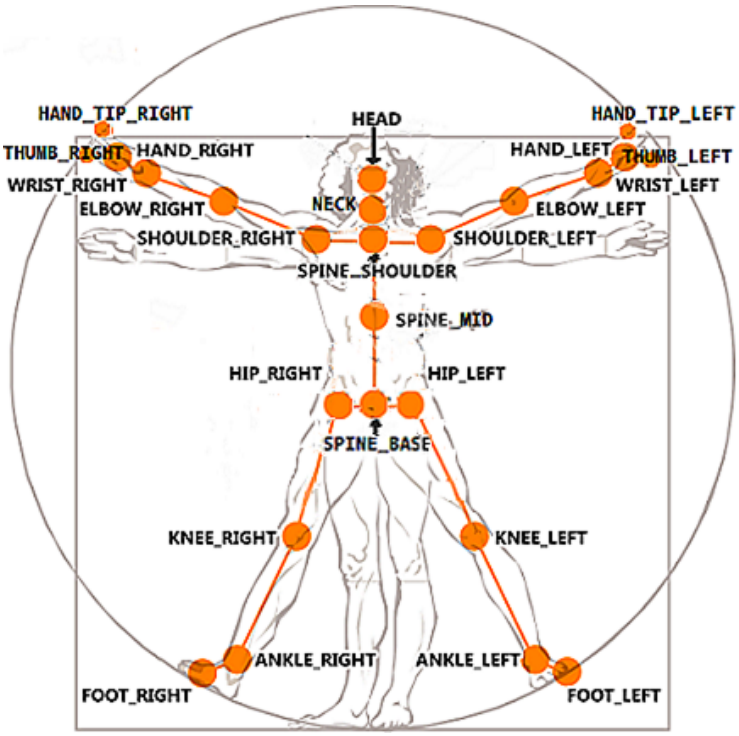

Abbildung 2.3: Kinect - Skeleton Joints (Microsoft2 (2017))

2.3.2 ART

Das ART ist ein Tracking-System, welches mit multiplen Infrarotkameras die Position und Rotation von Markern bestimmt. Hierbei soll das System maximale Abweichungen von 0.4 mm in der Rotation und 0.12 Grad in der Rotation bei einer Frequenz von 60 Hertz aufweisen (vgl. ART (2017)).

Eine Skelettverfolgung ist im ART im Gegensatz zur Kinect nicht integriert. Um dieses umzusetzen, müssten die Nutzer mit Markern, ein Marker für jede gewünschte Knochen- oder Gelenkposition, ausgestattet werden. Diese Lösung für Körpertracking ist einerseits aufwendiger bei der Berechnung und anderseits weniger immersiv für den Nutzer als bei der Kinect.

Deswegen bietet es sich an, die Skelettdaten der Kinect zu benutzen und diese auf das Koordinatensystem des ARTs zu transformieren. Für diese Koordinatentransformation kann eine Plattform für Personentracking in Anspruch genommen werden.

2.3.3 Plattform für Personentracking

Die Plattform für Personentracking basiert auf Sensorfusion und wurde in Kirchner (2017) entworfen und implementiert. Die Plattform bietet Entwicklern die Möglichkeit, Sensordaten verschiedener Sensoren über eine einheitliche Schnittstelle zu erfragen. Dabei kann die Plattform außerdem die Sensordaten im Zuge einer Koordinatentransformation vorverarbeiten oder diesen Personen zuordnen.

Hierbei wurde die Plattform speziell für Anwendungen in VR/AR Umgebungen konzipiert und parallel zu dem System dieser Arbeit entwickelt. Deshalb werden sowohl die Kinect als auch das ART von der Plattform unterstützt und ein Fokus auf Präzision und das Vermeiden von Latenzen gelegt.

2.3.4 Leap Motion

Ein Mangel, der auch bei den transformierten Skeletten bestehen bleibt, ist das die einzelnen Finger nicht getrackt und deswegen nicht dynamisch modelliert werden können. Deswegen wird in diesem System zusätzlich das Leap Motion Hand Tracking benutzt.

Die Leap Motion kann mehrere Hände in einem halbkugelförmigen Bereich mit einem Radius von einem Meter ausgehend von dem Sensor tracken. Hierbei wird für jede Hand die Position von 20 verschiedenen Finger Joints bestimmt. Zusätzlich dazu werden sieben weitere Joints der Hand und der Arme berechnet. Die Abweichung und die Latenz der Kamera betragen im Schnitt 39 Hertz und 0.5 mm (Guna u. a. (2014)).

Wegen der geringen Reichweite der Leap Motion ist es notwendig zwei Sensoren, einen für jeden Nutzer, zu verwenden. Die Leap Motion wird dabei an dem HMD des Nutzers befestigt.

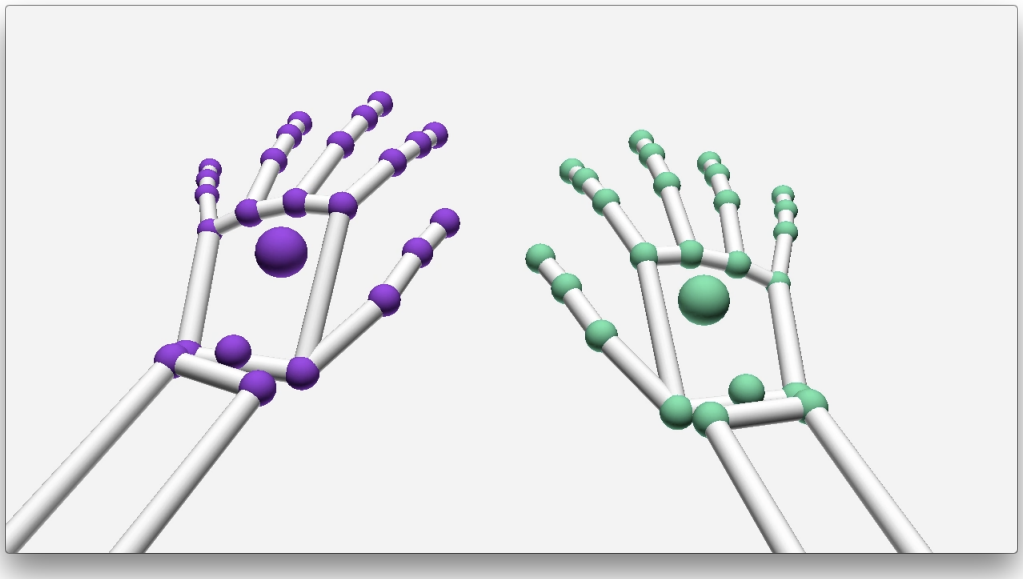

Abbildung 2.4: Leap - Hand Joints (VUO (2016))

2.3.5 Lighthouse

Das Kapitel Tracking abschließend wird die Bestimmung von Position und Rotation des HMDs erläutert. Hierfür wird das Lighthouse Tracking System genutzt. Das Lighthouse Tracking System kombiniert Infrarot-LEDs, Laser und Sensoren, um Position und Rotation kompatibler Controller zu berechnen. Ein kompatibler Controller für das Lighthouse ist die HTC Vive, die in diesem System als HMD zur Darstellung der virtuellen Welt genutzt wird (vgl. VRJump (2017)).

2.4 Koordinatentransformation

Geometrische Figuren werden durch Koordinaten in einem Koordinatensystem beschrieben. Dabei besteht die Möglichkeit die gleiche Figur durch verschiedene Koordinaten in unterschiedlichen Koordinatensystem darzustellen. Um die Koordinaten eines Punktes in einem Koordinatensystem in Koordinaten eines anderen Koordinatensystems zu überführen, wird eine Koordinatentransformation durchgeführt. Eine Koordinatentransformation beschreibt also die Beziehung zweier Koordinatensysteme.

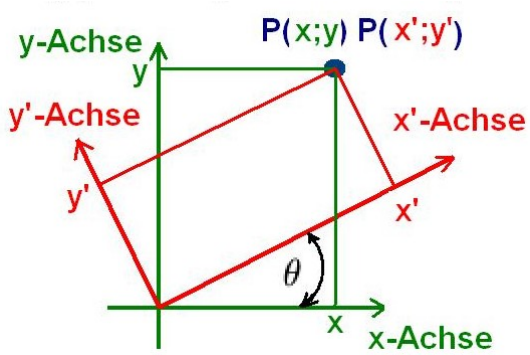

Abbildung 2.5: Punkt P in zwei verschiedenen Koordinatensystemen (TFWiki (2017))

In einem VR-System gibt es verschiedene Orte an denen eine Koordinatentransformation vorgenommen werden muss. So besitzen verschiedene Eingabegeräte des Trackings eigene Koordinatensysteme und liefern folglich bei der Positionsbestimmung für das gleiche Objekt unterschiedliche Koordinaten. Sollen beide Trackingergebnisse verwendet werden, müssen diese im Zuge der Sensorfusion in ein Koordinatensystem überführt werden. Die in Kapitel 2.3.3 aufgeführte Koordinatentransformation zwischen Kinect und ART bildet hierfür ein Beispiel.

Besonders schwierig gestaltet sich der Prozess der Sensorfusion wenn eines der Eingabegeräte ein lokales Koordinatensystem besitzt, also im Gegensatz zu einem globalen Koordinatensystem nicht einen festen Ursprungsort hat, sondern seine eigene Position im Rahmen des beobachteten Raumes und damit auch seinen Sichtbereich ändert.

Da die virtuelle Welt ebenfalls ein eigenes Koordinatensystem besitzt, muss in einem VR-Systen außerdem noch eine Koordinatentransformation zwischen den Tracking Systemen und der virtuellen Welt stattfinden.

Zur Umsetzung einer Koordinatentransformation existieren verschiedene Ansätze, die abhängig von den Anforderungen der Anwendung eingesetzt werden z. B. lineare Transformationen, affine Transformationen oder euklidische Transformationen.

Eine lineare Transformation setzt voraus, dass beide Koordinatensysteme den gleichen Ursprung haben und bildet die neuen Koordinaten aus linearen Funktionen der ursprünglichen. Affine Transformationen sind eine Teilmenge der linearen Transformationen und ergänzen diese um eine Translation. Außerdem dürfen bei einer affinen Transformation die Ursprünge der Koordinatensysteme unterschiedlich sein. Die euklidischen Transformationen sind wiederum eine Teilmenge der affinen Transformationen und besitzen die Eigenschaften abstands- und winkelerhaltend zu sein (vgl. Aichholzer und Jüttler (2013)).

Durch Koordinatentransformationen in einem VR-System entstehen zusätzliche Verzögerungen zwischen dem Erhalt von Koordinaten und der Verwendung dieser. Dieses hat Einfluss auf die Latenz des Systems.

2.5 Latenz

Unter Latenz versteht man die Zeit, die ein System benötigt, um auf eine Eingabe zu reagieren. Abhängig von der Latenz variiert also der zeitliche Abstand zwischen einer Handlung und der Wahrnehmung dieser. Dieses hat in dem Kontext eines VR-Systems einen großen Einfluss auf den Grad der Immersion des Nutzers.

Sind die Latenzen zu groß, sinkt der Grad der Immersion und es besteht die Möglichkeit, dass negative Auswirkungen auf das Wohlbefinden des Nutzers wie z. B. Übelkeit auftreten. Aus diesem Zusammenhang leitet sich der Anspruch auf Echtzeitfähigkeit des VR-Systems ab. Resultate einer Eingabe sollen zuverlässig in vorhersagbaren Zeiträumen geliefert werden und die Latenz des Systems unter der menschlichen Wahrnehmungsschwelle liegen (vgl. Meehan u. a. (2003)).

Um dieses zu gewährleisten wird bei der Verwendung von Head-Mounted Displays (HMDs) grundsätzlich eine Latenz unter 50 ms, also eine Aktualisierungsfrequenz von 20 Hz empfohlen (vgl. Dörner u. a. (2014)). Michael Abrash, der sich in seinem Artikel (Abrash (2012)) mit Latenzen in VR auseinandersetzt, verschärft diese Anforderungen nochmal. So treten laut Abrash bei einer Latenz von 50 ms immer noch Fehler bzw. Verzögerungen in der Darstellung auf, die ein Nutzer wahrnehmen kann. Erst bei einer Latenz von 20 ms oder niedriger ist das System echtzeitfähig. Solche Latenzen sind bei der Leistung aktueller HMDs auch mit Techniken und Tricks zur Optimierung des Systems kaum zu erreichen. Auch die Komplexität der Szenen zu verringern, die dargestellt werden sollen, schafft nur bedingt Abhilfe.

Latenzen entstehen an verschiedenen Orten in einem VR-System. Eine Übersicht dazu bietet Abbildung 2.6. Das Verhalten des Nutzers wird über verschiedene Eingabegeräte erfasst. Jedes dieser Geräte hat eine eigene Tracking-Latenz, also eine Verzögerung zwischen dem Zeitpunkt der Handlung des Nutzers und der Bereitstellung der Informationen dieser als Ereignis an die Weltsimulation. Da die Latenzen der einzelnen Sensoren sehr unterschiedlich sein können, müssen diese durch eine Sensorfusion ausgeglichen werden. Hierbei ist die höchste Latenz ausschlaggebend für die Gesamtlatenz des Trackings.

Weiterhin tritt bei jedem Transport der Informationen eine Transportlatenz auf. Sobald die Ereignisse die Weltsimulation erreichen, werden diese repräsentiert und die Interaktionseffekte mit Hilfe des verwendeten VR Frameworks simuliert. Die für die Simulation notwendigen Be-

rechnungen und die eventuelle Wartezeit auf andere Systemkomponenten sind verantwortlich für die hier vorhandene Simulationslatenz. Anschließend muss die Welt, deren Zustand sich durch die Simulation verändert hat, für die jeweiligen Ausgabegeräte aufbereitet (Generierungslatenz), zu den Ausgabegeräten transportiert und dort dargestellt werden (Darstellungslatenz). Die Gesamtlatenz des Systems bildet sich aus der Summe dieser Latenzen und wird auch Ende-zu-Ende-Latenz genannt.

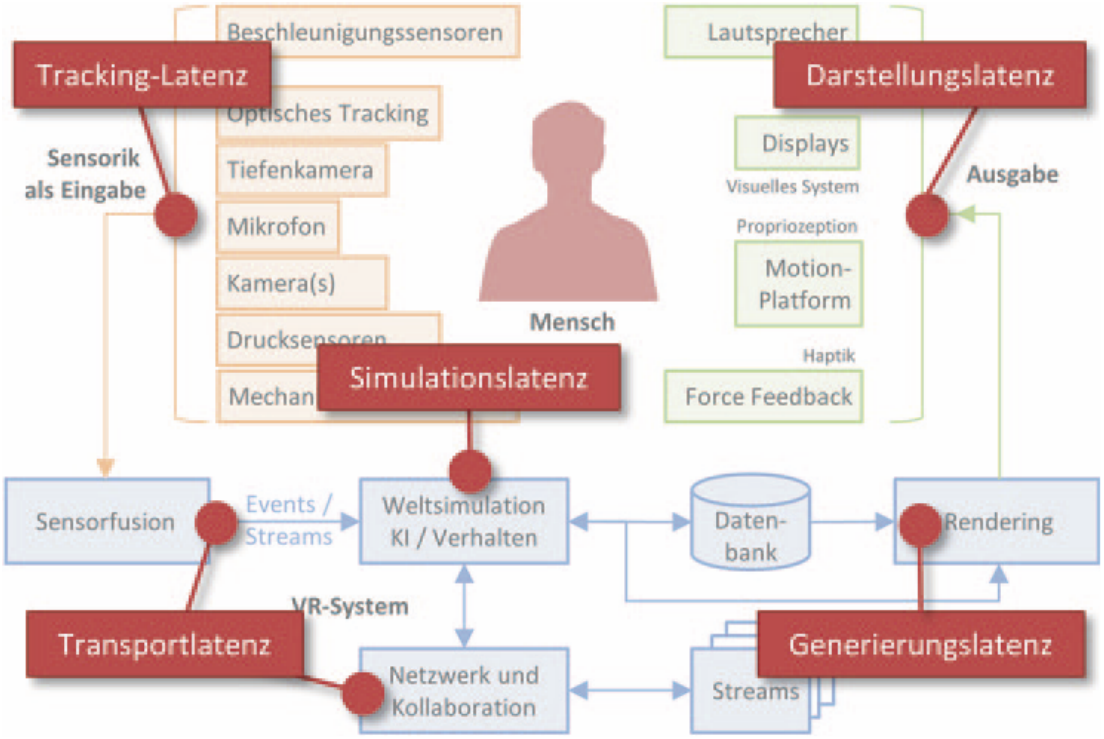

Abbildung 2.6: Latenzen in einem VR-System (Dörner u. a. (2014))

2.6 CSTI

Aus den Verwandten Arbeiten lässt sich ableiten, dass man für die Umsetzung eines Mehrbenutzer VR-Systems einen geeigneten Raum mit einer umfangreichen Infrastruktur benötigt. Dieses Projekt wird deswegen in der Laborumgebung des Creative Space for Technical Innovations (CSTI) durchgeführt.

Das CSTI ist ein interdisziplinäres Labor mit Fokus auf das Themengebiet Human-Computer Interaction und Smart Environments an der HAW Hamburg. Studenten verschiedener Studienrichtungen erhalten Raum, Equipment und Betreuung, um ihre Ideen als Projekte umzusetzen Besonderen Wert wird dabei auf agiles Prototyping und die Verbindung zur regionalen Wirtschaft gelegt (vgl. CSTI (2017)).

Zur Verfügung stehen neben mehreren Rechnern unter anderem diverse Tracking Systeme, die größtenteils in einer Traverse untergebracht sind sowie verschiedene Ausgabegeräte z.B. VR/AR Brillen. Weiterhin wird eine Netzwerkinfrastruktur zur Kommunikation zwischen den Akteuren bereitgestellt. Der Raumplan des CSTI (Stand 2017) wird in Abbildung 2.7 dargestellt.

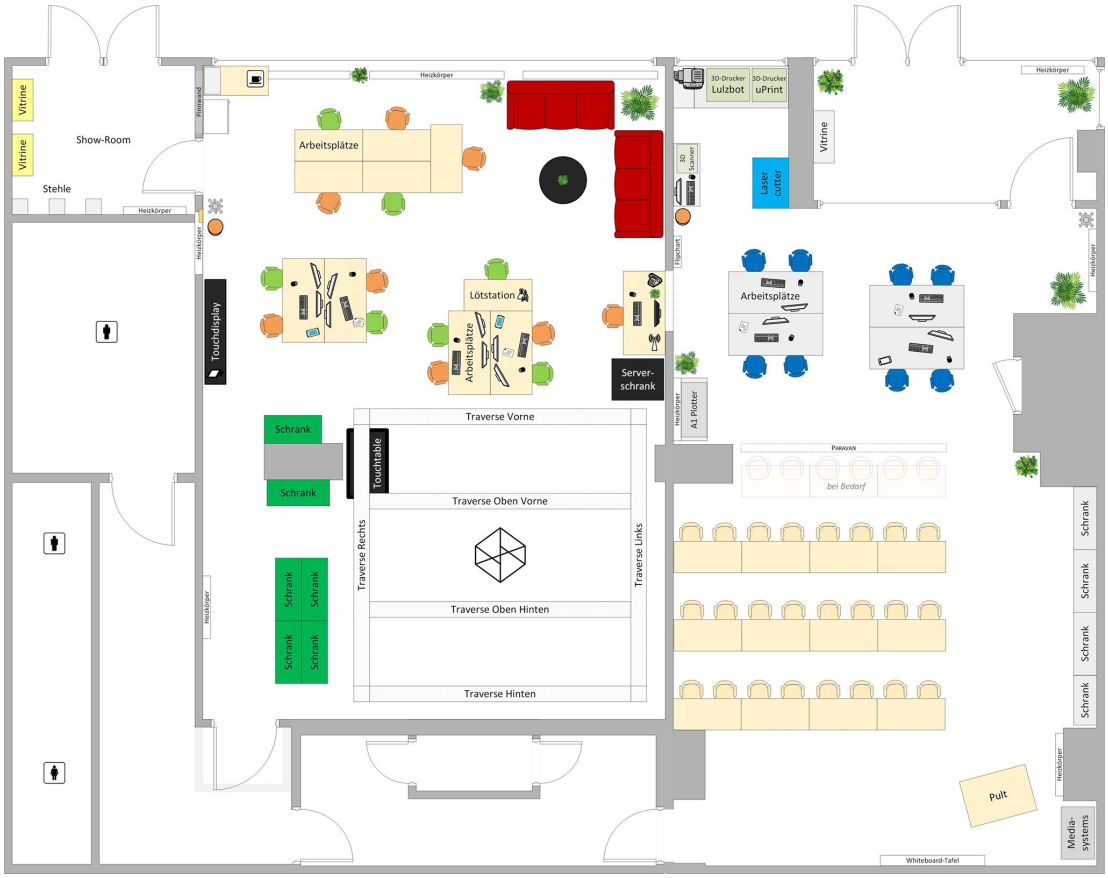

Abbildung 2.7: Raumplan des CSTI (Jeworutzki (2017))

2.7 Kommunikation

Die einzelnen Komponenten eines VR-Systems werden meist auf mehreren Rechnern verteilt ausgeführt. Dies erfordert die Verwendung einer Kommunikationsinfrastruktur.

2.7.1 Middleware

Im CSTI wird zur Kommunikation zwischen einzelnen Akteuren eine in Eichler (2014) entwickelte Middleware verwendet. Diese basiert auf einer Publish/Subscribe Architektur. Akteure können verschiedene Gruppen abonnieren oder in ihnen Nachrichten veröffentlichen, um miteinander zu kommunizieren. Weiterhin wurde im Zusammenhang mit der Middleware ein Framework zur Entwicklerunterstützung entworfen. Dieses vereinfacht die Anbindung von Akteuren an die Middleware und übernimmt die Serialisierung von Nachrichten. Durch die so gewonnene lose Kopplung ist es ebenfalls leichter möglich Komponenten wiederzuverwenden.

Die Middleware eignet sich gut für den Einsatz in einem VR-System, da beim Design der Middleware Anforderungen aus dem VR-Kontext berücksichtigt wurden. So produziert sie einerseits kaum Latenzen und ist andererseits in der Lage eine große Menge an Daten z. B. Trackingdaten zu verarbeiten (vgl. Eichler u. a. (2017)).

2.8 Anforderungen

Aus der Analyse ergeben sich verschiedene Anforderungen an das geplante Mehrbenutzer VR-System. Hierbei wird zwischen funktionalen und nicht-funktionalen Anforderungen differenziert. Funktionale Anforderungen beschreiben gewünschte Funktionalitäten des Systems, während nicht-funktionale Anforderungen ausdrücken in welcher Qualität diese Funktionalitäten erbracht werden sollen.

2.8.1 Funktionale Anforderungen

Für die erfolgreiche Durchführung des in Kapitel 1.1 beschriebenen Szenarios muss das System folgende Funktionalitäten bereitstellen:

1. *Tracking der Nutzer*

   Das System muss Position und Rotation beider Personen bestimmen. Dabei müssen die gesamten Körper getrackt werden, um die Benutzung der Ganzkörperavatare zu ermöglichen. Die gesammelten Daten müssen außerdem an die virtuelle Welt gesendet werden, um diese zu aktualisieren und die Bewegung der Personen sichtbar zu machen.

2. *Visualisierung der virtuellen Welt*

   Der virtuelle Raum und die Avatare der Nutzer müssen erzeugt und den Nutzern über ein passendes Medium präsentiert werden.

3. *Interaktion in der virtuellen Welt*

   Um die Interaktivität der virtuellen Welt zu gewährleisten, müssen Interaktionen registriert und verarbeitet werden. In diesem Szenario besteht die Interaktion aus der Bewegung der Nutzer, die durch Tracking erfasst und durch Tracking Daten repräsentiert wird. Diese Daten müssen entgegengenommen werden und durch die Aktualisierung der Avatare verwertet werden.

4. *Handschlag Erkennung*

   Zur Beurteilung, ob das Szenario erfolgreich war, muss das System eine Funktionalität anbieten, die bestimmt, ob der Handschlag gelungen ist.

2.8.2 Nicht-funktionale Anforderungen

Im Kontext des Szenarios gelten außerdem folgende Qualitätsansprüche an das System:

1. *Präzision*

   Damit der Handschlag erfolgreich verläuft, sollte die relative Position der beiden Nutzer in der virtuellen Welt möglichst genau der relativen Position in der realen Welt entsprechen. Die Differenz darf maximal einige Zentimeter groß sein, da sonst die Nutzer Gefahr laufen die Hand des Anderen zu verfehlen. Voraussetzung dafür sind präzise Geräte zur Positionsbestimmung und eine präzise Umrechnung im Zuge notwendiger Koordinatentransformationen.

2. *Latenz*

   Ebenfalls großen Einfluss auf den Erfolg des Handschlags hat die Höhe der Latenz im System. In Kapitel 2.5 wurden bereits Auswirkungen von zu hohen Latenzen in einem VR-System wie ein sinkender Grad an Immersion oder das Auftreten von Übelkeit beim Nutzer beschrieben. Weiterhin beeinflussen Latenzen die Präzision, da Verzögerungen bei der Aktualisierung der Avatare die Abweichung in der Position der Nutzer in realer und virtueller Welt erhöhen. Die Latenz des Gesamtsystem muss deswegen minimiert werden.

3. *Wiederverwendbarkeit*

   Projekte im CSTI werden grundsätzlich mit dem Anspruch entwickelt, dass diese und ihre Komponenten in anderen Projekten wiederverwendet werden können. Das gilt auch für das System dieser Arbeit, welches als Basis für weitere Mehrbenutzer VR Projekte im CSTI dienen soll.

3 Design

In diesem Kapitel wird der Entwurf des Mehrbenutzer VR-Systems, welches die Grundlage für den virtuellen Handschlag bildet, vorgestellt. Eine erste Abstraktion des Systems basierend auf der Anforderungsanalyse im Kapitel 2.8 findet sich in Abbildung 3.1. Aufbauend auf diesem Modell werden im Folgenden die verschiedenen Inhalte und Abläufe des Systems genauer erörtert.

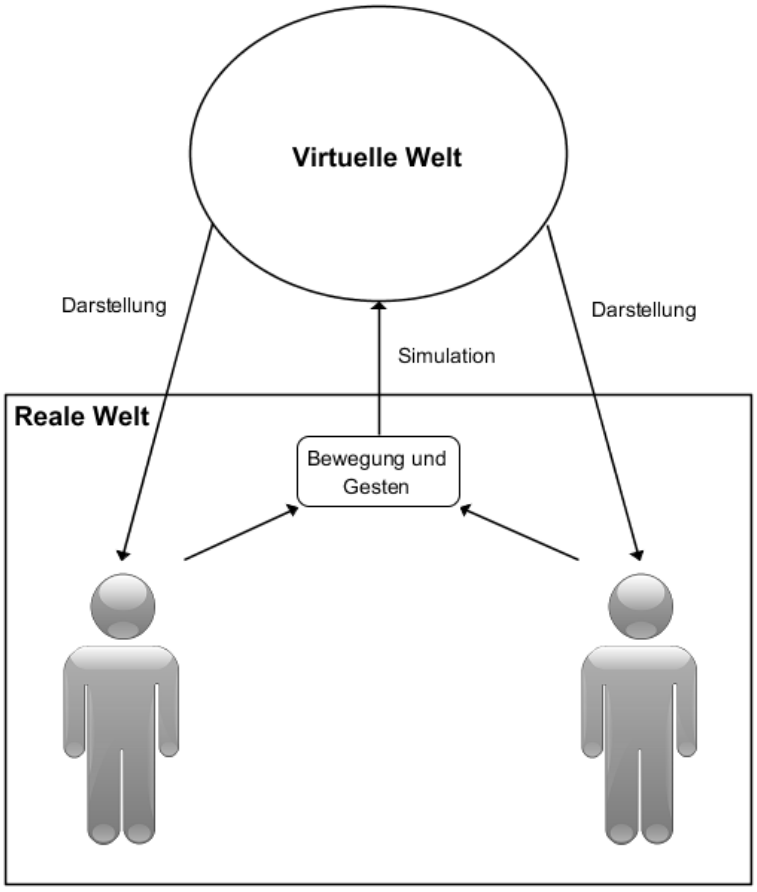

Abbildung 3.1: Übersicht des Gesamtsystems

3.1 Kommunikation

Aus der Analyse ergeben sich unterschiedliche Anforderungen an die in einem Mehrbenutzer VR-System verwendete Kommunikationsinfrastruktur. Diese sollte möglichst latenzarm sein, eine große Menge an Daten z. B. Trackingdaten verarbeiten können und flexibel sein, um weitere Eingabegeräte oder Nutzer integrieren zu können. In diesem System wird deswegen die in Kapitel 2.7.1 vorgestellte Middleware genutzt.

Die Flexibilität und Erweiterbarkeit der Middleware sind dabei nicht nur durch die Publish/Subscribe Architektur gegeben, sondern auch dadurch bedingt, dass die meisten Sensoren und Projekte in der Laborumgebung des CSTI bereits über eine Anbindung zur Middleware verfügen.

Weiterhin bietet das Framework der Middleware Libraries zur leichteren Integration in die Programmierumgebung in den Programmiersprachen Java, Scala, C++, Javascript und durch ein Plugin der C++ Version auch in C# an. Die Libraries ermöglichen Funktionalitäten der Middleware wie gewöhnliche Methoden der jeweiligen Programmiersprache zu verwenden und erlauben damit ebenfalls die Überprüfung dieser durch die Umgebung (vgl. Eichler u. a. (2017)).

Alle Komponenten in diesem System, welche über die Middleware kommunizieren, werden deswegen mit einer der unterstützen Programmierungsprachen umgesetzt. Außerdem muss jede dieser Komponenten in der Lage sein, das Dateiformat JSON zu parsen, welches das Standard Format der übermittelten Nachrichten in der Middleware ist. Dazu bietet sich der API-Generator aus dem Framework der Middleware an. Dieser übersetzt die gewünschten Nachrichtenformate, welche in einer Domain Specific Language spezifiziert werden, in die Sprachen Java, Scala, C++ und Javascript.

3.2 Komponenten des Systems

In Abbildung 3.2 werden die Komponenten des Systems und ihre Beziehungen in einem Komponentendiagramm dargestellt. Hierbei wurden die Komponenten der virtuellen Welt für beide Nutzer jeweils in einer Unity Komponente zusammengefasst, um die Komplexität des Diagramms zu verringern. Des Weiteren wurden die einzelnen Verbindungen zur Middleware missachtet. Betroffen davon ist die Kommunikation zwischen den Unity Komponenten untereinander und die gesamte Kommunikation der Plattform.

Für das Tracking der Nutzer werden die in der Analyse vorgestellten Tracking Systeme und die Plattform benutzt. Dabei sind die HTC Vive und die Leap Motion einmal für jeden Nutzer vorhanden. Der Ablauf des Trackings und die Schnittstellen der Tracking Systeme werden in Kapitel 3.3 genauer ausgeführt.

Die Tracking Daten werden im System zu den Unity Komponenten der Nutzer gesendet. Diese sind für die Visualisierung und Simulation der virtuellen Welt zuständig. Weitere Details dazu finden sich in Kapitel 3.4. Analog zum Tracking werden dort außerdem Ablauf und Schnittstellen der Komponente präsentiert.

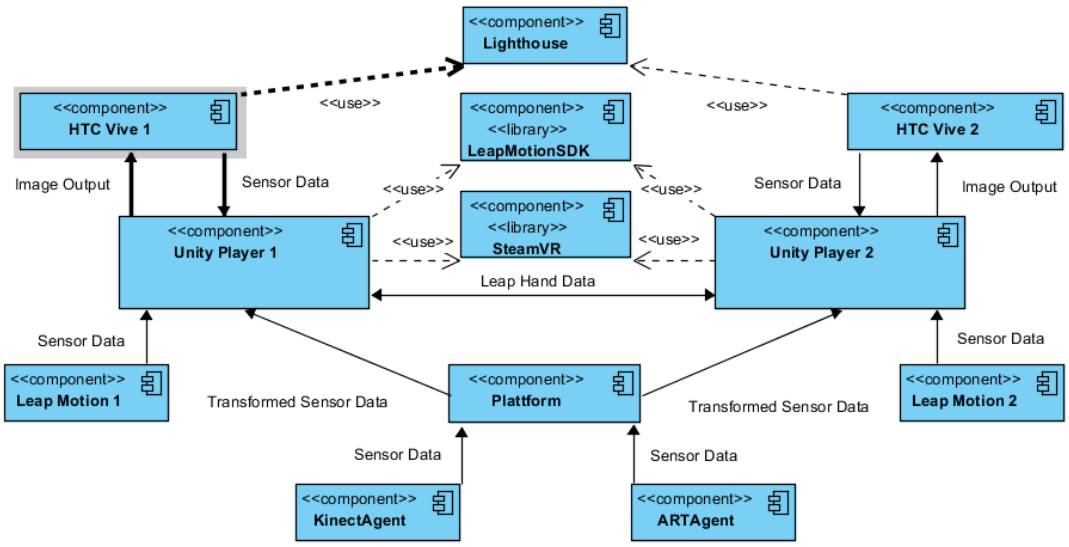

Abbildung 3.2: Komponentendiagramm des Gesamtsystems

3.3 Tracking

Der reale Raum, in dem sich die Nutzer während der Ausführung des Systems bewegen, befindet sich innerhalb des Traversen Systems des CSTI. Dies ist durch die Installation der verwendeten Tracking Systeme in der Traverse bedingt. Hierbei ist anzumerken, dass sowohl beim ART als auch beim Lighthouse der Ursprung des Koordinatensystems auf Bodenhöhe in der Raummitte ausgerichtet wurde.

3.3.1 Ablauf des Trackings

Das Tracking im System folgt dem im Sequenzdiagramm 3.3 visualisierten Ablauf. Zuerst wird eine Kalibration von ART und Kinect über die Plattform vorgenommen, um die Koordinatentransformation der Kinect Skelette in das Koordinatensystem des ART zu ermöglichen. Dabei bewegt sich eine Person mit einem ART Marker in der Hand im Trackingbereich. ART und Kinect erfassen diese Szene und senden an 20 Zeitpunkten ihre Daten über die Middleware an die Plattform. Diese berechnet anhand der korrespondierenden Punkte die Transformation und speichert diese.

Nach der Kalibration und der Berechnung der Transformation startet das Tracking der Nutzer. Hierbei wird das ART nicht mehr benötigt und deswegen ausgeschaltet, um die Leistung der anderen Tracking Systeme nicht mit Interferenzen im Infrarotbereich zu belasten. Die Sequenz zum Tracking der Nutzer wiederholt sich dann solange bis das System beendet wird.

In der Sequenz werden die von der Kinect erkannten Skelette über die Middleware an die Plattform gesendet. Diese führt die gespeicherte Transformation durch und benutzt dann die Middleware, um die transformierten Skelette an die virtuelle Welt beider Nutzer zu schicken. Weiterhin übermittelt die HTC Vive ihre Position und Rotation und die Leap Motion ihre bestimmten Hände an die virtuelle Welt des zugehörigen Nutzer.

Bedingt durch die kurze Reichweite der Leap Motion müssen anschließend beide Nutzer ihre Leap Motion Daten austauschen, um die Hände des jeweils anderen korrekt darstellen zu können. Diese Kommunikation findet über die virtuelle Welt statt und wird in Kapitel 3.4.2 beschrieben.

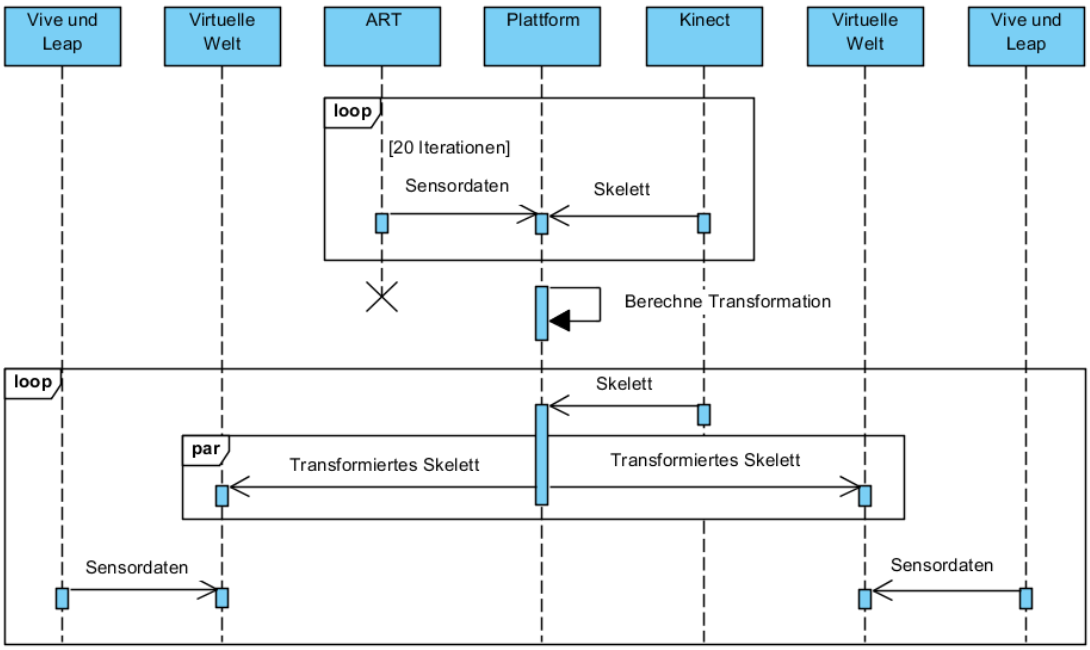

Abbildung 3.3: Ablauf des Trackings

3.3.2 Schnittstellen des Trackings

Da die eingesetzten Tracking Systeme und die Plattform ihre Daten auf sehr unterschiedliche Weise empfangen und senden, werden im Folgenden die Schnittstellen der einzelnen Komponenten des Trackings erläutert.

Kinect

Im CSTI wurde bereits mehrfach mit Kinect Kameras gearbeitet, weswegen eine existierende Middleware Anbindung wiederverwendet werden konnte. Hierbei wird ein Kinect Agent auf dem Rechner gestartet, an dem die Kinect über USB angeschlossen ist. Der Agent verpackt dann jedes getrackte Skelett einzeln in ein JSON und published es in einer definierten Gruppe an eine Middleware Instanz. Für die JSON-Serialisierung wird die Skeleton-API benutzt.

```
msg Vector3D(x: Double, y: Double, z: Double)
msg Joint(position: Vector3D, direction: Vector3D, confidence:
    Double)
msg Skeleton(id: Int, joints: Map[String, Joint])
```

Listing 3.1: Die Skeleton-API

ART

Ähnlich wie bei der Kinect, wurde auch das ART schon in einigen Projekten im CSTI genutzt und im Zuge dessen ein Agent für die Anbindung an die Middleware geschrieben. Dieser published die Position und Rotation aller in einem Frame gefundenen Marker in einem JSON an eine Gruppe der laufenden Middleware Instanz. Das Nachrichtenformat ist hier in der ART-API definiert.

```
msg Vector3D(x: Double, y: Double, z: Double)
msg Matrix3D(b0: Double, b3: Double, b6: Double,
             b1: Double, b4: Double, b7: Double,
             b2: Double, b5: Double, b8: Double)

msg ArtFrameId(id: Int)
msg ArtBodyId(id: Int)
msg ArtTimestamp(ts: Double)

msg Art6DStandardBody(id: ArtBodyId, quality: Double, position:
```

```
11 Vector3D, rotation: Vector3D, matrix: Matrix3D) extends ArtBody
12 msg ArtFrame(id: ArtFrameId, time: ArtTimestamp, bodies:
       Seq[ArtBody])
```

Listing 3.2: Die ART-API

Plattform

Durch eine Subscription der entsprechenden Gruppen erhält die Plattform die Daten von den ART und Kinect Agents. Um die eingehenden Nachrichten zu deserialisieren, werden diese mit der jeweiligen API geparsed. Weiterhin wird für die Serialisierung der transformierten Skelette erneut die Skeleton API verwendet. Die daraus resultierenden JSONs werden dann in einer separaten Gruppe gepublished.

Leap Motion und HTC Vive

Die Leap Motion und die HTC Vive eines Nutzers sind direkt mit dem Rechner verbunden, auf dem für diesen Nutzer die virtuelle Welt simuliert wird. Zugriff auf die Daten der Sensoren ist durch die Integration des Leap Motion SDKs für die Leap Motion und SteamVR für die HTC Vive möglich.

Das LeapSDK verwertet die Daten der Kamera, indem es für jedes Frame der Leap Motion ein Frame Objekt erstellt. Dieses hält eine Liste an Hand Objekten für die in dem Frame erkannten Hände. Ein Hand Objekt besteht aus 20 Vektoren, welche die Position der einzelnen Joints der Hand repräsentieren. SteamVR stellt die Position der HTC Vive durch einen Vektor und die Rotation durch einen Quaternion zur Verfügung.

3.4 Virtuelle Welt

Das Tracking ermöglicht die Bewegung der Nutzer in der realen Welt zu verfolgen und in Positions- und Rotationsdaten über verschiedene Schnittstellen zu veröffentlichen. Um die Bewegung und damit den Handschlag ins Virtuelle zu übertragen, wird nun eine virtuelle Welt benötigt. Diese muss Avatare für die Nutzer besitzen und in ihrer Simulation die Bewegung aus der realen Welt auf die Avatare übertragen. Weiterhin muss die virtuelle Welt dem Nutzer visualisiert werden.

Für die Erfüllung dieser Aufgaben wird in diesem System Unity verwendet. Unity ist eine 3D-Engine mit integrierter Entwicklungsumgebung, welche die Entwicklung von VR-Anwendungen unterstützt und in diesem Kontext sehr häufig Anwendung findet. Der Aufbau, die Simulation und die Darstellung der virtuellen Welt mit Unity werden im Folgenden genauer erläutert. Hierbei ist anzumerken, dass die entwickelte Unity Anwendung für beide Nutzer identisch ist.

3.4.1 Objekte der Virtuellen Welt

Um eine 3D-Welt zu erstellen bietet Unity die Möglichkeit simple 3D-Körper wie Kugeln, Quader oder Zylinder und abstrakte Objekte z. B. virtuelle Kameras oder Lichtquellen als sogenannte Gameobjekte einer Szene zu generieren. Des Weiteren lassen sich komplexere Objekte, die mit externen Programmen kreiert wurden, manuell importieren oder über einen Asset Store laden.

Position, Rotation und Skalierung eines Gameobjekts sind Bestandteil der Transform Komponente, die jedes Objekt in Unity besitzt. Die Gameobjekte einer Szene können dabei in Hierarchien strukturiert werden. Innerhalb einer Hierarchie ist die Transform Komponente eines Gameobjekts immer relativ zu dem direkt übergeordneten Gameobjekt. Existiert kein übergeordnetes Gameobjekt steht diese in Relation zum Koordinatensystem der virtuellen Welt. Gameobjekte sind ebenfalls durch weitere Komponenten wie Materialien und Texturen erweiterbar.

Grundsätzlich ist in Unity also eine vielfältige Gestaltung der virtuellen Welt möglich. Dies gilt auch für das Handschlag Szenario, welches im Virtuellen nicht an einen bestimmten Raum gebunden ist. Allenfalls sollte der verwendete Raum die gleichen Bewegungseinschränkungen wie der reale Raum besitzen, da sonst die Nutzer bei der Exploration der virtuellen Welt auf

Hindernisse in der realen Welt stoßen könnten.

Bei der Konstruktion der virtuellen Welt dieses Szenarios wurde deswegen der virtuelle Raum auf die Maße des zugänglichen Trackingbereichs beschränkt. Dabei wurde ein bereits vorhandenes 3D-Modell der Traverse, welche in der realen Welt den Trackingbereich begrenzt, in die Szene integriert. Die so gewonnene Übereinstimmung der realen und virtuellen Welt dient ebenfalls als Hilfe bei Anwendungstests.

Weiterhin wurde das Koordinatensystem der virtuellen Welt an das Koordinatensystem vom ART und vom Lighthouse angepasst. Hierbei wurde der Koordinatenursprung der virtuellen Welt auf die Position gesetzt, welche den Ursprung der Trackingsysteme in der virtuellen Welt repräsentiert. Durch diese Gleichsetzung wird eine weitere Koordinatentransformation von ART und Lighthouse Daten überflüssig.

Für die Avatare der Nutzer wird ein selbst konstruiertes Skelettmodell benutzt. Dieses besteht aus Kugeln, die den Joints des Kinect Skelettmodells entsprechen und Zylindern, welche die Verbindungen der einzelnen Joints darstellen. Die einzelnen Gameobjekte des Skelettmodells sind unter einem leeren Gameobjekt gruppiert. Ein leeres Gameobjekt besitzt eine Transform Komponente, aber keine sichtbare Form. Durch das Speichern dieser Struktur als Vorlage können beliebig viele Skelette der Szene hinzugefügt werden oder diese in anderen Projekten wiederverwendet werden.

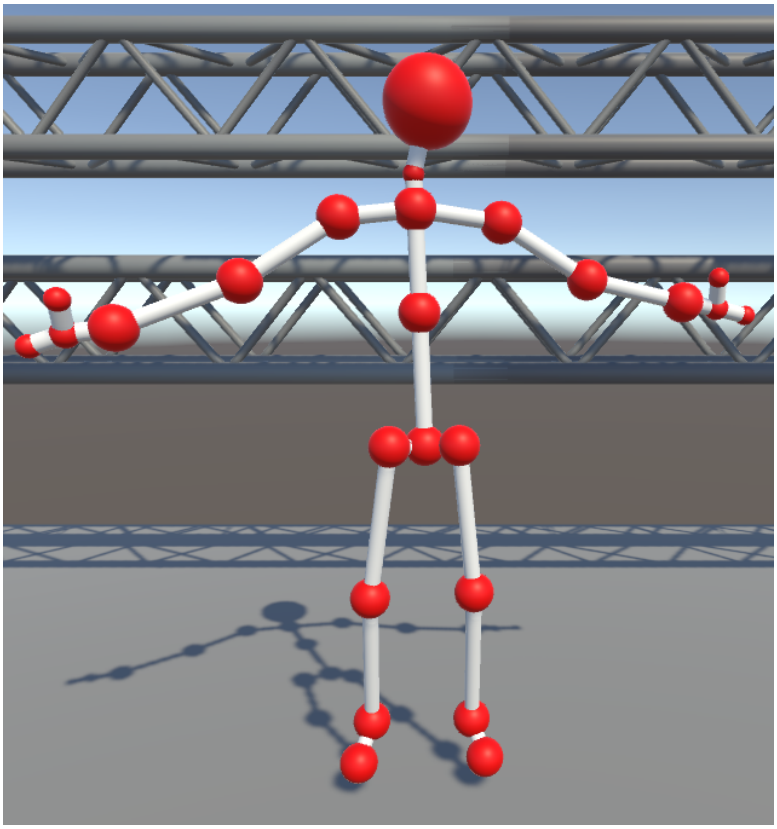

Abbildung 3.4: Das Skelettmodell

Zusätzlich dazu werden Hände und Arme der Nutzer durch die Capsule Hand Vorlage aus dem Leap Motion SDK visualisiert. Das Leap Motion SDK ist hierbei durch eine Kooperation zwischen den Entwicklern von Unity und der Leap Motion über den Asset Store verfügbar und kann direkt in die Umgebung eingebunden werden. Die Capsule Hand Vorlage ist ähnlich strukturiert wie die Skelett Vorlage. Auch hier werden einem leeren Gameobjekt Kugeln untergeordnet. Die Kugeln repräsentieren in diesem Fall die Hand und Arm Joints einer von der Leap Motion getrackten Hand. Jedoch existieren bei der Capsule Hand Vorlage keine Gameobjekte für die Verbindungen der Joints.

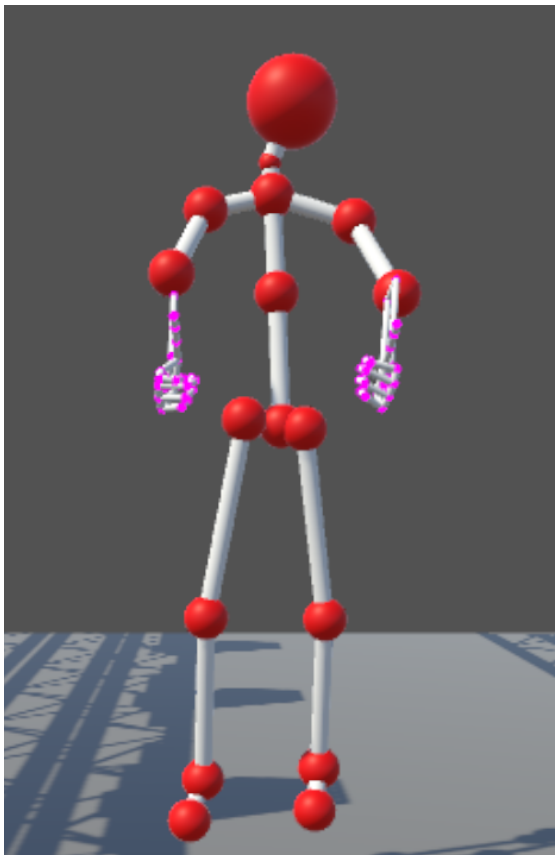

Abbildung 3.5: Das Skelettmodell mit Capsule Hands

Da jeder Nutzer beide Avatare sehen soll, sind also insgesamt zwei Skelette und vier Capsule Hands in der Szene vorhanden. Das Design der Avatare ist dabei in Relation zum Uncanny Valley Effekt bewusst einfach gehalten. Der Uncanny Valley Effekt beschreibt das Verhältnis von Akzeptanz einer Figur in Bezug auf den Grad der Änhlichkeit, welche die Figur mit einem realen Menschen hat. Hierbei sinkt, laut dem Effekt, die Akzeptanz, wenn der Grad der Änhlichkeit steigt. Erst wenn eine perfekte Abbildung erreicht wird, steigt die Akzeptanz auf ein Maximum (vgl. Tinwell und Grimshaw (2009)). Eine perfekte Abbildung ist in VR durch Leistungsgrenzen der Hardware und der Forderung geringer Latenzen jedoch nicht möglich.

Den Abschluss der virtuellen Welt bildet das LMHeadMountedRig. Dieses stammt aus einer Vorlage des Leap Motion SDKs und beinhaltet mitunter die virtuelle Kamera, welche den für den Nutzer sichtbaren Bildausschnitt reguliert. Andere Bestandteile der Vorlage sind mehrere leere Gameobjekte, die der Integration von Komponenten der Simulation dienen. Die Simulation der virtuellen Welt in Unity wird folgend erläutert.

3.4.2 Simulation der virtuellen Welt

In der virtuellen Welt der Unity Szene befinden sich jetzt alle Objekte, die für die Ausführung des Handschlags nötig sind. Diese sind jedoch noch statisch und weisen zur Laufzeit des System nicht das gewünschte Verhalten auf. Um das Verhalten von Objekten zu implementieren, können diese in Unity um C# oder JavaScript Skripte erweitert werden. Für die Simulation der virtuellen Welt dieses Systems wurden deswegen mehrere C# Skripte geschrieben oder importiert und als Komponenten an die entsprechenden Objekte gebunden.

Hierbei ist eine Besonderheit in Unity, dass alle C# Skripte von der Klasse MonoBehaviour erben müssen. Durch diese Beziehung werden Methoden, welche automatisch in einer festgelegten Reihenfolge vom System ausgeführt werden, in alle Skripte integriert (vgl. Unity (2017)).

In einigen der folgend beschriebenen Skripten werden davon die Methoden Start und Update benutzt. Start wird einmal zum Beginn der Anwendung ausgeführt. Danach folgt die Update Methode, die bis zum Ende der Anwendung kontinuierlich jedes Frame einmal aufgerufen wird. Die Reihenfolge der Start und Update Methoden verschiedener Skripte wird dabei von Unity bestimmt.

Der Großteil des Verhaltens der virtuellen Welt wird in dem PlayerController Skript simuliert, welches Komponente eines der Skelette ist. Das Skript baut in seiner Start Methode mit Hilfe

des C# Plugins der Middleware eine Verbindung zur Middleware Instanz des Systems auf und subscribed anschließend den Gruppen, in denen der Transform Agent die transformierten Skelette und der andere Nutzer seine Hände published. Die Identifizierung der richtigen Nutzergruppe erfolgt hierbei über eine Variable, die in den Anwendungen der Nutzer unterschiedlich gesetzt wird.

Weiterhin wird in der Start Methode der Callback für die Nachrichtenevents implementiert. Dabei werden die ankommenden Skelette des Transform Agenten und die Positionsdaten der Hände des anderen Nutzers entpackt und in eigens dafür definierten Klassen zwischengespeichert. Die Zwischenspeicherung der Daten ist notwendig, da Unity es nicht erlaubt die Position von Gameobjekten außerhalb der Update Methode zu bearbeiten.

In der Update Methode des PlayerController Skripts werden die Positionsdaten der eigenen Hände in dem Koordinatensystem der virtuellen Welt in ein JSON verpackt und über die Middleware an die Gruppe für die Hände des jeweiligen Nutzers gepublished. Die Wahl der entsprechenden Gruppe basiert auch hier wieder auf der Identifikations Variable des Nutzers. Danach werden die Hände des anderen Nutzers und die beiden Skelette der Nutzer anhand der zwischengespeicherten Daten aktualisiert. Die Differenzierung welches Skelett zu welchem Nutzer gehört erfolgt hierbei über einen Vergleich der Position des Kopfes vom Skelett mit der aktuellen Position des HMDs des Nutzers, welche aus dem in der Szene existierenden LMHeadMountedRigs abgeleitet wird.

Ein weiterer Bestandteil der Update Methode ist die Erkennung eines Handschlags. Hierfür wird die Position der Handmittelpunkte beider Nutzer verglichen. Sind diese in einem definierten Zeitraum nicht weiter als ein paar Zentimeter voneinander entfernt, wird dies als erfolgreicher Handschlag gewertet.

Um den Zugriff auf die Joints der Skelette und Capsule Hands zu erleichtern, besitzen die den Joints übergeordneten leeren Gameobjekte Skripte, in denen die Gameobjekte der Joints Variablen zugewiesen werden. Das PlayerController Skript benötigt für die Lese- und Schreiboperationen auf den Transform Komponenten der Joints dadurch nur Referenzen dieser Skripte. Darüber hinaus verfügen die Capsule Hands über ein Skript aus der Leap Motion SDK Vorlage, welches während seiner Update Methode die fehlenden Verbindungen zwischen den Joints zeichnet.

Die Bewegung der eigenen Hände und der virtuellen Kamera werden ebenfalls durch Skripte aus dem Leap Motion SDK umgesetzt. Diese sind an den Gameobjekten des LMHeadMounted-Rigs angebracht. Für die Aktualisierung der Kamera wird das Unity SteamVR Plugin verwendet. Dieses stellt die Daten der HTC Vive als globale Umgebungsvariablen von Unity zur Verfügung. Das Kamera Skript fragt diese kontinuierlich ab und richtet den Ergebnissen entsprechend den Bildausschnitt des Nutzers aus. Um die Frames der Leap Motion zu erhalten wird ein LeapServiceProvider Skript benutzt. Ein LeapController Skript filtert dann die Hände aus den bereitgestellten Frames. Die Hände werden mit Hilfe eines weiteren Skripts und den Daten der virtuellen Kamera auf das Koordinatensystem der Kamera umgerechnet. Anschließend aktualisiert das LeapController Skript über zwei gesetzte Referenzen die Capsule Hände des Nutzers.

3.4.3 Darstellung der virtuellen Welt

Für die Darstellung der virtuellen Welt muss der Ausschnitt der virtuellen Kamera für die HTC Vive des Nutzers aufbereitet werden. Diese Aufbereitung kann Unity selbst durchführen wenn die Option Virtual Reality Supported aktiviert ist, da Unity die HTC Vive als VR-Ausgabegerät unterstützt. Das Endergebnis wird dann über das SteamVR Plugin an die HTC Vive gesendet und dort dem Nutzer dargestellt.

3.4.4 Ablauf der Simulation und Darstellung

In Abbildung 3.6 ist der zeitliche Ablauf der Simulation und Darstellung der virtuellen Welt eines Nutzers durch ein Sequenzdiagramm veranschaulicht. Hierbei wurden die verschiedenen Skripte des LMHeadMountedRigs in einer Komponente zusammengefasst und die Skripte für die Zeichnung der Capsule Hands missachtet. Deutlich erkennbar ist die von Unity vorgegebene Reihenfolge der genutzten Start und Update Methoden. Dabei wird zuerst die Start Methode des PlayerController Skripts einmalig ausgeführt und dann in einer Endlosschleife für jedes Frame die Update Methoden der verschiedenen Skripte. In dieser Schleife findet ebenfalls die Visualisierung der virtuellen Welt für den Nutzer statt. Die Framerate wird in diesem System nicht künstlich begrenzt und ist deswegen abhängig von der Zeit, die für eine Iteration der Schleife benötigt wird. Eine Ausnahme in dieser vorherbestimmten Reihenfolge bildet der callback des PlayerController Skripts, welcher durch die Nachrichtenevents der Middleware ausgelöst wird.

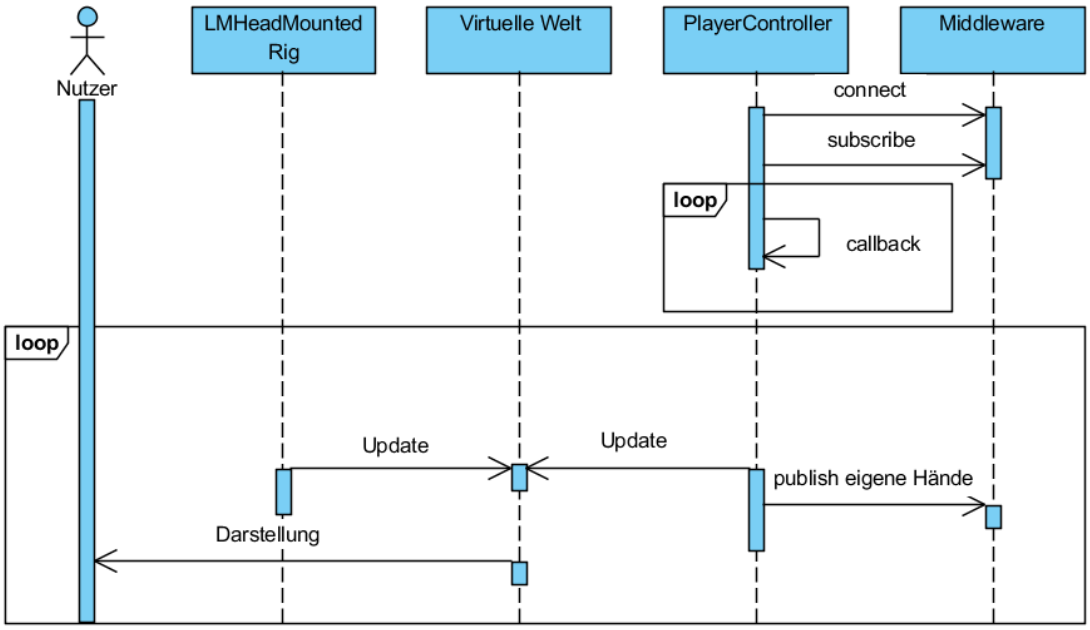

Abbildung 3.6: Ablauf der Simulation und Darstellung der virtuellen Welt

3.4.5 Schnittstellen der virtuellen Welt

Die virtuelle Welt besitzt verschiedene Schnittellen, um die eingehenden Tracking Daten zu verarbeiten und die Leap Motion Daten der eigenen Hände zu veröffentlichen. Hierbei wird für die Verarbeitung der HTC Vive und Leap Motion Daten die in 3.4.2 beschriebenen Skripte der LMHeadMountedRig Vorlage und SteamVR genutzt.

Für die Serialisierung der Leap Motion Hände, die über die Middleware gesendet werden, wird die Skeleton API verwendet. Dabei werden die Hand Joints der Leap Motion Hands, die einen korrespondierenden Joint im Skelettmodell der Kinect besitzen, mit der gleichen ID wie der entsprechende Joint der Kinect versehen. Dies ermöglicht es, die transformierten Skelette der Plattform und die Hände des anderen Nutzers auf gleiche Weise zu entpacken. Da der API-Generator aus dem Framework der Middleware keine Unterstützung für C# bietet, werden die JSONs der Skeleton API mit Hilfe eines generischen JSON Frameworks serialisiert und deserialisiert.

ID	Joint
4	Shoulder Left
5	Elbow Left
6	Wrist Left
7	Hand Left
8	Shoulder Right
9	Elbow Right
10	Wrist Right
11	Hand Right
21	Hand Tip Left
22	Thumb Left
23	Hand Tip Right
24	Thumb Right

Abbildung 3.7: Korrespondierende Joints zwischen Leap Motion und Kinect

4 Evaluation

In diesem Kapitel wird das entwickelte Mehrbenutzer VR-System in Bezug auf die in 2.8 gestellten Anforderungen evaluiert. Außerdem wird ein Ausblick auf mögliche Verbesserungen und Erweiterungen des Systems im Kontext des Handschlag Szenarios gegeben.

4.1 Bewertung der Anforderungen

4.1.1 Funktionalität

Das System wurde, wie im vorigen Kapitel dargestellt, funktional vollständig implementiert und das Handschlag Szenario mit verschiedenen Personen erfolgreich durchgeführt. Um eine Grundlage für die Bewertung der Qualität zu schaffen wurden verschiedene Einstellungen getestet. Dabei wurde einerseits die Position im Raum und andererseits die Form der Avatare variiert. Insgesamt wurden drei verschiedene Modelle als Avatare getestet.

1. *Skelettmodell*

   Beide Nutzer sehen nur die Kinect Skelettmodelle.

2. *Skelettmodell mit Leap*

   Beide Nutzer sehen sowohl die Kinect Skelettmodelle als auch die Capsule Hands der Leap Motion.

3. *Modifiziertes Skelettmodell mit Leap*

   Beide Nutzer sehen sowohl die Kinect Skelettmodelle als auch die Capsule Hands der Leap Motion. Hierbei werden jedoch die Joints der Arme beider Skelettmodelle und die entsprechenden Verbindungen ausgeblendet.

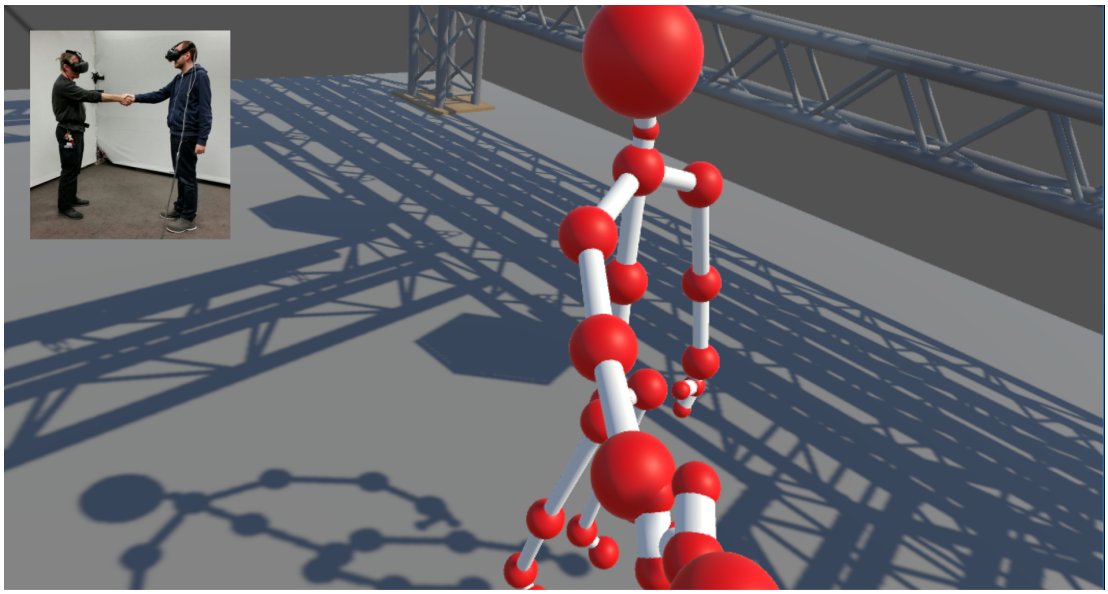

Abbildung 4.1: Handschlag mit Skelettmodellen

4.1.2 Präzision

Die Präzision des Systems ist hauptsächlich abhängig von der Genauigkeit der Sensoren und den Koordinatentransformationen. Diese weisen jeweils eine ausreichende Genauigkeit auf. Im Zusammenspiel der Sensoren unter realen Bedingungen treten jedoch Probleme auf. So entstehen Ungenauigkeiten durch Interferenzen im Infrarotbereich, da alle verwendeten Sensoren infrarot basiert sind. Verstärkt werden diese Interferenzen durch Lichtreflexionen in der Traverse. Das Ausmaß der Interferenzen wird aber durch in den Sensoren eingebaute Ausgleichsfunktionen und die unterschiedliche Ausrichtung, Frequenz und Reichweite der Sensoren begrenzt, weshalb diese wenig Einfluss auf den Erfolg des Handschlags haben.

Eine weiteres Problem ist die Qualität der Kinect in Bezug auf die Position und Rotation der Nutzer in ihrem Trackingbereich. Steht ein Nutzer mit dem Rücken zur Kinect oder am Rand des Trackingbereichs kann die Kinect nicht alle Joints bestimmen. Gleiches tritt auf wenn ein Nutzer den Anderen verdeckt. Aktuell ist deswegen die Präzision des Handschlags nur gewährleistet wenn sich beide Nutzer in einem gewissen Winkel und Abstand zur Kinect befinden.

4.1.3 Latenz

Durch die Verwendung von Komponenten, die selbst latenzarm sind und für den VR-Kontext entworfen wurden, wird der Anforderung nach geringen Latenzen im Gesamtsystem nachgekommen.

4.1.4 Wiederverwendbarkeit

Fast alle Features und Komponenten des Systems benutzen die Middleware zur Kommunikation und besitzen generisch und klar definierte Schnittstellen, die über den API-Generator verfügbar sind. Dadurch sind diese für Entwickler im CSTI leicht wiederverwendbar. Weiterhin wurden aus entwickelten 3D-Modellen Vorlagen generiert, die über Unity direkt eingebunden werden können.

4.2 Systemausblick

4.2.1 Kombinierte Avatare

Das Skelettmodell und das Capsule Hand Modell der Avatare weisen bereits eine ähnliche Struktur und Darstellung auf. Um die Verbindung beider natürlicher zu gestalten, könnte das Modell der Capsule Hand in das Skelettmodell fest integriert werden. Hierbei ist jedoch zu beachten, dass die Capsule Hands des anderen Nutzers nur korrekt positioniert werden, wenn dessen Hände sich im Blickfeld seiner Leap Motion befinden. Eine mögliche Lösung für dieses Problem ergibt sich aus einer dynamischen Schnittstelle bei der Wahl der Positionsdaten für Hände und Arme. Diese wechselt bei einem persistenten Mangel an Leap Motion Daten zu den Daten der Kinect um die Hände und Arme des anderen Nutzers darzustellen. Der dabei entstehende Verlust an Genauigkeit der Hände ist für die Interaktion in diesem System nicht von großer Relevanz, da Nutzer grundsätzlich bei einem Handschlag ihre Hände im Blick haben und deswegen Daten der Leap Motion zur Verfügung stehen.

4.2.2 Multiple Kinects

Um die Probleme im Zusammenhang mit dem Trackingbereich der Kinect zu lösen, könnten mehrere Kinects für die Skelettverfolgung genutzt werden. Dabei müsste zuerst eine Kalibration aller Kinects stattfinden. Außerdem müsste ein Algorithmus implementiert werden, welcher die Qualität der Skelette der einzelnen Kameras auswertet und ein Skelett auswählt oder aus den einzelnen Teilen der Skelette ein Vollständiges zusammensetzt.

Weiterhin müsste evaluiert werden, inwiefern die Abweichungen, welche durch die notwendige Koordinatentransformation entstehen, die Qualität des Handschlags beeinflussen. Dies gilt ebenfalls für die erhöhte Wahrscheinlichkeit von Interferenzen durch die zusätzlichen Kinects.

5 Zusammenfassung und Ausblick

Im Folgenden werden die vorangegangen Kapitel zusammengefasst und diese Arbeit mit einem Ausblick auf weitere Anwendungsgebiete von VR und Mehrbenutzer VR-Systemen abgeschlossen.

5.1 Zusammenfassung

In dieser Arbeit wurde ein Mehrbenutzer VR-System entworfen, welches die Interaktion in Form eines virtuellen Handschlags zwischen zwei Nutzern in einer virtuellen Welt ermöglicht.

Zuerst wurden in der Analyse (s. Kapitel 2) die Anforderungen an das System identifiziert. Das System muss für die Durchführung des Handschlags verschiedene Funktionalitäten bereitstellen. Die Nutzer müssen durch geeignete Tracking System im Raum verfolgt, die Bewegung in einer virtuellen Welt simuliert und die Ergebnisse dem Nutzer präsentiert werden. Dabei sind für die Qualität des Handschlags eine niedrige Latenz und eine hohe Präzision Ansprüche an das System. Damit andere Entwickler auf dem System aufbauen können, sollte dieses ebenfalls wiederverwendbar sein.

Ausgehend von den identifizierten Anforderungen wurde in Kapitel 3 das System entworfen. In dem System werden die Daten der Tracking Systeme durch entsprechende Schnittstellen an die virtuellen Welten der Nutzer gesendet. Die virtuellen Welten wurden in Unity erstellt und sind für beide Nutzer identisch. Unity simuliert zur Laufzeit des Systems die virtuellen Welten, indem es die Tracking Daten über die Schnittstellen entgegennimmt und die Avatare der Nutzer aktualisiert. Dabei findet auch eine Bewertung statt, ob ein Handschlag erfolgt ist. Weiterhin werden die virtuellen Welten in Unity aufbereitet und dem Nutzer über ein HMD visualisiert.

Die Evaluation des Systems (s. Kapitel 4) basiert auf einer Implementation des Systems, welche im Rahmen des Designs stattfand. Dabei wurde die Erfüllung der Anforderungen durch eine

Analyse der Implementation und einige Testdurchläufe mit verschiedenen Einstellungen bewertet. Aus der Analyse und den Tests hat sich ergeben, dass die geforderte Latenz und Präzision des Systems bei einer optimalen Einstellung durch die Qualität der einzelnen Komponenten gegeben ist. Auch die Wiederverwendbarkeit ist durch generische Schnittstellen und eine Kapselung von Inhalten gewährleistet. Abschließend wurden in der Evaluation noch mögliche Erweiterungen des Systems aufgeführt.

5.2 Ausblick

Das in dieser Arbeit entwickelte System zeigt den konzeptionellen Aufbau eines Mehrbenutzer VR-Systems. Hierbei ist das Handschlag Szenario nur als ein Beispiel für eine Interaktion anzusehen. Grundsätzlich lassen sich eine Vielzahl von Interaktionen implementieren, wenn dafür passende Eingabe- und Ausgabegeräte vorhanden sind. Die Entwicklung neuer oder qualitativ besserer Geräte in den letzten Jahren hat dabei den Einsatz von VR in unterschiedlichen Anwendungsgebieten ermöglicht, die von der natürlichen Art der Interaktion und der immersiven Darstellung von 3D-Welten in VR profitieren.

In der Entertainment-Branche beispielsweise bieten Videospiele in VR den Spielern ein viel immersiveres Erlebnis, indem die Avatare durch Bewegung der Spieler gesteuert werden und den Spielern das Abenteuer aus der Sicht der Avatare präsentiert wird. Ebenso bietet VR Vorteile in der Industrie. Ingenieure können Arbeiten an großen Maschinen oder in Gefahrensituationen in VR üben. Dabei werden Kosten gespart und die Sicherheit der Ingenieure garantiert. Ähnlich gestaltet sich die Verwendung von VR in der Medizin. Ärzte trainieren Operationen an virtuellen Patienten und nutzen virtuelle Simulationen für Therapien, um reale Patienten nicht zu gefährden. Ein weiteres Anwendungsgebiet von VR findet sich im Marketing. Produkte können in VR besser visualisiert werden und Kunden können diese direkt erkunden oder ausprobieren.

All diese Anwendungsgebiete teilen sich die Eigenschaft, dass sie ebenfalls einen Vorteil aus einem Mehrbenutzer VR-System ziehen können. Spiele können zusammen mit Freunden erlebt werden. In der Industrie und in der Medizin können Ausbilder den Auszubildenden komplexe Schritte vorführen und im Marketing der Verkäufer seine Produkte dem Kunden präsentieren. Außerdem erschließen sich aus der Unterstützung von mehreren Benutzern in einem VR-System wiederum neue Anwendungsgebiete. Ein Beispiel dafür ist die Durchführung ei-

nes Meetings in einem virtuellen Konferenzraum, in dem alle Teilnehmer einen Avatar besitzen.

Folglich existieren viele potenzielle Anwendungsmöglichkeiten für ein interaktives Mehrbenutzer VR-System. Dabei ist jedoch bei dem Entwurf des Systems oder der Integration der entsprechenden Interaktionen in ein bestehendes System zu beachten, dass die Anwendungsgebiete unterschiedliche Anforderungen an das System stellen. Ebenfalls besitzen die Eingabegeräte, welche für die Interaktionen der jeweiligen Anwendungsgebiete benötigt werden, verschiedene Schnittstellen. Für zukünftige Arbeiten in diesem Kontext sollte deswegen der erste Schritt eine Analyse der Anforderungen und Eingabegeräte des gewählten Anwendungsgebiets sein.

Literaturverzeichnis

[Abrash 2012] Abrash, Michael: *Latency – the sine qua non of AR and VR.* http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/. 2012. – Zugriff: 28.09.2017

[Aichholzer und Jüttler 2013] Aichholzer, Oswin ; Jüttler, Bert: *Einführung in die angewandte Geometrie.* Berlin Heidelberg New York : Springer-Verlag, 2013. – ISBN 978-3-034-60651-6

[ART 2017] ART: *ART - Technical Details.* http://ar-tracking.com/technology/technical-details/. 2017. – Zugriff: 18.11.2017

[CSTI 2017] CSTI: *Creative Space for Technical Innovations.* https://csti.haw-hamburg.de. 2017. – Zugriff: 02.09.2017

[Dörner u. a. 2014] Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul ; Jung, Bernhard: *Virtual und Augmented Reality (VR / AR) - Grundlagen und Methoden der Virtuellen und Augmentierten Realität.* Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978-3-642-28903-3

[Eichler 2014] Eichler, Tobias: *Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor.* Hamburg, Germany, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 10 2014

[Eichler u. a. 2017] Eichler, Tobias ; Draheim, Susanne ; Grecos, Christos ; Wang, Qi ; von Luck, Kai: Scalable Context-Aware Development Infrastructure for Interactive Systems in Smart Environments. In: *Fifth International Workshop on Pervasive and Context-Aware Middleware 2017 (PerCAM'17).* Rome, Italy, Oktober 2017

[Guna u. a. 2014] Guna, J ; Jakus, G ; Pogačnik, M ; Tomažič, S ; Sodnik, J: *An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking.* Feb 2014. – URL https://www.ncbi.nlm.nih.gov/pubmed/24566635

[ISO92411.11 2017] DIN: *Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte.* 2017

[Jeworutzki 2017] Jeworutzki, André: *Raumplan des CSTI.* 2017

[Kirchner 2017] Kirchner, Marc: *Eine auf Sensorfusion basierte Plattform für Personentracking in VR/AR-Umgebungen.* Hamburg, Germany, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 12 2017

[Meehan u. a. 2003] Meehan, M. ; Razzaque, S. ; Whitton, M. C. ; Brooks, F. P.: Effect of latency on presence in stressful virtual environments. In: *IEEE Virtual Reality, 2003. Proceedings.*, March 2003, S. 141–148. – ISSN 1087-8270

[Microsoft1 2017] Microsoft1: *Kinect-Hardware.* https://developer.microsoft.com/de-de/windows/kinect/hardware. 2017. – Zugriff: 18.11.2017

[Microsoft2 2017] Microsoft2: *Kinect Skeleton Joints.* https://msdn.microsoft.com/de-de/library/windowspreview.kinect.jointtype.aspx. 2017. – Zugriff: 18.11.2017

[Sra und Schmandt 2015] Sra, Misha ; Schmandt, Chris: *MetaSpace II: Object and full-body tracking for interaction and navigation in social VR.* Dec 2015. – URL https://arxiv.org/abs/1512.02922. – Zugriff: 06.11.2017

[TFWiki 2017] TFWiki: *Koordinatentransformation.* http://de.theoriefinder.wikia.com/wiki/Koordinatentransformation. 2017. – Zugriff: 27.11.2017

[Tinwell und Grimshaw 2009] Tinwell, A. ; Grimshaw, M.: Bridging the Uncanny: An Impossible Traverse? In: *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era.* New York, NY, USA : ACM, 2009 (MindTrek '09), S. 66–73. – URL http://doi.acm.org/10.1145/1621841.1621855. – ISBN 978-1-60558-633-5

[Unity 2017] Unity: *Execution Order of Event Functions.* https://docs.unity3d.com/Manual/ExecutionOrder.html. 2017. – Zugriff: 23.11.2017

[VRJump 2017] VRJump: *Lighthouse von VALVE erklärt.* https://vrjump.de/lighthouse-erklaert. 2017. – Zugriff: 28.11.2017

[VUO 2016] VUO: *Leap Motion hand skeleton points*. http://vuo.org/component/vuo.leap. 2016. – Zugriff: 19.11.2017

[Young u. a. 2015] Young, Mary K. ; Rieser, John J. ; Bodenheimer, Bobby: Dyadic Interactions with Avatars in Immersive Virtual Environments: High Fiving. In: *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception*. New York, NY, USA : ACM, 2015 (SAP '15), S. 119–126. – URL http://doi.acm.org/10.1145/2804408.2804410. – 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, 08. Dezember 2017 Niklas Alexander Gerwens