

Multimodale Telepräsenz mit dem humanoiden Roboter NAO und VR-Brille

Studienarbeit

des Studienganges Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg, Karlsruhe

Ersteller:	Isabella Schmidt (7927456), Fabian Dogendorf (8053885)
Abgabedatum:	17.05.2019
Abgabeort:	Duale Hochschule Baden-Württemberg, Karlsruhe
Kursbezeichnung DHBW:	TINF16B4
Betreuer:	Prof. Dr. Hans-Jörg Haubner

Eidesstattliche Versicherung

Hiermit erklären wir an Eides statt, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Inhaltlich und wörtlich aus den Quellen entnommene Stellen wurden als solche kenntlich gemacht.

Abstract

This student research project examines the possibilities of a multimodal telepresence of a Nao-Robot. Telepresence is used in laboratories that work with hazardous substances and thus pose a safety risk for humans. The robot can be remotely controlled by a human from a safe distance using Virtual Reality technologies. Telepresence allows the environment of the robot to be transmitted to the human operator. By reproducing the image and sound information inside a virtual environment, the operator can react accordingly to events.

Within this student research project, a telepresence application was created. The application serves as an interface between the human operator and the robot. Using Virtual Reality technologies any user can enter the virtual environment that offers certain robot control options. In addition to the remote control of the robot the created environment enables the rendering of the current robot's camera image. The user can therefore directly track the interactions with the robot and further adjust its behavior.

The application can be used to demonstrate the control of the Nao-Robot. The advantage of this project is that the human operator does not have to be in the same room as the robot. Therefore, a remote control can be performed, and the actions of the robot can be chosen according to the operator's requirements in the virtual environment.

Abstract (Deutsch)

In diesem Studienprojekt werden die Möglichkeiten einer multimodalen Telepräsenz eines Nao-Roboters untersucht. Die Technologie der Telepräsenz wird in Laboren eingesetzt, die mit Gefahrstoffen arbeiten und damit ein Sicherheitsrisiko für den Menschen darstellen. Mit Hilfe von Virtual Reality-Technologien kann ein Roboter von einem Menschen aus sicherer Entfernung gesteuert werden. Telepräsenz ermöglicht hierbei die Übertragung der Umgebung des Roboters an den menschlichen Operator. Durch die Wiedergabe der Bild- und Toninformationen in einer virtuellen Umgebung kann der Operator entsprechend auf Ereignisse reagieren.

Im Rahmen dieser Studienarbeit wurde eine Telepräsenzanzwendung erstellt. Die Anwendung dient als Schnittstelle zwischen einem menschlichen Operator und einem Roboter. Durch den Einsatz von Virtual Reality-Technologien kann der Operator die virtuelle Umgebung betreten. Diese virtuelle Welt bietet Optionen zur Steuerung des Roboters an. Neben der Fernsteuerung des Roboters ermöglicht die geschaffene Umgebung weiterhin die Darstellung des Kamerabildes des Roboters. Der Operator kann somit die Interaktionen mit dem Nao-Roboter direkt verfolgen und sein Verhalten steuern.

Die Anwendung kann verwendet werden, um die Steuerung des Nao-Roboters zu demonstrieren. Der Vorteil dieses Projektes besteht darin, dass sich der menschliche Operator nicht im selben Raum wie der Roboter befinden muss. Somit können die Aktionen des Roboters entsprechend den Anforderungen des Bedieners in einer virtuellen Umgebung gewählt werden.

Inhaltsverzeichnis

Abkürzungsverzeichnis.....	7
Abbildungsverzeichnis.....	8
Tabellenverzeichnis.....	9
1 Einführung.....	10
1.1 Aufgabenstellung und Zielsetzung.....	10
1.2 Vorgehensweise.....	11
2 Grundlagen.....	12
2.1 Theoretische Grundlagen.....	12
2.1.1 Telepräsenz.....	12
2.1.2 Robotik.....	14
2.1.3 Virtual Reality.....	16
2.1.4 Schichtenarchitektur.....	18
2.2 Technische Grundlagen.....	20
2.2.1 Nao-Roboter.....	20
2.2.2 Choreographie Suite.....	22
2.2.3 HTC Vive Pro.....	24
2.2.4 Unity.....	25
2.2.5 Maya.....	29
3 Entwurf.....	31
3.1 Theoretische Vorgehensweise.....	31
3.1.1 Anforderungsanalyse.....	31
3.1.2 Geplantes Vorgehen.....	32
3.2 Konzeption.....	34
3.2.1 Aufbau der Anwendung.....	34
3.2.2 Entwurf der virtuellen Umgebung.....	36
4 Implementierung.....	40
4.1 Erstellung der virtuellen Umgebung.....	40
4.1.1 Design der 3D-Objekte.....	40
4.1.2 Aufbau der virtuellen Umgebung.....	43
4.1.3 Ordnerstruktur.....	49
4.2 Kommunikation mit dem Nao-Roboter.....	51
4.3 Steuerung des Nao-Roboters.....	56

4.3.1	Wrapper Schicht für C#.....	56
4.3.2	Unity Skripte für Steuerung	58
4.4	Umsetzung der Telepräsenz	60
5	Ergebnis.....	62
6	Zusammenfassung und Ausblick	64
6.1	Persönliches Feedback.....	64
6.2	Erweiterungsmöglichkeiten.....	65
	Literaturverzeichnis.....	67

Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Application Programming Interface
AR	Augmented Reality
DLL	Dynamic Link Library
FPS	Frames Per Second
MEL	Maya Embedded Language
GUI	Graphical User Interface
MR	Mixed Reality
SDK	Software Development Kit
VDI	Verein Deutscher Ingenieure
VR	Virtual Reality

Abbildungsverzeichnis

Abbildung 1: Teleroboter	12
Abbildung 2: Industrieroboter in der Fertigung eines Automobilherstellers	14
Abbildung 3: Gegenüberstellung VR, AR und MR	17
Abbildung 4: Nao-Roboter	21
Abbildung 5: Oberfläche Choreographie Suite	22
Abbildung 6: HTC VIVE Pro	24
Abbildung 7: Benutzeroberfläche Unity	26
Abbildung 8: Architektur-Entwurf	34
Abbildung 9: Entwurf Head-Display	36
Abbildung 10: Entwurf Labor	37
Abbildung 11: Entwurf Willkommensbildschirm	38
Abbildung 12: Design-Prozess in Maya	41
Abbildung 13: Willkommensbildschirm	44
Abbildung 14: Cubemap Skybox Willkommensbildschirm	45
Abbildung 15: Menü IP-Settings	46
Abbildung 16: Nao-Lab	48
Abbildung 17: Menü im Nao-Lab	49
Abbildung 18: Ordnerstruktur Unity-Projekt	49

Tabellenverzeichnis

<i>Tabelle 1: Building-Blöcke in Unity</i>	28
<i>Tabelle 2: Ordnerstruktur Unity-Projekt</i>	50
<i>Tabelle 3: Methoden C#-Wrapper</i>	56

1 Einführung

1.1 Aufgabenstellung und Zielsetzung

Im Rahmen dieser Studienarbeit wurde eine Anwendung konzipiert und implementiert, die es einem menschlichen Benutzer ermöglicht, einerseits einen entfernten Roboter fernzusteuern und andererseits in der entfernten Umgebung visuell präsent zu sein. Für die Umsetzung der Studienarbeit standen ein Nao-Roboter der Firma Aldebaran sowie die Virtual Reality Brille HTC Vive Pro zur Verfügung. Mit Hilfe dieser Hardware wurde im Verlauf der Studienarbeit eine virtuelle Umgebung konzipiert und in Unity erstellt. Der Operator der VR-Brille kann sich in dieser Umgebung frei bewegen. Relevant war hierbei insbesondere die Umsetzung einer multimodalen Telepräsenz. Die Übertragung von Befehlen und Sensordaten sollte in beide Richtungen erfolgen können: sowohl vom Roboter zum Anwender als auch umgekehrt. Der Telepräsenzroboter überträgt hierbei die Wahrnehmung seiner Sensoren an den Anwender und die übermittelten Daten werden dem Benutzer in der virtuellen Umgebung aufbereitet angezeigt. Über die Virtual Reality Brille wird es dem Benutzer ermöglicht, in die entfernte Umgebung einzutauchen und den Effekt der Immersion zu spüren. Über die Controller der HTC Vive Pro besteht für ihn zusätzlich die Möglichkeit, die Aktionen des Nao-Roboters zu manipulieren und diesen aus einer entfernten Umgebung zu steuern.

Diese Arbeit dient dazu, dem Leser einen Überblick über das Thema dieser Studienarbeit zu verschaffen und die Projektdurchführung zu dokumentieren. Dadurch soll ermöglicht werden, dass andere Projekte von dem gewonnenen Wissen profitieren.

1.2 Vorgehensweise

Diese Studienarbeit dokumentiert die Konzeption und Umsetzung eines Programms zur multimodalen Telekommunikation zwischen einem menschlichen Benutzer und einem entfernten Nao-Roboter. Der Benutzer begibt sich dabei mit Hilfe einer Virtual Reality Brille in eine virtuelle Umgebung. Diese mit Unity erstellte virtuelle Welt stellt Interaktions- und Kommunikationsmöglichkeiten für den Roboter bereit.

Zunächst werden in Kapitel 2 die theoretischen und technischen Grundlagen erläutert. Dieses Kapitel enthält neben theoretischen Grundprinzipien auch die Funktionsweise und den Aufbau der verwendeten Software. In Kapitel 3 wird der Entwurf der Softwarelösung vorgestellt. Dieser beinhaltet die theoretische Vorgehensweise und die Konzeption der entstandenen Anwendung. Dabei werden auch einige Designentwürfe des virtuellen Raumes vorgestellt. Anschließend wird in Kapitel 4 die Implementierung und Umsetzung der Anwendung dokumentiert. Darin enthalten ist sowohl der Prozess der Softwareentwicklung als auch der Erstellprozess der virtuellen Umgebung. Die Kommunikation zwischen Roboter und Anwendung wird ebenfalls erläutert. In Kapitel 5 wird das Ergebnis der Studienarbeit vorgestellt. Das abschließende Kapitel 6 enthält einen Ausblick zu möglichen Einsatzzwecken und ein persönliches Feedback zu dem Ergebnis der Studienarbeit.

2 Grundlagen

In diesem Kapitel werden grundlegende Fachbegriffe erläutert, die zum Verständnis der Thematik erforderlich sind. Dabei wird zwischen theoretischen und technischen Grundlagen unterteilt.

2.1 Theoretische Grundlagen

2.1.1 Telepräsenz

Die Technologien der Telepräsenz im Bereich der Robotik ermöglichen es einem menschlichen Benutzer in einer virtuell erzeugten, entfernten Umgebung visuell präsent zu sein. Der Anwender kann durch den Einsatz von Technologien der Virtual Reality die künstliche Wirklichkeit realistisch erleben und in diese vollständig eintauchen. Das Besondere dieser Technologie ist, dass der Roboter und Mensch eins werden, indem die Sensor-Signale des Roboters an den Operator übertragen werden. Über ein Head-Mounted-Display können die erzeugten Video-Signale in einer 3D-Umgebung visualisiert werden. Die Kopf- und Armpositionen des Benutzers in der realen Umgebung können direkt an den Roboter mit Hilfe einer VR-Brille sowie Datenhandschuhen übertragen werden.

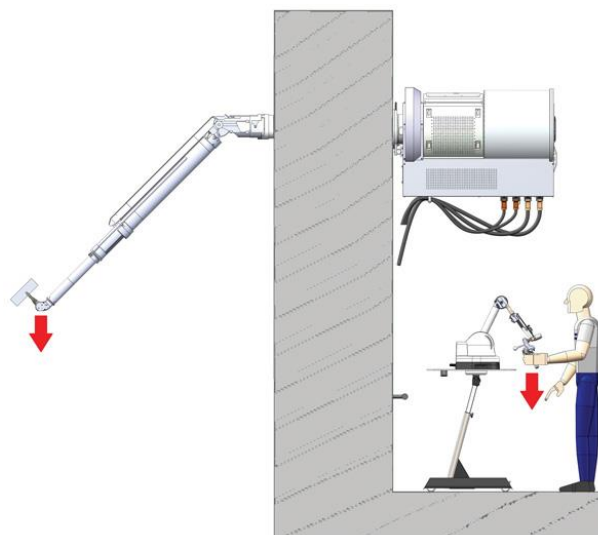


Abbildung 1: Teleroboter¹

¹ Siehe <https://www.neimagazine.com/features/featurehot-cell-robot-4483658/featurehot-cell-robot-4483658-462777.html> [Letzter Zugriff: 12.01.2019]

Eine wichtige Komponente der Technologie ist das sogenannte Telemanipulations-System. Hierunter versteht man Werkzeuge, mit denen es einem Menschen ermöglicht wird, Gegenstände zu manipulieren, ohne sie mit eigenen Händen anzufassen. Anwendung finden diese Systeme vor allem in Laboren, in denen mit Gefahrenstoffen gearbeitet wird und die somit ein Sicherheitsrisiko für den Menschen darstellen. Die Technik der Telerobotik ermöglicht, dass Bewegungskommandos komfortabel von einem ausgebildeten Anwender ausgeführt werden können, ohne dass dieser sich dabei in direkte Gefahr begibt. Der Operator agiert aus sicherer Entfernung heraus, gegebenenfalls hinter einer zusätzlichen Schutzvorrichtung oder Abschirmung. Er kann alle Aktionen des Roboters mit Hilfe von Controllern „remote“ steuern und die Aktionen über einen Videobildschirm verfolgen. Oft werden die Telemanipulatoren dem Menschen direkt nachempfunden, damit eine intuitive Steuerung möglich ist.² Ein Beispiel hierfür ist ein Roboterarm, wie in Abbildung 1 dargestellt. Wesentlicher Vorteil dieser Systeme ist, dass der Mensch zwar Aufgaben durch einen autonomen Roboter durchführen lassen kann, jedoch weiterhin stets die Kontrolle über das System hat. Falls unerwartete Ereignisse eintreten, kann er jederzeit direkt eingreifen und nach eigenem Ermessen eine Entscheidung über die nächste Aktion des Teleroboters treffen.³

Im Rahmen dieser Studienarbeit soll der Spezialfall der multimodalen Telepräsenz umgesetzt werden. Darunter versteht man die Kommunikation zwischen Mensch und Roboter, die über verschiedene Kanäle durchgeführt wird. Es werden somit nicht nur Signale vom Menschen zum Roboter gesendet, um den Roboter fernzusteuern, sondern zusätzlich Signale vom Roboter an den Operator, wie beispielsweise Bild- oder Tonmaterial. Die Umsetzung einer multimodalen Telepräsenz hat den Vorteil, dass durch die beidseitige Übertragung der Sensordaten die Steuerung des Roboters unmittelbar nachverfolgt werden kann.

² Vgl. Bruder, Jan „Praktische Realisierung einer haptischen Telerobotik-Steuerung für eine interaktive Nutzung“ [2009]

³ Vgl. https://www.dlr.de/desktopdefault.aspx/tabid-3228/5011_read-26483/5011_page-2/ [Letzter Zugriff: 15.01.2019]

2.1.2 Robotik

Unter einem Roboter versteht man eine Maschine, die zur Übernahme von menschlichen Aufgaben konzipiert wurde. Je nach Art des Roboters werden die Aufgaben an einem festen Ort (stationär) oder in einer dynamischen Umgebung (mobil) ausgeführt. Laut der VDI-Definition 2860 von 1990 müssen Roboter, auch bezeichnet als Manipulatoren, mindestens drei unabhängige Achsen besitzen um bestimmte Dinge wie Werkzeuge, Teile und Materialien bewegen zu können.⁴

Der Einsatz von mobilen Robotern gestaltet sich in der Praxis schwieriger als der von stationären Robotern. Bei einem stationären Roboter ist „die Umgebung, in welcher der Roboter arbeitet, vollständig bekannt und kontrolliert“.⁵ Dies trifft vor allem auf Industrieroboter zu, welche sich meist in einem eigenen, abgeäuerten und gesicherten Bereich befinden. Industrieroboter sind deswegen oft „nackt“ und haben kaum Verkleidung.

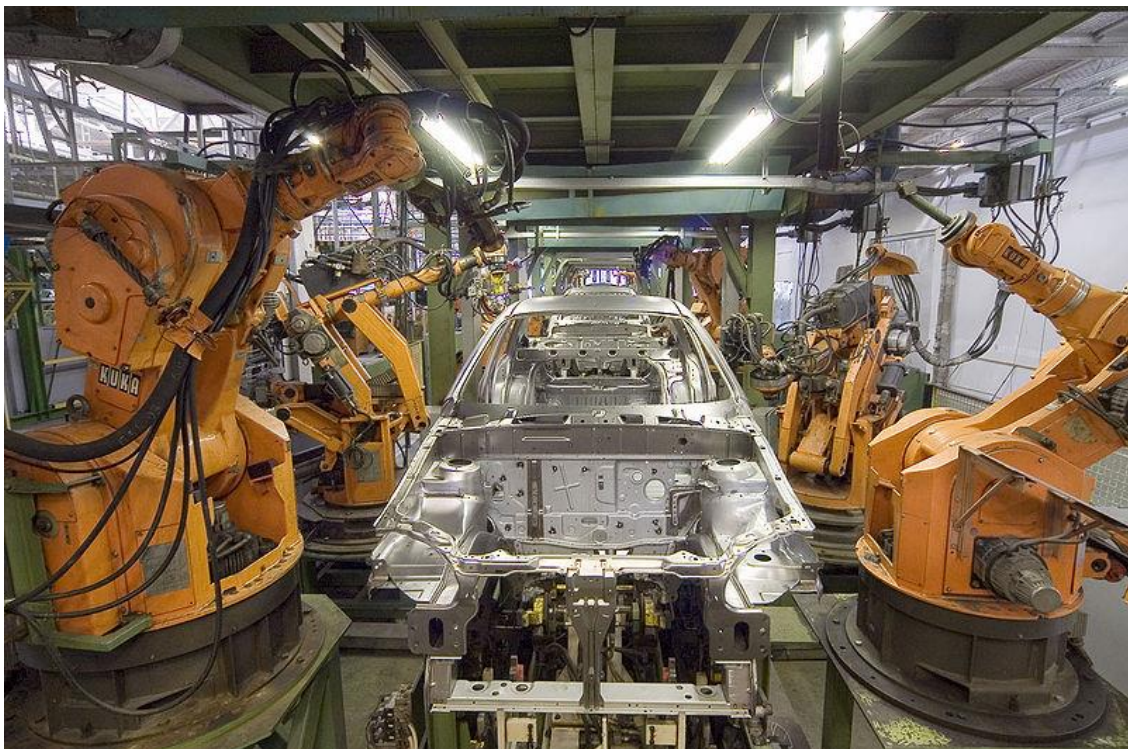


Abbildung 2: Industrieroboter in der Fertigung eines Automobilherstellers⁶

⁴ Vgl. (Joachim Hertzberg, 2012, S. 1f)

⁵ Siehe (Joachim Hertzberg, 2012, S. 2)

⁶ Siehe (Mixabest, 2007)

Bei mobilen Robotern ist die Umgebung variabel und kann sich jederzeit verändern. Der Roboter muss sich orientieren und auf veränderte Umwelteinflüsse reagieren können. Für das Erfassen der Umgebung sind verschiedene Sensoren notwendig. Um optimal umgebungsabhängig operieren zu können, werden die Sensordaten ausgewertet und anschließend die nächsten Aktionen ausgewählt. Im Vergleich zu Industrierobotern ist die auszuführende Aktion dadurch erst kurz vor dem Ausführungszeitpunkt im Detail bekannt.

Neben mobilen und stationären Robotern kann man zusätzlich die Unterscheidung nach Einsatzarten heranziehen⁷:

- Autonome mobile Roboter
- BEAM (**B**iology, **E**lectronics, **A**esthetics and **M**echanics)
- Humanoide Roboter
- Kognitive Roboter
- Laufroboter
- Portalroboter

Der für diese Studienarbeit eingesetzte Nao-Roboter ist von der Klassifizierung her ein humanoider Roboter. Das Aussehen eines humanoiden Roboters ist dem eines Menschen nachempfunden: Die meisten Humanoiden besitzen zwei Arme und zwei Beine. Das wirkt sich auf etwaige Gelenke und vor allem deren Positionen aus. Dadurch sind auch die Bewegungsabläufe an die eines Menschen angepasst, etwa das Gehen, Hinsetzen oder Aufstehen. Eine weitere Form des humanoiden Roboters ist der Androide, welcher dem Menschen täuschend ähnlich sieht.⁸

Einige der humanoiden Roboter, wie beispielsweise der in dieser Studienarbeit verwendete Nao-Roboter, können sprechen und gesprochene Sätze von einem Menschen verstehen.

⁷ Vgl. <https://de.wikipedia.org/wiki/Roboter> [Letzter Zugriff 07.05.2019]

⁸ Vgl. https://de.wikipedia.org/wiki/Humanoider_Roboter [Letzter Zugriff 07.05.2019]

2.1.3 Virtual Reality

Unter dem Begriff Virtual Reality versteht man die Darstellung einer digital erstellten, künstlichen Wirklichkeit. Eingesetzt wird die Technik sowohl in der Entertainment-Branche als auch in den Medien oder der Medizin. Der Trend prägt inzwischen zunehmend unseren Alltag und die Einsatzgebiete der Technologie werden immer vielseitiger. Beispielsweise kann heutzutage ein Flug in 360 Grad durch den Einsatz von Virtual Reality Hardware simuliert werden. Interessant ist außerdem der Einsatz dieser Technologie für die Durchführung von Schulungen für Operationstechniken der Medizin über eine virtuelle Realität.⁹

Mit der Erstellung einer virtuellen Umgebung wird es uns ermöglicht, in eine Realität computergenerierter Bilder und Sounds einzutauchen. Übertragen werden die Daten meist über Head-Mounted-Displays, die sogenannten Virtual Reality Brillen. Das in der VR-Brille eingebaute Display stellt künstlich erzeugte Bilder dar, die den Anwender vollständig umgeben. Weiterhin verfügen sie über Sensoren, welche die Lage und die Position des Anwenders bestimmen können und somit das freie Bewegen in der Umgebung ermöglichen.⁹ Oft gibt es neben der Bildübertragung zusätzlich Datenhandschuhe, durch welche der Anwender sich in der virtuellen Umgebung frei bewegen und mit den vorhandenen Objekten interagieren kann.¹⁰ Durch das Ermöglichen der virtuellen Interaktion wird der Effekt der Immersion erzeugt. Dieser beschreibt das vollständige Eintauchen in die virtuelle Realität und somit das Wahrnehmen der computergenerierten Wirklichkeit als reale Welt. Die Grenzen zwischen realer und virtueller Umgebung verschwimmen und die Realität sowie die Wahrnehmung der eigenen Person treten in den Hintergrund.¹¹

⁹ Vgl. <https://www.zukunftsinstitut.de/artikel/virtual-reality-die-erschaffung-neuer-welten/> [Letzter Zugriff: 20.02.2019]

¹⁰ Vgl. <https://wirtschaftslexikon.gabler.de/definition/virtuelle-realitaet-54243> [Letzter Zugriff: 20.02.2019]

¹¹ Vgl. <http://www.inztitut.de/blog/glossar/immersion/> [Letzter Zugriff: 20.02.2019]



Abbildung 3: Gegenüberstellung VR, AR und MR ¹²

Das Konzept der künstlich erzeugten Realität kann in verschiedene Ausprägungen auftreten: Virtual Reality, Augmented Reality und Mixed Reality. Abgegrenzt werden kann die Technologie der zuvor erklärten Virtual Reality von der verwandten Augmented Reality. Diese kennzeichnet sich durch die Erweiterung der realen Welt durch digitale Objekte in Echtzeit. Im Gegensatz zur virtuellen Umgebung befindet sich der Anwender in der Realität, die lediglich durch virtuelle Informationen erweitert wird. Bei der Virtual Reality hingegen wird die ganze Umgebung virtuell dargestellt.¹³ Eine weitere Ausprägungsmöglichkeit ist die sogenannte Mixed Reality. Hierunter versteht man eine Zusammenführung der realen Welt mit einer virtuellen Realität. Dabei verschwimmen die Grenzen zwischen synthetischen und natürlichen Elementen und werden nicht wie bei der Augmented Reality lediglich zusätzlich eingeblendet.¹⁴ Diese Studienarbeit beschäftigt sich ausschließlich mit der Umsetzung der Virtual Reality. Die Vorgehensweise zur Erstellung der virtuellen Umgebung wird in Kapitel 3.2.2 (Entwurf der virtuellen Umgebung) beschrieben.

¹² Siehe <http://arvr3dmodelling.com/virtual-reality-vs-augmented-reality-vs-mixed-reality/> [Letzter Zugriff: 12.02.2019]

¹³ Vgl. <https://www.virtual-reality-magazin.de/themen/augmented-reality-vr> [Letzter Zugriff: 12.02.2019]

¹⁴ Vgl. <https://www.elektronik-kompodium.de/sites/com/2210231.htm> [Letzter Zugriff: 12.02.2019]

2.1.4 Schichtenarchitektur

Schon in der Planungsphase von Software können Architekturen genutzt werden, um die einzelnen Softwarekomponenten besser zu strukturieren. Die Schichtenarchitektur strukturiert eine Anwendung durch einzelne Schichten, die sich durch ihren Aufgaben- und Zuständigkeitsbereich unterscheiden. Die Komponenten werden der Schicht zugeordnet, die zu ihren Aufgaben am besten passt. Im Normalfall dürfen Schichten nur auf die darunter liegenden Schichten zugreifen. Schnittstellen zwischen den Schichten dienen der Kommunikation und dem Zugriff auf die Funktionen einer Schicht. Durch die Unterteilung und die Einigung auf Schnittstellen können Schichten im späteren Verlauf geändert werden, ohne die gesamte Architektur anpassen zu müssen.¹⁵

Im Folgenden wird speziell die Fünf-Schichtenarchitektur (Five-Tier-Model), beginnend von der höchsten Schicht, vorgestellt.¹⁶

Client Tier

Die Clientschicht (Client Tier) ist die höchste Schicht und repräsentiert alle Geräte oder Systeme, die als Client auf das System zugreifen. Beispiele für Clients sind Web-Browser, Applikationen in Java und weiteren Sprachen, Netzanwendungen, WAP-Telefone und weitere Geräte.¹⁷

Presentation Tier

Die Präsentationsschicht (Presentation Tier) enthält die Logik für den Zugriff auf das System, indem es Anfragen von Clients entgegennimmt und Funktionalität wie Single-Sign-On oder Session Management bereitstellt. Diese Schicht greift auf die darunter liegende Businessschicht zu, um Antworten für die Clients zu generieren und zu senden. In der Präsentationsschicht können auch Elemente enthalten sein, die UI-Elemente generieren.¹⁷

¹⁵ Vgl. Alexander Schatten, Stefan Biffel, Markus Demolsky, Erik Gostischa-Franta, Thomas Östreicher, Dietmar Winkler, „Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen“, S. 211f

¹⁶ Vgl. Deepak Alur, John Crupi, Dan Malks, „Core J2EE Patterns: Best Practices and Design Strategies“, S. 120

¹⁷ Vgl. Deepak Alur, John Crupi, Dan Malks, „Core J2EE Patterns: Best Practices and Design Strategies“, S. 121

Business Tier

Die Businessschicht (Business Tier) enthält die meisten Geschäftsdaten und zentralisiert die Businesslogik der Anwendung. Dafür werden unter anderem Businessobjekte eingesetzt. Die Daten der Business-Objekte und Operationen werden über die darunter liegende Integrationsschicht angefordert beziehungsweise ausgeführt.¹⁸

Integration Tier

Die Integrationsschicht (Integration Tier) ist zuständig für die Kommunikation mit der darunter liegenden Ressourcenschicht. In der Integrationsschicht befinden sich Adapter und Konnektoren, die auf die Systeme in der Ressourcenschicht zugreifen, um Daten zu übertragen. Beispiele für Konnektoren sind JDBC, J2EE und andere Middleware.¹⁸

Resource Tier

Innerhalb der Ressourcenschicht (Resource Tier) befinden sich externe Systeme und Geschäftsdaten. Diese können in Mainframes, Legacy Systems und Datenbanken gespeichert sein, aber auch über Business-to-Business (B2B) Systeme oder Services abgerufen werden.¹⁸

¹⁸ Vgl. Deepak Alur, John Crupi, Dan Malks, „Core J2EE Patterns: Best Practices and Design Strategies“, S.121

2.2 Technische Grundlagen

2.2.1 Nao-Roboter

Nao ist ein programmierbarer humanoider Roboter, der von der Firma Aldebaran Robotics entwickelt wurde und im Jahr 2006 zum ersten Mal auf den Markt kam. Eingesetzt wird er vor allem zu Bildungs- sowie Forschungszwecken. Dies liegt insbesondere an der benutzerfreundlichen Entwicklungsumgebung des Nao-Roboters, welche die Programmierung des Roboters über Bausteine auch für Programmieranfänger ermöglicht. Der Roboter ist ein mittelgroßer humanoider Roboter mit einer Größe von circa 60 cm. Der Roboter hat sich seit seiner Einführung stets weiterentwickelt. Zum Zeitpunkt der Untersuchung im Frühjahr 2019 ist bereits die sechste Version auf dem Markt. Der NAO-Roboter hat sechs wichtige Charakteristika:

- 25 Freiheitsgrade: Der Roboter verfügt über eine hohe Bewegungsfreiheit, da er aus 25 Gelenken besteht, die es ihm ermöglichen, sich in seiner Umgebung zu bewegen und sich dieser anzupassen.
- Touch-Sensoren: Der Nao-Roboter ist mit sieben Touch-Sensoren ausgestattet. Diese befinden sich am Kopf, an den Händen und den Füßen. Der Roboter kann seine Umgebung somit wahrnehmen und sich im Raum lokalisieren.
- Vier Mikrophone und Lautsprecher: Die Mikrophone und Lautsprecher ermöglichen die direkte Interaktion mit den Anwendern.
- Spracherkennung: Der Roboter verfügt über Spracherkennungssysteme, die insgesamt zwanzig verschiedene Sprachen unterstützen.
- 2D-Kameras: Nao ist mit zwei Kameras ausgestattet, die Umrisse, Objekte und Menschen erkennen können.
- Voll programmierbare Plattform: Der Roboter ist frei programmierbar in den Sprachen C# und Python.¹⁹

¹⁹ Vgl. <https://www.softbankrobotics.com/emea/en/nao> [Letzter Zugriff: 14.04.2019]



Abbildung 4: Nao-Roboter ²⁰

In den Roboter ist das Betriebssystem „NAOqi“ integriert, welches das Verhalten des Roboters kontrolliert. Das Betriebssystem wird bei allen Robotern der Firma Aldebaran eingesetzt und bietet eine zuverlässige und plattformunabhängige Robotik-Umgebung. Der Roboter kann seinen ganzen Körper bewegen und auf Ereignisse in seiner Umgebung reagieren. Sein Bewegungsmodul beruht auf der inversen Gesamtkinematik, bei welcher verschiedene Parameter, wie beispielsweise das Gleichgewicht, bei der Bewegung der Gelenke berücksichtigt werden. Über die Kameras kann der Roboter seine Umgebung erkennen und mit Hilfe von Algorithmen zur Ortung und Erkennung von Gesichtern die Umgebung analysieren und mit ihr interagieren.²¹

²⁰ Siehe <https://www.generationrobots.com/de/401617-humanoider-roboter-nao-evolution-rot.html>
[Letzter Zugriff: 24.03.2019]

²¹ Vgl. Seo, Kisung „Using NAO – Introduction to interactive humanoid robots“, S. 136f

2.2.2 Choreographie Suite

Die „Choreographie Suite“ ist eine Desktop-Anwendung, die zur Steuerung des Nao-Roboters verwendet werden kann. Über die Software kann nach Angabe der zugehörigen IP-Adresse des Naos der Roboter direkt angebunden und sein Verhalten kontrolliert und gesteuert werden. Es können individuelle Applikationen erstellt werden, die beispielsweise Dialoge oder Verhalten enthalten, ohne selbst die Funktionalität in Code zu implementieren. Die Anwendung verfügt über die folgenden Funktionen:

- Erstellen von Animationen, Verhalten und Dialogen
- Testen der erstellten Funktionen auf einem simulierten oder realen Roboter
- Überwachung und Kontrolle eines Nao-Roboters
- Anreicherung der vordefinierten Verhaltensskripte mit selbst implementiertem Python-Code.²²

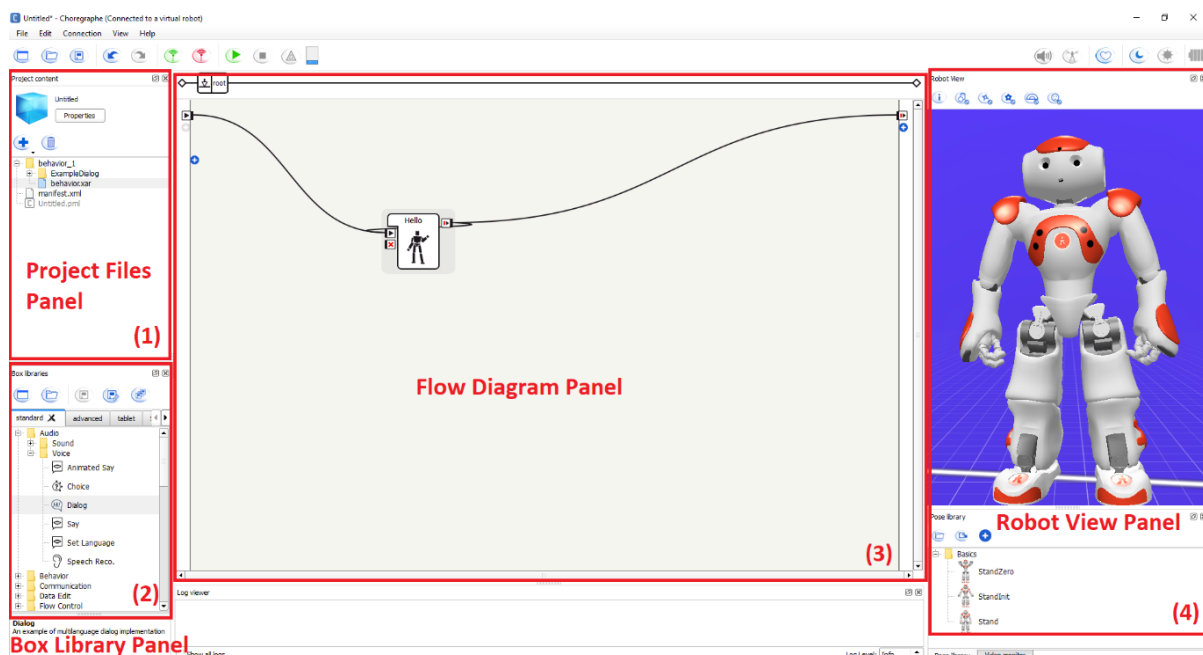


Abbildung 5: Oberfläche Choreographie Suite

²² Vgl. http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe_overview.html
[Letzter Zugriff: 11.04.2019]

Die Abbildung 5 zeigt die Struktur des Software-Tools. Die Benutzeroberfläche lässt sich in vier grundlegende Bereiche gliedern. Das Project Files Panel (1) zeigt die vom Anwender konfigurierten Einstellungen des angelegten Projektes sowie alle gespeicherten Dateien innerhalb des Projekts. Das Box Library Panel (2) dient zur Übersicht aller verfügbaren Box-Bibliotheken. Der Anwender kann diese vordefinierten Skripte per Drag-and-Drop in das Flow Diagram Panel (3) ziehen. Die Box-Bibliotheken stellen wiederverwendbare Komponenten dar, die das Verhalten des Nao-Roboters steuern können. Viele Standard-Funktionen, wie beispielsweise Reden oder Gehen, sind bereits vordefiniert und können direkt verwendet werden. Sollte der Anwender spezifischere Aktionen brauchen, besteht die Möglichkeit, eigene Box-Bibliotheken anzulegen und das Verhalten des Roboters über ein Python-Skript zu definieren. In dem Flow Diagram Panel kann das gewünschte Verhalten des Nao-Roboters definiert werden. Die Box-Bibliotheken können in einem Flussdiagramm miteinander verknüpft und an den Roboter gesendet werden. Der Nao-Roboter führt dann in der definierten Reihenfolge die Aktionen aus. Das Verhalten des Roboters kann ebenso in dem Robot View Panel (4) direkt nachvollzogen werden. Hier werden die genauen Gelenkpositionen des Roboters angezeigt.

2.2.3 HTC Vive Pro

Die „HTC Vive Pro“ ist ein Virtual Reality Headset des für Smartphone bekannten Herstellers HTC, das in Kollaboration mit dem Softwareunternehmen „Valve“ entwickelt wurde. Vorgestellt wurde der Vorgänger HTC Vive erstmalig im Jahr 2015 bei dem Mobile World Congress, die Markteinführung erfolgte im nachfolgenden Jahr. Das neue Modell HTC Vive Pro, welches bei der Durchführung dieser Studienarbeit verwendet wurde, kam drei Jahre später mit verbesserter Technologie auf den Markt.



Abbildung 6: HTC VIVE Pro²³

Ausgestattet ist die HTC Vive Pro mit einer High-Fidelity-Grafik, womit realistisch Grafiken virtueller Umgebung dargestellt werden können. Das eingebaute Dual-OLED-System stellt pro Auge 1440 x 1600 Pixel dar. Über die 32 eingebauten Headset-Sensoren wird eine 360 Grad Bewegungsverfolgung ermöglicht, sodass der Anwender ringsum von den virtuell erstellten Bildern umgeben ist. Die virtuelle räumliche Ebene kann bis zu einer Größe von 5 x 5 Metern angezeigt werden. Um das Erlebnis abzurunden, sind zusätzlich zu dem Bildschirm Kopfhörer im Kopfbügel der neuen Version des Headsets integriert. ²⁴

²³ Siehe <https://www.vive.com/de/product/vive-pro/> [Letzter Zugriff: 24.04.2019]

²⁴ Vgl. https://www.vive.com/de/product/vive-pro-starter-kit/?gclid=Cj0KCQjwyoHIBRCNARIsAFjKJ6Cu3-S1YzX5OfBx3iI5psCH8nb8suJPLiMYi_mo6je9G4fVAgkNCWcaAvXvEALw_wcB [Letzter Zugriff: 05.04.2019]

2.2.4 Unity

Die Software „Unity“ ist eine Game-Engine, die „Spieleersteller mit dem notwendigen Satz von Funktionen versorgt, um schnell und effizient Spiele erstellen zu können“ ²⁵. Game-Engines bieten das Gerüst für die Entwicklung von 2D- sowie 3D-Spielen. Die Anwendung Unity bietet Ressourcen, wie zum Beispiel Grafiken oder Audio-Dateien, um grafische Oberflächen zu entwerfen. Zusätzlich zu den von Unity angebotenen Ressourcen, können 3D-Modelle von Maya oder Photoshop oder anderen Anwendungen importiert werden. Weiterhin können beispielsweise Animationen, Beleuchtungen und Sound-Effekte zu den Szenen hinzugefügt werden, um diese beliebig nach den Anforderungen des Entwicklers zu gestalten. Ein wichtiges Feature des Software-Tools ist das Scripting. Scripts ermöglichen es, dem Spiele-Entwickler die Logik von den Komponenten des Spiels zu definieren. Unity vereinfacht die Entwicklung, indem es vordefinierte Scripts anbietet, die beispielsweise das Umschauen des Operators in dem virtuellen Raum durch eine VR-Brille ermöglichen. ²⁶

2.2.4.1 Struktur der Benutzeroberfläche

Die Abbildung 7 zeigt die Standardoberfläche von Unity. Die Benutzeroberfläche ist untergliedert in verschiedene Bereiche, welche das Erstellen einer Spieleumgebung erleichtern.

²⁵ Siehe <https://unity3d.com/de/what-is-a-game-engine> [Letzter Zugriff: 05.04.2019]

²⁶ Vgl. Ebenda

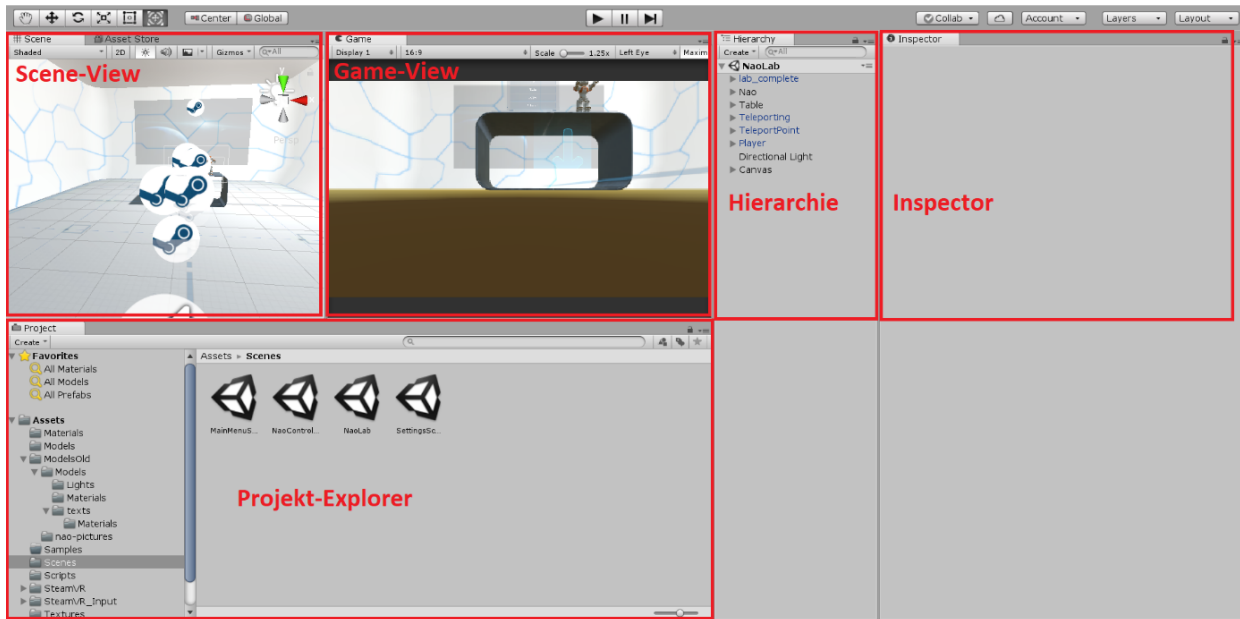


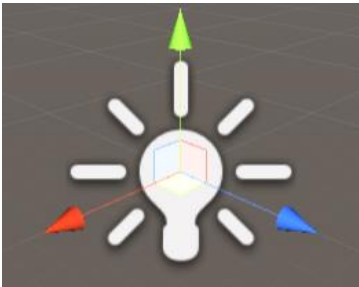
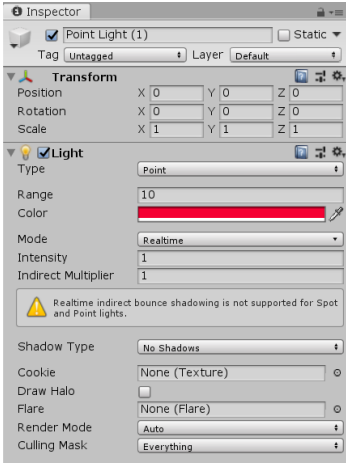
Abbildung 7: Benutzeroberfläche Unity

Mit Hilfe der Toolbar von Unity kann der Anwender verschiedene Aktionen in der Szene ausführen, beispielsweise kann das Bild geschwenkt oder skaliert werden. Die Scene-View stellt die designte Szene dar. In diese Szene können nach Belieben GameObjects und weitere Komponenten hinzugefügt werden. Die Eigenschaften dieser Objekte werden im folgenden Kapitel erläutert. Durch das Betätigen des Play-Knopfes in der Toolbar wird das Spiel gestartet und in der Game-View angezeigt. Die Game-View ist der sogenannte Spielmodus, in welchem das konzipierte Spiel getestet werden kann. Hier kann sowohl ein 2D-Spiel direkt über die Anwendung oder ein 3D-Spiel nach dem Verbinden einer VR-Brille über das Head-Mounted-Display getestet werden. Der Projekt-Explorer stellt eine Übersicht der Projektdateien dar und ermöglicht es, neue Dateien wie beispielsweise ein mit Maya erstelltes 3D-Modell per Drag-and-Drop einzufügen. Die Hierarchie hingegen zeigt alle Spielobjekte der dargestellten Szene an und ermöglicht es, neue Game-Objekte hinzuzufügen. In dem Inspector werden die Eigenschaften der ausgewählten Hierarchie-Objekte angezeigt, so dass diese nach den Wünschen des Anwenders konfiguriert werden können.²⁷

²⁷ Vgl. <https://msdn.microsoft.com/de-de/magazine/dn759441.aspx> [Letzter Zugriff: 12.03.2019]

2.2.4.2 Grundlegende Building-Blöcke

Unity hat drei grundlegende Building-Blöcke, die zum Aufbau von Spieleumgebungen verwendet werden können. Die nachfolgende Tabelle erklärt die verschiedenen Arten von Blöcken, die in Unity zur Verfügung stehen. Zusätzlich wird anhand eines Beispiels die Funktionalität jedes Blocks dargestellt.

<p>GameObjects</p>	<p>GameObjects sind fundamentale Objekte in Unity. Sie umfassen alle Arten von Objekten in dem erstellten Spiel. Beispielsweise gehören dazu Lichtquellen, Requisiten und Spezialeffekte. Sie fungieren dabei lediglich als Container für Komponenten und verfügen noch über keine Funktionalität.</p> <p>Beispiel: Point Light</p> 
<p>Komponenten</p>	<p>Komponenten können zu den Spiel-Objekten hinzugefügt werden. Sie ermöglichen es, dass das Verhalten der verknüpften GameObjects gesteuert werden kann.</p> <p>Beispiel: Eigenschaften des Point Lights</p> 

Variablen	<p>Variablen sind Eigenschaften, die von den Entwicklern von Spieleumgebungen frei nach ihren Bedürfnissen angepasst werden können. Sie werden im Inspector-Fenster angezeigt und können dort editiert werden.</p> <p>Beispiel: Farbe, Position und Intensität des Point Lights</p>
------------------	---

Tabelle 1: Building-Blöcke in Unity ²⁸

2.2.4.3 SteamVR-Plugin

Unity verfügt außerdem über die Möglichkeit, Plugins zu installieren. Zur Erstellung virtueller Umgebungen ist das Virtual Reality System „SteamVR“ notwendig. Das Plugin wurde von dem Softwareunternehmen Valve entwickelt und erlaubt es Entwicklern, die Game-Engine Unity mit dem System SteamVR zu koppeln. Das Plugin ermöglicht es, virtuelle Umgebungen unter Verwendung von VR-Hardware zu erleben. Die Hauptfeatures von SteamVR sind das Laden von Modellen der VR-Controllern, Input-Handling der Controller und die Abschätzung der Bewegung der Hände des Operators bei der Benutzung der Controller.²⁹

²⁸ Vgl. <https://unity3d.com/de/programming-in-unity> [Letzter Zugriff: 06.04.2019]

²⁹ Vgl. <https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647> [Letzter Zugriff: 06.04.2019]

2.2.5 Maya

„Maya“ ist ein Softwareprodukt der Firma Alias, welche am 10. Januar 2006 von Autodesk übernommen wurde. Maya wird für die 3D-Visualisierung und Animation von der Film- und Fernsehindustrie sowie von Computerspiel-Herstellern eingesetzt. Das Erstellen von 3D-Modellen wird auch in der Industrie, Architektur und Forschung eingesetzt. Maya zählt zu den bekanntesten Softwareprodukten aus den Bereichen 3D-Modellierung, Computeranimation und Rendering.³⁰ Die Software bietet unter anderem folgenden Funktionsumfang:

Steuerungssprache MEL

Die interne Steuerungssprache MEL (Maya Embedded Language) ist eine Skriptsprache und ermöglicht die Automatisierung bestimmter Aufgaben, die grafische Umgestaltung des Editors und weitere Anpassungen.³¹

Python Unterstützung

Seit Version 8.5 wird die Programmiersprache Python mit einem eigenen Interpreter unterstützt. Dadurch können Python gestützte Plugins und Skripte zur Funktionserweiterung entwickelt werden. Über Python können außerdem fremde Bibliotheken eingebunden werden, um auf deren Funktionen zuzugreifen und somit Schnittstellen zu operativen Systemen zu schaffen.³¹

Modularer Aufbau

Mayas modularer Aufbau ermöglicht das Hinzufügen und Nutzen einer Vielzahl von Funktionen. Durch das Modul „Maya Fur“ lassen sich zum Beispiel realistisch aussehendes Fell, Haarflächen oder Gras darstellen. Die Simulation von Flüssigkeiten oder Gasen kann über das Modul „Maya Fluids“ bewerkstelligt werden. Ein weiteres Modul ist zum Beispiel „Maya Cloth“, mit dem Kleidungsstücke und Stoff simuliert werden können.³¹

³⁰ Vgl. https://www.autodesk.de/products/maya/overview_ [Letzter Zugriff: 02.05.2019]

³¹ Vgl. https://en.wikipedia.org/wiki/Autodesk_Maya [Letzter Zugriff: 02.05.2019]

Rendering

Das Rendering von Maya kann durch die Wahl verschiedener implementierter Renderers angepasst werden. Maya bietet auch einen nativen Renderer namens „Maya Software“, der für die Berechnung aller Objekte zuständig ist. Dieser ist qualitativ hochwertig, aber arbeitet im Vergleich zu anderen Renderern vergleichsweise langsam und nicht immer physikalisch korrekt. Mit Maya lässt sich auch ein hardwarebasierter Renderer nutzen, dieser hat den Namen „Maya Hardware“. Der Renderer bindet die 3D-Grafikkarte in den Rendering-Prozess ein und ist dadurch erheblich schneller, bei jedoch geltenden Hardwarelimitierungen, die zum Beispiel die Texturgröße einschränken. Ein weiterer Renderer ist der aus einem deutschen Entwicklerstudio stammende „Mental Ray“. Dieser ermöglicht die annähernd physikalische Darstellung von Beleuchtung, Tiefen- und Bewegungsunschärfe und die Verwendung von Ray Tracing. Maya bietet auch einen Renderer für Vektor Rendering, der meistens für das Erstellen von Webanimationen mit Flash genutzt wird. Seit 2017 ist der Renderer „Arnold“ in Maya integriert und liefert bessere Ergebnisse für Ray Tracing, Fur-Rendering sowie Bewegungsunschärfe und Volumenrendering.^{32 33}

Es gibt einige bekannte Filme, die mithilfe von Maya erstellt wurden. Diese sind unter anderem „Findet Nemo“ und „Shrek“. Maya wird auch zur Berechnung von Fantasy Figuren in realen Filmaufnahmen verwendet, wie beispielsweise das Geschöpf Gollum bei „Herr der Ringe“.³²

³² Vgl. [https://de.wikipedia.org/wiki/Maya_\(Software\)](https://de.wikipedia.org/wiki/Maya_(Software)) [Letzter Zugriff: 02.05.2019]

³³ Vgl. https://en.wikipedia.org/wiki/List_of_Maya_plugins [Letzter Zugriff: 02.05.2019]

3 Entwurf

Dieses Kapitel dokumentiert die Vorgehensweise für die Entwicklung der Lösung und verschiedene Konzepte. Zu den Konzepten gehören unter anderem die Softwarearchitektur und Designentwürfe des virtuellen Raums.

3.1 Theoretische Vorgehensweise

3.1.1 Anforderungsanalyse

Wie in der Einführung (siehe 1.1 Aufgabenstellung und Zielsetzung) dargestellt, ist das Ziel dieser Studienarbeit die Schaffung einer multimodalen Telepräsenz zwischen einem menschlichen Benutzer und einem entfernten Roboter. Dabei soll eine virtuelle Umgebung geschaffen werden, in welcher der Benutzer einerseits Steuerungseingaben an den Roboter senden und andererseits Bild- und Toninformationen vom entfernten Roboter erhält. Über eine Virtual Reality Brille soll der Benutzer in die Umgebung eintauchen und darüber interagieren können.

Die virtuelle Umgebung soll folgende Features bereitstellen:

- Die IP-Adresse des zur Verfügung stehenden Roboters soll angepasst werden können.
- Der Anwender soll eine Verbindung zu einem beliebigen Nao aufbauen können.
- Der Nao soll über den virtuellen Raum angesteuert werden können, um Aktionen wie Hinsetzen, Aufstehen oder Laufen zu ermöglichen.
- Das Kamerasignal des Nao soll im virtuellen Raum angezeigt werden. Dadurch kann der Benutzer durch die Umgebung navigieren, in der sich der Nao befindet.
- Das Tonsignal des Nao-Roboters soll im virtuellen Raum wiedergegeben werden.
- Die Sprachausgabe auf dem Nao soll über einen Menüpunkt im virtuellen Raum ermöglicht werden.

3.1.2 Geplantes Vorgehen

Die Anforderungen werden mit dem Team besprochen, um Unklarheiten frühzeitig zu beseitigen. Für die Entwicklung wird ein Projektplan entworfen und wichtige Meilensteile festgelegt.

Für den frühzeitigen Entwurf der Anwendung werden Designentwürfe (Mockups) entwickelt, um die unterschiedlichen Vorstellungen im Team besser visualisieren zu können. Diese können einfacher diskutiert werden und sollen Missverständnissen vorbeugen. Anhand der Mockups wird ein Modell für den virtuellen Raum konzipiert, der die Steuerung für den Roboter enthält. Die gewählte Architektur soll die Anwendung in Komponenten untergliedern, so dass diese unabhängig voneinander verändert und angepasst werden können. Gleichzeitig ermöglicht die Architektur eine bessere Übersicht der Anwendung. Die zum Planungszeitpunkt feststehenden Funktionen werden in Komponenten eingeteilt. Darüber hinaus werden Schnittstellen zwischen den Komponenten festgelegt, so dass diese durch Erweiterungen oder Rückschläge nachträglich verändert werden können. Die Aufgaben aus dem Projektplan werden in kleinere Aufgaben zerlegt, so dass diese den Komponenten zugeordnet werden können. Jeder Entwicklungsstand wird in ein Git-Repository hochgeladen, so dass der Projektfortschritt bei jedem Teammitglied immer auf dem aktuellen Stand ist. Dies soll die Parallelentwicklung fördern und einem möglichen Datenverlust vorbeugen. Durch regelmäßige Teammeetings soll über den aktuellen Entwicklungsfortschritt kommuniziert werden, damit jedes Teammitglied über die aktuellen Entwicklungen informiert wird. Durch die Treffen können auch Ratschläge und Verbesserungen durch das Team eingeholt werden.

Zur Einarbeitung in die Thematik werden zunächst Informationen über die vorhandene Hardware und die Ansteuerung dieser gesammelt. Anschließend werden erste Programmiersuche mit dem Nao über die in Kapitel 2.2.2 beschriebene Software Choreographie Suite gestartet und eine virtuelle Testumgebung in der Game-Engine Unity erstellt. Durch Tests mit der Virtual Reality Brille, der HTC Vive Pro, soll die korrekte

Funktionsweise der Hardware verifiziert werden. Nachdem erste Erfahrungen in der Ansteuerung des Nao-Roboters sowie der Erstellung von virtuellen Umgebungen mit Hilfe von Unity gemacht werden, beginnt die Projektkonzeption. In der Konzeptionsphase werden Vorstellungen zum Design der virtuellen Umgebung festgehalten, verglichen und diskutiert. Des Weiteren werden Überlegungen zum Verbindungsaufbau und Übertragen der Daten zum Roboter gemacht.

Die Entwicklung der virtuellen Umgebung und die Ansteuerung des Nao-Roboters werden anschließend in drei Implementierungsphasen umgesetzt. Nachdem Grundkenntnisse über die Ansteuerung des Nao-Roboters sowie die Erstellung virtueller Umgebungen gesammelt wurden, kann die Phase der Konzeption einer virtuellen Umgebung eingeleitet werden. In dieser Phase liegt der Fokus auf der Erstellung einer virtuellen Umgebung mit Hilfe der Software Unity für die HTC Vive Pro. Außerdem soll über vorhandene Schnittstellen eine Verbindung der virtuellen Umgebung mit dem Nao-Roboter ermöglicht werden. Die erste Implementierungsphase soll als Grundlage für die Ermöglichung der Telepräsenz dienen.

Die nächste Implementierungsphase hat den Schwerpunkt, die Telepräsenz umzusetzen. Hierbei soll das Kamera- sowie Tonsignal des Nao-Roboters an den Anwender über die VR-Brille übertragen werden. Innerhalb der virtuellen Umgebung wird ein Videostream des Nao-Roboters angezeigt, so dass der Operator nachverfolgen kann, welche Aktionen der Roboter ausführt. Über die virtuelle Umgebung kann der Operator dem Roboter Anweisungen geben, wie beispielsweise „Gehen“ oder „Hinsetzen“, die der Nao umsetzen soll. In der finalen Implementierungsphase wird die Steuerung des Roboters implementiert. Innerhalb der virtuellen Umgebung soll es möglich sein, den Roboter über die Datenhandschuhe der VR-Brille zu steuern. Durch das Abfangen der Signale der Controller kann der Nao-Roboter direkt von dem Anwender gesteuert werden.

3.2 Konzeption

3.2.1 Aufbau der Anwendung

Die Anwendung wurde nach der Fünf-Schichtenarchitektur (siehe 2.1.4 Schichtenarchitektur) konzipiert. In der nachfolgenden Abbildung ist der Architektur-Entwurf zu sehen.

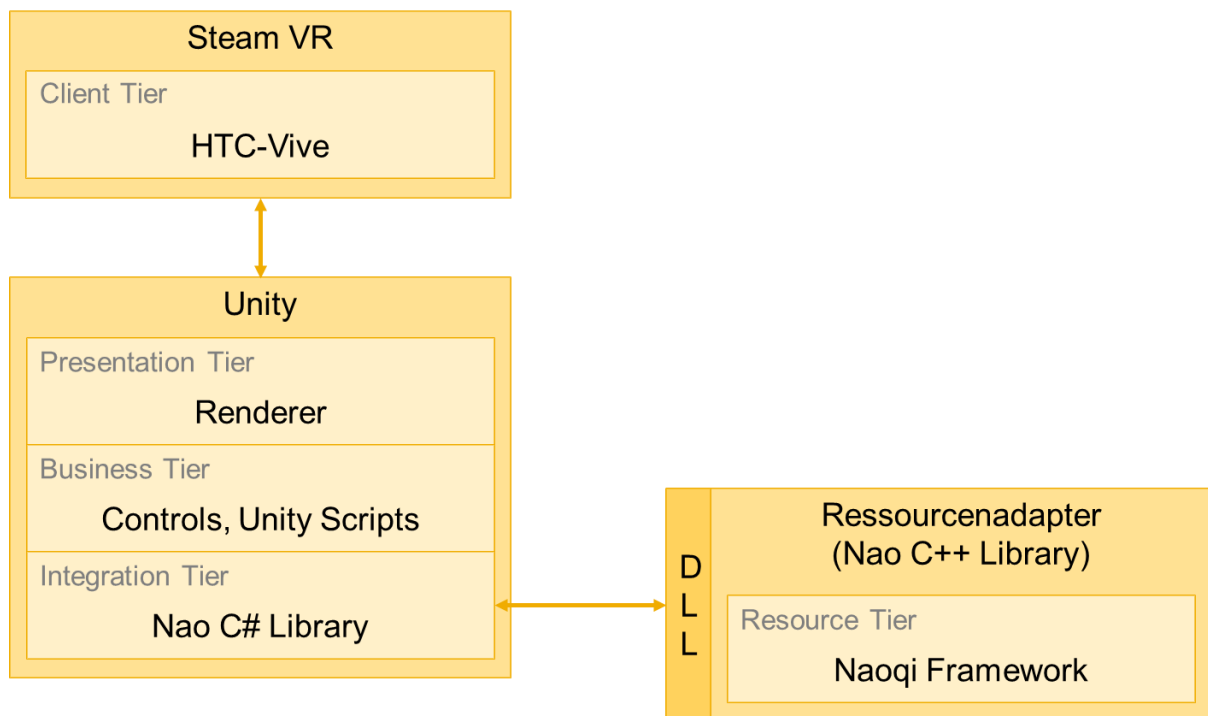


Abbildung 8: Architektur-Entwurf

Resource Tier

Die Ressourcenschicht (Resource Tier) ist über einen Ressourcenadapter für das Naoqi Framework integriert. Dieser Adapter realisiert eine DLL-Schnittstelle, über welche Aktionen auf dem Nao gestartet werden können. Diese sind u.a. Laufen, Gehen oder Sprechen. Über den Adapter wird auch auf die Kamera und das Mikrofon des Nao zugegriffen.

Integration Tier

Die Integrationsschicht (Integration Tier) ist für die Kommunikation mit dem Ressourcenadapter zuständig. Dabei soll die Einbindung des Frameworks in Unity sichergestellt werden. Funktionen in dieser Schicht rufen Routinen in der DLL auf und fungieren als Erweiterung von Unity.

Business Tier

Die Businesslogik (Business Tier) übernimmt die Datenlogik und Steuerung in Unity. Speziell entwickelte Skripte in dieser Schicht ermöglichen die Interaktion des Benutzers mit der in der Integrationsschicht implementierten Erweiterung von Unity.

Presentation Tier

Die Präsentationsschicht (Presentation Tier) wird über Unity realisiert und enthält sowohl sämtliche Objekte zur Generierung des Virtuellen Raumes als auch Elemente für die Grafische Benutzeroberfläche (GUI).

Client Tier

Die Clientschicht (Client Tier) übernimmt die Darstellung der in der Präsentationschicht generierten Oberfläche und wird über die HTC-Vive Brille realisiert. Die Verbindung übernimmt das SteamVR Framework.

3.2.2 Entwurf der virtuellen Umgebung

Für das Design der virtuellen Umgebung wurden verschiedene Entwürfe entwickelt. Diese werden im folgenden Abschnitt genauer vorgestellt.

Head-Display

Bei diesem Entwurf wird der virtuelle Raum so dargestellt, als befände sich der Benutzer im Kopf des Nao und würde durch seine Augen sehen. Links unten ist ein Abbild des Nao, das Informationen über die einzelnen Gelenkpositionen darstellen soll. Der Benutzer soll dadurch erkennen können, in welcher Lage sich der Nao befindet und welche Aktionen aktuell ausgeführt werden. Rechts oben ist ein Menü für die Steuerung zu sehen. Durch dieses Menü soll der Nao bewegt werden können.

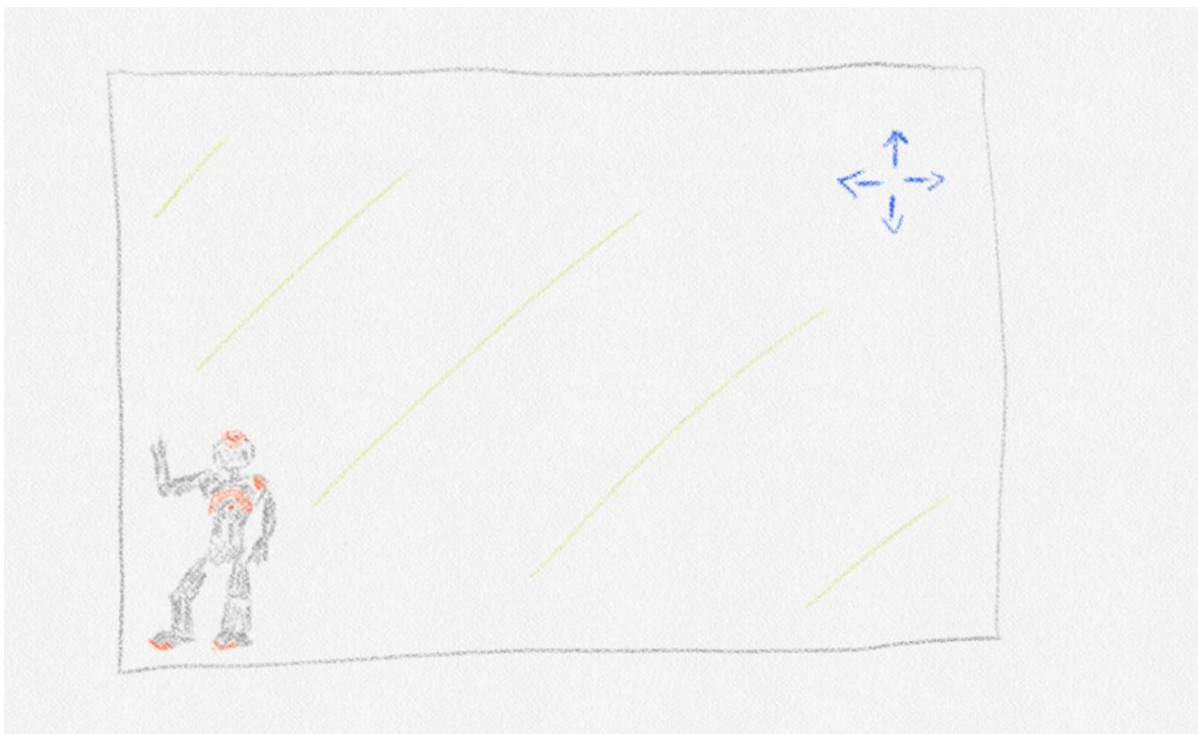


Abbildung 9: Entwurf Head-Display

Nao-Labor

Bei diesem Entwurf wurde ein Labor entwickelt, in welchem sich der Benutzer aufhält, während er den Nao steuert. In der Mitte des Raums befindet sich ein Tisch. Über ein Menü auf dem Tisch soll der Benutzer Steuerungseingaben an den Nao senden können, um diesen zu bewegen. Neben dem Menü steht ein Modell des Nao, welches ein gegenwärtiges Abbild des Roboters darstellt. Dieses soll dem Benutzer ermöglichen, die aktuell ausgeführte Aktion und Lage des Roboters zu erkennen. An der Wand hinter dem Tisch befindet sich ein Bildschirm, auf dem das aktuelle Kamerabild des Nao-Roboters dargestellt wird. Der Benutzer kann direkt sehen, was der Nao aufzeichnet. Im virtuellen Raum kann sich der Benutzer durch Teleportation über die Steuerung der HTC Vive bewegen.

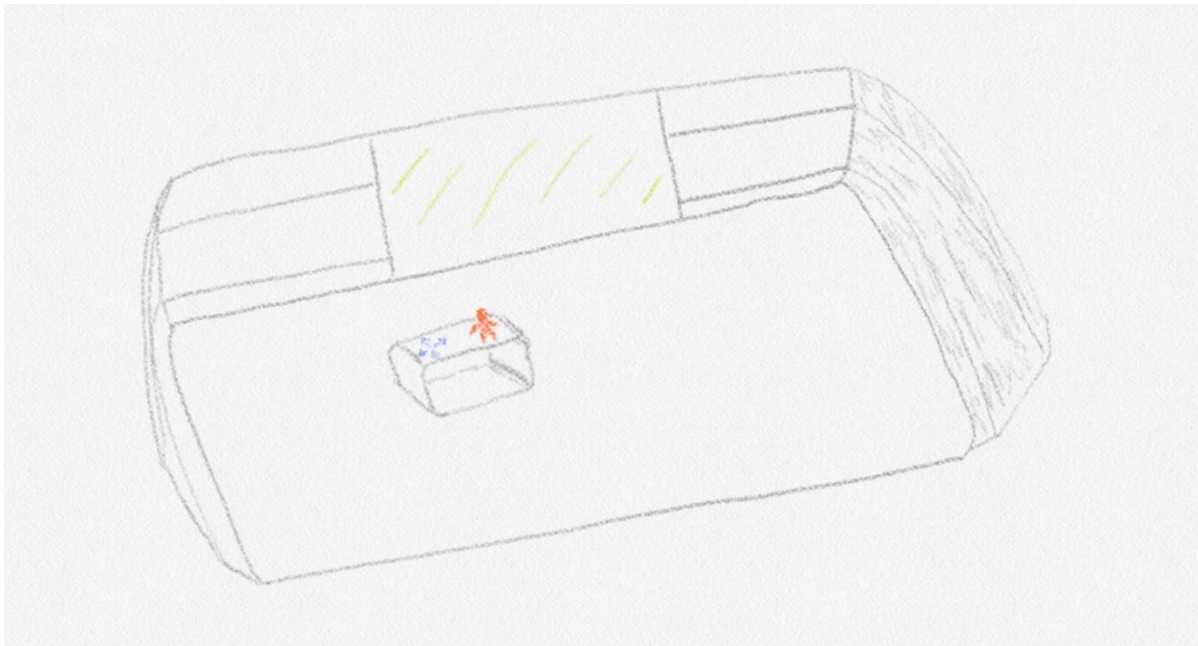


Abbildung 10: Entwurf Labor

Willkommensbildschirm (Hauptmenü)

In diesem Entwurf ist ein Hauptmenü auf einer einzelnen Plattform zu sehen. Der virtuelle Raum soll einen Ort zwischen den Welten darstellen, an welchen der Benutzer gebracht wird, wenn er die Anwendung startet. Zu diesem Zeitpunkt wurde noch keine Verbindung mit einem Nao-Roboter hergestellt. Das Menü soll dem Benutzer einen übersichtlicheren Einstieg ermöglichen, da Verbindungsparameter wie beispielsweise die IP-Adresse eingegeben werden können. Dadurch kann sich der Benutzer mit verschiedenen Nao-Robotern verbinden.

Über einen Menüeintrag wird eine Verbindung zum Nao erstellt und der Benutzer wird in einen weiteren virtuellen Raum für die Fernsteuerung des Roboters teleportiert.

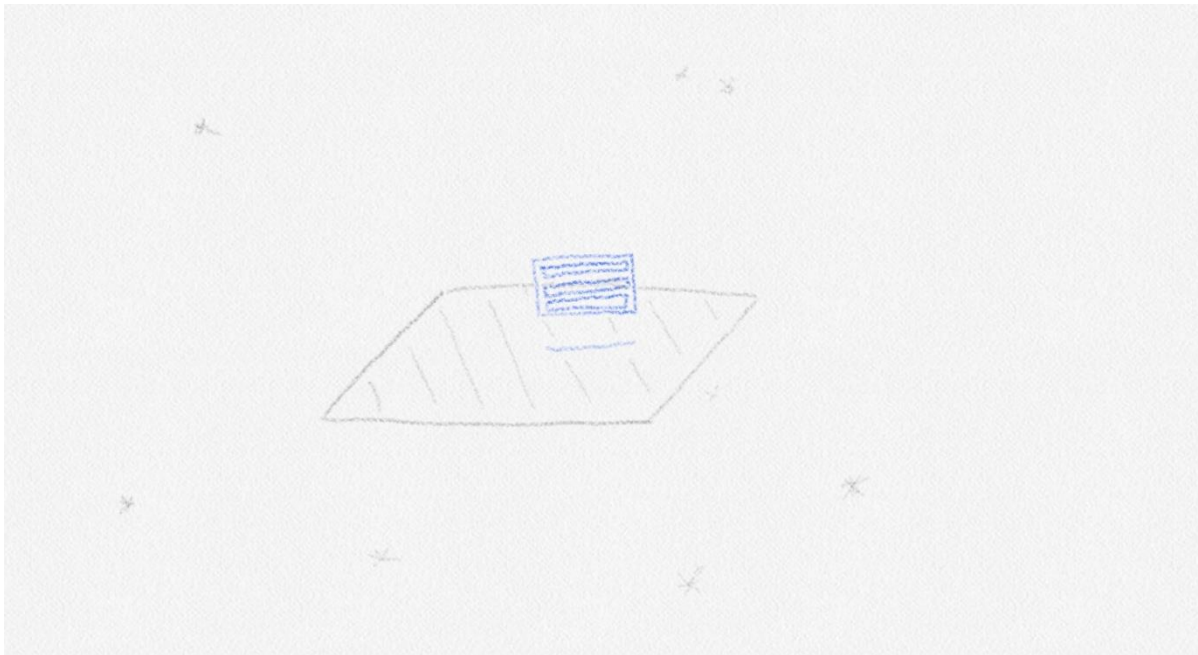


Abbildung 11: Entwurf Willkommensbildschirm

Entscheidung

Umgesetzt werden soll der eben dargestellte Willkommensbildschirm, da er einen Einstiegspunkt in die Anwendung bietet. Der Entwurf bietet zusätzlich einen passenden Übergang zwischen der realen und der virtuellen Welt. Bestehend aus einem Menü und einer Plattform, auf welcher der Benutzer steht, ist er ein übersichtlicher Raum. Der Einstieg in die virtuelle Welt wirkt sich somit nicht belastend oder überfordernd auf den Benutzer aus.

Als virtuelle Umgebung für die Steuerung fiel die Entscheidung auf das Nao-Labor. Durch die Schaffung eines eigenen Raumes, erhält der Benutzer einen Bezugspunkt für das Auge. Im Vergleich zum Head-Display ist dies zwar ein virtueller Raum, jedoch hängt der Blickwinkel vom Benutzer und nicht vom Roboter ab. Fällt der Roboter auf den Boden oder bewegt sich schnell, kann das zu Schwindelgefühl und Gleichgewichtsstörungen führen. Der Bezugspunkt des Auges ist in diesem Fall das bewegte Kamerabild. Beim Nao-Labor wird die Kamera auf einem Bildschirm dargestellt. Somit handelt es sich um einen getrennten Bereich und der Bezugspunkt des Auges bleibt das Labor. Ein weiterer Punkt ist die Auflösung des übertragenen Bildes: Bei schlechter Qualität sind die Pixel des Bildes sichtbar. Beim Head-Display könnte ein schlechtes Bild zu Kopfweh führen, da die Umgebung nicht wie gewohnt scharf, sondern in grobe Raster unterteilt wird. Auf dem Bildschirm im Nao-Labor fällt die schlechte Bildqualität zwar negativ auf, hat aber keine gesundheitlichen Schäden zur Folge.

4 Implementierung

Dieses Kapitel befasst sich mit der Umsetzung der geplanten Anwendung und der zuvor konzipierten virtuellen Räume.

4.1 Erstellung der virtuellen Umgebung

4.1.1 Design der 3D-Objekte

Für die virtuellen Räume wurden dreidimensionale Objekte entworfen. Die Räume wurden mithilfe dieser Objekte zusammengesetzt. Dabei wurde gezielt ein futuristisches Design gewählt und die Objekte mit Texturen eingefärbt. Die erstellten Texturen sind fortlaufend und können ohne Unterbrechung nebeneinandergelegt werden. Texturen, die diese Eigenschaft aufweisen, werden auch *seamless* oder *tileable* genannt. Die Texturen sind durch verschiedene Science-Fiction Filme wie zum Beispiel Starwars, Star Trek beziehungsweise Computerspiele wie ELEX inspiriert.

Labor

Das Labor wurde aus mehreren Objekten zusammengesetzt. Diese lassen sich konkret in Wand-, Boden- und Deckenstücke, sowie Dekorobjekte unterteilen.

- Zu den Wandstücken gehören gerade Wände und Wände mit 90°-Ecken. Diese können durch Rotation einen viereckigen Raum abbilden. Die blauen Linien auf der Textur (siehe Abbildung 12: Design-Prozess in Maya), gehen an den Rändern nahtlos ineinander über.
- Zu den Bodenstücken gehören drei Objekte: Mittel-, Rand- und Eckstücke. Die Randstücke besitzen einen durchgehenden Streifen, der die Randstellen markiert. Dieser grenzt an den unteren Teil der Wandstücke und soll den Übergang hervorheben.
- Das Deckenstück besteht aus einem einzigen Objekt und besitzt keine spezielle Textur. Das Element ist daher komplett weiß. Wie bei den Wänden wurden die Seitenkanten nach innen gebogen, um die entstehenden Ecken abzurunden.

- Die Dekorobjekte bestehen aus einem abgerundeten Tisch, einem Menü und einem Modell des Nao-Roboters. Bis auf das Menü wurden alle Objekte aus bestehenden Objekten zusammengesetzt. Der Tisch ist ein Standardobjekt aus SteamVR. Der Roboter stammt von der Herstellerfirma des Nao, Aldebaran.

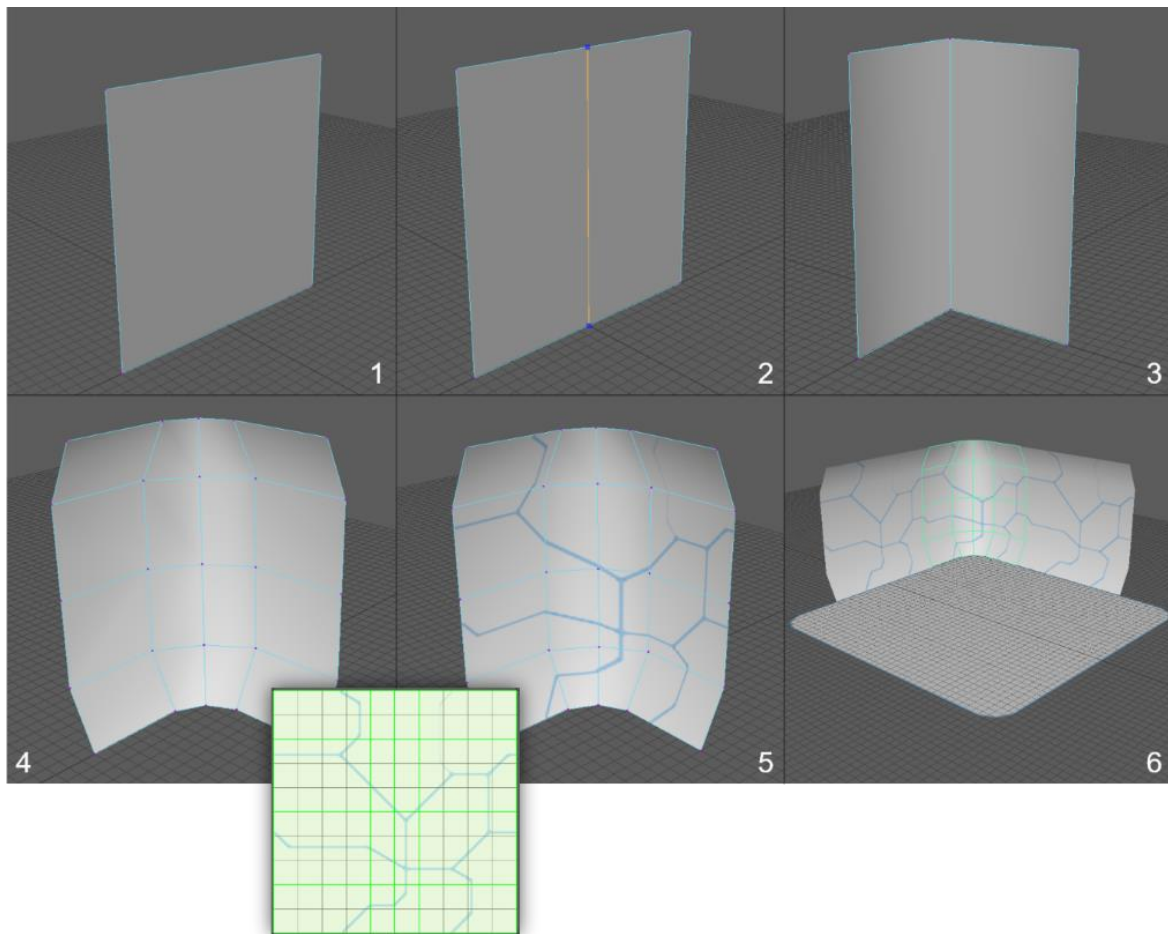


Abbildung 12: Design-Prozess in Maya

Die Objekte für die Wände, den Boden und die Decke wurden mit Maya erstellt. Das Ausgangsobjekt war für alle Objekte ein elementares Quadrat. Im Designprozess wurden weitere Punkte eingeführt und Kanten geschnitten. Durch Rotationen und Transformationen wurden die Objekte in ihre Form gebracht. Zuletzt wurde eine Textur aufgetragen und auf die Punkte des dreidimensionalen Körpers gelegt. Dieser Prozess nennt sich „Texture-Mapping“. Im Folgenden ist der gesamte Designprozess in Maya für das Wand-Eckstück beschrieben und wird in Abbildung 12: Design-Prozess in Maya dargestellt.

1. Mithilfe der Polygonfunktion wird eine quadratische Fläche erstellt. Anschließend wird die Größe angepasst und das Objekt zu einer Wand rotiert.
2. In das Objekt werden Mittelpunkte der oberen und unteren Kante eingefügt. Das Objekt wird in der Mitte „geschnitten“. Es entstehen zwei zusammenhängende, rechteckige Hälften.
3. Eine Hälfte des Objektes wird um 90° gedreht, so dass eine Ecke entsteht. Dafür werden die Eckpunkte an die richtige Stelle verschoben.
4. Es folgen weitere Anpassungen, so dass das Eckstück an eine gerade Wand gesetzt werden kann. Um die Übergänge zur Decke und zum Boden etwas abzurunden, werden die oberen und unteren Kanten zur Innenseite gebogen. Des Weiteren wird die Innenkante durch das Einfügen von zwei weiteren Kanten abgerundet. Es entstehen zwei abgerundete Segmente in der Mitte.
5. Die Wandtextur wird auf das Objekt gelegt und an die Punkte des Körpers geheftet. Hierbei muss beachtet werden, dass die Regelmäßigkeit der Textur erhalten bleibt, da die Wandstücke beim Aneinanderheften keine Unterbrechungen aufweisen sollen.
6. Abschließender Test, ob das Eckstück auf ein gerades Wandstück passt. Zu sehen ist ebenfalls, dass der Boden an das Wandstück passt.

4.1.2 Aufbau der virtuellen Umgebung

Um es dem Anwender zu ermöglichen, in die Welt des Roboters einzutauchen, wurde im Rahmen dieser Studienarbeit eine virtuelle Umgebung unter Verwendung der Game-Engine Unity erstellt. Die virtuelle Umgebung besteht aus zwei Szenen, welche in dem Kapitel Konzeption grundlegend beschrieben werden. Hierfür muss das Plugin SteamVR in die Anwendung integriert werden, damit statt zweidimensionaler Umgebungen auch dreidimensionale virtuelle Umgebungen dargestellt werden können. Dadurch kann die Anwendung über Unity gebaut und gestartet werden und direkt über eine VR-Brille visualisiert werden. Das Plugin verfügt über eine Render-Komponente, welche das Rendern des Signals der VR-Kamera übernimmt. Eine zweite wichtige Komponente für das Erstellen virtueller Umgebungen in Unity ist das *CameraRig*. Sie übernimmt die Steuerung des Headsets und der Controller der HTC Vive. Über Unity kann damit die Position, an der sich der Operator zum Start der Anwendung befindet, definiert werden. Der Anwender kann seinen Kopf bewegen und die Position wird unmittelbar über das Signal der HTC Vive erfasst und das Bild entsprechend der Position und Blickrichtung gerendert.

4.1.2.1 Aufbau der erstellten Szenen

In den beiden erstellten Szenen hat der Anwender die Möglichkeit, sich frei im Raum zu teleportieren. Dies wurde realisiert über die Einbindung des von Unity zur Verfügung gestellten Teleport-Prefabs. Um sich anschließend mit Hilfe des Prefabs frei in der Umgebung bewegen zu können, müssen zusätzlich Teleport-Punkte in der Szene angelegt werden. Diese definieren für das System, an welche Stellen des Raumes der Anwender sich teleportieren kann. Sie bieten somit dem Entwickler die Möglichkeit, die Anwendung passend zu den spezifizierten Anforderungen zu gestalten. Weiterhin ist es für das Nutzererlebnis von großer Bedeutung, dass der Operator durch die Controller Aktionen ausführen kann und somit auch die Möglichkeit der Teleportation nutzen kann. Der Input der VR-Controller muss in der Unity-Anwendung getrackt werden können, damit die Position der Hände wahrheitsgetreu in der virtuellen Umgebung angezeigt und die Benutzereingabe über die Controller erkannt und umgesetzt werden kann. Hierzu werden sobald die Controller von dem SteamVR-Plugin

erkannt werden, virtuelle Versionen der Controller in Unity erstellt und zu den bereits vorhandenen Controllern in CameraRig gemapped.

4.1.2.2 Willkommensbildschirm (Hauptmenü)

Sobald der Operator die Anwendung startet, befindet er sich in der Willkommensszene. Der Anwender steht auf einer Plattform im Weltall, welches als eine Art Zwischenwelt fungiert. Zu diesem Zeitpunkt ist er zwar noch nicht mit dem Nao-Roboter verbunden, jedoch bereits rundum von virtuell erstellten Bildern umgeben. In der nachfolgenden Abbildung sieht man den Aufbau des Willkommensbildschirms. Der Willkommensbildschirm hat die Funktion, dem Anwender Konfigurationsmöglichkeiten anzubieten. In der Mitte dieser Szene befindet sich ein Menü, durch das es dem Anwender ermöglicht wird, das Nao-Lab zu starten. Außerdem kann der Benutzer bevor er die Kontrolle des Nao-Roboters in dem Lab übernimmt, Voreinstellungen bezüglich der IP-Adresse des Roboters treffen.

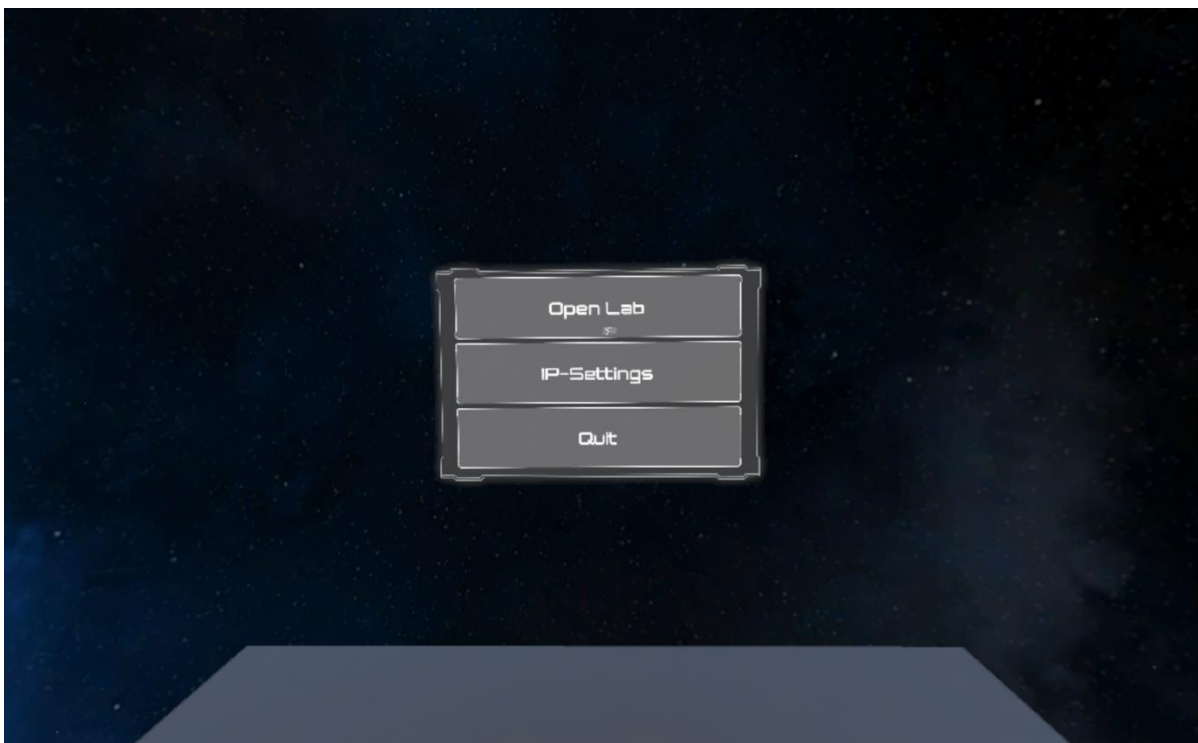


Abbildung 13: Willkommensbildschirm

Realisiert wurde die Umgebung der ersten Szene durch das Verwenden einer Skybox von Unity. Skyboxen sind Umgebungs-Beleuchtungs-Effekte, die über die gesamte Peripherie der Szene gelegt werden können. Somit kann der Eindruck erweckt werden, dass der Anwender sich in der virtuellen Umgebung befindet. Beim Bewegen des Head-Mounted-Displays werden die Bilder gerendert und entsprechend der Position und Blickrichtung angezeigt.³⁴ Die Panoramasicht wird bei einer Skybox in die sechs Richtungen der Achsen oben, unten, links, rechts, vorwärts und rückwärts geteilt. Die Bilder sollen nahtlos ineinander übergehen und dem Anwender eine kontinuierliche Umgebung anzeigen, die aus jeder Perspektive dargestellt werden kann. Durch Verwenden dieses Effekts, kann Realismus in die Szene mit minimaler Belastung der Grafikhardware hinzugefügt werden.³⁵

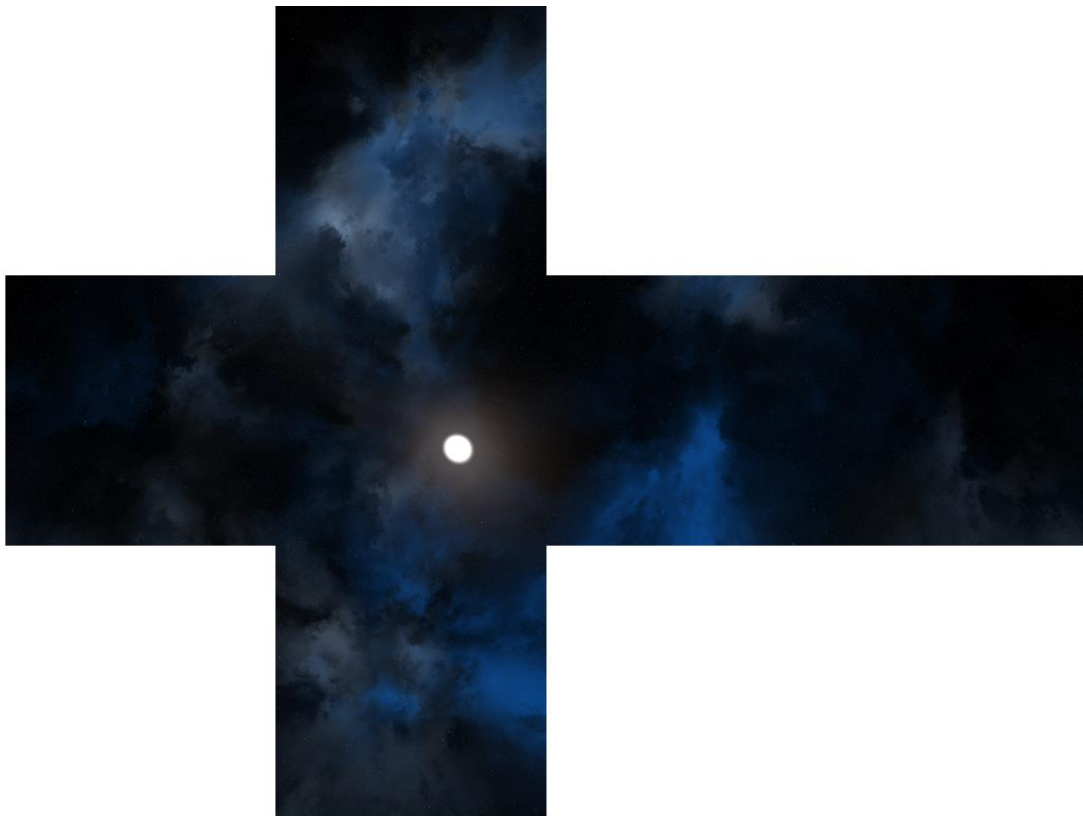


Abbildung 14: Cubemap Skybox Willkommensbildschirm

³⁴ Vgl. <https://medium.com/aol-alpha/how-to-design-vr-skyboxes-d460e9eb5a75> [Letzter Zugriff: 31.03.2019]

³⁵ Vgl. <https://unity3d.com/de/learn/tutorials/topics/graphics/using-skyboxes> [Letzter Zugriff: 31.03.2019]

Es gibt hierbei verschiedene Möglichkeiten, Skyboxen zu erstellen. Beispielsweise kann ein walzenförmiges Panorama benutzt werden, das jedoch lediglich die horizontalen Achsen darstellt und somit nicht für unseren Einsatzzweck geeignet war. Die beste Möglichkeit für die Umsetzung der virtuellen Umgebung dieser Studienarbeit war der Einsatz einer Cubemap. Bei einer Cubemap wird der inverse rotatorische Anteil der Augpunkttransformation auf automatisch generierte Texturkoordinaten angewendet. Vorteil des Cube Mappings ist, dass das Verfahren unabhängig vom Blickwinkel Texturen mit geringer Verzerrung darstellen kann. Abbildung 14 zeigt den Aufbau der erstellten Cubemap zur Darstellung der Umgebung der Willkommensszene. Wie zu sehen ist, besteht eine Cubemap aus sechs Bildern, die als Seiten eines Würfels angeordnet sind. Unity rendert diese Bilder während der Laufzeit als Skybox. Die Cubemap ist eine sehr gute Variante für eine Skybox, da sie einen hohen Detailgrad hat und sowohl die horizontale als auch die vertikale Ebene vollständig darstellt.³⁶



Abbildung 15: Menü IP-Settings

³⁶ Vgl. <https://medium.com/aol-alpha/how-to-design-vr-skyboxes-d460e9eb5a75> [Letzter Zugriff: 31.03.2019]

Über den Menüpunkt „IP-Settings“ gelangt der Operator zu einem anderen Menü. In diesem besteht die Möglichkeit, die IP-Adresse des zur Verfügung stehenden Roboters anzugeben. Damit nicht jedes Mal beim Start der Anwendung die IP-Adresse neu konfiguriert werden muss, wird standardmäßig die zuletzt definierte IP-Adresse übernommen und angezeigt. Über ein Nummernpad kann in dieser Szene die Adresse über die Controller geändert werden. Diese wird zur Ansteuerung des Nao-Roboters nach Bestätigung übernommen. Er gelangt zurück zum Startmenü und kann sich in das erstellte Lab zur Ansteuerung des Roboters teleportieren.

4.1.2.3 Nao-Lab

Die zweite erstellte Szene stellt das Nao-Lab dar. Das Labor dient als Basis zur Fernsteuerung des Roboters. Der Anwender ist nun mit dem Roboter verbunden, kann ihn als Teleroboter verwenden und ihn aus der entfernten Umgebung steuern. In der erstellten Szene ist es nicht nur möglich, den Roboter zu steuern, vielmehr wird eine multimodale Kommunikation zwischen Mensch und Roboter ermöglicht. Über ein Menü in der Mitte des Raumes kann der Operator bestimmen, welche Aktionen der Teleroboter ausführen soll. Der Aufbau des Menüs ist äquivalent zu dem Menü der Willkommensszene. Da dem Anwender direkt angezeigt werden soll, was der Roboter macht, wird das Kamerabild des Roboters übertragen und als Display in einer Wand dargestellt. Die folgende Abbildung 16 zeigt die grundlegenden Elemente und den Aufbau der Szene.

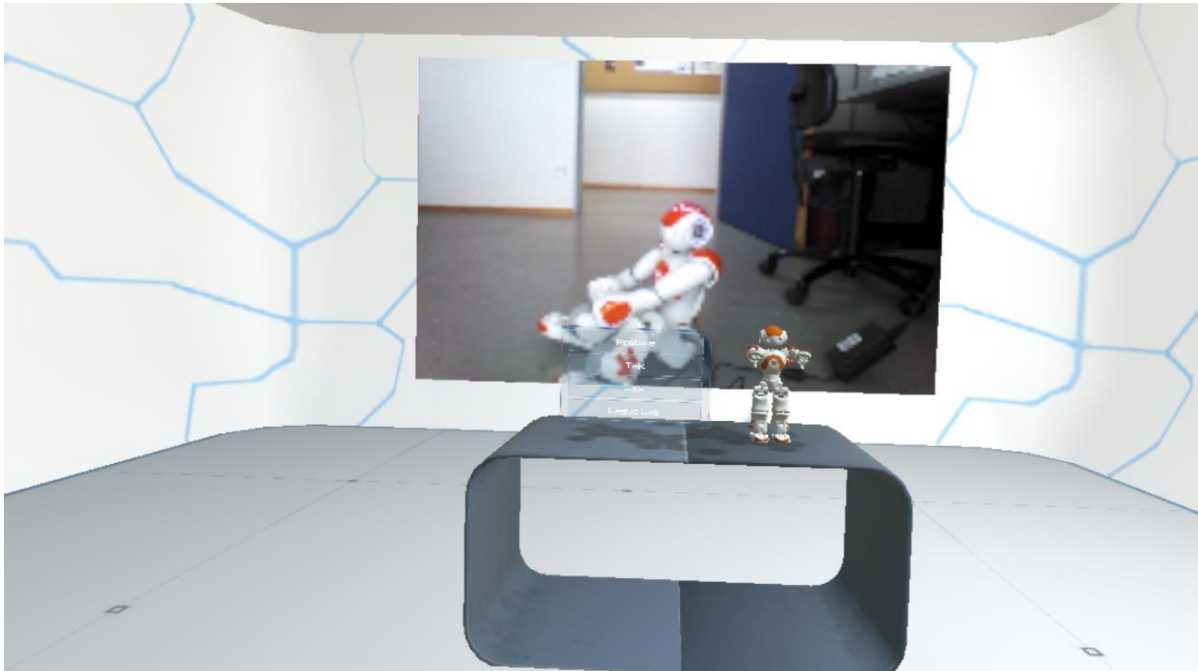


Abbildung 16: Nao-Lab

Das Hauptmenü verfügt über verschiedene Auswahlmöglichkeiten (siehe Abbildung 17):

- Posture: Nach Betätigen dieses Buttons, wird das Posture-Untermenü anstelle des Hauptmenüs angezeigt. Über dieses Menü können verschiedene Posen durch den Anwender ausgewählt werden. Somit kann das Verhalten des Roboters gesteuert werden und die Aktion eingeleitet werden (Stehen, Liegen, Sitzen).
- Talk: Der zweite Menüpunkt ist „Talk“. Hierbei sind bereits Standard-Sätze im Skript hinterlegt, die zufällig ausgesucht und vom Nao nachgesprochen werden.
- Walk: Zu dem Menüpunkt „Walk“ sind wiederum verschiedene Möglichkeiten in einem Untermenü hinterlegt. Der Anwender kann auswählen, in welche Richtung der Roboter sich bewegen soll.
- Leave Lab: Über diesen Menüpunkt kann der Anwender die Verbindung zu dem Nao trennen und zurück in die Willkommensszene gelangen. Dort hat er die Möglichkeit, sich mit einem anderen Roboter zu verbinden.

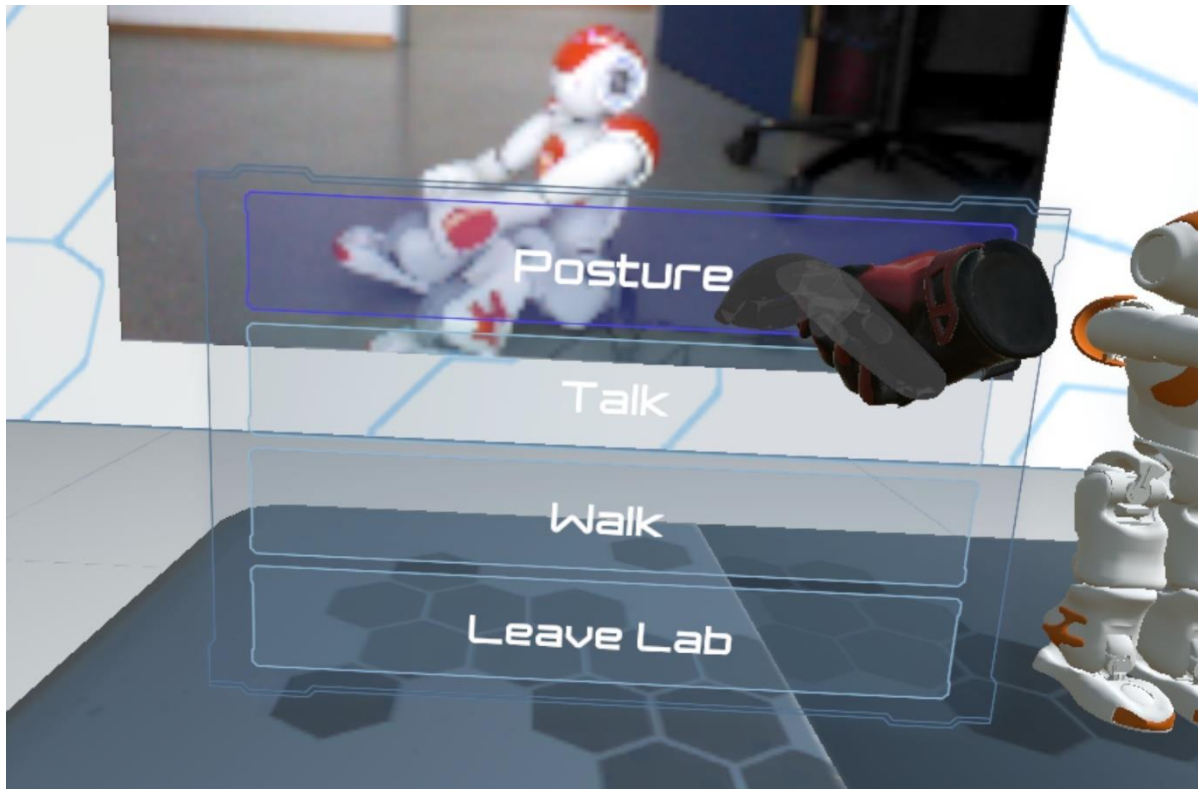


Abbildung 17: Menü im Nao-Lab

4.1.3 Ordnerstruktur

Die nachfolgende Abbildung 18 stellt die Ordnerstruktur des erstellten Unity-Projektes dar. Die übersichtliche Struktur ermöglicht ein effizientes Vorgehen und dient der besseren Orientierung. Die Bausteine der Anwendung konnten dadurch in verschiedene Kategorien unterteilt werden.

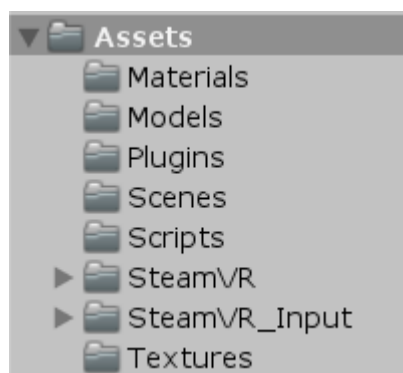


Abbildung 18: Ordnerstruktur Unity-Projekt

Die nachfolgende Tabelle erklärt die Funktionalität der angelegten Ordner.

Materials	In diesem Unterordner sind alle erforderlichen Materialien gespeichert. Materialien können beispielsweise auf Farben oder Texturen gelegt werden.
Models	Der Ordner enthält die mit Maya erstellten 3D-Objekte.
Plugins	Hier sind alle erforderlichen Plugins gespeichert. Zum Beispiel befindet sich hier die erstellte Bibliothek, die die Kommunikation mit dem Nao-Roboter ermöglicht.
Scenes	In dem Ordner Scenes sind alle erstellten Szenen (Willkommensbildschirm und Nao-Lab) der Anwendung gespeichert.
Scripts	Dieser Ordner enthält alle implementierten Skripte.
SteamVR	Der Ordner wird automatisch von dem Plugin SteamVR erstellt. Die dort enthaltenen Objekte sind für das Erstellen dreidimensionaler Umgebungen erforderlich.
SteamVR_Input	Dieser Ordner wird automatisch von SteamVR hinzugefügt, damit die Steuerung der Controller getrackt werden kann.
Textures	Texturen können beispielsweise auf Materialien gelegt werden.

Tabelle 2: Ordnerstruktur Unity-Projekt

4.2 Kommunikation mit dem Nao-Roboter

Der wichtigste Aspekt dieser Studienarbeit war die Ermöglichung der Kommunikation mit einem Nao-Roboter. Die Kommunikation wurde umgesetzt über die Implementierung eines C++-Projektes. Über diese Klasse finden die Anbindung sowie die Steuerung des Roboters statt. Damit auf die Funktionen des NAOqi-Frameworks zugegriffen werden kann, muss das NAOqi-SDK in das Projekt eingebunden werden. Dieses Kapitel beschreibt die implementierten Funktionalitäten des C++-Projektes. Das Projekt dient als Grundlage für die Anbindung des Roboters in der virtuellen Umgebung. Die Klasse wurde als Klassenbibliothek (.dll) in Unity eingebunden und stellt so die implementierten Funktionalitäten zur Verfügung. Im nächsten Kapitel wird beschrieben, wie ein Zugriff auf die implementierten Methoden in Unity erfolgen kann.

Das NAOqi-Framework beinhaltet verschiedene Kernmodule, auf die bei der Implementierung einer Anwendung zur Steuerung des Roboters zugegriffen werden kann. Die zu implementierenden Funktionen wurden über die Software Choreographie Suite (siehe 2.2.2) getestet. Einige Bausteine sind dort bereits als Python-Skript enthalten und können zur Steuerung des Nao verwendet werden. Diese wurden als Grundlage für die umgesetzte Ansteuerung nach C++ übersetzt und implementiert. Nachfolgend eine Übersicht über die Programmierschnittstellen der NAOqi, auf welche in dem C++-Projekt zur Umsetzung der Anforderungen der Anwendung zugegriffen wurde:

ALRobotPosture

Das Modul enthält diverse Methoden, um den Roboter vordefinierte Posen einnehmen zu lassen. Dafür bestimmt der Roboter zunächst seine eigene Pose und wendet anschließend die definierte Pose des Anwenders an. Jede Pose ist spezifiziert durch eine eindeutige Gelenk- und Sensorkonfiguration.

Der Aufruf erfolgt über folgende Methode: *ALRobotPostureProxy::goToPosture*

ALMotion

Das Modul ALMotion definiert Methoden, die die Steuerung der Bewegungen des Roboters vereinfachen. Beispielsweise kann die Position der Gelenke, die Steuerung von Bewegungen oder die Vermeidung von Kollisionen reguliert werden.³⁷

Der Aufruf zur Steuerung des Roboters erfolgt über: *ALMotionProxy::moveTo*

ALVideoDevice

Dieses Modul ist für die Kameraübertragung des Roboters zuständig. Eingesetzt werden kann das Modul beispielsweise für die Gesichtserkennung. Bei der Erstellung von einem Proxy, um Zugriff auf die Methoden zu bekommen, können verschiedene Parameter definiert werden (Auflösung, Farbraum, etc.).

Wichtige Methoden:

1. *ALVideoDeviceProxy::subscribeCamera:*
Registrieren eines Videomoduls zu ALVideoDevice. Die Rohdaten des Bildes werden in dem definierten Format gespeichert.
2. *ALVideoDeviceProxy::getImageRemote:*
Abrufen des letzten Bildes des Roboters.
3. *ALVideoDeviceProxy::releaseImage:*
Freigabe des Bild-Puffers, welcher von subscribeCamera blockiert wurde.
4. *ALVideoDeviceProxy::unsubscribe:*
Abmeldung des angemeldeten Video-Moduls von ALVideoDevice.³⁸

ALTextToSpeech

Das Modul ermöglicht es dem Roboter, zu sprechen. Der Benutzer kann Strings definieren, die als Kommandos an den Nao geschickt werden. Der Roboter spricht anschließend die Wörter nach.³⁹

³⁷ Vgl. <http://doc.aldebaran.com/2-5/naoqi/motion/almotion.html#almotion> [Letzter Zugriff: 16.04.2019]

³⁸ Vgl. <http://doc.aldebaran.com/2-5/naoqi/vision/alvideodevice-api.html#alvideodevice-api> [Letzter Zugriff: 16.04.2019]

³⁹ Vgl. <http://doc.aldebaran.com/2-5/naoqi/audio/alttexttospeech.html#alttexttospeech> [Letzter Zugriff: 16.04.2019]

In der C++-Klasse wurde folgende Methode eingesetzt: *ALTextToSpeechProxy::say*

Über die zuvor dargestellten NAOqi-Module erfolgt in der implementierten C++-Klasse die Steuerung des Roboters. Die Anbindung des Roboters wird jeweils durch die Erstellung eines Proxys des jeweiligen Moduls ermöglicht. Dabei müssen lediglich die IP-Adresse des zur Verfügung stehenden Roboters und die Portnummer, über die das NAOqi-Framework zuhört, angegeben werden. Durch diese Angabe kann der Roboter eindeutig identifiziert und an das Programm angebunden werden. Zur Realisierung der gewünschten Funktionalitäten zur Ansteuerung des Roboters, wurden verschiedene C++-Methoden implementiert. Nachfolgend werden die umgesetzten Funktionen detailliert beschrieben.

NaoGoToPosture

Über die Erstellung einer *ALRobotPostureProxy* kann eine Verbindung zu dem Modul zur Steuerung der Posen des Roboters hergestellt werden. Da in der erstellten virtuellen Umgebung verschiedene Posen (Stehen, Liegen, Sitzen) zur Verfügung stehen, besitzt die Methode einen Parameter zur Angabe der gewünschten Pose. Somit kann gewährleistet werden, dass die Methode für die Umsetzung der Funktionalität des Nao-Labs mehrfach verwendet werden kann. Über den Aufruf *goToPosture* des Proxy-Objekts können diverse Posen durch den Roboter direkt eingenommen werden. Neben den in der virtuellen Umgebung verfügbaren Posen *Sit* (Setzen), *Stand* (Stehen) und *LyingBack* (Zurücklehnen) gibt es weitere Haltungen, die von der API zur Verfügung gestellt werden. Auf die Umsetzung weiterer Funktionen in der Umgebung wurde verzichtet, um die Übersichtlichkeit für den Anwender zu gewährleisten. Die implementierte Methode kann jedoch einfach wiederverwendet werden, um weitere gewünschte Posen umzusetzen.

NaoSpeak

In dieser Methode wird ein `ALTextToSpeechProxy` erstellt, das zur Ansteuerung des Lautsprechers des Roboters eingesetzt werden kann. Die Methode `say` nimmt einen String als Parameter entgegen. Zu Vorführzwecken wurden hierbei verschiedene Default-Sätze hinterlegt, die zufällig nach Betätigen des Menüpunktes vom Roboter gesagt werden.

NaoWalkTo

Diese Methode dient dazu, dem Operator die Möglichkeit zur Verfügung zu stellen, den Roboter zu bewegen. Sie verfügt über drei Parameter zur Spezifikation der Richtung in x- und y-Richtung und der Drehung. Bevor die Methode mit den definierten Parametern aufgerufen wird, muss der Roboter in die Pose `StandInit` fahren, damit keine Beschädigungen durch falsche Gelenkwinkelkonfigurationen auftreten können. In der virtuellen Umgebung kann derzeit zwischen Vorwärts- und Rückwärtsgehen ausgewählt werden. Die Methode kann zusätzlich auch zum seitlichen Fortbewegen oder zum Drehen verwendet werden.

SubscribeVideo

Eine wichtige Funktionalität der virtuellen Umgebung ist die Live-Übertragung des Video-Signals des Nao-Roboters. Der erste Schritt hierbei ist das Anmelden des Kameramoduls des Roboters, damit anschließend über dieses Objekt der Videobuffer geladen werden kann. Beim Aufruf dieser Methode kann ein frei wählbarer Name des Kameramoduls, die Kamera-Auflösung, der Farbraum und die Anzahl der Einzelbilder pro Sekunde (fps) definiert werden. Die richtige Auswahl der Auflösung und des Farbraums sind elementar für die Anzeige des Videos in der virtuellen Umgebung. In diesem Projekt wurde die Auflösung `kQQVGA` gewählt. Zwar würde eine höhere Auflösung zu einem schärferen Bild führen, jedoch muss ein Kompromiss zwischen Genauigkeit des Bildes und der Ladegeschwindigkeit getroffen werden. Bei dieser Einstellung wird das Bild in den Maßen 160 x 120 übertragen. Durch diese Einstellung ist es möglich, ein flüssiges Bild mit relativ hoher Genauigkeit anzuzeigen.

GetImageData

Nachdem das Kameramodul des Nao-Roboters angemeldet wurde, kann der Videobuffer über diese Methode geladen werden. Über den Aufruf der Methode *getImageRemote* können Rohdaten des zuletzt erzeugten Bildes geladen werden. Über den in dem nächsten Kapitel beschriebenen C#-Wrapper kann daraus ein Byte-Array der Rohdaten des Bildes erzeugt werden und auf einer Wand des Nao-Labs angezeigt werden. Im nächsten Kapitel wird die Vorgehensweise zur Darstellung der Rohdaten detaillierter erläutert. Wichtig ist zudem, dass nach dem Aufruf von *getImageRemote* zusätzlich die Methode *releaseImage* aufgerufen wird. Über diesen Methodenaufruf wird der reservierte Image-Buffer freigegeben. Zwar ist es nicht vorgeschrieben diesen nach der Verwendung freizugeben, sollte aber bevorzugt verwendet werden. Ein Wechsel von einem entfernten zu einem lokalen Verhalten wird dadurch ohne Verwendung der CPU erleichtert.

UnsubscribeVideo

Beim Verlassen des Nao-Labs wird automatisch diese Methode aufgerufen. Die Anzahl der möglichen Verbindungsinstanzen zu einem Nao-Roboter ist begrenzt. Somit ist es empfehlenswert, die Verbindung zu dem Roboter nach der Ansteuerung zu trennen. In dieser Methode wird das Modul des ALVideoDevice von dem Roboter abgemeldet.

4.3 Steuerung des Nao-Roboters

In diesem Kapitel wird erläutert, wie die Steuerung des Roboters in Unity umgesetzt wurde.

4.3.1 Wrapper Schicht für C#

Für bessere Kompatibilität wurde eine Zwischenschicht eingebaut, die Methoden aus der C++ DLL einbindet und für Aufrufe von C# zur Verfügung stellt. Die Namen der Methoden gleichen denen aus der DLL. Die Schicht stellt die Kompatibilität zu C# und Unity sicher. Die daraus erzeugte DLL wird in Unity importiert und stellt als *NaoCSharpLibrary* Funktionen für den Zugriff auf den Nao zur Verfügung.

Die folgenden Methoden importieren Funktionen aus der C++ DLL und wurden über den DLL-Import-Befehl realisiert. Alle Methoden haben die Zugriffsmodifizierer *public static extern*.

Methode	Kurzbeschreibung
<code>void NaoWalkTo(float x, float y, float theta);</code>	Befehl für Laufen
<code>void NaoSpeak(string phraseToSay);</code>	Befehl für Sprachausgabe
<code>void NaoSit();</code>	Befehl für Animation „Setzen“
<code>void NaoStand();</code>	Befehl für Animation „Stehen“
<code>void SetIPAddress(string ipAddress);</code>	IP-Adresse für Verbindung setzen
<code>void NaoLieDown();</code>	Befehl für Animation „Liegen“
<code>void SubscribeVideo();</code>	Befehl für Videoaufzeichnung
<code>int GetImageData(out IntPtr data, out int size, out int width, out int height);</code>	Befehl zum Übertragen der Bilddaten
<code>void UnsubscribeVideo();</code>	Befehl für Ende der Videoaufzeichnung
<code>string GetIPAddress();</code>	IP-Adresse abfragen

Tabelle 3: Methoden C#-Wrapper

Die folgenden Methoden implementieren neue Funktionalität:

GetRawPictureByte

Diese Methode stellt eine Hilfsmethode für *GetImageData* bereit, um die Bilddaten auszulesen und an den Aufrufer zu übergeben. Zuerst wird die Methode *GetImageData* aufgerufen, um die Bilddaten in den Speicher zu kopieren. Anhand der übergebenen Pointer wird der Speicherbereich ausgewertet. Die Bilddaten werden ausgelesen und ein Byte-Array kopiert. Die so genannten „Bild-Rohdaten“ aus dem Byte-Array werden an den Aufrufer übergeben und können nach dem Laden als Bild oder als Textur repräsentiert werden.

GetRawPictureIntPtr

Diese Methode stellt ähnlich wie *GetRawPictureByte* eine Hilfsmethode dar. Im Gegensatz zu den Bilddaten liefert sie einen Pointer, der auf den Speicherbereich der Bilddaten zeigt. Wie zuvor wird dafür die Methode *GetImageData* aufgerufen und der Pointer der Speicheradresse zurückgegeben.

4.3.2 Unity Skripte für Steuerung

Es wurden mehrere C#-Skripte für Unity erstellt, um verschiedene Anwendungsfälle abzudecken. Die folgenden Skripte wurden implementiert:

IPSettings

Dieses Skript steuert die Eingabe und Übernahme der IP-Einstellungen im Hauptmenü. Es benötigt dafür das Textfeld aus der virtuellen Umgebung, auf dem die aktuelle IP-Adresse ausgegeben wird. Die Unity-Start-Methode wird zum Initialisieren der NaoCSharpLibrary genutzt, damit die IP-Adresse abgerufen werden kann. Die Funktion *OnClick* nimmt eine weitere Zahl entgegen und fügt sie an die gespeicherten Zahlen an, solange die Länge einer gültigen IP-Adresse nicht überschritten wird. Danach ruft sie die Validierungsfunktion *UpdateShownAddress* auf, welche im richtigen Abstand Punkte zur Trennung der Zahlenblöcke einfügt und die generierte Adresse speichert. Diese wird über das Textfeld ausgegeben. Die Methode *OnDelete* entfernt die letzte Zahl der aktuellen Adresse und ruft ebenfalls *UpdateShownAddress* auf. Die Methode *OnConfirm* übermittelt die IP-Adresse an die NaoCSharpLibrary.

LoadSceneOnClick

Dieses Skript stellt lediglich die Methode *LoadSceneByIndex* zur Verfügung, in welcher eine indexierte Szene von Unity geladen werden kann. Das Skript wird beim Drücken eines Knopfs verwendet, um den Spieler in eine andere Szene zu bringen. Eingesetzt wird es, um eine Verknüpfung zwischen dem erstellten Willkommensmenü und dem Nao-Lab herzustellen.

NaoController

Das Skript enthält Methoden für die Steuerung des Nao-Roboters, die unter anderem Aktionen zum Aufstehen, Hinsetzen, Hinlegen, Reden und Laufen beinhalten. Die Unity-Start-Methode wird hier ebenfalls zum Initialisieren der NaoCSharpLibrary genutzt, welche die Funktionen der Bewegungssteuerung zur Verfügung stellt. Die Aktionen werden jeweils über einen Thread gestartet, so dass die Kommunikation mit dem Nao nicht das Rendern des

Bildes beeinflusst. Die Methode *CallNaoSit* löst die Aktion für das Hinsetzen aus, *CallNaoLieDown* für Hinlegen und *CallNaoStand* für Hinstehen.

Die Methode *CallNaoWalk* steuert das Laufverhalten des Nao über Koordinaten und einen Drehwinkel. Die Methode *NaoWalkHelper* vereinfacht die Steuerung fürs Laufen, indem sie eine Zahl entgegennimmt: „0“ für vorwärts und „1“ für rückwärts. Sie greift dabei auf die Methode *CallNaoWalk* zu. Zum Sprechen wird die Methode *CallNaoTalk* verwendet, welche einen aus einer Liste zufälligen ausgewählten Satz an den Nao überträgt.

NaoVideoController

Stellt eine Hilfsklasse dar, welche die Video-Verbindung zum Nao über die NaoCSharpLibrary verwaltet. Sie merkt sich, ob bereits eine Verbindung geöffnet wurde und sichert darüber die Vorgänge für Verbindungsaufbau und -abbau: Eine Verbindung kann erst wieder geöffnet werden, wenn sie zuvor geschlossen wurde. Ebenfalls kann keine Verbindung geschlossen werden, die nicht existiert. Die Methode *ConnectCamera* stellt eine Verbindung her, während die Methode *DisconnectCamera* die Verbindung abbaut.

QuitOnClick

Dieses Skript beendet Unity, wenn die Methode *Quit* aufgerufen wird. Das Skript wird in Zusammenhang mit dem „Exit“ Knopf im Hauptmenü verwendet.

RenderNaoCamera

Dieses Skript ist für das Rendern des Kamerabildes des Nao in Unity zuständig. Dafür benötigt ein Objekt der Klasse *RawImage*, welches die Bildrohdaten laden kann. Über die Methode *ConnectCamera* wird eine Verbindung zur Kamera hergestellt. Diese kann über *DisconnectCamera* beendet werden. In der Unity-Update-Methode werden die Bilddaten über die NaoCSharpLibrary abgerufen. Dies passiert in einem separaten Thread, damit das Bild nicht pausiert wird. Wenn die Bilddaten vorhanden sind, wird das Objekt vom Typ *RawImage* aktualisiert. Anschließend kann das nächste Bild der Kamera übertragen werden.

Die Update Methode wird vor dem Zeichnen jedes Frames aufgerufen. Die Übertragung des Kamerabildes dauert in der Regel länger als die Zeit zwischen den Frames.

TeleportIPSettings

Dieses Skript wird im Hauptmenü verwendet, um zwischen den Plattformen für die IP-Einstellungen und dem eigentlichen Hauptmenü zu wechseln. Dabei wird der Spieler in einem „fliegenden“ Effekt von einer Plattform zur anderen bewegt. Das Skript nimmt zwei Transform-Objekte entgegen: eins für die Ziel-Koordinaten und das Andere als aktuelle Position des Spielers. Beim Aufruf der Methode *TeleportToTarget* wird der Spieler durch den Raum zu den Ziel-Koordinaten bewegt. Der Bewegungseffekt streckt sich über einen bestimmten Zeitraum. Die aktuelle Position während dem Bewegen wird in der Unity-Update-Methode für jeden Frame neu berechnet.

4.4 Umsetzung der Telepräsenz

Die multimodale Telepräsenz wurde mithilfe der implementierten Klassen umgesetzt. Der Benutzer betritt dabei einen dafür speziell entwickelten Raum: das Nao-Lab (siehe Kapitel 4.1.2.1 Aufbau der erstellten Szenen). Dort kann sich der Benutzer frei im virtuellen Raum teleportieren, um die beiden Hauptelemente, den Bildschirm und das Aktions-Menü, besser erreichen zu können. Durch den eingeschränkten Interaktionsraum der Virtual Reality Brille wird es dem Benutzer dadurch erleichtert mit den Elementen zu interagieren.

Bildschirm

Über den Bildschirm im virtuellen Raum wird die Telepräsenz umgesetzt, indem das Kamerabild des Nao angezeigt wird. Dieses wird in Farbe angezeigt, permanent übertragen und auf dem Bildschirm im virtuellen Raum aktualisiert. Dem Benutzer ist es dementsprechend möglich, eine Vielzahl von Objekten und Personen über den Bildschirm zu unterscheiden. Die Größe des Bildschirms wurde angepasst, damit Details auch aus Distanz gut zu erkennen sind.

Aktions-Menü

Die Befehlsübermittlung und die Steuerung des Roboters können über ein Menü im virtuellen Raum abgewickelt werden und sorgen für die multimodale Telepräsenz. Für die Steuerung begibt sich der Benutzer an einen markierten Teleportpunkt im Raum. Das Menü basiert auf einem oberen Menü und zwei Untermenüs, wodurch die obere Schicht eine bessere Übersicht über die einzelnen Funktionen darstellt. Die Menüpunkte „Posture“ und „Walk“ bieten ein detailliertes Untermenü. Über ersteres können vordefinierte Haltungen wie zum Beispiel Sitzen, Stehen oder Liegen eingenommen werden. Das Menü „Walk“ bietet Funktionen zum Vorwärts- und Rückwärtslaufen. Der Menüpunkt „Talk“ ruft kein weiteres Menü auf, sondern startet die Sprachausgabe auf dem Nao. Die Sätze werden dabei zufällig aus einer vordefinierten Liste ausgewählt. Abschließend kann über den Menüpunkt „Leave Lab“ das Labor verlassen werden.

5 Ergebnis

Im Rahmen dieser Studienarbeit wurde eine lauffähige, virtuelle Applikation entwickelt. Die Anwendung ermöglicht es einem Operator, einen entfernten Nao-Roboter über Einsatz von Virtual Reality Technologien fernzusteuern. Die in Kapitel 3.1.1 (Anforderungsanalyse) dargestellten Anforderungen konnten weitestgehend erfüllt werden. Dieses Kapitel stellt das Ergebnis dieser Studienarbeit dar und erläutert die umgesetzte Funktionalität.

Das Hauptziel dieser Studienarbeit war die Schaffung einer multimodalen Telepräsenz zwischen einem menschlichen Benutzer und einem entfernten Roboter. Dies wurde realisiert über die zuvor beschriebene entwickelte Applikation. Um die Komfortabilität für den Anwender zu erhöhen, wurden zwei virtuelle Szenen erstellt. Innerhalb dieser kann der Operator sich frei teleportieren und mit den Objekten über die Steuerung der Controller virtuell interagieren. Die Willkommens-Szene dient dazu, Vorkonfigurationen bezüglich des Roboters zur Verfügung zu stellen. Über das bereitgestellte IP-Menü kann der Anwender auf einfache Weise die IP-Adresse des Roboters auswählen. Die zweite Szene hingegen dient der Ansteuerung des Roboters. Die Szene stellt einerseits die empfangenen Bildinformationen vom Nao-Roboter dar. Andererseits kann der Operator diverse Steuerungsmöglichkeiten auswählen und diese Eingaben an den Roboter senden. Der Nao führt die ausgewählten Aktionen augenblicklich aus und das Geschehen kann über das übertragene Video nachvollzogen werden. Zusätzlich zur Steuerung der Posen des Roboters, kann eine Sprachausgabe ausgelöst werden. Hierbei sind Standardsätze hinterlegt, die von dem Roboter nachgesprochen werden. Aufgrund der geringen Übertragungsrate des Roboters, konnte das Tonsignal nicht zusätzlich zu dem dargestellten Kamerabild übertragen werden. Die Technologie des Nao-Roboters ist nicht darauf ausgelegt, eine simultan stattfindende Übertragung von Ton- und Bildinformationen zu ermöglichen. Für eine flüssige Darstellung des Kamerabildes ohne Verzögerung musste bereits die Bildqualität deutlich heruntergesetzt werden. Auf die Umsetzung der Übertragung des Tonsignals wurde verzichtet, damit das Bild erkennbar bleibt und zeitnah gerendert werden kann.

Die entwickelte Applikation erfüllt somit den gewünschten Leistungsumfang dieser Studienarbeit. Besonders herauszuheben ist hierbei, die bidirektionale Signalübertragung zwischen dem menschlichen Anwender und dem Roboter, sowie die erreichte räumliche Entkopplung. Damit müssen sich Anwender und Roboter nicht im selben Raum befinden. Das Ergebnis der Studienarbeit ermöglicht vielfältige Einsatzmöglichkeiten. Beispielsweise kann die virtuelle Umgebung eingesetzt werden, um Schülern die Funktionsweise des Nao-Roboters näherzubringen.

6 Zusammenfassung und Ausblick

6.1 Persönliches Feedback

Die im Laufe dieser Studienarbeit realisierte Anwendung stellte die Autoren und Entwickler vor eine große Herausforderung: Es galt eine komplexe Virtual Reality Anwendung mit Abhängigkeiten zu einem externen Roboter-System zu entwerfen. Die Schwierigkeit bestand daher nicht nur im Design einer glaubwürdigen virtuellen, dreidimensionalen Umgebung, sondern zusätzlich noch in der Anbindung und Steuerung eines Roboters.

Die Autoren konnten vor allem bei der Planung auf das in den Vorlesungen „Projektmanagement“ und „angewandtes Projektmanagement“ erworbene Wissen zurückgreifen. Dadurch konnte ein realistischer Projektplan entworfen und Unklarheiten einzelner Aufgaben schon frühzeitig geklärt werden. Die Vorlesungen „Softwareengineering“ und „Advanced Softwareengineering“ erleichterten den Autoren die Konzeption und Planung der Anwendung. Dabei kamen speziell durch die Vorlesung gelernte Vorgehensweisen zum Einsatz. Beide Autoren haben im Laufe des Studiums bereits mehrere Anwendungen entwickelt und konnten die vorhandenen Erfahrungen einbringen und weiterentwickeln. Ein Transfer dieses Wissens in die unbekannten Bereiche der Virtual Reality und Robotik erleichterte ihnen den Einstieg.

Die Anwendung wurde mit mehreren Programmiersprachen realisiert: Unity-Skripte in C# und die Anbindung des Nao-Roboters über C++. Dieser Umstand sorgte bis zuletzt für kleinere Schwierigkeiten, wurde abschließend jedoch durch sogenanntes „Refactoring“, dem Verbessern der Codequalität, behoben. Die Programmiersprachen waren den Autoren nur zum Teil bekannt und mussten zu Beginn der Studienarbeit erlernt werden. Für die Umsetzung der Anwendung mit Unity war eine Einarbeitung erforderlich, die jedoch durch die von den Entwicklern angebotenen Tutorials wesentlich erleichtert wurde. Außerdem weist Unity ein einsteigerfreundliches Bedienkonzept auf.

Durch Regelmeetings mit dem zuständigen Betreuer wurde Feedback eingeholt und die weitere Vorgehensweise besprochen. Auch dadurch wurde eine ständige Verbesserung der Anwendung und ein hoher Lernerfolg erreicht.

6.2 Erweiterungsmöglichkeiten

Die implementierte Funktionalität der Applikation ist durch die Schichtenarchitektur jederzeit leicht erweiterbar. Somit ist es möglich, weitere Steuerungsmöglichkeiten des Nao-Roboters auf simple Weise einzubinden und die Funktionalität der Applikation den Anforderungen entsprechend zu erweitern.

Zusätzlich zu dem implementierten Funktionsumfang sind folgende Erweiterungsmöglichkeiten denkbar:

Darstellung der aktuellen Gelenkpositionen anhand des Nao-Modells in Unity

Derzeit befindet sich ein Modell des Nao-Roboters auf dem Tisch des Nao-Labors. Das Modell kann durch Einsatz der Controller bewegt werden. Um neben der Videoübertragung dem Anwender zusätzlich zu zeigen, in welcher Pose sich der Roboter aktuell befindet, könnte dieses Modell erweitert werden. Denkbar wäre es, die Gelenkpositionen des echten Nao-Roboters zu erfassen und diese anhand des Modells darzustellen. Dadurch kann die Bewegung des Roboters in der virtuellen Umgebung durch den menschlichen Operator genauer nachvollzogen werden.

Direkte Steuerung des Roboters über die VR-Controller

Die Applikation bietet die Möglichkeit, den Roboter fernzusteuern. Die Auswahl der Steuerungsaktionen ist hierbei praktikabel über ein Menü möglich. Interessant wäre es, ob eine direkte Steuerung der Armgelenke über die Bewegung der VR-Controller umsetzbar ist. Um den Effekt der Immersion weiter zu verstärken, könnte hierbei auf die Umgebung verzichtet und direkt das Kamerabild übertragen werden. Somit wirkt es auf den menschlichen Operator, als würde er sich im Körper des Roboters befinden und die Grenzen zwischen virtueller und realer Umgebung würden kleiner werden.

Ermöglichung der Übertragung von Tonsignalen

Die Umsetzung der gleichzeitigen Übertragung von Bild- und Tonsignalen wäre eine weitere sinnvolle Erweiterungsmöglichkeit. Hierbei wäre eine ausführliche Recherche erforderlich, um die Qualität der übertragenen Informationen zu gewährleisten.

Sprachsteuerung

Aktuell können die Befehle über eine Auswahl im Menü gesteuert werden. Zusätzlich wäre es denkbar, die Sprachsteuerung des Roboters zu verwenden. Problematisch ist hierbei lediglich, dass der Vorteil der entfernten Steuerung somit hinfällig wird und der Anwender die verfügbaren Befehle kennen muss. Der Roboter muss sich für diese Funktionalität im selben Raum wie der Anwender befinden und könnte direkt über Sprachbefehle gesteuert werden.

Neben den zuvor beschriebenen Erweiterungsmöglichkeiten sind weitere Einsatzgebiete denkbar. Die dargestellten Aspekte sollen lediglich einen Ausblick auf mögliche Erweiterungen der Funktionalität des Projektes sein. Die Applikation soll als Grundlage für weitere Projekte dienen, die Virtual Reality Technologien und Nao-Roboter einsetzen. Die leichte Erweiterbarkeit der Anwendung und die Dokumentation der Vorgehensweise unterstützen dies.

Literaturverzeichnis

- Aldebaran Robotics. (o.D.). Documentation - ALMotion. Von <http://doc.aldebaran.com/2-5/naoqi/motion/almotion.html#almotion> abgerufen
- Aldebaran Robotics. (o.D.). Documentation - ALTextToSpeech. Von <http://doc.aldebaran.com/2-5/naoqi/audio/altexttospeech.html#altexttospeech> abgerufen
- Aldebaran Robotics. (o.D.). Documentation - ALVideoDevice API. Von <http://doc.aldebaran.com/2-5/naoqi/vision/alvideodevice-api.html#alvideodevice-api> abgerufen
- Aldebaran Robotics. (o.D.). Documentation - Choreographie. Von http://doc.aldebaran.com/2-4/software/choregraphie/choregraphie_overview.html abgerufen
- Alur, D., Crupi, J., & Malks, D. (2003). *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall Computer.
- Autodesk. (o.D.). Maya. Von <https://www.autodesk.de/products/maya/overview> abgerufen
- Bendel, P. D. (o.D.). Virtuelle Realität. (G. Wirtschaftslexikon, Hrsg.) Von <https://wirtschaftslexikon.gabler.de/definition/virtuelle-realitaet-> abgerufen
- Bruder, J. (2009). *Praktische Realisierung einer haptischen Telerobotik-Steuerung für eine interaktive Nutzung*. Universität Hamburg, Department Informatik.
- DLR. (2010). Dem Weltraum zum Greifen nah: Die DLR-Forschung zur Telepräsenz. Von <https://www.neimagazine.com/features/featurehot-cell-robot-4483658/featurehot-cell-robot-4483658-462777.html> abgerufen
- Elektronik Kompendium. (o.D.). Virtual Reality / Augmented Reality / Mixed Reality. Von <https://www.elektronik-kompendium.de/sites/com/2210231.htm> abgerufen
- Génération Robots GmbH. (o.D.). NAO6: Der Roboter aus dem Hause Aldebaran. Von <https://www.generationrobots.com/de/403100-programmierbarer-humanoider-roboter-nao-version-6.html> abgerufen
- Gingold, S. (16. Juli 2018). Virtual Reality Vs Augmented Reality Vs Mixed Reality. Von <http://arvr3dmodelling.com/virtual-reality-vs-augmented-reality-vs-mixed-reality/> abgerufen
- Hammer, P. (2016). Virtual Reality: Die Erschaffung neuer Welten. Von <https://www.zukunftsinstitut.de/artikel/virtual-reality-die-erschaffung-neuer-welten/> abgerufen
- Inztitut. (o.D.). Immersion. Von <http://www.inztitut.de/blog/glossar/immersion/> abgerufen
- Joachim Hertzberg, K. L. (2012). *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Springer-Verlag.
- Microsoft. (o.D.). Entwickeln Ihres ersten Spiels mit Unity und C#. Von <https://msdn.microsoft.com/de-de/magazine/dn759441.aspx> abgerufen

- Mixabest. (17. Juli 2007). *KUKA Industrial Robots*. Von https://de.wikipedia.org/wiki/Datei:KUKA_Industrial_Robots_IR.jpg abgerufen
- Nuclear Engineering International. (o.D.). *Hot Cell Robot*. Von <https://www.neimagazine.com/features/featurehot-cell-robot-4483658/featurehot-cell-robot-4483658-462777.html> abgerufen
- Schatten, A., Biffel, S., Demolsky, M., Gostischa-Franta, E., Östreicher, T., & Winkler, D. (2010). *Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Springer.
- Seo, P. K. (o.D.). *Using Nao: Introduction To Interactive Humanoid Robots*. Aldebaran Robotics & NT Research, INC.
- SoftBank Robotics. (o.D.). NAO. Von <https://www.softbankrobotics.com/emea/en/nao> abgerufen
- Unity Technologies. (o.D.). Der Unity-Satz. Von <https://unity3d.com/de/programming-in-unity> abgerufen
- Unity Technologies. (o.D.). Game-Engines – wie funktionieren sie? Von <http://doc.aldebaran.com/2-5/naoqi/audio/altexttospeech.html#altexttospeech> abgerufen
- Unity Technologies. (o.D.). SteamVR Plugin. Von <https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647> abgerufen
- Virtual Reality Magazin. (o.D.). Augmented Reality & VR. Von <http://arvr3dmodelling.com/virtual-reality-vs-augmented-reality-vs-mixed-reality/> abgerufen
- VIVE. (o.D.). VIVE Pro. Von <https://www.vive.com/de/product/vive-pro/> abgerufen
- Wikipedia. (o.D.). Autodesk Maya. Von https://en.wikipedia.org/wiki/Autodesk_Maya abgerufen
- Wikipedia. (o.D.). *Humanoider Roboter*. Von Wikipedia: https://de.wikipedia.org/wiki/Humanoider_Roboter abgerufen
- Wikipedia. (o.D.). List of Maya plugins. Von https://en.wikipedia.org/wiki/List_of_Maya_plugins abgerufen
- Wikipedia. (o.D.). Maya. Von <https://de.wikipedia.org/wiki/Maya> abgerufen
- Wikipedia. (o.D.). *Roboter*. Von Wikipedia: <https://de.wikipedia.org/wiki/Roboter> abgerufen