HAW HAMBURG

Bachelorarbeit

Pascal Worreschk

Sprache versus Controller - Eignung von sprachgesteuerten Assistenten zur Interaktion im 3D-Raum

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informatik

Hamburg University of Applied Sciences
Faculty of Computer Science and Engineering
Department Computer Science

Pascal Worreschk

Sprache versus Controller - Eignung von sprachgesteuerten Assistenten zur Interaktion im 3D-Raum

Betreuender Prüfer: Prof. Birgit Wendholt
Zweitgutachter: Prof. Dr. Philipp Jenke

Eingereicht am: 25. April 2019

Pascal Worreschk

Thema der Arbeit

Sprache versus Controller - Eignung von sprachgesteuerten Assistenten zur Interaktion im 3D-Raum

Stichworte

Kommandobasierte Spracherkennung, 3D-Raum, Interaktion, Inklusion, Virtual Reality

Kurzzusammenfassung

Die vorliegende Bachelorarbeit beschreibt die Entwicklung und Evaluierung eines durch Spracherkennung steuerbaren 3D Virtual Reality Videospiels. Durch die Evaluierung wird herausgestellt, ob sich ein kompetitives Mehrspieler-Spiel genauso effizient durch Sprachkommandos steuern lässt, wie durch einen klassischen Videospiel-Controller. Dies ist besonders für Menschen mit körperlichen Einschränkungen interessant, da diese in der Regel Probleme haben ein Videospiel mit Hilfe eines Videospiel-Controllers zu bedienen.

Pascal Worreschk

Title of Thesis

Speech versus Controller - Suitability of speech based personal assistants to interact in 3D space.

Keywords

Command based Speec Recognigtion, 3D-Space, Interaction, Inclusion, Virtual Reality

Abstract

This bachelor thesis deals with the development and evaluation of a 3D virtual reality videogame based on speech recognition. The goal is to answer the question if it's possible to control a competitive multiplay-game via speech control as efficient as with a conventional videogame controller. This is especially important for people with motor disabilities since they usually tend to have problems controlling these kind of videogames.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Seit den frühen 60er Jahren wird an der automatisierten Spracherkennung geforscht. Dieses Forschungsfeld war seit seit jeher eines, welches eng an die Bedürfnisse von Menschen geknüpft ist, die mit körperlichen Einschränkungen zu leben haben. Menschen, denen ihre Gliedmaßen nicht oder nur eingeschränkt zur Verfügung stehen, können oft nur durch Sprache mit ihrer Umwelt interagieren. Das ist in Zeiten zunehmender Technologisierung einerseits ein Hindernis, andererseits aber auch eine Chance. Telefone, Computer und andere elektronische Gerätschaften sind oftmals nicht auf die Bedürfnisse von Menschen mit körperlichen Einschränkungen ausgelegt. Dadurch wird es diesen schwer bis unmöglich gemacht, am gesellschaftlichen Leben teilzunehmen. Auf der anderen Seite bieten sich gerade durch den technologischen Fortschritt Möglichkeiten diese Menschen besser in unsere Gesellschaft zu inkludieren. Dazu gehört insbesondere die automatisierte Spracherkennung.

Videospiele sind seit jeher ein Medium, welches Menschen die keinen Controller bedienen können in der Regel nicht zugänglich ist. Dabei gab es auch hier schon in den 70er Jahren Arbeiten zum Thema Spracherkennung, die Videospiele als Grundlage hatten. Als Beispiel ist hier „Voice Chess“[1] zu nennen, eine Videospiel-Umsetzung des Brettspiels Schach, die per Sprachkommandos gespielt werden konnte und im Jahr 1973 von Forschern der Carnegie Mellon University veröffentlicht wurde. Allerdings hat Spracherkennung nicht als gängige Alternative zu Videospiel-Controllern durchgesetzt. Vielmehr wird sie lediglich als abwechslungsreiche Ergänzung in Videospiele eingebaut, um diese aufzulockern oder immersiver zu gestalten.

Besonders in 3D-Spielen, die eine schnelle Reaktion und präzise Steuerung erfordern, findet Spracherkennung praktisch keine Beachtung. Negative Erfahrungen mit ungenauer

Spracherkennung und hohen Latenzen haben in der Vergangenheit selten die Frage aufkommen lassen, ob man diese Art von Spielen nicht auch ähnlich effizient per Spracheingabe steuern kann. Da auf dem Gebiet der Spracherkennung aber stets Fortschritte erzielt werden, ist es vielleicht an der Zeit sich diese Frage zu stellen und diese anhand einer prototypischen Anwendung zu beantworten.

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist es eine Videospiel-Anwendung zu entwickeln, die einerseits per Spracheingabe und andererseits per konventioneller Controller-Steuerung zu bedienen ist. Um die Spielerfahrung dabei abwechslungsreich und kompetitiv zu gestalten, soll diese Anwendung als Mehrspieler-Spiel konzipiert werden.

Da in den letzten Jahren die „Virtual Reality"-Technologie eine neue Blüte erlebt und Einzug in die Haushalte von Endkonsumenten findet, soll diese Anwendung ebenfalls als VR-Spiel entwickelt werden. So kann gleichzeitig untersucht werden, inwiefern Spracherkennung dazu beitragen kann VR-Erfahrungen noch intensiver und immersiver zu gestalten.

Die im Zuge dieser Arbeit entwickelte Videospiel-Anwendung soll anschließend auch evaluiert werden. Dafür soll eine Gruppe von Testpersonen das Spiel testen und im Rahmen einer Befragung die Frage beantworten, inwiefern Spracherkennung in diesem Kontext eine valide Alternative zum klassischem Videospiel-Controller darstellt.

1.3 Aufbau der Arbeit

Der Hauptteil dieser Bachelorarbeit teilt sich in die folgenden vier Kapitel auf.

- Kapitel 2 erörtert die Grundlagen zum Thema Spracherkennung und Inklusion. Dabei wird ein besonderer Fokus auf kommandobasierte Spracherkennung gelegt. Es werden vergleichbare Arbeiten vorgestellt, die sich ebenfalls mit kommandobasierter Spracherkennung auseinandersetzen. Weiterhin werden Arbeiten vorgestellt, die sich dem Thema Spracherkennung und Inklusion, sowie Spracherkennung in Videospielen widmen. Abschließend wird Spracherkennung in Videospielen im Laufe der Zeit betrachtet.

- Kapitel 3 betrachtet die funktionalen und nichtfunktionalen Anforderungen an das zu entwickelnde Spiel. Dabei werden die konkreten Rollen innerhalb der Anwendung spezifiziert.
- Kapitel 4 stellt den Entwurf des zu entwickelnden Videospiels vor, sowie die konkrete Umsetzung in der 3D-Engine „Unity3D“ mit der „HTC Vive“ als VR-System.
- Kapitel 5 erörtert die stattgefundene Evaluierung des Videospiels und bewertet die daraus hervorgegangenen Ergebnisse.

Im Abschluss wird in Kapitel 6 die Arbeit noch einmal zusammengefasst und ein kleiner Ausblick in die Zukunft gegeben.

2 Grundlagen

In den folgenden Unterkapiteln werden die Grundlagen ausgewählter Felder der Sprach-, Stimmen- und Ton-Erkennung erörtert. Die in Abschnitt 2.1 vorgenommene Klassifizierung des Anwendungsfeld in drei Kategorien dient dazu im weiteren Verlauf der Arbeit gezielt innerhalb der verschiedenen Technologien zur Spracherkennung zu differenzieren. Abschnitt 2.2 liefert eine historische Übersicht über die Meilensteine der kommandobasierten Spracherkennung (CSR) und erörtert Beispiele zum Thema. Abschnitt 2.3 setzt sich mit dem Thema Spracherkennung und Inklusion auseinander. Dabei werden Arbeiten zu diesem Thema vorgestellt, welche sich ebenfalls mit der Fragestellung auseinandergesetzt haben, wie man Menschen mit körperlichen Einschränkungen mittels Spracherkennung helfen kann Computer-Software zu steuern. Im Unterabschnitt 2.3.3 wird die Fragestellung im Kontext von Videospielen weiter vertieft. Abschließend werden die Grundlagen in Abschnitt 2.4 noch einmal zusammengefasst.

2.1 Die verschiedenen Arten von Spracheingabe als Software-Eingabeinterface

Das breite Forschungsfeld der digitalen Sprach-, Stimmen- und Ton-Erkennung, wird innerhalb der Literatur aus Gründen der Unterscheidbarkeit in verschiedene Bereiche unterteilt, wobei die jeweils verwendete Terminologie häufig nicht einheitlich ist. Im Folgenden Abschnitt wird dieses Feld daher von mir in drei Kategorien klassifiziert. Diese drei Kategorien werden dem jeweiligen Kerngebiet entsprechend sinnvoll bezeichnet, damit im restlichen Teil der Arbeit erfolgreich zwischen ihnen differenziert werden kann.

2.1.1 Speech Voice Recognition

Unter Speech Voice Recognition (SVR) versteht man die softwareseitige Analyse und Interpretation kontinuierlicher Sprache. Die Semantik der gesprochenen Sprache steht hier im Fokus, entsprechend werden die erfassten Wörter und Sätze einer Grammatik entsprechend, sinngemäß interpretiert und verarbeitet. Dies führt in der Theorie zu einem sehr mächtigen Werkzeug, innerhalb des Bereichs der Spracherkennung, da einer Software komplexe Sachverhalte und Eingaben auf einer Art und Weise vermittelt werden können, die der *Mensch zu Mensch*-Kommunikation sehr nahe kommt. Diverse Eigenarten der verschiedenen menschlichen Sprachen limitieren diese Art der Spracherkennung aber immens. So können z.B. Homophone (Wörter die gleich ausgesprochen werden, aber eine unterschiedliche Bedeutung haben) nur relativ unzuverlässig semantisch interpretiert werden.

2.1.2 Non Speech Voice Recognition

Die Non Speech Voice Recognition (NSVR) beschäftigt sich, wie der Name es schon verrät, nicht mit der menschlichen Sprache. Vielmehr werden hier die akustischen Eigenschaften gesprochener Laute oder auch gesummter Töne analysiert und entsprechend dem Verwendungszweck verarbeitet. Die maßgeblichen Eigenschaften, die dabei beobachtet werden sind die Tonhöhe (Frequenz), die Länge des Tons, sowie die Lautstärke (Amplitude). Diese analogen Eingabeparameter können softwareseitig vielseitig eingesetzt werden.

Abbildung 2.1 zeigt, wie in [19] mit Hilfe von Non Speech Voice Recognition eine Steuerung des Arcade Klassikers Tetris umgesetzt wurde. Innerhalb dieser Arbeit werden die vier verschiedenen Aktionen die es braucht um Tetris zu steuern, auf vier Tonmuster abgebildet. So bewegt ein in der Tonhöhe fallender Ton den aktuell im Spiel befindlichen Stein nach links, ein in der Tonhöhe steigender Ton bewegt ihn nach rechts. Ein kurzer in der Tonhöhe gleichbleibender Ton dreht den Stein und ein längerer in der Tonhöhe gleichbleibender Ton lässt ihn fallen.

Im Zuge dieser Arbeit wurde diese Umsetzung von NSVR mit einer kommandobasierten Spracherkennung (CSR) verglichen, wobei sich herausstellte, dass sich CSR zwar gerade zu Beginn für einen unerfahrenen Anwender intuitiver anwenden ließ, nach einer gewissen

Trainingsdauer die NSVR aber bessere Ergebnisse in Bezug auf Effizienz und Spielerfolg verzeichnen konnte.

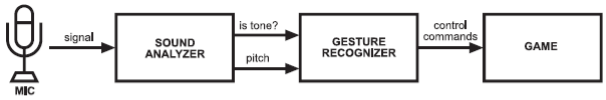

Figure 3: Non-speech sounds processing pipeline

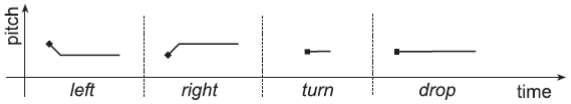

Abbildung 2.1: Verschiedene Tonhöhen-Verläufe und Tonlängen sind den Eingabe-Aktionen von Tetris zugeordnet. [19]

2.1.3 Command based Speech Recognition

In der Command based Speech Recognition (CSR) wird nicht wie in der NSVR der semantische Inhalt der gesprochenen Sprache interpretiert, sondern es werden spezifische Wörter verarbeitet, die dann als Kommandos zur Steuerung einer Applikation interpretiert werden. Diese Wörter entstammen einem endlichen Wörterbuch, mit verhältnismäßig geringem Umfang.

CSR ist deutlich weniger komplex und entsprechend weniger fehleranfällig als die SVR. Die im Zuge dieser Arbeit entwickelte Anwendung arbeitet mit CSR, da es nicht nötig ist mit der Software in einen komplexen Dialog zu treten. Einzelne Wörter, welche präzise und schnell verarbeitet werden können, eignen sich besser für die Steuerung einer Applikation, als eine Spracheingabe mittels SVR, in welcher die Bedeutung der Eingabe erst aufwendig verarbeitet und semantisch analysiert werden muss.

Eine der ersten Anwendungen, die eine entsprechende kommandobasierte Spracherkennung umgesetzt hat, wurde in [8] präsentiert. In den Bell Telephone Labroratories haben K.H. Davis, R. Biddulph und S. Balashek einen Schaltkreis „Audrey“ entwickelt, welcher in der Lage ist die in englischer Sprache gesprochen Zahlen 0-9 zu erkennen, mit einer Genauigkeit von 97 bis 99 Prozent.

2.2 Command based Speech Recognition

Unterabschnitt 2.2.1 stellt einige Meilensteine der Forschung zum Thema CSR in Form eines historischen Abrisses vor. Im Anschluss werden in diesem Kapitel vergleichbare Arbeiten vorgestellt, die sich ebenfalls mit der CSR auseinandergesetzt haben. Unterabschnitt 2.2.2 stellt eine Arbeit vor, die sich mit der Steuerung von Robotern durch CSR beschäftigt. Im Unterabschnitt 2.2.3 wird eine Arbeit präsentiert, die sich mit CSR-Schnittstellen innerhalb von Automobilen auseinandersetzt und gleichzeitig die dadurch entstehenden Auswirkungen auf den Fahrer erörtert. Im abschließenden Unterabschnitt 2.2.4 werden verschiedene Videospiele vorgestellt, die eine Eingabe auf CSR basierter Spracherkennung beinhalten.

2.2.1 Historie der Command based Speech Recognition

[8] gilt gemeinhin als Startpunkt der kommandobasierten elektronischen Spracherkennung. Der Schaltkreis war, nach Anpassung auf den individuellen Sprecher, in der Lage die gesprochen einstelligen Zahlen von 0-9 mit einer Genauigkeit von 97-99% zu erkennen.

Im Jahr 1962 veröffentliche IBM eine experimentelle Maschine mit dem Namen „Shoebox“. Diese Maschine war nicht nur in der Lage, die Zahlen von 0-9 zu erkennen, sondern konnte auch arithmetische Operatoren wie „Plus“ oder „Minus“ verstehen und somit eingesprochene Rechenoperationen verstehen, ausrechnen und das Ergebnis ausgeben.

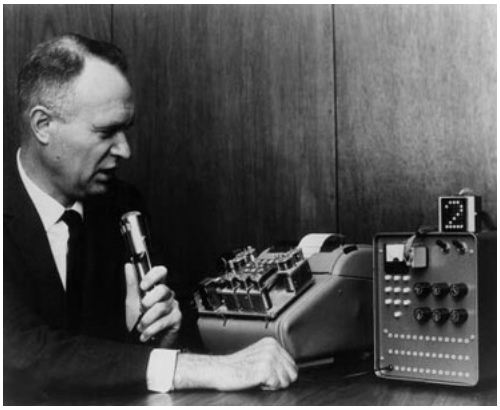

Abbildung 2.2: Dr. E. A. Quade demonstriert Shoebox. Quelle: https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html

In der ersten Hälfte der 1970er Jahre, hat das U.S. Department of Defense wesentlich zur Weiterentwicklung der Spracherkennung beigetragen. Von 1971 bis 1976 haben sie das DARPA SUR (Speech Understanding Research) Programm finanziert. Dies mündete 1976 in der Veröffentlichung von [15], entwickelt an der Carnegie-Mellon University, in Pittsburgh. Dieses System integriert Eigenschaften zweier vorangegangener Systeme, „Hearsay-1“ und „Dragon“ und war in der Lage komplette Sätze einer Grammatik, gebildet aus 37 Wörtern, zu verarbeiten. Dabei erreichte es eine Genauigkeit von über 95% beim Erkennen der einzelnen Wörter.

Ein weiterer wesentlicher Durchbruch auf dem Forschungsgebiet der Spracherkennung, war die Anwendung von Hidden Markov Modellen (HMM) innerhalb von Spracherkennungssystemen. Mithilfe eben dieser, ließ sich durch auf Statistiken basierenden Schätzungen, die Wahrscheinlichkeit untrainierte Spracheingaben den korrekten Einträgen eines Wörterbuchs zuzuordnen, immens steigern. [14]

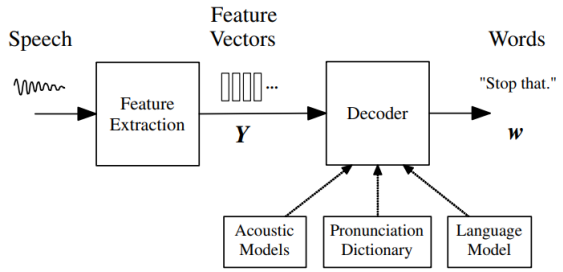

Abbildung 2.3: Architektur einer auf Hidden Markov Modellen basierenden Spracherkennung [9]

Die aus dieser Erkenntnis resultierenden Fortschritte, sowie die kontinuierliche Weiterentwicklung leistungsfähiger und immer günstiger werdender Hardware, hatten zur Folge, dass in den 1980er Jahren erste Produkte mit kommandobasierter Spracherkennung für Endkonsumenten auf den Markt gekommen sind. Abbildung 2.4 zeigt Julie, eine 1987 von Worlds of Wonder veröffentlichte Puppe, die als eines der ersten Spielzeuge eine rudimentäre Art von Sprach-Interaktion ermöglicht hat.

Endgültig auf dem Markt für Endkonsumenten angekommen, ist die Spracherkennung dann in den frühen 1990er Jahren. Die Firma Dragon veröffentlichte im Jahr 1990 die Software „DragonDictate 30K“, welche es Menschen erlaubte mit einem 5000 Wörter umfassenden Wörterbuch ihren Heimcomputer zu steuern. Schnell wurde offensichtlich, dass diese neue und moderne Technologie besonders Menschen mit Behinderungen helfen

Abbildung 2.4: Julie (1987) - Eines der ersten Spielzeuge mit Spracherkennung. Quelle: http://planetofthedolls.blogspot.com/2017/11/doll-day-2017-316worlds-of-wonder-julie.html

könnte, von nun an auch erfolgreich einen Computer zu bedienen. [7]

Über den Verlauf der folgenden Jahre, wurde die Spracherkennung immer weiter entwickelt und verbessert. Einen wesentlichen Hype hat das Thema erst in jüngster Vergangenheit erfahren. Seitdem Apple im Jahr 2011 ihren auf Sprach-Interaktion basierenden, digitalen persönlichen Assistenten „Siri“ veröffentlicht hat, wurde der Markt mit weiteren Sprach-Assistenten geradezu überflutet. Besonders hervorzuheben ist dabei, der von Amazon 2014 veröffentlichte Sprach-Assistent „Alexa“, welcher maßgeblich dafür entwickelt wurde, nun nicht nur einen Computer, sondern einen ganzen Haushalt per Sprachkommandos steuern zu können. Andere Firmen wie Google veröffentlichten Konkurrenz-Produkte (Google Home) und so existiert aktuell ein vielfältiger Markt basierend auf persönlichen Sprach-Assistenten.

Neben Anwendungen, die sich rein und ausschließlich auf Spracherkennung berufen, wurde auch umfangreich an sogenannten multimodalen Interaktions-Systemen geforscht. Dies sind Systeme, die neben der Stimme des Interagierenden, noch andere unkonventionelle körperliche Eigenschaften des Anwenders mit in die Computer-Mensch-Interaktion einbeziehen. Das prominenteste Beispiel hierfür ist mit Sicherheit die von Richard A. Bolt veröffentlichte Arbeit [5], von 1980. Am Massachusetts Institute of Technology wurde hier ein Aufbau entwickelt, welcher es Anwendern ermöglichte mit dem Finger auf eine Stelle einer Leinwand zu zeigen und parallel dazu per kommandobasierter Spracheingabe dem System zu sagen, an welcher Stelle es welches Symbol platzieren soll.

Abbildung 2.5: Amazon Echo, in schwarz. 1. Generation. Quelle: https://de.wikipedia.org/wiki/Amazon_Echo

Auch wenn CSR als Spracherkennung in modernen Sprachassistenten wie Amazons „Alexa“ nicht oder nur selten eingesetzt wird, hat CSR dennoch auch heutzutage noch eine Berechtigung. Besonders Systeme die schnelle Reaktionen erfordern, vertrauen auch heute noch auf CSR. Aus genau diesem Grund wurde auch CSR für diese Bachelorarbeit ausgewählt.

2.2.2 CSR zur Steuerung von Robotern

Im Zuge der 2008 veröffentlichten Arbeit [2] haben chinesische Forscher eine auf CSR basierende Spracherkennungs-Schnittstelle entwickelt, welche dazu dient einen auf vier Rädern fahrenden Roboter Kommandos ausführen zu lassen. Der Anwender spricht in ein Mikrofon, welches an einen Computer angeschlossen ist. Wird vom Computer eines der sieben Kommandos des Wörterbuchs erkannt, wird der entsprechend hinterlegte Befehl per WiFi an den Roboter übertragen, der im Anschluss daran den besagten Befehl ausführt. Um den praktischen Nutzen dieser Anwendung einzuordnen, wurde das System in einer ruhigen Umgebung getestet, aber auch in einer Umgebung mit starken Hintergrundgeräuschen. In Abbildung 2.6 können wir sehen, dass sämtliche Kommandos innerhalb der ruhigen Umgebung zu 100% korrekt erkannt wurden. Sobald aber starke Geräusche im Hintergrund vorhanden sind, nimmt die Genauigkeit ab. Teilweise werden Kommandos nur noch in 7 von 10 Fällen richtig erkannt.

Anhand dieser Ergebnisse lässt sich deutlich sehen, dass Anwendungsszenarios die eine Umgebung mit starken Nebengeräuschen beinhalten mit Einschränkungen rechnen müssen, sollte hier mit CSR gearbeitet werden. Besonders wenn eine besonders hohe

TABLE I
EXPERIMENT RESULTS IN NORMAL ENVIRONMENT

No.	Command	Test times	Correct times
1	go forward	10	10
2	go backward	10	10
3	turn left	10	10
4	turn right	10	10
5	extend	10	10
6	reposition	10	10
7	stop	10	10

TABLE II
EXPERIMENT RESULTS IN NOISE ENVIRONMENT

No.	Command	Test times	Correct times
1	go forward	10	8
2	go backward	10	9
3	turn left	10	7
4	turn right	10	7
5	extend	10	9
6	reposition	10	8
7	stop	10	10

Abbildung 2.6: Testergebnisse in ruhiger Umgebung, sowie in Umgebung mit starken Hintergrundgeräuschen. [2]

Genauigkeit von maßgeblicher Bedeutung ist, ist der erfolgreiche Einsatz von CSR vermutlich kaum zu realisieren. Eine im Jahr 2011 veröffentlichte Arbeit [18] kommt zu ähnlichen Ergebnissen, im Bezug auf die Umgebungslautstärke. Im Zuge dieser Arbeit wurde ein ebenfalls auf CSR basierendes Spracherkennungssystem entwickelt, welches speziell auf den flexiblen Einsatz in Industrie-Umgebungen zugeschnitten ist. Während das System in einer ruhigen Umgebung noch 92 von 100 Sprach-Kommandos korrekt verarbeiten konnte, wurden in einer Umgebung mit anhaltenden Nebengeräuschen nur noch 82 von 100 Sprach-Kommandos korrekt erkannt. Als Möglichkeit diese Ergebnisse zu verbessern, wurde vorgeschlagen einen besonderen Wert auf hochwertige Mikrofone, mit eingebautem Noise-Filter zu setzen.

2.2.3 CSR innerhalb von Automobilen

Sprach-Schnittstellen haben innerhalb der letzten Jahre auch ihren Platz innerhalb von modernen Automobilen gefunden. Der grundlegende Gedanke dabei ist, dass der Fahrer seinen Blick nicht von der Straße und die Hände nicht vom Lenkrad nehmen muss, während er beispielsweise das Radio oder auch das Navigationssystem konfiguriert. Auf Basis dieses Gedanken, sollen Sprach-Systeme innerhalb von Automobilen nicht nur dem Komfort des Fahrers dienen, sondern weiterhin auch das Unfall-Risiko senken.

Eine 2009 veröffentlichte Studie [16] hat eruiert inwiefern sich die Verwendung eines auf CSR basierenden Sprach-Systems auf verschiedene Verhaltensweisen eines Autofahrers auswirkt. Dafür wurde ein Test-Aufbau geschaffen, in welchem die Test-Fahrer einem Straßenverlauf folgen sollten, welcher in drei Spuren unterteilt ist. In unregelmäßigen Abständen bekommen sie dabei durch Straßenschilder die Aufgabe die aktuelle Spur zu

wechseln. Während dieser Fahrt bekommen sie vom Beifahrer sogenannte sekundäre Aufgaben gestellt, die sie mal manuell und mal mit Hilfe von Spracheingabe lösen sollten. Zu diesen Aufgaben gehörten unter anderem die Konfiguration eines Navigationssystems, das Bedienen des Autotelefons oder das Wechseln des aktuellen Songs im Musik-System.

Eine der untersuchten Verhaltensweisen war die Abweichung von einer vorher festgelegten Idealstrecke, während des Spurwechsels. Die Ergebnisse in Abbildung 2.7 zeigen, dass die Abweichung von der Ideallinie(Baseline) abgenommen hat, wenn die Aufgaben mittels Sprachkommandos erledigt wurden. POI steht für *Point Of Interest* und die dazugehörige Aufgabe im Navigationssystem einen solchen als Zielort zu konfigurieren. Bei der Aufgabe Multiple musste der Testfahrer eine Adresse im Navigationssystem eingeben und dabei unabhängig von der Eingabemethode vielfach auf das visuelle Feedback des Systems achten. Die Aufgabe Single wurde mit einem Navigationssystem ausgeführt, welches ausschließlich per Sprache zu steuern ist und welches einmalig ein Feedback per Audio-Ausgabe an den Fahrer übermittelt hat.

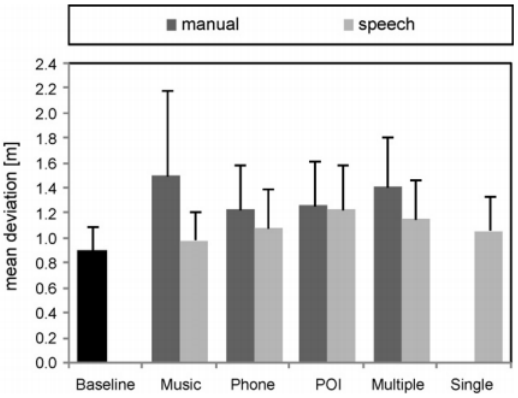

Abbildung 2.7: Abweichung von der Ideallinie [16]

Abbildung 2.8 zeigt uns die Ergebnisse im Bezug auf die Reaktionszeit der Fahrer. Hier lässt sich auch deutlich erkennen, dass die Fahrer über alle Aufgaben hinweg deutlich schneller reagiert haben, wenn sie die Aufgaben per Sprache gelöst haben. Im Fall der Radio-Aufgabe, bei welcher ein vom Beifahrer genannter Song im Musik-System eingestellt werden sollte, hat sich die Reaktionszeit sogar halbiert. Die als ideale Reaktionszeit(Baseline) definierte Zeit von 0.05 Sekunden konnte aber auch bei Verwendung der Sprachsteuerung nicht erreicht werden.

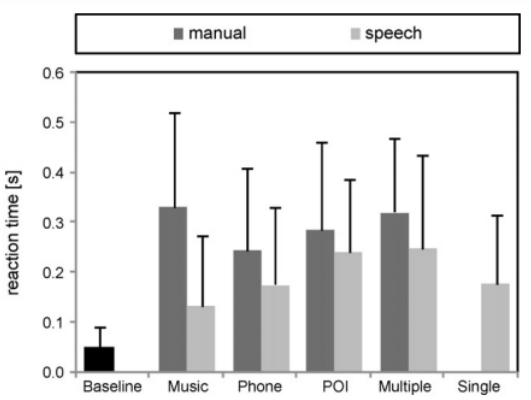

Abbildung 2.8: Reaktionszeiten der Testfahrer [16]

Die deutlichsten Unterschiede beim Fahrverhalten wurden bei der Betrachtung der *gaze duration* gemessen. Dabei wurde untersucht zu wie viel Prozent der Zeit der Fahrer seinen Blick von der Straße genommen hat. In Abbildung 2.9 lässt sich erkennen, dass hierbei mitunter drastische Unterschiede zu verzeichnen sind. Während der Musik-Aufgabe haben die Fahrer ihren Blick praktisch nie von der Straße genommen wenn sie das Musik- System mithilfe der Sprach-Steuerung bedient haben. Bei manueller Bedienung des Musik-Systems haben sie ihren Blick hingegen während fast 40% der Aufgabenerfüllung von der Straße genommen.

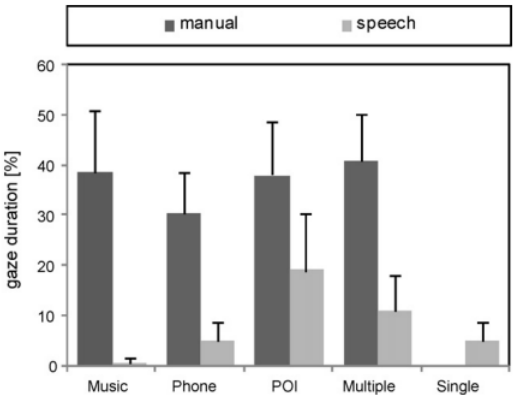

Fig. 4. Mean (bar) and standard deviation (whiskers) of the percentage of gaze time away from the LCT screen for the different IVIS comparing manual vs. speech control. The baseline condition is not shown as it was 0%.

Abbildung 2.9: Zeit die die Fahrer ihren Blick von der Straße genommen haben [16]

Somit wurde in [16] festgestellt, dass der Nutzen von auf CSR basierender Spracherkennung in Automobilen definitiv vorhanden ist. Die Fahrer konnten schneller reagieren, bessere Spurwechsel-Manöver ausführen und mussten deutlich seltener ihren Blick von der Straße nehmen.

2.2.4 CSR zur Steuerung von Videospielen

Videospiele waren seit jeher ein gern benutztes Medium um neue Technologien, wie auf CSR basierende Sprach-Schnittstellen, auszutesten.

Im Jahr 1973 veröffentlichten Forscher der Carnegie Mellon University ihr Spiel „Voice Chess“[1], welches auf Basis des „Hersay-1“ Spracherkennung-System entwickelt wurde. In diesem frühen Videospiel ließ sich auf rudimentäre Art und Weise ein Schach-Spiel mit Sprachkommandos steuern. Die Anwendung verarbeitete Befehle wie „Bishop to Queen Three“ und bewegte die Schachfiguren entsprechend.

Im Jahr 1985 wollte die Firma RDI Video Systems eine Videospiel-Konsole mit dem Namen „Halcyon“[1] auf den Markt bringen. Auch wenn diese nie den Endkonsumenten erreicht hat, war der Prototyp für die damalige Zeit beeindruckend. Die Konsole ließ sich komplett per Sprachkommandos steuern. Insgesamt sollten nach einer Trainingsphase über 1000 Kommandos von „Halycon“ verstanden und verarbeitet werden können.

[1] http://www.videogameconsolelibrary.com/pg80-rdi.htm

Allerdings hat sich die Spracherkennung des Systems als nicht besonders ausgereift herausgestellt und hat die damaligen Testanwender eher frustriert als bereichert.

Abbildung 2.10: Die RDI Halcyon hat niemals die Marktreife erreicht Quelle: https://www.videogameconsolelibrary.com/pg80-rdi.htm

Im Jahr 1998 veröffentlichte Nintendo für ihre damalige „Nintendo 64“ Videospielkonsole ein Spracherkennung-Kit mit dem Namen *VRU - Voice Recognition Unit*[2]. Nur zwei Spiele(*Hey You, Pikachu!*, *Densha De Go! 64*) machten Gebrauch von dieser auf CSR-Spracherkennung basierenden Erweiterung, aber andere Videospiel Entwickler wie Sega sollten in den folgenden Jahren ebenfalls Gebrauch vom Prinzip dieser Technologie machen.

Abbildung 2.11: Die Nintendo Voice Recognition Unit Quelle: https://nintendo.fandom.com/wiki/VRU

So hat Sega für ihre 1999 erschienene Videospielkonsole *Sega Dreamcast* ebenfalls ein Mikrofon-Kit entwickelt, von dem einige Spiele Gebrauch machten. Besonders hervorzu-

[2] https://nintendo.fandom.com/wiki/VRU

heben ist dabei das Spiel „Seaman“[3], in welchem der Spieler per Spracheingabe Fragen beantwortet, die ihm die im virtuellen Aquarium lebenden Protagonisten des Spiels gestellt haben.

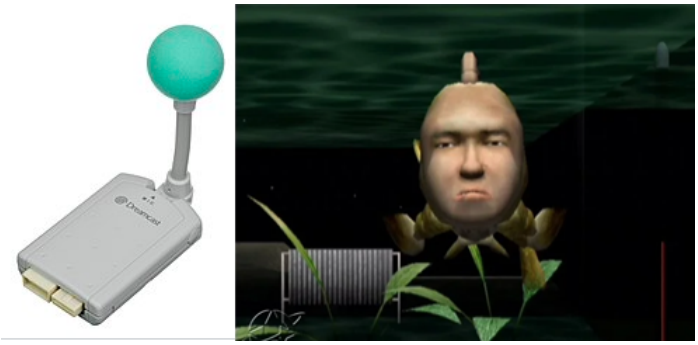

Abbildung 2.12: Links - das Sega Mikrofon-Kit Quelle: https://commons.wikimedia.org/wiki/File:Dreamcast-Microphone.jpg
Rechts - einer der Protagonisten aus Seaman (1999) Quelle: https://www.ign.com/images/games/seaman-dc-14720

Im Jahr 2005 veröffentlichte Nintendo für ihre tragbare Videospielkonsole *Nintendo DS* das Spiel *Nintendogs*[4]. Hier pflegt und betreut der jeweilige Spieler einen virtuellen Hund und kann diesen Befehle per Sprachkommando(„Sitz“, „Platz“) ausführen lassen. [1]

Die Möglichkeit gewisse Aspekte von Videospielen per Sprachkommandos zu steuern, hat sich in den folgenden Jahren in der Videospiel-Industrie etabliert. Moderne Videospielkonsolen brauchen dafür keine exklusive Peripherie, da diese in der Regel mit jedem gängigen Headset kompatibel sind. Ein besonders populäres Beispiel ist dabei das 2012 veröffentlichte Videospiel *Mass Effect 3*. In diesem Spiel stehen dem Spieler eine Reihe von Sprachkommandos zur Verfügung um Nicht-Spieler-Charaktere zu befehligen.[5]

In den letzten Jahren hat die *Virtual-Reality* Technologie einen massiven Fortschritt zu verzeichnen. VR-Systeme wie die *HTC Vive* oder die *Oculus VR* haben zahlreiche Anwendungen und Spiele hervorgebracht, deren maßgeblicher Fokus die Immersion des Anwenders ist. Der Spieler soll in die Welt abtauchen können und sich möglichst wenig bewusst sein, dass er gerade mit einer Software interagiert. Dabei ist Spracherkennung auch ein maßgeblicher Faktor, der zur Immersion beitragen kann.

[3] https://en.wikipedia.org/wiki/Seaman_(video_game)
[4] https://de.wikipedia.org/wiki/Nintendogs
[5] https://www.ign.com/wikis/mass-effect-3/Kinect_Controls

Abbildung 2.13: Nintendogs - Spieler können per Sprachkommandos mit virtuellen Hunden interagieren. Quelle: https://www.gamerstemple.com/vg/games18/002010/002010s.asp?screen=6

Im Jahr 2017 hat der Entwickler *Human Interact* zusammen mit Microsoft das VR-Spiel *Starship Commander*[6] angekündigt, in welchem der Spieler ein Raumschiff steuert und ausschließlich per Spracherkennung mit den Nicht-Spieler-Charakteren und der Spielumgebung interagiert.

Die im Zuge dieses Unterkapitels betrachteten Arbeiten haben anschaulich demonstriert, dass CSR für verschiedene Anwendung einen immensen Nutzen haben kann. Ebenfalls hat sie in vielen Videospielen Einzug gefunden, wenn auch nur selten zur alleinigen Steuerung eben dieser Spiele. Aus genau diesem Grund soll innerhalb dieser Arbeit untersucht werden, inwiefern sich CSR dafür eignet ein Videospiel ausschließlich mit Hilfe dieser Art der Spracheingabe zu steuern und ob sie eine ebenbürtige Alternative zum klassischen Videospiel-Controller darstellt.

[6] http://human-interact.com/

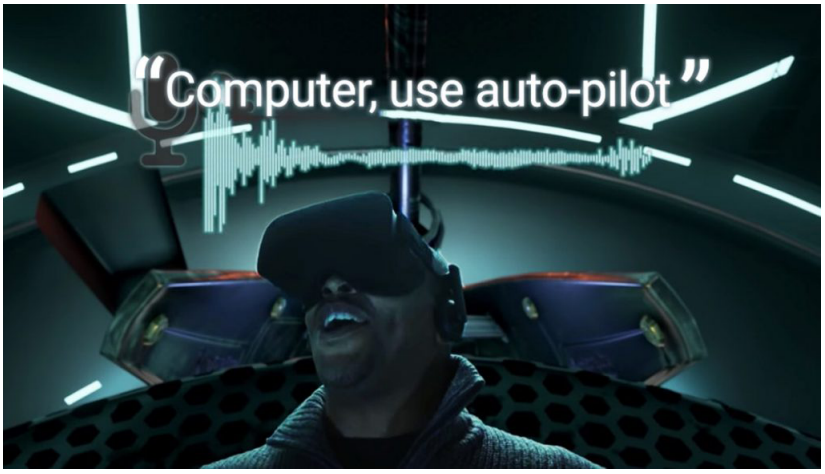

Abbildung 2.14: Starhip Commander - Immersion durch Spracherkennung. Quelle: https://www.roadtovr.com/conversational-gameplay-interactive-narrative-starship-commander/

2.3 Inklusion mittels Spracherkennungssystemen

In den folgenden Kapiteln wird erläutert, auf welche Art und Weise Spracherkennung Menschen mit körperlichen Einschränkungen eine Möglichkeit bieten kann Applikationen zu bedienen, welche sie mit konventionellen Eingabe-Methoden nur schwer bis gar nicht steuern können. Unterabschnitt 2.3.1 diskutiert in dem Zusammenhang die grundsätzliche Relevanz der Thematik, sowie auf Sprach-, Stimmen- und Tonerkennung basierende Lösungsansätze. Unterabschnitt 2.3.2 stellt Arbeiten vor, die sich mit der Thematik Inklusion im Zusammenhang mit Spracherkennung auseinandergesetzt haben. Da in dieser Arbeit ein spezieller Fokus auf Inklusion durch Spracherkennung in Videospielen gelegt wird, stellt der abschließende Unterabschnitt 2.3.3 vergleichbaren Arbeiten vor, die ihren Fokus ebenfalls auf Videospiele gesetzt haben.

2.3.1 Spracherkennung als Möglichkeit zur Inklusion

Die World Health Organization hat im Jahr 2011 einen Report [4] veröffentlicht, welcher berichtet, dass Schätzungen zufolge 15% der menschlichen Bevölkerung eine körperliche oder geistige Einschränkung haben. Verschiedene weiter genannte Schätzungen vermuten, dass zwischen 2.2% und 3.8% der menschlichen Bevölkerung mit einer sogenannten schweren Behinderung leben müssen. Als eine maßgebliche Maßnahme, um diesen Menschen

ihr Leben einfacher und lebenswerter zu gestalten, empfiehlt der Report unter anderem Barrieren im Alltag zu beseitigen. Speziell für Menschen mit motorischen Einschränkungen, die beispielsweise ihre Hände nicht oder nur sehr eingeschränkt verwenden können, stellt die Bedienung konventioneller Eingabe-Schnittstellen sehr häufig eine unüberwindliche Barriere dar. Somit besteht also ein Bedarf alternativer Eingabe-Möglichkeiten, die es Menschen mit motorischen Einschränken ermöglicht Software- und Hardware-Systeme ebenfalls steuern und bedienen zu können.

Dass Spracherkennungssysteme ein variabler Kandidat für solche alternativen Eingabe-Schnittstellen sind, wurde schon früh innerhalb der Forschung von Spracherkennung deutlich. [10] präsentiert schon im Jahr 1977 ein Spracherkennungssystem, welches auf die Bedürfnisse von im Krieg schwer verwundeter Veteranen zugeschnitten war. Mit Hilfe von CSR basierter Spracheingabe, kann der jeweilige Anwender durch Hilfe des vorgestellten Systems sein Haustelefon steuern, eine elektrische Schreibmaschine bedienen oder sich auch simple Grundrechenoperationen ausrechnen lassen.

Im Jahr 1980 wurde innerhalb von [6] ein Sprach-Assistent vorgestellt, welcher ebenfalls maßgeblich für Menschen mit körperlichen Einschränkungen konzipiert wurde. Neben der Möglichkeit ein Telefon, oder eine Schreibmaschine zu bedienen, ist es mit Hilfe des im Zuge der Arbeit entwickelten Systems auch möglich einen elektrischen Rollstuhl zu steuern. Bei der Entwicklung wurden spezielle Design-Kriterien aufgestellt, welche es bei der Entwicklung solcher Assistenten zu berücksichtigen gilt:

- Größe und Gewicht des Systems
- Energieverbrauch
- Größe des Wörterbuchs
- Verarbeitungszeit
- Genauigkeit der Spracherkennung
- Flexibilität

Besonders im Zusammenhang mit Kriterien wie Verarbeitungszeit und Genauigkeit wird deutlich, dass SVR für dieses Einsatzgebiet weniger geeignet ist als CSR, da aufgrund der hohen Komplexität eben genau diese Faktoren in der Regel leiden und dafür auch praktisch keinen Mehrwert liefern.

2.3.2 Inklusion durch Spracherkennung in Software

Im Zuge der *SICE Annual Conference 2007* veröffentlichten japanische Forscher ihre Arbeit [17], im Zuge welcher sie ein Sprach-Interface für einen Rollstuhl entwickelt haben, welches sich per Sprachkommandos steuern lässt. Die Motivation war es älteren Menschen eine Schnittstelle zu bieten, mit welcher sie auf natürliche Weise mit ihrer Umwelt interagieren können.

Das Sprach-Interface *Julian* versteht dabei 14 verschiedene Sprachkommandos, die sich Abbildung 2.15 entnehmen lassen. Neben Kommandos wie *run forward*, die solange ausgeführt werden bis der Anwender sie ebenfalls mittels Sprachkommando(*stop*) abbricht, sind weiterhin Kommandos vorhanden um den Rollstuhl nur graduell zu steuern. So kann der Rollstuhl mittels Sprachkommando um 30 Grad nach links oder rechts rotiert werden, oder auch nur um 30cm nach vorne oder hinten gefahren werden lassen. Weiterhin sind *acceptance commands* und *rejection commands* vorhanden, die dazu dienen die von Julian erkannten Eingaben entweder zu verifizieren oder abzubrechen. Diese dienen der Sicherheit und sollen verhindern, dass sich der Anwender aufgrund von falsch erkannten Kommandos in Gefahr bringt.

Table 1 Voice command and reaction.

	command	reaction (mode)
①	tomare	stop
②	susume	run forward
③	sagare	run backward
④	migi	turn right / rotate right
⑤	hidari	turn left / rotate left
⑥	sukoshi-susume	run forward about 30cm
⑦	sukoshi-sagare	run backward about 30cm
⑧	sukoshi-migi	rotate right about 30 degrees
⑨	sukoshi-hidari	rotate left about 30 degrees
Ⓐ	OK / yes	acceptance command
Ⓑ	torikeshi / no / cancel	rejection command

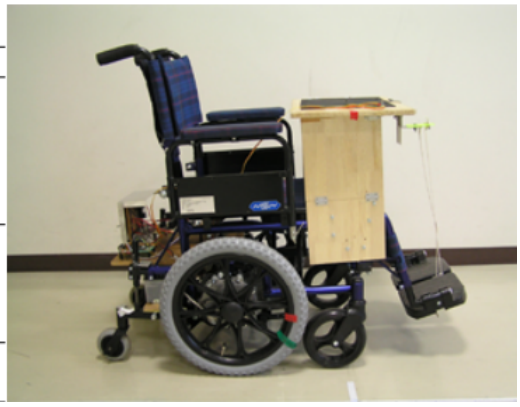

Abbildung 2.15: Sprachkommandos die Julian verarbeiten kann [16]

Getestet wurde Julian mit Hilfe von insgesamt 15 Test-Personen, die verschiedene Teststrecken mit einem elektronischem Rollstuhl abgefahren sind. Dabei sind die Probanden die gleiche Strecke mehrfach gefahren, einmal mit Julian als Sprach-Schnittelle und einmal mit manueller Tasten-Steuerung. Julian konnte im Zuge dieser Untersuchung Sprachkommandos mit einer Genauigkeit von 98,3% erkennen. Untersucht wurden im Zuge der Testfahrten jeweils drei Qualitäts-Faktoren: Die Zeit die es benötigt die Teststrecke zu-

rückzulegen, die Strecke die der Rollstuhl insgesamt gefahren ist und die Anzahl der Eingabe-Aktivitäten.

In Abbildung 2.16 sind die Ergebnisse dreier Probanden abgebildet. Proband A und Proband B verwenden Julian dabei zum ersten mal, während Proband C einer der Entwickler von Julian ist und im Vorwege entsprechend viel mit dem System geübt hat. Betrachtet man die Zeit, die unter Verwendung von Julian gebraucht wurde um die Strecke zurückzulegen, lässt sich deutlich erkennen dass der trainierte Proband C, im Vergleich zu Proband A und B, nur einen Bruchteil derer benötigt hat. Daraus kann man schließen, dass sich durch Übung im Umgang mit Julian eine deutliche Verbesserung in der Anwendung des Systems herstellen lässt. So konnte Proband B zum Beispiel auch unter Verwendung der manuellen Eingabe, die Teststrecke nicht schneller zurücklegen als der trainiere Proband C mit Hilfe von Julian.

Im Durchschnitt haben die Probanden unter Verwendung von Julian nur marginal mehr Strecke mit dem Rollstuhl zurücklegen müssen um ihr Ziel zu erreichen, als wie sie es unter Verwendung der manuellen Steuerung getan haben. Die Anzahl der Eingabe-Aktivitäten ist im Durchschnitt geringer, wenn die Probanden den Rollstuhl manuell gesteuert haben. Wobei hier der trainierte Proband C mit Julian in der Lage war insgesamt zwei Eingabe-Aktivitäten einzusparen, im Vergleich zu seiner manuell gesteuerten Testfahrt.

(a) Voice control

person	running time [s]	running distance [cm]	the number of input command
A	143.95	1120	8
B	256.87	984	30
C	49.43	959	7
average	150.08	1021	15

(b) Key control

person	running time [s]	running distance [cm]	the number of input command
A	34.15	1042	7
B	49.70	947	19
C	38.51	983	9
average	40.79	991	11.7

Abbildung 2.16: Ergebnisse der Probanden im Vergleich [16]

2.3.3 Inklusion durch Spracherkennung in Videospielen

Als Unterhaltungsmedium, sind Videospiele aus der heutigen Gesellschaft nicht mehr wegzudenken. Ein aktueller Report [3] der in Washington sitzenden Entertainment Software Association berichtet, dass alleine 60 Prozent aller US-Amerikaner Videospiele spielen. Es wäre wünschenswert, wenn ein solches Breitenphänomen auch für alle Menschen zugänglich ist, die gerne daran teilhaben möchten. Leider gibt es viele Faktoren, die es besonders körperlich benachteiligte Gruppen der Gesellschaft schwer machen, einen Zugang zu finden. Menschen mit fehlenden Gliedmaßen oder Behinderungen, ist es oft unmöglich mittels eines klassischen Controllers, oder auch einer Maus und Tastatur-Schnittstelle ein Videospiel zu spielen. Daher stellt sich die Frage, ob unkonventionelle Methoden der Eingabe, wie ein Spracherkennungssystem, solchen Menschen in diesem Fall nicht helfen können.

Neben der reinen Ermöglichung, einen körperlich benachteiligtem Menschen ein Videospiel mittels Spracheingabe spielen zu lassen, ist es auch wichtig zu untersuchen inwiefern sich Präzision, Effektivität und Erfolg bei unterschiedlicher Eingabemethode verändert. Viele der populärsten Videospiele spielt man nicht alleine, sondern mit oder gegen andere Spieler. Damit hier Chancengleichheit sichergestellt wird, sollten unterschiedliche Arten der Eingabe, einzelne Spieler weder maßgeblich bevorteiligen, noch benachteiligen.

Im Jahr 2006 veröffentlichten Forscher der University of Washington ihre Arbeit [11]. Mittels NSVR haben sie im Zuge dieser eine Software entwickelt, die wie ein virtueller Joystick funktioniert, welcher sich Mithilfe von gesummten oder gesungenen Tönen steuern lässt. Der Anwender kann somit einen Mauszeiger steuern, indem er unterschiedliche Vokale erklingen lässt, welche wiederum verschiedenen Richtungen im 2D-Raum zugeordnet sind. Über die Lautstärke des Tons lässt sich parallel dazu die Bewegungsgeschwindigkeit kontrollieren. Abbildung 2.17 zeigt, welche Vokale gesprochen, gesummt oder gesungen werden müssen um dem virtuellen Joystick eine entsprechende Eingabe vollführen zu lassen.

Getestet wurde der Vocal Joystick gegen zwei auf CSR basierenden Kontroll-Methoden, namentlich Mouse Grid und Speech Cursor. Bei Benutzung von Mouse Grid ist der Bildschirm stets in neun gleichmäßig aufgeteilte Abschnitte unterteilt, in welche der Anwender seine Maus per Sprachkommando hineinsteuern kann um dann im Anschluss diesen Abschnitt abermals zu unterteilen und so weiter. Speech Cursor ist eine Anwendung, bei welcher der Anwender den Mauszeiger über Sprachkommandos wie „Maus links“, „Maus

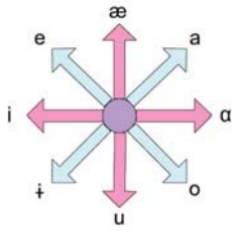

Abbildung 2.17: The Vocal Joystick. Mittels verschiedener Vokale, steuert man einen Cursor in unterschiedliche Richtungen. [11]

hoch“ oder „Maus Stop“ kontinuierlich steuern kann. Dem Test-Ergebnis in Abbildung 2.18 lässt sich entnehmen, dass der Vocal Joystick nur marginal schlechter abschneidet, als Mouse Grid und dafür signifikant besser als Speech Control.

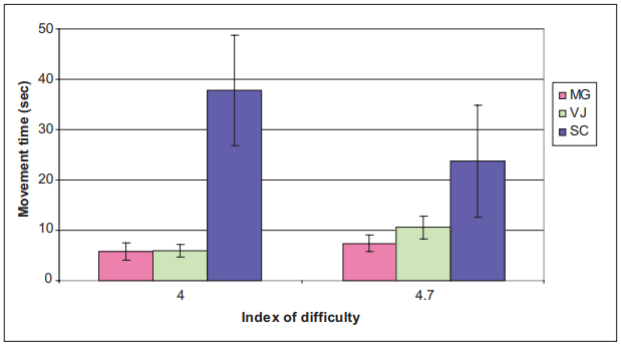

Abbildung 2.18: The Vocal Joystick nur unwesentlich langsamer als Mouse Grid[11]

Basierend auf der Technologie des Vocal Joysticks haben die größtenteils identischen Forscher sich in der Arbeit [12] mit der Frage auseinandergesetzt, wie effektiv sich diese Technologie einsetzen ließe um Videospiele inklusiver zu gestalten. Dabei wurde ein besonderes Augenmerk auf die Reaktionszeiten und Verarbeitungszeiten verschiedener Eingabemethoden gelegt, da diese elementar sind, für die erfolgreiche Steuerung vieler moderner Videospiele.

Ein entsprechender Testaufbau, ließ eine konventionelle Eingabe per Tastendruck, eine kommandobasierte Sprachsteuerung und den Vocal Joystick[11] gegeneinander antreten. Die Probanden mussten im ersten Durchlauf auf einen visuellen Befehl hin, so schnell wie möglich eine vorher vereinbarte Aktion mit der entsprechenden Eingabemethode durchführen. Im zweiten Durchlauf wurde den Probanden plötzlich eine Entscheidungsaufgabe gestellt, auf welche sie ebenfalls so schnell wie möglich mit der korrekten Eingabe rea-

gieren sollten. Neben Probanden ohne körperliche Beeinträchtigungen, wurden auch zwei Probanden mit motorischen Einschränkungen getestet. Proband MI1 war ausschließlich in der Lage eine konventionelle Tastatur mit einem Mund-Stab zu bedienen, während Proband MI2 lediglich seinen Handrücken benutzen konnte.

Das Resultat in Abbildung 2.19 zeigt deutlich, dass sich CSR und NSVR im Bezug auf Reaktionszeit nur marginal unterscheiden, der Unterschied im Bezug auf die Verarbeitungszeit aber sehr deutlich zu sehen ist. Trotz Einschränkungen beider Probanden, war die jeweilige physische Eingabe per Tastendruck im Bezug auf Reaktions- sowie Verarbeitungszeit der Spracherkennung in allen Disziplinen überlegen.

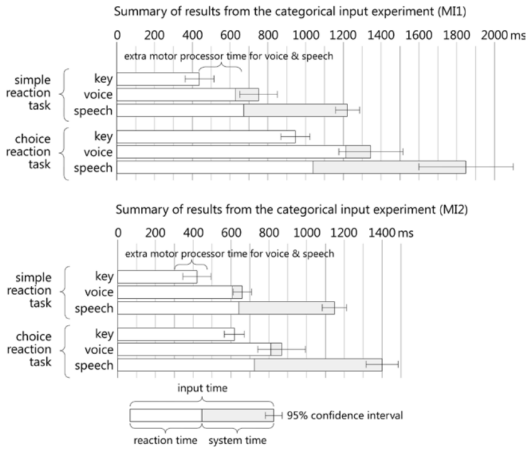

Abbildung 2.19: Ergebnisse der körperlich beeinträchtigten Probanden. [12]

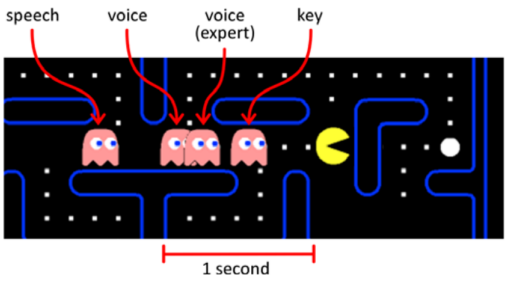

Abbildung 2.20: Wie weit ein Geist entfernt sein muss, um noch rechtzeitig reagieren zu können. [12]

Als Schlussfolgerung dieser Arbeit lässt sich vermuten, dass auf CSR und NSVR basierende Eingabemethoden durchaus für den Einsatz in Videospielen in Frage kommen, man aber in Bezug auf Reaktionszeit und Input-Delay einen Nachteil zu berücksichti-

gen hat. Videospielen, die eine sehr schnelle Reaktion, sowie eine schnelle Verarbeitung der Eingabe erfordern, sind vermutlich schlechter für diese unkonventionellen Arten der Eingabe geeignet. Dies wurde auch noch einmal anschaulich am Beispiel des Arcade-Klassikers Pac-Man demonstriert. In Abbildung 2.20 wird veranschaulicht, wie weit ein Gegner (Geist) von Pac-Man jeweils entfernt sein müsste, damit eine entsprechende Art der Eingabe es ermöglicht, noch rechtzeitig mit einem Richtungswechsel zu reagieren.

2.4 Zusammenfassung

Dieses Kapitel hat die Grundlagen zusammengefasst, welche für die restliche Arbeit essentiell sind. Zu Beginn wurde eine Klassifizierung von unterschiedlichen Teilgebieten der Sprach-, Stimmen- und Ton-Erkennung vorgenommen, innerhalb welcher die jeweiligen Teilgebiete kurz und bündig erläutert wurden. Das Ziel dieser Klassifizierung war es, eine Grundlage für den weiteren Verlauf der Arbeit zu schaffen, auf Basis derer gezielt innerhalb dem doch sehr komplexen Forschungs- und Anwendungsfeld der Sprach-, Stimmen- und Ton-Erkennung differenziert werden kann.

Im Anschluss daran wurde ein historischer Abriss über die Forschung zur Spracherkennung und die entsprechenden Meilensteine gegeben. Dabei wurde ein besonderer Fokus auf die auf CSR basierende Spracherkennung gelegt, da diese auch für die im Zuge dieser Arbeit entwickelte Anwendung verwendet wurde.

Das darauf folgende Unterkapitel beschäftigt sich dann noch einmal intensiv mit der auf CSR basierenden Spracherkennung. Es werden vergleichbare Arbeiten vorgestellt, die auf Basis dieser Technologie Anwendungen erschaffen oder untersucht haben. Weiterhin wird die Historie von CSR basierter Spracherkennung innerhalb von Videospielen beleuchtet, da im Zuge dieser Arbeit ebenfalls eine Videospiel-Anwendung entsteht.

Das abschließende Unterkapitel beschäftigt sich mit dem Thema Spracherkennung und Inklusion. Zunächst wird einmal erörtert, inwiefern Spracherkennung zur Inklusion von Menschen mit körperlichen Beeinträchtigungen beitragen kann. Daraufhin werden vergleichbare Arbeiten vorgestellt, die dieses Ziel verfolgt haben. Dabei wird auch noch einmal ein Fokus auf Inklusion durch Spracherkennung innerhalb von Videospielen gelegt.

3 Anforderungsanalyse

Dieses Kapitel spezifiziert die Anforderungen, welche die zu entwickelnde System erfüllen soll. Dazu wird als erstes (Abschnitt 3.1) die zu erstellende Spielwelt beschrieben. Innerhalb dieser Beschreibung werden auch die einzelnen Rollen ermittelt, für welche die Anforderungen definiert werden. Die konkreten Anforderungen werden im Anschluss aus funktionaler Sicht (Abschnitt 3.2), sowie aus nichtfunktionaler Sicht (Abschnitt 3.3) detailliert beschrieben.

3.1 Aufbau der Spielwelt

Entwickelt werden soll ein asynchrones 2-Spieler 3D-Spiel, welches in einem Weltall-Szenario stattfindet. Ein *Defensiv-Spieler* hat das Ziel für einen Zeitraum von fünf Minuten einen Zielpunkt in Form eines Planeten vor einem *Offensiv-Spieler* zu verteidigen. Dem Defensiv-Spieler soll ein Bildschirm als Ausgabe- und eine Maus als Eingabe-Gerät zur Verfügung stehen, während der *Defensiv-Spieler* in einer VR-Umgebung mit einer VR-Brille auf dem Kopf spielt. Das Spiel soll seinen spielmechanischen Fokus auf die Aspekte Reaktion, Geschicklichkeit und strategisches Handeln ausrichten. Entsprechend soll ein Spiel entwickelt werden, welches dem Arcade/Action-Genre zugeordnet werden kann und nicht Genres wie Rollenspiele, oder Wissensspiele.

Der *Offensiv-Spieler* soll von einer Menge an Start-Punkten Asteroiden erscheinen lassen können, die sich vom gewählten Start-Punkt in konstanter Geschwindigkeit und auf direktem Weg zum Zielpunkt bewegen. Die Start-Punkte sind auf einer Sphäre verteilt, deren Mittelpunkt der Zielpunkt ist. Auf dem Bildschirm kann er die Sphäre mit der Maus drehen und so auf die verschiedenen Start-Punkte klicken um von dort die Asteroiden zu starten. Dem *Offensiv-Spieler* steht ein begrenztes Arsenal an Asteroiden zur Verfügung, welches sich über den Verlauf der Zeit wieder auffüllt.

Der Defensiv-Spieler befindet sich auf einem Raumschiff, dass sich innerhalb der Sphäre befindet. Durch den Einsatz einer VR-Technologie kann sich dieser auf dem Raumschiff umsehen und bewegen. Steuern soll dieser das Raumschiff entweder mit einem VR-Controller oder über Sprachkommandos. In beiden Fällen stehen dem Defensiv-Spieler folgende Möglichkeiten zur Verfügung:

- Der Defensiv-Spieler kann das Raumschiff auf der Hochachse und Querachse rotieren lassen.
- Der Defensiv-Spieler kann das Raumschiff positiv, wie negativ beschleunigen lassen. Es gibt eine Maximal-Geschwindigkeit.
- Der Defensiv-Spieler kann von seinem Raumschiff aus Projektile abfeuern, die sich fortan in positiver Richtung entlang der Frontachse des Raumschiffes bewegen.

Für die Eingabe mit dem Spracheingabe-Assistent steht dem Spieler exklusiv folgende Möglichkeit zur Verfügung das Raumschiff zu steuern:

- Der Defensiv-Spieler kann das Raumschiff per Kommando in die aktuelle Blickrichtung des Defensiv-Spielers rotieren lassen.

Weiterhin soll der Defensiv-Spieler die Möglichkeit haben, die Eingabe-Möglichkeiten vom VR-Controller und Spracheingabe-Assistent zu kombinieren. Hierbei stehen ihm sämtliche Eingabe-Möglichkeiten gleichzeitig zur Verfügung. Den jeweilig zu verwendenden Eingabe-Modus wählt der Defensiv-Spieler stets per Sprachkommando.

Das Ziel des Defensiv-Spielers ist es die Asteroiden davon abzuhalten den Zielpunkt zu erreichen. Dafür setzt er die Projektile ein, die einen Asteroiden beim Aufeinandertreffen zerstören. Dem Defensiv-Spieler steht ein begrenztes Arsenal an Projektilen zur Verfügung, welches sich über Zeit wieder auffüllt. Ebenfalls muss der Defensiv-Spieler verhindern, dass die Asteroiden mit dem Raumschiff kollidieren. Sollte dies passieren, verliert der Defensiv-Spieler das Spiel. Erreichen drei Asteroiden den Zielpunkt, verliert der Defensiv-Spieler ebenfalls. Schafft er es über einen Zeitraum von fünf Minuten nicht zu verlieren, gewinnt der Defensiv-Spieler.

Der Defensiv-Spieler hat eine weitere Möglichkeit das Spielgeschehen aktiv zu beeinflussen. Die Start-Punkte, von welchen der Offensiv-Spieler seine Asteroiden starten lässt, sind auch für den Defensiv-Spieler sichtbar und können auf dem gleichen Wege wie die Asteroiden mit Projektilen zerstört werden. Trifft ein Projektil einen Start-Punkt, so wird

dieser aus dem Spiel entfernt und steht dem Offensiv-Spieler von nun an nicht mehr zur Verfügung um Asteroiden erscheinen zu lassen. So kann der Defensiv-Spieler strategisch dafür sorgen, dass der Offensiv-Spieler nur noch aus bestimmten Richtungen Asteroiden starten lassen kann. Rein theoretisch kann der Defensiv-Spieler das Spiel somit auch vor Ablauf der fünf Minuten zu seinen Gunsten entscheiden, sollte er es schaffen alle Start-Punkte aus dem Spiel zu entfernen. Der Offensiv-Spieler kann dies wiederum durch strategisches Ablenken mit Hilfe von Asteroiden zu unterbinden versuchen.

Dem Defensiv-Spieler steht zur Übersicht über das aktuelle Spielgeschehen eine 3D-Karte innerhalb des Raumschiffs zur Verfügung. Hier sieht er die Position und Ausrichtung seines Raumschiffs innerhalb der Spielwelt, sowie alle zur Zeit existierenden Start-Punkte und Asteroiden.

3.1.1 Rollen

Dieser Beschreibung der Spielwelt lassen sich folgende Rollen entnehmen, für welche Anforderungen definiert werden müssen. Dies sind:

User Rollen:

- Spieler 1 - *Defensiv-Spieler*
- Spieler 2 - *Offensiv-Spieler*

Technische Rollen:

- *System*

3.2 Funktionale Anforderungen

3.2.1 Defensiv-Spieler

Im folgendem Unterkapitel werden die Anforderungen beschrieben, die im unmittelbaren Zusammenhang zur Rolle „Defensiv-Spieler“ stehen. Diese sind

FA1 - Ausgabe der Spielwelt

Die Ausgabe der Spielwelt soll für den Defensiv-Spieler über eine VR-Brille erfolgen. Mit der VR-Brille ist der Defensiv-Spieler in der Lage sich ohne Einschränkungen im 3D-Raum auf dem Raumschiff umzusehen. Die für den Defensiv-Spieler wesentlichen Informationen werden mittels eines sogenannten „Diegetic Interface“[13] in der Spielwelt abgebildet. Das bedeutet, dass die Informationen innerhalb der Spielwelt angezeigt werden und nicht fest im Sichtfeld des Anwenders verankert sind. So befindet sich im Raumschiff ein Display in Form eines Monitors, welcher folgende Informationen anzeigt:

- *Lebenspunkte des Zielpunkts*
- *Aktuell verfügbare Projektile*
- *Aktuelle Geschwindigkeit des Raumschiffs*

Zusätzlich findet der Defensiv-Spieler im Raumschiff eine Minimap, in Form eines transparenten Miniatur-Modells der Spielwelt, vor. Sämtliche zur Zeit existierenden Start-Punkte, sowie alle aktuell im Spiel befindlichen Asteroiden sind entsprechend ihrer Position abgebildet. Auch das Raumschiff ist entsprechend aktueller Position und Rotation abgebildet. Diese dreidimensionale Karte hilft dem Defensiv-Spieler unabhängig vom Sichtfeld, welches durch die Fenster im Raumschiff festgelegt ist, akute Bedrohungen auszumachen und entsprechend strategisch zu handeln.

Weiterhin sind auf einem zweiten Monitor im Raumschiff alle ausführbaren Sprachkommandos aufgeführt. Diese entsprechen den Kommandos in der nachfolgenden Anforderung (Abschnitt 3.2). Somit muss sich der Defensiv-Spieler nicht alle zur Verfügung stehenden Sprachkommandos merken, sondern kann sie jederzeit nachlesen.

FA2 - Eingabe per Sprachkommandos

Dem Defensiv-Spieler stehen eine Reihe an Sprachkommandos zur Verfügung, mit Hilfe derer er das Raumschiff steuern kann. Diese stehen ihm lediglich im Eingabe-Modus *Voice* und *Complete* zur Verfügung, mit der Ausnahme der Sprachkommandos um den Eingabemodus zu wechseln. Folgende Sprachkommandos dienen dazu den Eingabemodus zu wechseln und stehen in jedem Eingabemodus zur Verfügung:

- ***Complete*** *- wechselt in den Eingabemodus Complete (Sprach- und Controller-Steuerung*
- ***Voice*** *- wechselt in den Eingabemodus Voice (Sprachsteuerung)*
- ***Manual*** *- wechselt in den Eingabemodus Manual (Controller-Steuerung)*

Im Eingabemodus Voice sowie im Eingabemodus Complete stehen dem Defensiv-Spieler folgende Sprachkommandos zur Verfügung:

- ***Stop*** *- Setzt die Geschwindigkeit und die aktuelle Rotationsbewegung des Raumschiffs auf 0*
- ***Speed [1..99]*** *- Setzt die Bewegungs-Geschwindigkeit des Raumschiffs auf den entsprechenden Wert*
- ***Turn [right/left/up/down]*** *- Startet die schnelle Rotation des Raumschiffs in die entsprechende Richtung. Diese dient der zügigen Rotation des Raumschiffs um generelle Richtungswechsel vorzunehmen.*
- ***Slow [right/left/up/down]*** *- Startet die langsame Rotation des Raumschiffs in die entsprechende Richtung. Diese dient dem genauen Zielen und der filigranen Ausrichtung des Raumschiffs.*
- ***Look Me*** *- Rotiert das Raumschiff in Blickrichtung des Defensiv-Spielers*
- ***Release*** *- Startet ein Projektil*

FA3 - Eingabe per VR-Controller

In den Eingabe-Modi Manual und Complete kann der Defensiv-Spieler sein Raumschiff mit dem VR-Controller steuern. Für jedes Sprachkommando steht dem Defensiv-Spieler hier eine entsprechende Möglichkeit zur Verfügung, die jeweilige Eingabe ebenfalls auf manuelle Weise zu tätigen. Die Bewegungs-Geschwindigkeit, sowie die Rotations-Richtung und Rotations-Geschwindigkeit sollen mittels kontinuierlich zu betätigenden Mechaniken des VR-Controllers gesteuert werden. Für die Ausgangsstellung eben dieser Mechaniken ist die jeweilige Geschwindigkeit, oder Rotation auf 0 festgelegt. Befehle, wie *Stop*, oder *Release* werden über Betätigungen diskreter Eingabe-Mechaniken des VR-Controllers umgesetzt.

3.2.2 Offensiv-Spieler

Im folgenden werden die Anforderungen beschrieben, die sich speziell auf die Rolle *Offensiv-Spieler* beziehen.

FA4 - Ausgabe der Spielwelt

Der Offensiv-Spieler soll die Spielwelt auf einem Bildschirm präsentiert bekommen. Dort sieht er alle zur Verfügung stehenden Startpunkte, das Raumschiff des Defensiv-Spielers, sowie den Zielpunkt. Weiterhin werden am Rand des Bildschirms folgende Informationen präsentiert:

- *Lebenspunkte des Zielplaneten*
- *Vergangene Zeit seit Spielbeginn*
- *Aktuell verfügbare Asteroiden*

FA5 - Eingabe per Maus

Der *Offensiv-Spieler* soll das System ausschließlich per Maus steuern. Durch gedrückt halten der linken Maustaste sowie gleichzeitigem Bewegen der Maus soll der Spieler die Kamera um die Spielwelt rotieren lassen, so dass er einen guten Überblick über das Spielgeschehen erhält und auf alle Start-Punkte zugreifen kann. Klickt der Spieler mit der linken Maustaste auf einen Start-Punkt und steht ihm mindestens ein Asteroid zur Verfügung, so soll am entsprechenden Start-Punkt ein Asteroid generiert werden, welcher sich fortan Richtung Zielpunkt bewegt.

3.2.3 System

Dieses Unterkapitel beschreibt die funktionalen Anforderungen die an die Rolle *System* gestellt werden.

FA6 - Aktualisierung der Spielwelt

Die Spielwelt wird 60 mal pro Sekunde aktualisiert. Dabei werden folgende Komponenten der Spielwelt neu berechnet und aktualisiert:

- *Die Position des Raumschiffs auf Basis der aktuellen Geschwindigkeit und Rotation*
- *Die Rotation des Raumschiffs auf Basis der aktuellen Rotations-Eingabe*
- *Die Position sämtlicher im Spiel befindlichen Asteroiden und Projektile auf Basis der jeweiligen Geschwindigkeit*
- *Vorhandene Kollision zwischen Spielobjekten*
- *Die seit Spielbeginn vergangene Zeit*
- *Die dem Offensiv-Spieler zur Verfügung stehenden Asteroiden*
- *Die dem Defensiv-Spieler zur Verfügung stehenden Projektile*

Sämtliche UI-Elemente werden ebenfalls entsprechend ihrer unter Umständen neuen Werte 60 mal pro Sekunde aktualisiert.

FA7 - Regeln der Spielwelt

Für das zu entwickelnde Spiel gelten folgende Regeln, welche vom System entsprechend implementiert werden müssen.

- *Wenn innerhalb der Spielwelt ein Projektil mit einem Asteroiden kollidiert, werden beide Objekte zerstört.*
- *Wenn innerhalb der Spielwelt ein Projektil mit einem Start-Punkt kollidiert, werden beide Objekte zerstört.*
- *Kollidiert das Raumschiff mit einem Asteroiden wird das Spiel beendet und der Defensiv-Spieler hat verloren.*
- *Verlässt ein Projektil die Sphäre, auf welcher sich die Start-Punkte befinden, dann wird das Projektil aus der Spielwelt entfernt. Dies dient dazu die Spielwelt von Spielobjekten zu bereinigen, die von nun an keinen Einfluss auf das Spielgeschehen mehr ausüben können.*

- *Trifft ein Asteroid auf den Zielpunkt, wird der Asteroid entfernt und die Anzahl der Lebenspunkte des Zielpunkts um einen verringert.*
- *Fallen die Lebenspunkte des Zielpunkts auf 0, dann wird das Spiel beendet. Der Offensiv-Spieler hat das Spiel gewonnen.*
- *Sind seit Spielbeginn mehr als 5 Minuten vergangen und der Defensiv-Spieler hat das Spiel seitdem nicht verloren, dann wird das Spiel beendet. Der Defensiv-Spieler hat das Spiel gewonnen.*

3.3 Nichtfunktionale Anforderungen

In diesem Kapitel werden sämtliche nichtfunktionalen Anforderungen an das zu entwickelnde System beschrieben.

NFA1 - Qualität des Systems als Videospiel

Das System soll dem Qualitätsstandard von einem zeitgemäßen Videospiel genügen. Es ist wichtig, dass die Anwender nicht einen Mangel an Qualität als störenden Faktor wahrnehmen, im Bezug auf die Bewertung der jeweiligen Eingabe-Methoden. Dazu gehört eine ansprechende Aufmachung in Hinsicht der grafischen Elemente, sowie ein dem Szenario entsprechendes Audio-Feedback mittels passender Sound-Effekte. Ebenfalls elementar ist die generelle Latenz der Spielwelt. Eingaben beider Spieler müssen in „gefühlter Echtzeit“ und unmittelbar verarbeitet werden, damit hier keine Latenz vom Anwender wahrgenommen wird.

NFA2 - Latenz der Verarbeitung der Sprachkommandos

Für eine möglichst anwenderfreundliche Spracheingabe ist es maßgeblich wichtig, dass die Latenz zwischen dem Aussprechen des Sprachkommandos und dem Ergebnis innerhalb des Systems so gering wie möglich gehalten werden sollte. Im Optimalfall sollte keine Latenz wahrgenommen werden. Daher ist an dieser Stelle ein besonderes Optimierungspotential vorhanden.

NFA3 - Korrektes Erkennen der Sprachkommandos

Für eine funktionierende Steuerung per Sprachkommando ist es elementar, dass das System die ausgesprochenen Sprachkommandos zu einer sehr hohen Wahrscheinlichkeit korrekt erkennt. Daher ist es wichtig, verschiedene Aussprachen der zu erwartenden Sprachkommandos zu berücksichtigen. Zum Beispiel kann ein Sprachkommando *right* von vielen gängigen Spracherkennungssystemen auch als *light* missverstanden werden. Solche zu erwartenden Missverständnisse müssen innerhalb des Systems berücksichtigt und entsprechend dennoch korrekt verarbeitet werden. Die Zielsetzung soll es sein, das zu verwendende Spracherkennungs-System so zu konfigurieren, dass eine nach Möglichkeit perfekte Erkennung der eingesprochenen Sprachkommandos stattfindet.

3.4 Zusammenfassung

Dieses Kapitel hat die funktionalen und nichtfunktionalen Anforderungen an das System beschrieben, welche im nächsten Kapitel (Entwurf und Umsetzung) umgesetzt werden.

Zunächst wurde die Spielwelt und der Spielablauf umfassend beschrieben (Abschnitt 3.1) um ein konkretes Verständnis für das herzustellende System zu entwickeln. Dabei wurden die wesentlichen Rollen herausgestellt und entsprechend benannt und definiert (Unterabschnitt 3.1.1).

Im Anschluss wurden sämtliche funktionalen Anforderungen auf die entsprechenden Rollen verteilt und ausführlich beschrieben (Abschnitt 3.2). Unabhängig von den Rollen wurden dann zum Ende die nichtfunktionalen Anforderungen an das zu entwickelnde System festgelegt und beschrieben (Abschnitt 3.3).

4 Entwurf und Umsetzung

In diesem Kapitel wird der Software-Entwurf für die zu entwickelnde Anwendung beschrieben. Anhand der im Kapitel 2 aufgeführten funktionalen und nicht funktionalen Anforderungen werden im Unterkapitel Software-Entwurf (Abschnitt 4.1) zunächst die benötigten Teilsysteme, sowie die zugehörigen Schnittstellen herausgearbeitet und beschrieben. Basierend auf dieser System-Übersicht wird eine Software-Architektur beschrieben, die unabhängig von der konkreten Implementierungsumgebung ist.

Im Anschluss werden im Abschnitt 4.2 die Entscheidungen für die zur Realisierung verwendete Soft- und Hardware getroffen und begründet.

Im Unterkapitel Umsetzung (Abschnitt 4.3) folgt die Beschreibung der konkreten Umsetzung des Entwurfs, auf Basis der gewählten Soft- und Hardware. Dabei wird analysiert, in welchem Maße die einzelnen Anforderungen aus Kapitel 2 erfolgreich umgesetzt werden konnten.

4.1 Software-Entwurf

Abbildung 4.1 zeigt den abstrakten physischen Aufbau des zu entwickelnden Spiels. Man sieht den Offensiv-Spieler, welcher das Spielgeschehen an einem Monitor verfolgt und mit einer Computer-Maus Eingaben tätigen kann. Weiterhin wird der Defensiv-Spieler gezeigt, welcher das Spiel über ein VR-Headset ausgegeben bekommt und mit einem Sprach-Assistenten und einem Controller seine Eingaben tätigt. Die Software-Anwendung selber läuft auf einem lokalen Desktop-Computer, an welchem sämtliche Ein- und Ausgabe Gerätschaften angeschlossen sind.

Aus dieser Übersicht leiten sich folgende Teilsysteme ab, aus denen sich das zu entwickelnde Spiel zusammensetzt:

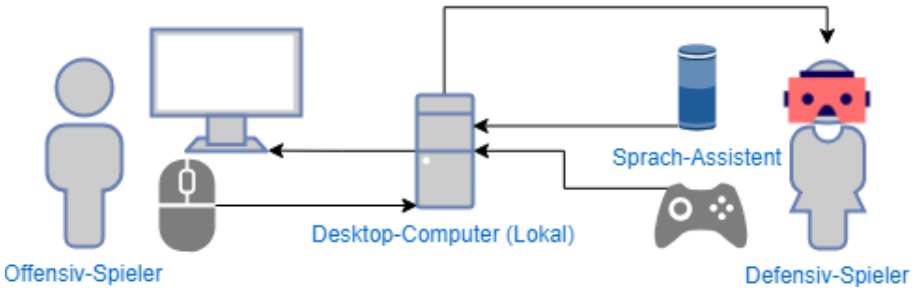

Abbildung 4.1: Verteilung der Systemkomponenten im Gesamtsystem. (Quelle: Eigene Arbeit)

Das für diese Arbeit elementare Teilsystem ist der Sprach-Assistent, an welchen der Defensiv-Spieler seine Spracheingabe richtet und welches daraufhin die entsprechenden Sprachkommandos an die Software-Anwendung überträgt. Dieses Teilsystem wird von nun an als ***CSR-System*** bezeichnet, da es auf kommandobasierter Spracherkennung basiert. Als nächstes Teilsystem lässt sich das ***VR-System*** identifizieren. Dieses System umfasst sämtliche Hardware die notwendig ist um die VR-Erfahrung für den Defensiv-Spieler bereitzustellen. Da VR-Headsets in der Regel mit individuellen Game-Controllern zusammenarbeiten, wird der Game-Controller ebenfalls zum *VR-System* gezählt. Ein weiteres Teilsystem ist die ***Software-Anwendung***. Diese wird auf dem lokalen Desktop-Computer ausgeführt, verarbeitet Eingaben der anderen Systeme auf Basis der implementierten Spiellogik, aktualisiert die Spielwelt und rendert die Ausgabe für beide Spieler. Das letzte Teilsystem ist das **Offensiv-Interface**. Dazu gehört der Monitor an welchem der *Offensiv-Spieler* seine Ausgabe der Spielwelt betrachtet. Weiterhin zählt dazu die Computer-Maus, welche als Eingabegerät für den *Offensiv-Spieler* fungiert.

Abbildung 4.2 zeigt eine Struktursicht der beschriebenen Teilsysteme, sowie die benötigten Schnittstellen um die Teilsysteme miteinander kommunizieren zu lassen.

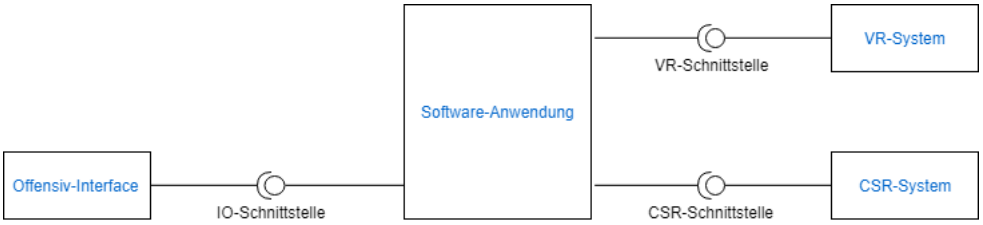

Abbildung 4.2: Übersicht der Teilsysteme und der benötigten Schnittstellen. (Quelle: Eigene Arbeit)

Für das Teilsystem *Software-Anwendung* wird nun eine Software-Architektur beschrieben. Die Schnittstellen der Teilsysteme *VR-System* und *CSR-System* sind maßgeblich von den zur Realisierung getroffenen Soft- und Hardware-Entscheidungen abhängig und sind unter Umständen schon vorhanden. Daher wird auf diese Schnittstellen an dieser Stelle nicht eingegangen.

Die hier präsentiere Software-Architektur dient dazu, sie mit einer beliebigen 3D-Engine in eine konkrete Software überführen zu können. Da der Entwurf in einer 3D-Engine umgesetzt werden soll wird davon ausgegangen, dass eine entsprechende Render-Pipeline bereits vorhanden ist und nicht implementiert werden muss. Abbildung 4.3 zeigt die Architektur des Teilsystems *Software-Anwendung* anhand eines Komponentendiagramm. Dabei steht eine Komponente nicht sinnbildlich für eine zu implementierende Klasse, sondern für eine logische Einheit innerhalb der Anwendung. Verschiedene 3D-Engines basieren auf unterschiedlichen Architekturen, die individuelle Implementierungen zulassen.

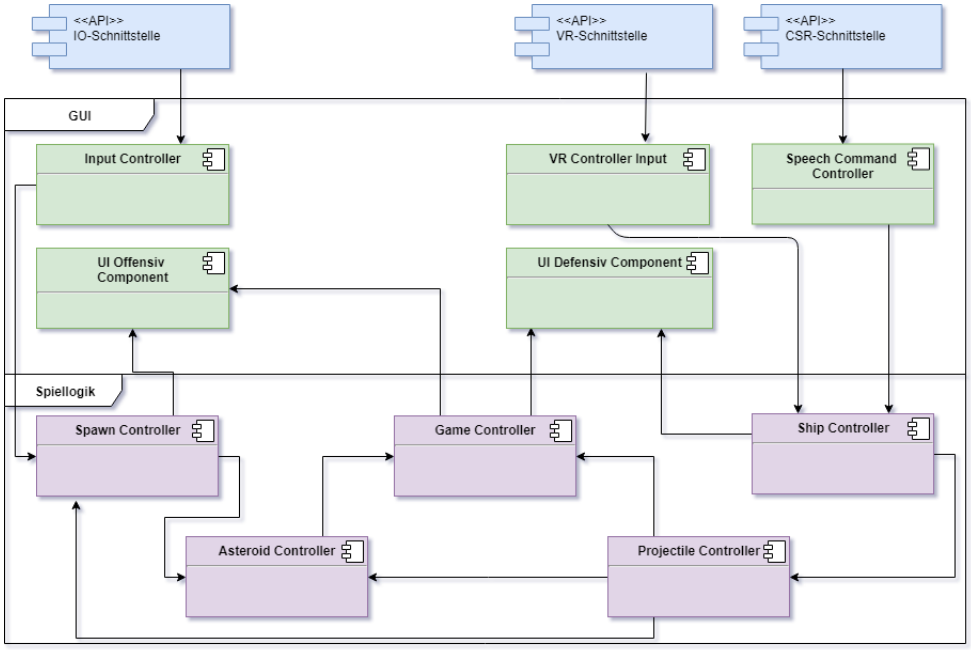

Abbildung 4.3: Software-Architektur in Form eines Komponenten Diagramms. (Quelle: Eigene Arbeit)

Diese Architektur ist in zwei Schichten unterteilt. Die *GUI*-Schicht enthält alle Software-Komponenten, die sich dem User-Interface zuordnen lassen. Entsprechend sind diese Komponenten dafür vorgesehen Eingaben zu verarbeiten, oder visuelle Elemente des User-Interfaces zu aktualisieren. In der *Spiellogik*-Schicht finden sich die Komponenten, die die eigentliche Spiellogik umsetzen. Gleichzeitig werden auf dieser Schicht die Spielregeln implementiert. Über den beiden Schichten finden wir die Schnittstellen, die von der Software benötigt werden um mit den anderen Teilsystemen zu kommunizieren.

Im Folgenden werden die einzelnen Komponenten dieses Architektur-Entwurfs beschrieben.

4.1.1 UI Controller

Der UI Controller empfängt die Eingaben, die der *Offensiv-Spieler* mit der Maus tätigt. Klickt dieser auf einen Spawn-Punkt wird die Eingabe an die Spawn-Controller Komponente übergeben und entsprechend verarbeitet. Ebenfalls ist diese Komponente für die Rotation der Kamera des *Offensiv-Spielers* zuständig.

4.1.2 UI Offensiv Component

Diese Komponente enthält alle UI-Elemente, die dem Offensiv-Spieler seine benötigten Information anzeigen und ist dafür zuständig, diese auf Befehl des *Game Controllers* zu aktualisieren.

4.1.3 VR Controller Input

Diese Komponente empfängt die Eingaben des VR-Controllers. Entsprechend der Controller-Belegung interpretiert diese Komponente den Zweck dieser Eingaben und leitet sie an den *Ship Controller* weiter.

4.1.4 Speech Command Controller

Der *Speech Command Controller* ist dafür zuständig, die erkannten Sprachkommandos der CSR-Schnittstelle zu empfangen. Entsprechend ihrer hier konfigurierten Bedeutung werden diese interpretiert und an den *Ship Controller* weitergeleitet.

4.1.5 UI Defensiv Component

Diese Komponente enthält alle UI-Elemente, die dem Defensiv-Spieler seine benötigten Information anzeigen und ist dafür zuständig, diese auf Befehl des *Game Controllers* zu aktualisieren.

4.1.6 Spawn Controller

Der *Spawn Controller* ist dafür zuständig Asteroiden in die Spielwelt zu setzen. Gleichzeitig hält er den aktuellen Vorrat an Asteroiden und aktualisiert diesen entsprechend der vergangenen Zeit. Weiterhin hält er eine Liste aller aktuell im Spiel befindlichen Spawn-Punkte und kann individuelle Spawn-Punkte entfernen, sollte der *Projectile Controller* dies befehlen.

4.1.7 Asteroid Controller

Diese Komponente aktualisiert die Position der Asteroiden. Wenn ein Asteroid den Zielpunkt erreicht, entfernt der *Asteroid Controller* den entsprechenden Asteroiden aus der Spielwelt und meldet dies dem *Game Controller*.

4.1.8 Projectile Controller

Im *Projectile Controller* wird die Position aller zur Zeit im Spiel befindlichen Projektile aktualisiert. Gleichzeitig wird beobachtet ob ein Projektil mit einem im Spiel befindlichen Asteroiden, oder Spawn-Punkt kollidiert. Dafür hat er Zugriff auf die Komponenten *Asteroid Controller* und *Spawn Controller*. Sollte eine Kollision stattfinden, meldet der *Projectile Controller* dies der entsprechenden Komponente und lässt diese aus dem Spiel entfernen.

4.1.9 Ship Controller

Diese Komponente widmet sich der Steuerung des Raumschiffs. Sie hält den aktuellen Eingabe-Modus und reagiert entsprechend auf Eingaben der jeweiligen Input-Komponenten *Speech Command Controller* und *VR Controller Input*. Je nach Eingabe aktualisiert diese

Komponente die Position und Rotation des Raumschiffs. Weiterhin hält sie den aktuellen Vorrat an Projektilen und aktualisiert diesen entsprechend.

4.1.10 Game Controller

Der *Game Controller* implementiert die grundsätzlichen Sieg-Bedingungen für beide Spieler. Er aktualisiert die seit Spielbeginn vergangene Zeit und die verbleibenden Lebenspunkte des Zielpunkts. Diese Informationen gibt die Komponente an die UI-Komponenten weiter, damit diese ihre UI-Elemente entsprechend aktualisieren. Weiterhin beendet der Game Controller das Spiel, wenn eine der Sieg-Bedingungen eingetreten ist.

4.2 Hard- und Software Entscheidungen

In diesem Kapitel wird beschrieben welche Hard- und Software für die Teilsysteme *Software-Anwendung*, *VR-System* und *CSR-System* gewählt wurde. Weiterhin wird eine verworfene Umsetzung des *CSR-Systems* beschrieben, welche vom Autoren als ungenügend bewertet wurde.

4.2.1 Software-Anwendung

Die Anwendung selbst wird in der 3D-Engine „Unity3D“[1] entwickelt. „Unity3D“ bietet mit dem „SteamVR“-Plugin eine konkrete Implementierung der „OpenVR“-API an. Darüber kann die Software-Anwendung mit einem *VR-System* kommunizieren und zur Anwendungsentwicklung an dieser Stelle eingesetzt werden. Als weitere Begründung zur Wahl von „Unity3D“ ist zu erwähnen, dass der Autor über Vorerfahrung in der Entwicklung mit dieser 3D-Engine verfügt.

4.2.2 VR-System

Zur Umsetzung des Teilsystems *VR-System* kommt die von HTC entwickelte „HTC Vive“[2] zum Einsatz. Diese besteht aus einem VR-Headset(„Vive Headset“), zwei „Vive Base

[1] https://unity3d.com/de/unity
[2] https://www.vive.com/de/product/vive-pro-full-kit

Stations“ und zwei „Vive Controllers“. Das VR-Headset wird per Kabel an eine „Link-Box“ angeschlossen, die wiederum mit dem lokalen Desktop-Computer verbunden ist. Die „Vive Base Stations“ und die „Vive Controllers“ sind kabellos mit der „Link-Box“ verbunden.

Die „HTC Vive“ kommuniziert mittels „OpenVR“-API mit der jeweiligen VR-Anwendung, welche uns wie schon erwähnt durch „Unity3D“ zur Verfügung gestellt wird. Diese Tatsache und die Verfügbarkeit der Hardware für den Autor, sind der maßgebliche Grund, dass diese VR-Hardware zum Einsatz kommt.

Die „Vive Base Stations“ des Systems stecken einen Raum ab, in welchem sich der Anwender physisch bewegen kann. Theoretisch ist die Möglichkeit vorhanden, diese Bewegung auch in der virtuellen Umgebung abzubilden. Da dies für die zu entwickelnde Anwendung aber kein benötigtes Feature ist, wird darauf verzichtet. Abbildung 4.4 demonstriert den Aufbau des VR-Systems anschaulich.

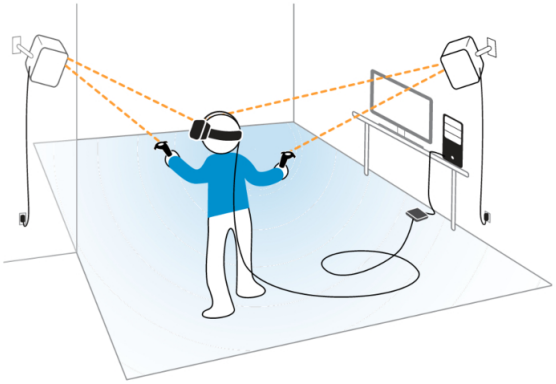

Abbildung 4.4: Aufbau einer HTC Vive (Quelle: https://springschool-thueringen.de/programm-2017/workshop-immersive-medien-2)

4.2.3 CSR-System

Für die Erkennung und Verarbeitung der Sprachkommandos, kommt Microsofts Software-Lösung „Cortana“[3] zum Einsatz. Diese setzt das Betriebssystem „Windows 10“ voraus, welches entsprechend auf dem lokalen Desktop-Computer als Plattform dienen muss. Als Mikrofon wird das im „Vive Headset“ enthaltene Mikrofon eingesetzt. Die verwendete 3D-Engine „Unity3D“ liefert in der aktuellen Version (Unity 2018) eine Cortana-Schnittstelle mit, so dass diese aus der Anwendung heraus ohne Umwege verwendet werden kann.

In einer früheren Version dieses Entwurfs, wurde auch Amazons „Amazon Echo“ als CSR-System getestet. Die aus dem Test hervorgegangenen Eingabe-Latenzen, waren aber im Vergleich zu Cortana zu hoch um einen Einsatz zu rechtfertigen.

4.3 Umsetzung

In diesem Kapitel wird die Umsetzung des Software-Entwurfs erläutert. Dabei wird die erfolgreiche oder nicht erfolgreiche Umsetzung ausgewählter funktionaler Anforderungen aus dem Kapitel Kapitel 2 untersucht. Die Umsetzung der Rollen *Offensiv-Spieler* und *System* wird dabei nicht ausführlich erläutert, da diese für den wissenschaftlichen Aspekt und die anschließende Evaluierung dieser Arbeit nicht sonderlich relevant ist. Es lässt sich aber sagen, dass die Überführung des Entwurfs in eine konkrete Software-Anwendung auch für die Rollen *Offensiv-Spieler* und *System* erfolgreich funktioniert hat, was sich auch dem abschließenden Unterabschnitt 4.3.4 entnehmen lässt.

4.3.1 Umsetzung FA1 - Ausgabe der Spielwelt

Die Ausgabe der Spielwelt wird dem *Defensiv-Spieler* über das „Vive Headset“ geboten. Dabei steht dieser in einem durch die „Vive Base Stations abgegrenztem Bereich und hält einen „Vive Controller“ in der rechten Hand. Nun kann sich der *Defensiv-Spieler* innerhalb der VR-Umgebung auf dem Raumschiff umschauen und durch die Fenster die Spielwelt außerhalb des Raumschiffs beobachten (Abbildung 4.5).

[3] https://www.microsoft.com/de-de/windows/cortana

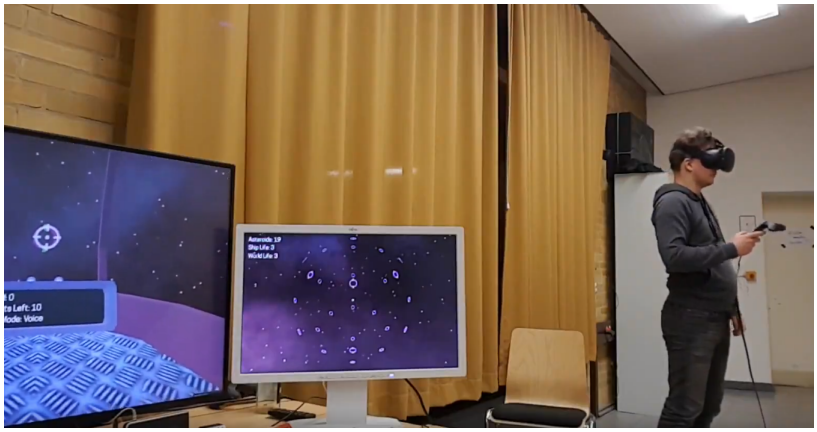

Abbildung 4.5: Autor der Arbeit als Defensiv-Spieler - Früher Prototyp (Quelle: Eigene Arbeit)

Das geforderte „Diegetic Interface“ befindet sich im unteren, vorderen Teil des Raumschiffs in Form eines Bildschirms und liefert dem *Defensiv-Spieler* die geforderten Spiel-Informationen. Im oberen vorderen Teil des Raumschiffs befindet sich ein weiterer Bildschirm, welcher dem Spieler die verfügbaren Sprachbefehle aufzeigt (Abbildung 4.6).

Abbildung 4.6: Raumschiff mit Diegetic Interface (Quelle: Eigene Arbeit)

Die im hinteren Teil des Raumschiffs befindliche Minimap zeigt dem *Defensiv-Spieler* eine dreidimensionale Übersicht der Spielwelt. Dadurch kann der Spieler potentielle Gefahren erkennen, die ihm der aktuelle Bildausschnitt, welcher ihm durch die Fenster des Raumschiffs gewährt wird, nicht zeigen kann. (Abbildung 4.7)

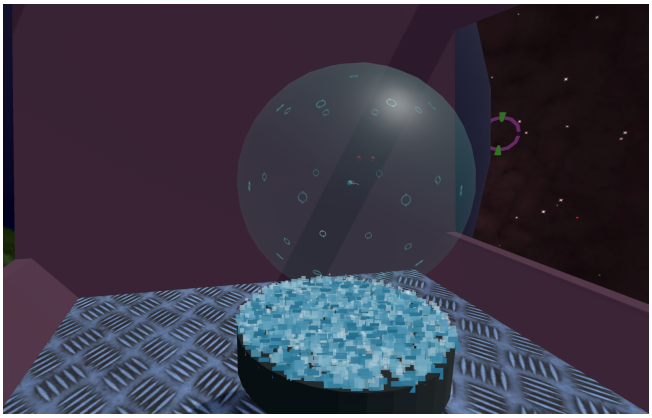

Abbildung 4.7: Minimap auf dem Raumschiff (Quelle: Eigene Arbeit)

4.3.2 Umsetzung FA2 - Eingabe per Sprachkommandos

Der *Defensiv-Spieler* spricht seine Sprachkommandos in das Mikrofon des „Vive Headsets“. Diese Befehle werden an die Software-Anwendung übertragen und dort über die von Unity zur Verfügung gestellte „Cortana“-API analysiert. Werden von „Cortana“ Befehle erkannt, die entsprechend der Anforderung in einem Wörterbuch hinterlegt sind, führt das Raumschiff die entsprechende Operation aus.

Die in Unity3D zur Verfügung gestellten Bibliotheken zur Cortana-Spracherkennung erlauben verschiedene Arten der Spracherkennung. So lassen sich mittels der Klasse „GrammarRecognizer“ auch Grammatiken implementieren, mit denen man eine SVR-Spracherkennung realisieren kann. Da eine CSR-Spracherkennung angefordert ist, wurde die Klasse „KeywordRecognizer“ verwendet. Von dieser Klasse wird ein Objekt instanziiert und dieses anschließend mit allen zu erwartenden „KeyWords“ in Form von Strings versorgt. Sobald eines dieser „KeyWords“ von Cortana erkannt wird, wird eine entsprechende Methode aufgerufen die dann denn indidivudellen Sprachbefehl weiterverarbeitet.

Dabei hat sich herausgestellt dass es in speziellen Fällen hilfreich sein kann im Wörterbuch nicht nur den kompletten Sprachbefehl zu hinterlegen. Falls die erste Silbe eines Sprachkommandos nicht gleichzeitig die erste Silbe eines anderen Sprachkommandos ist, empfiehlt es sich zusätzlich nur diese Silbe im Wörterbuch zu hinterlegen und mit der entsprechenden Logik zu verknüpfen. Dadurch kann „Cortana“ schneller eine Übereinstimmung feststellen, weil es den gesprochenen Befehl nicht bis zur letzten Silbe kontrol-

lieren muss. Diese Erkenntnis wurde in der Umsetzung dieser Anforderung berücksichtigt und für die Sprachkommandos „Release“ und „Look Me“ umgesetzt.

Sämtliche in der Anforderung spezifizierten Sprachkommandos wurden während der Umsetzung der Sprachsteuerung implementiert und funktionieren wie im Kapitel Kapitel 2 beschrieben.

4.3.3 Umsetzung FA3 - Eingabe per VR-Controller

Als VR-Controller kommt der zur „HTC Vive“ zugehörige „Vive Controller“ zum Einsatz (Abbildung 4.8). Folgende Tastenbelegung ist dabei konfiguriert, um dem *Defensiv-Spieler* alle Eingabe-Möglichkeiten zur Verfügung zu stellen.

- **Trackpad** - Raumschiff rotieren lassen (Querachse und Hochachse)
- **Trigger** - Beschleunigen [0..99]
- **Grip Button** - Projektil erzeugen

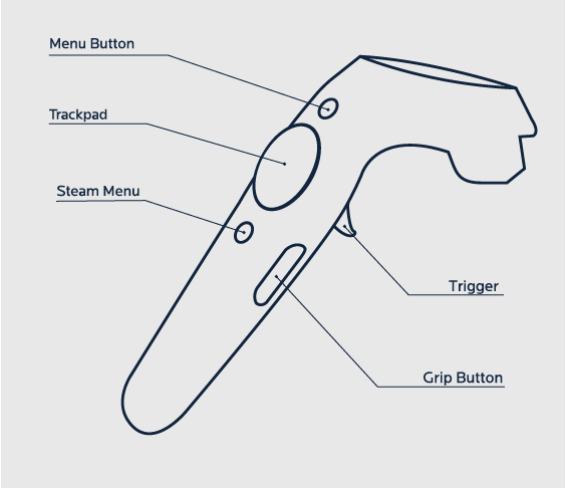

Abbildung 4.8: „Vive Controller“ (Quelle: https://survios.com/rawdata/guide/)

Entgegen der Anforderung, dass der Befehl „Stop“ über eine diskrete Betätigung einer Eingabe-Mechanik erfolgt, ist die Nichtbetätigung vom „Trigger“ gleichbedeutend mit dem Stillstand des Raumschiffs, beziehungsweise mit dem Geschwindigkeitswert 0. Die Rotation des Raumschiffs ist aber auch bei Geschwindigkeit 0 möglich.

4.3.4 Umfang der Umsetzung

Die Anforderungsanalyse konnte in großen Teilen erfolgreich und vollständig umgesetzt werden. Im Bereich der nichtfunktionalen Anforderungen mussten aber einige Einschränkungen gemacht werden. Die Anforderung „NFA1 - Qualität des Systems als Videospiel“ konnte nur zum Teil erfolgreich umgesetzt werden. Das Echtzeitgefühl der Spielwelt wurde zweifelsfrei erfolgreich umgesetzt, aber die audiovisuellen Komponenten haben nicht ganz die Qualität eines zeitgemäßen Videospiels erreicht. Dies ist begründet durch Zeitmangel und zum Teil auch technischen Schwierigkeiten.

Weiterhin wurde die Latenz beim Verarbeiten der Sprachkommandos nicht in dem Maße minimiert worden, wie vom Autor ursprünglich gewünscht. Zwar wurde eine durchaus erträgliche Latenz erreicht, aber sie ist zweifelsohne als solche spürbar.

Abbildung 4.9 zeigt eine Übersicht der Anforderungen aus Kapitel Kapitel 2 und den Grad der vollständigen Umsetzung eben dieser.

	Grad der Umsetzung		
Anforderung Funktional	**Vollständig**	**Teilweise**	**Nicht vorhanden**
FA1 - Ausgabe der Spielwelt	x		
FA2 - Eingabe per Sprachkommandos	x		
FA3 - Eingabe per VR-Controller	x		
FA4 - Ausgabe der Spielwelt	x		
FA5 - Eingabe per Maus	x		
FA6 - Aktualisierung der Spielwelt	x		
FA7 - Regeln der Spielwelt	x		
Anforderung Nichtfunktional			
NFA1 - Qualität des Systems als Videospiel		x	
NFA2 - Latenz der Verarbeitung der Sprachkommandos		x	
NFA3 - Korrektes Erkennen der Sprachkommandos		x	

Abbildung 4.9: Grad der Umsetzung (Quelle: Eigene Arbeit)

4.4 Zusammenfassung

Der Software-Entwurf des VR-Spiels wurde in diesem Kapitel ausführlich erörtert. Im Unterkapitel Software-Entwurf (Abschnitt 4.1) wurde sich schrittweise dem schlussendlichem Entwurf genähert. Zuerst wurde der abstrakte physische Aufbau der zu entwi-

ckelnden Anwendung erörtert, aus welchem sich im Anschluss die Teilsysteme ableiten ließen. Auf der Basis dieser Teilsysteme wurde mit Hilfe eines Komponentendiagramms eine Architektur entworfen, die für die darauffolgende Umsetzung als Software-Entwurf dienlich war.

Das anschließende Unterkapitel Hard- und Software-Entscheidungen (Abschnitt 4.2) erörtert, welche Hard- und Software Produkte für die Umsetzung ausgewählt wurden und begründet diese Entscheidungen.

Abschließend wurde im Unterkapitel Umsetzung (Abschnitt 4.3) die Umsetzung ausgewählter funktionaler Anforderungen beschrieben. Weiterhin wurde eine Übersicht über den jeweiligen Grad der Umsetzung im Bezug auf sämtliche Anforderungen geliefert.

5 Evaluierung

Dieses Kapitel erörtert die Ergebnisse einer Evaluierung der im vorherigen Kapitel umgesetzten Anwendung. Dabei haben 15 Testpersonen aus dem Bekanntenkreis des Autors in der Rolle des *Defensiv-Spielers* für zwei Runden gegen den Autoren als *Offensiv-Spieler* das VR-Spiel gespielt. Dabei wurden sie aufgefordert den Eingabe-Modus regelmäßig zu wechseln. Im Anschluss haben die Testpersonen einen Fragebogen ausgefüllt, der sich dem Vergleich der Sprachsteuerung im Gegensatz zur Controller-Steuerung widmet.

5.1 Auswertung

Die 15 Testpersonen waren im Alter zwischen 23 und 52 Jahren alt. Der Großteil der Testpersonen hat vorher schon Erfahrungen mit vergleichbaren Videospielen gemacht. Keine der Testpersonen hatte körperliche oder sprachliche Einschränkungen. Jede Testperson war physisch und geistig in der Lage beide der zu vergleichenden Eingabe-Methoden zu bedienen.

Der erste Abschnitt des Fragebogens widmet sich der Qualität der Sprachsteuerung. Folgende Frage wurde an die Testpersonen gestellt:

„Wie sehr konntest du die unterschiedlichen Aspekte der Raumschiff-Steuerung mit Hilfe des Sprach-Interfaces zu deiner Zufriedenheit kontrollieren?“

Die Testpersonen konnten auf einer Skala von 1(Sehr gut) bis 5(Gar nicht) ankreuzen, wie gut sie mit Hilfe der Sprachsteuerung das Raumschiff steuern konnten. Abbildung 5.1 lassen sich die Ergebnisse dieser Frage entnehmen.

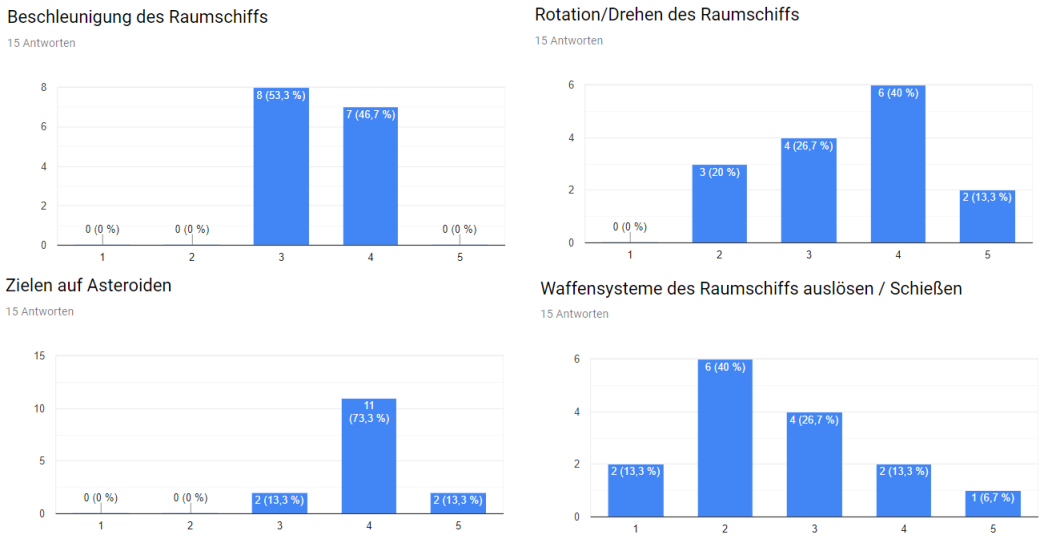

Abbildung 5.1: Wie sehr konntest du die unterschiedlichen Aspekte der Raumschiff-Steuerung mit Hilfe des Sprach-Interfaces zu deiner Zufriedenheit kontrollieren? (Quelle: Eigene Arbeit)

Die Ergebnisse zeigen deutlich, dass die wahrgenommene Qualität der Steuerung durch Spracherkennung vom jeweiligen Sprachkommando abhängt. Die Auslösung der Waffensysteme hat ein tendenziell eher positives Feedback der Testpersonen erhalten, während die anderen Sprachkommandos offensichtlich wenig gut dazu dienten das Raumschiff zu steuern. Dies lässt sich nach Vermutung des Autors dadurch erklären, dass das Absetzen von Projektilen eine vergleichsweise triviale Aktion ist und unabhängig von etwaigen Latenzen gut funktioniert, solange das Sprachkommando korrekt erkannt wurde.

Im nächsten Abschnitt wurden die Testpersonen zur Qualität der konventionellen Controller-Steuerung befragt:

„Wie sehr konntest du die unterschiedlichen Aspekte der Raumschiff-Steuerung mit Hilfe der Controller Steuerung zu deiner Zufriedenheit kontrollieren?“

Abbildung 5.2 lassen sich die Ergebnisse dieser Frage entnehmen.

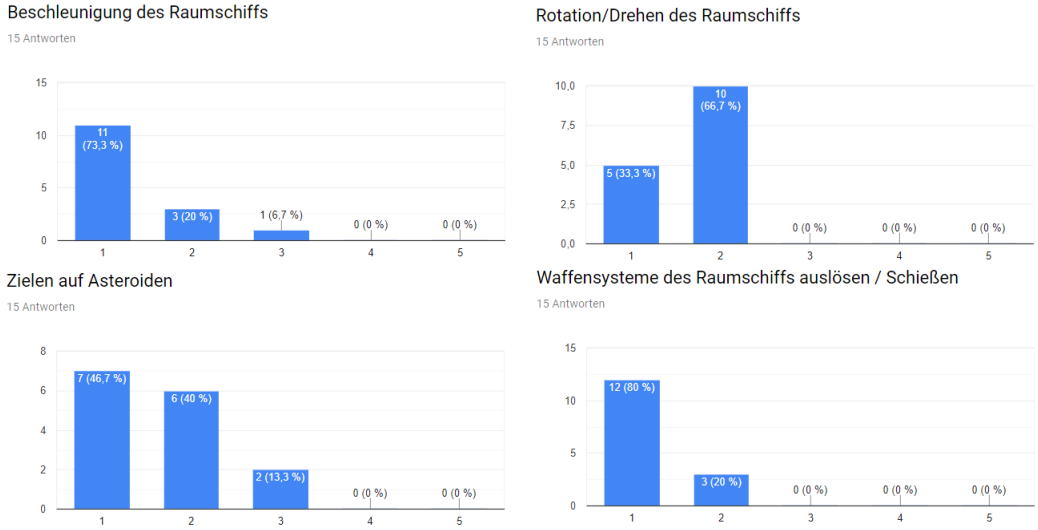

Abbildung 5.2: Wie sehr konntest du die unterschiedlichen Aspekte der Raumschiff-Steuerung mit Hilfe der Controller Steuerung zu deiner Zufriedenheit kontrollieren? (Quelle: Eigene Arbeit)

Diese Ergebnisse sind im Vergleich zum vorherigen Abschnitt wesentlich positiver. Grundsätzlich waren die Testpersonen mit der konventionellen Controller-Steuerung zufrieden bis sehr zufrieden und hatten das Gefühl das Raumschiff mit dieser Eingabe-Methode gut steuern zu können.

Im dritten und letztem Abschnitt des Fragebogens wurden die beiden Eingabe-Methoden gezielt verglichen. Den Testpersonen wurden Aussagen präsentiert und sie sollten auf einer Skala von 1(Stimme sehr zu) bis 5(Stimme gar nicht zu) auswählen, inwiefern sie diesen Aussagen zustimmen.

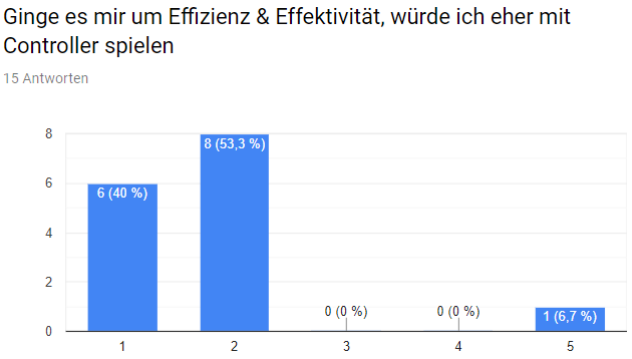

Abbildung 5.3: (Quelle: Eigene Arbeit)

Abbildung 5.3 zeigt, dass die Testpersonen eindeutig zur Controller-Steuerung greifen würden, wenn sie das Spiel möglichst effizient und effektiv spielen wollen. Daraus lässt sich spätestens schließen, dass im direkten Vergleich die konventionelle Controller-Steuerung die effizientere Eingabe-Methode ist.

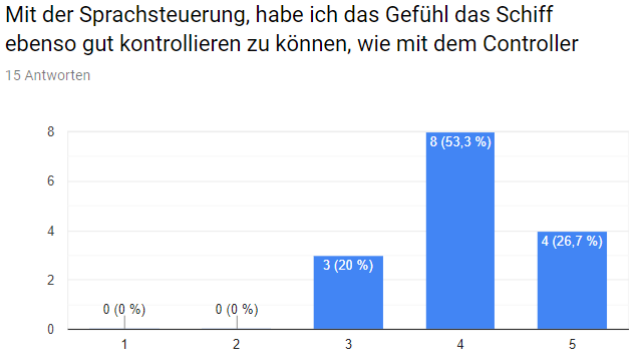

Abbildung 5.4: (Quelle: Eigene Arbeit)

Diese Annahme wird durch Abbildung 5.4 noch einmal bestätigt. Keine der Testpersonen hat der Aussage, dass sich das Spiel durch die Sprachsteuerung ebenso gut kontrollieren lässt, mehr als teilweise zugestimmt.

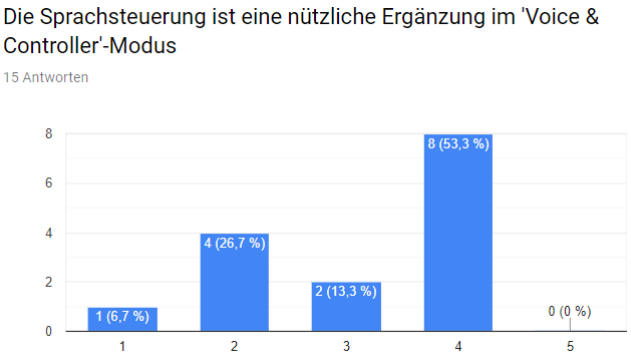

Abbildung 5.5: (Quelle: Eigene Arbeit)

Uneindeutiger fallen die Ergebnisse aus Abbildung 5.5 aus. Die Testpersonen sind sich eher uneinig darüber, ob die Sprachsteuerung im Eingabe-Modus „Complete“ eine nützliche Ergänzung darstellt. Obwohl die Tendenz hier eher gegen die Aussage spricht, stimmen ihr doch einige Testpersonen zu.

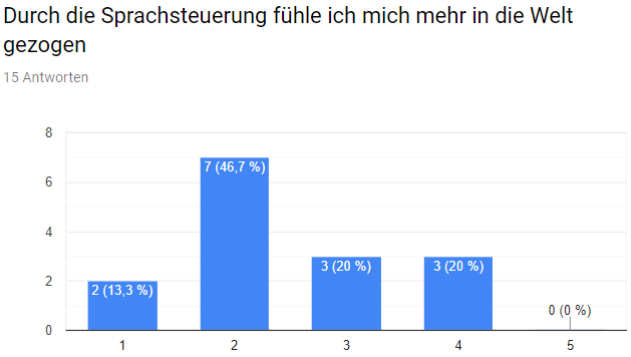

Abbildung 5.6: (Quelle: Eigene Arbeit)

Als nächstes sollten die Testpersonen beurteilen, ob sie sich durch die Sprachsteuerung mehr in die Welt gezogen gefühlt haben. Dieser Aussage wurde tendenziell eher zugestimmt (Abbildung 5.6). Daraus lässt sich vermuten, dass Sprachsteuerung unabhängig von Effizienz und Effektivität andere Qualitäten aufweist, die zu einer intensiveren VR-Erfahrung beitragen kann.

Zum Abschluss wurde den Testpersonen die offene Frage gestellt, was an der Sprachsteuerung verbessert werden könnte. Die meisten Antworten bezogen sich dabei auf die

Latenz und die korrekte Erkennung der Sprachbefehle. Besonders in Momenten wo eine schnelle Reaktion gefordert war haben sich die Testpersonen daran gestört, dass die Latenz zwischen Sprachkommando und Reaktion des Spiels zu groß war. Weiterhin wurden die Sprachbefehle besonders in diesen hektischen Momenten unzureichend genau erkannt. Dies lässt sich dadurch begründen, dass die Testpersonen aus Stress generell schneller und undeutlicher gesprochen haben, wenn sie eine sehr zügige Reaktion vom Spiel benötigten.

5.2 Fazit

Als Fazit dieser Evaluierung lässt sich sagen, dass die auf CSR basierende Sprachsteuerung für die hier entwickelte Art von Spielen tendenziell weniger geeignet ist. Zwar lässt sich das Spiel prinzipiell komplett per Sprachkommandos steuern, aber der Spieler muss Einbussen im Bezug auf Effektivität und Effizienz hinnehmen. So kann der Spieler schlichtweg nicht so schnell reagieren und Eingaben tätigen, wie mit einem konventionellem Spiele-Controller.

Für Spieler mit körperlichen Einschränkungen, die auf eine unkonventionelle Eingabe-Methode angewiesen sind, müsste das Spiel angepasst und verlangsamt werden um mit Spracherkennung eine vergleichbar effiziente Spielerfahrung herzustellen. Das würde wiederum einem sehr dynamischen und schnellen Spielablauf im Weg stehen. Da die herausgestellten Störfaktoren grundsätzlich technischer Natur sind, ist aber nicht ausgeschlossen, dass eine vergleichbare Arbeit in der Zukunft ein positiveres Ergebnis ergeben könnte.

Als interessante Beobachtung lässt sich weiterhin feststellen, dass einige Testpersonen die Sprachsteuerung durchaus als nützliche Ergänzung betrachtet haben. Dies lässt zumindest vermuten, dass ungeachtet der eigentlichen Fragestellung dieser Arbeit, ein noch weitgehend ungenutztes Potential in der auf CSR basierenden Sprachsteuerung im Zusammenhang mit Videospielen liegt.

6 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel werden die Ergebnisse dieser Bachelorarbeit vorgestellt. Im Abschnitt 6.1 werden die Inhalte der vorherigen Kapitel zusammengefasst. In Abschnitt 6.2 wird ein Blick in die Zukunft gewagt und überlegt was Spracherkennung in Videospielen in Zukunft leisten kann.

6.1 Zusammenfassung der Arbeit

In dieser Bachelorarbeit wurde ein VR-Spiel mit Spracherkennung entworfen und umgesetzt. Dieses diente dazu herauszufinden ob man mittels Spracherkennung eine Steuerung entwerfen kann, die im Bezug auf Effizienz und Effektivität mit einer konventionellen Controller-Steuerung mithalten kann. Der Gedanke hinter dieser Überlegung war es Menschen mit körperlichen Einschränkungen eine Möglichkeit zu bieten, auf Augenhöhe an Mehrspieler-Videospiel-Erfahrungen teilhaben zu lassen.

Dafür wurden zunächst die Grundlagen (Kapitel 2) zum Thema Spracherkennung aufgearbeitet. Innerhalb dieser Aufarbeitung wurde auf vergleichbare Arbeiten eingegangen, die sich ebenfalls dem Thema Inklusion durch Spracherkennung widmen. Auch wurde sich generell der Einsatz von Spracherkennung in Videospielen angeschaut.

Im Anschluss daran fand eine Anforderungsanalyse statt (Kapitel 3). Hier wurde die Idee des Spiels in konkrete Formen gegossen und es haben sich die individuellen Rollen innerhalb des zu entwickelnden Spiels herauskristallisiert.

Der Entwurf und die Umsetzung (Kapitel 4) beschreibt die Überführung eben dieser Anforderungen in eine funktionstüchtige Anwendung. An dieser Anwendung wurde mit einer Gruppe von Testpersonen eine Evaluierung (Kapitel 5) durchgeführt, deren Ergebnisse ausgewertet wurden. Abschließend wurde das Fazit gezogen, dass sich zum aktuellen Zeitpunkt eine kommandobasierte Sprachsteuerung nicht als gleichwertiger Ersatz für eine konventionelle Controller-Steuerung einsetzen lässt.

6.2 Ausblick

Ein Blick in die Grundlagen aus Kapitel 2 zeigt, dass im Forschungsfeld der Spracherkennung innerhalb der letzten Jahrzehnte große Fortschritte gemacht wurden. Es ist davon auszugehen, dass durch zunehmende Rechenleistung privater Computer und besserer Algorithmen Spracherkennung in Zukunft noch besser und effizienter vonstatten gehen wird. Ob dadurch kommandobasierte Spracherkennung jemals die Effizienz und Effektivität eines klassischen Controllers erreichen wird, ist zwar äußert fraglich, aber nicht ausgeschlossen. Daher erscheint es vernünftig vergleichbare Arbeiten auch in Zukunft durchzuführen, um etwaiges Potential rechtzeitig zu erkennen.

A CD-Inhalt

Die beiligende CD dieser Bachelorarbeit enthält folgende Ordner bzw. Datein:

- **Bachelorarbeit-SpracheVsController.pdf** ist die vorliegende Bachelorarbeit als PDF-Datei.
- **Unity**/ enthält das Unity Projekt der Umsetzung des VR-Spiels.
- **Build**/ enthält den aktuellsten Build der Umsetzung des VR-Spiels.

Literaturverzeichnis

[1] Allison, Fraser ; Carter, Marcus ; Gibbs, Martin: Word Play: A History of Voice Interaction in Digital Games. In: *Games and Culture* (2017), S. 1555412017746305

[2] and and: Robot control based on voice command. In: *2008 IEEE International Conference on Automation and Logistics*, Sep. 2008, S. 2490–2494. – ISSN 2161-8151

[3] Association, Entertainment S.: *2018 Sales, Demographic and Usage Data. Essential Facts about the computer and video game industry.* 2018. – URL http://www.theesa.com/wp-content/uploads/2018/05/EF2018_FINAL.pdf

[4] Bickenbach, Jerome: The World Report on Disability. In: *Disability & Society* 26 (2011), Nr. 5, S. 655–658. – URL https://doi.org/10.1080/09687599.2011.589198

[5] Bolt, Richard A.: &Ldquo;Put-that-there&Rdquo;: Voice and Gesture at the Graphics Interface. In: *SIGGRAPH Comput. Graph.* 14 (1980), Juli, Nr. 3, S. 262–270. – URL http://doi.acm.org/10.1145/965105.807503. – ISSN 0097-8930

[6] Cohen, Arnon ; Graupe, Daniel: Speech recognition and control system for the severely disabled. In: *Journal of Biomedical Engineering* 2 (1980), Nr. 2, S. 97 – 107. – URL http://www.sciencedirect.com/science/article/pii/0141542580900606. – ISSN 0141-5425

[7] Dalton, Jacqueline R. ; Peterson, Cindee Q.: The Use of Voice Recognition as a Control Interface for Word Processing. In: *Occupational Therapy In Health Care* 11 (1997), Nr. 1, S. 75–81. – URL https://doi.org/10.1080/J003v11n01_05

[8] Davis, K. H. ; Biddulph, R. ; Balashek, S.: Automatic Recognition of Spoken Digits. In: *The Journal of the Acoustical Society of America* 24 (1952), Nr. 6, S. 637–642. – URL https://doi.org/10.1121/1.1906946

[9] Gales, Mark ; Young, Steve: The Application of Hidden Markov Models in Speech Recognition. In: *Found. Trends Signal Process.* 1 (2007), Januar, Nr. 3, S. 195–304. – URL http://dx.doi.org/10.1561/2000000004. – ISSN 1932-8346

[10] Grunza, E. ; Cohen, S.: A voice activated control system for the severely disabled. In: *ICASSP '77. IEEE International Conference on Acoustics, Speech, and Signal Processing* Bd. 2, May 1977, S. 257–260

[11] Harada, Susumu ; Landay, James A. ; Malkin, Jonathan ; Li, Xiao ; Bilmes, Jeff A.: The Vocal Joystick:: Evaluation of Voice-based Cursor Control Techniques. In: *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility.* New York, NY, USA : ACM, 2006 (Assets '06), S. 197–204. – URL http://doi.acm.org/10.1145/1168987.1169021. – ISBN 1-59593-290-9

[12] Harada, Susumu ; Wobbrock, Jacob O. ; Landay, James A.: Voice Games: Investigation Into the Use of Non-speech Voice Input for Making Computer Games More Accessible. In: Campos, Pedro (Hrsg.) ; Graham, Nicholas (Hrsg.) ; Jorge, Joaquim (Hrsg.) ; Nunes, Nuno (Hrsg.) ; Palanque, Philippe (Hrsg.) ; Winckler, Marco (Hrsg.): *Human-Computer Interaction – INTERACT 2011.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, S. 11–29. – ISBN 978-3-642-23774-4

[13] Iacovides, Ioanna ; Cox, Anna ; Kennedy, Richard ; Cairns, Paul ; Jennett, Charlene: Removing the HUD: The Impact of Non-Diegetic Game Elements and Expertise on Player Involvement. In: *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play.* New York, NY, USA : ACM, 2015 (CHI PLAY '15), S. 13–22. – URL http://doi.acm.org/10.1145/2793107.2793120. – ISBN 978-1-4503-3466-2

[14] Juang, B. H. ; Rabiner, L. R.: Hidden Markov Models for Speech Recognition. In: *Technometrics* 33 (1991), Nr. 3, S. 251–272. – URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484833

[15] Lowerre, Bruce T.: The HARPY speech recognition system / CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE. 1976. – Forschungsbericht

[16] Maciej, Jannette ; Vollrath, Mark: Comparison of manual vs. speech-based interaction with in-vehicle information systems. In: *Accident Analysis Prevention*

41 (2009), Nr. 5, S. 924 – 930. – URL http://www.sciencedirect.com/science/article/pii/S0001457509001080. – ISSN 0001-4575

[17] Masato Nishimori, Takeshi S. ; Konishi, Ryosuke: Voice Controlled Intelligent Wheelchair. In: *SICE Annual Conference 2007*, Sep. 2007, S. 336–340

[18] Rogowski, Adam: Industrially oriented voice control system. In: *Robotics and Computer-Integrated Manufacturing* 28 (2012), Nr. 3, S. 303 – 315. – URL http://www.sciencedirect.com/science/article/pii/S0736584511001189. – ISSN 0736-5845

[19] Sporka, Adam J. ; Kurniawan, Sri H. ; Mahmud, Murni ; Slavík, Pavel: Non-speech Input and Speech Recognition for Real-time Control of Computer Games. In: *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*. New York, NY, USA : ACM, 2006 (Assets '06), S. 213–220. – URL http://doi.acm.org/10.1145/1168987.1169023. – ISBN 1-59593-290-9

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Sprache versus Controller - Eignung von sprachgesteuerten Assistenten zur Interaktion im 3D-Raum

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original