HAW HAMBURG

Bachelorarbeit

Robert Bernhof

Augmented Reality mit der Microsoft HoloLens im Indoor Bereich

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Robert Bernhof

Augmented Reality mit der Microsoft HoloLens im Indoor Bereich

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

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Philipp Jenke

Eingereicht am: 03. Juni 2019

Robert Bernhof

Thema der Arbeit

Augmented Reality mit der Microsoft HoloLens im Indoor Bereich

Stichworte

Augmented Reality, Interaktion, Registration, Okklusion, Head-Mounted-Display, Microsoft Hololens

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Möglichkeit, mit Hilfe der Microsoft Hololens positionsbezogene Objekte anzeigen zu lassen und mit Ihnen zu interagieren. Hierbei steht im Vordergrund, dass nur das nächstgelegene Objekt angezeigt wird, um den Benutzer nicht mit zu vielen Informationen abzulenken. Zudem sollen die angezeigten Objekte dynamisch durch den Nutzer veränderbar sein, damit für diesen nur Informationen angezeigt werden, welche individuelle Relevanz haben (Recommender Systems). Es werden Grundbegriffe definiert die für AR von Bedeutung sind und die AR selbst in den Mixed Reality Kontext eingeordnet. Dafür wird die Geschichte, die aktuellen Möglichkeiten und Einsatzgebiete von AR erläutert und mit Hilfe der Microsoft Hololens ein Prototyp entwickelt und implementiert. Beim Prototyp werden die Steuerung der AR-Elemente mit Hilfe von Sprache und Gesten evaluiert und darauf geachtet ob es mit dem aktuellen Stand der Technik möglich ist eine authentische Immersion zu erzeugen.

Robert Bernhof

Title of Thesis

Augmented Reality with Microsoft HoloLens for Indoor Usage

Keywords

Augmented Reality, interaction, registration, occlusion, Head-Mounted-Display, Microsoft Hololens

Abstract

This work examinates the possibilities to display and interact with location based objects with the Microsoft Hololens. The main focus is that only the nearest object is displayed, to not distract the user with too many information at once. In addition, the displayed objects should be dynamically changeable by the user, so that only information that has individual relevance is displayed for him (Recommender Systems). Basic terms are defined that are important for AR and then AR itself is placed within the context of Mixed Reality. For this the history, the current possibilities and application areas of AR are explained and a prototype will be developed and implemented with the help of the Microsoft Hololens. Within the prototype, the controls of the AR elements are evaluated with the help of speech and gestures and consideration is given to whether it is possible with the current state of the art to create an authentic immersion.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Es gibt heutzutage immer mehr Informationen, die gefiltert, zugeordnet und angezeigt werden müssen. Dies geschieht unter anderem durch die Augmented Reality, welche fast nicht mehr aus dem normalen Leben wegzudenken ist. Ob bewusst oder unbewusst ist sie mittlerweile allgegenwärtig und dies in fast allen Bereichen. Im privaten Bereich gibt es Spiele auf dem Handy, beim Fernsehen werden Hilfsinformationen eingeblendet wie die Abseitslinie beim Fußball und bei öffentlichen Verkehrsmitteln existieren elektronische Anzeigen mit Abfahrtszeiten. Dass die AR so allgegenwärtig ist, ist aber keine gezwungene Entwicklung, sondern orientiert sich an der Nachfrage der Gesellschaft nach solchen Systemen. Der Anstieg des Interesses begann im Zeitraum von 1980-1989. In diesem Zeitraum sind 389 wissenschaftliche Publikationen zum Thema AR veröffentlicht worden. Im Jahrzehnt danach ist die Zahl der Veröffentlichungen auf 1571 angestiegen. Heutzutage (Zeitraum 2010-2019, bisher mindestens 12500 Publikationen) haben sich die Publikationensveröffentlichungen fast verzehnfacht [18]. Einen großen Einfluss auf diese Entwicklung hatte in den letzten Jahren *Pokemon GO*, denn diese Applikation für Smartphones und Tablets wurde bisher circa 800 Millionen mal runtergeladen [27]. Dort gibt es die Option, AR einzuschalten, um die *Pokemons* per Smartphone Display so anzeigen zu lassen, dass sie wie reale Objekte in der realen Welt wirken. Zudem ist *Pokemon GO* das erste location-based AR Spiel, welches bei der breiten Masse der Gesellschaft Anklang gefunden hat [25]. Hierbei handelt es sich, um es genauer zu definieren, um eine *global location-based* Anwendung, welche mit GPS-Daten an bestimmten Orten der Umwelt virtuelle Objekte platziert. Für *local location-based* Anwendungen gibt es verschiedene Ansätze, denn GPS-Daten sind auf und vor allem in kleineren Räumen für die Positionsbestimmung zu ungenau. Zwar könnten zum Beispiel Bluetooth Beacons oder Wi-Fi Sender benutzt werden, um die Positionsbestimmung im Indoor Bereich umzusetzen, dies würde aber bedeuten, extra Hardware anschaffen zu müssen. Google arbeitet schon länger an einer Lösung der Indoor Navigation mit Smartphones und hat 2017 die Technologie ARCore vorgestellt. Diese Technologie kann zur Entwicklung von Anwendungen eingesetzt werden und erkennt mit Hilfe der Sensoren und Kamera von Smartphones

die eigene Position und versucht ein Verständnis des Raumes zu erstellen. So können zusätzliche virtuelle Informationen in der realen Welt durch das Smartphone Display angezeigt werden [5]. Auch im professionellen Umfeld sieht man das steigende Interesse an Indoor AR. So verwendet Thyssenkrupp das Head-Mounted-Display *Microsoft HoloLens* zur Wartung von Fahrstühlen. Hierbei werden dem Techniker, ohne die Abdeckungen entfernen zu müssen, zusätzliche Informationen zu den technischen Teilen angezeigt. Dies hat unter anderem den Effekt, dass sich die Dauer von Service Wartungen um das vierfache reduziert hat [14]. Somit gibt es je nach Anforderung verschiedene Ansätze der Erkennung und Bestimmung der Position von realen und virtuellen Objekten. Virtuelle Objekte, welche sich nicht anhand der Position des Nutzers verändern, sondern ihre Rotation durch die Umgebung festgelegt wird, werden als Hologramme bezeichnet. Für diese Arbeit entscheidend sind die Erkennung der Umgebung und die lokale Bestimmung der Hologramme durch Sensoren der *Microsoft HoloLens* im Indoor Bereich. Aus Gründen der leichteren Lesbarkeit wird in dieser Arbeit nur von Hologrammen gesprochen und der Name der *Microsoft HoloLens* abgekürzt.

1.1 Motivation

Mit neu entwickelter Hardware im Bereich der AR und besonders bei Head-Mounted-Displays (HMD), wird die private Anschaffung immer erschwinglicher. So steigt die Verbreitung jener Geräte immer weiter an. Dies hat zur Folge, dass sich immer mehr Optionen zur Nutzung von AR ermöglichen lassen, was vor einigen Jahren noch nicht möglich gewesen wäre. Dadurch entstand die Idee, das Prinzip der personenbezogenen Werbung, welche schon länger in bestimmten Bereichen vorhanden ist, in der realen Welt umzusetzen. Angefangen hat dieser Trend im Internet, wo mit Hilfe von gespeicherten Informationen des Nutzers gezielt Werbung für diesen ausgespielt werden kann. Diese Informationen orientieren sich an bestimmten Eckdaten wie Alter, Geschlecht, persönliche Präferenzen und auch an spezifische Geolokationsdaten. Diese Daten werden auch anderweitig, auf dynamischen Anzeigen außerhalb vom Internet verwendet. Bei *Advanced TVs* (*Addressable TV*, *Connected TV* und *Programmatic TV*) werden beispielsweise Werbeeinblendungen beziehungsweise Werbespots über das Internet gekauft und so vom Käufer dynamisch bespielt. Im öffentlichen Raum gibt es *Digital Out-of-Home*, bei dem große Monitore und Leinwände verwendet werden, um dynamische Werbung auszuspielen. Der Nachteil dieser Systeme ist jedoch, dass diese Werbung nicht auf das individuelle Interesse einzelner Personen zugeschnitten ist, welche auf diese Anzeige schauen. Somit besteht

momentan die Nachfrage nach einer Lösung für individuell angezeigte Informationen, die sowohl abhängig von Lokation, als auch anderen definierten Umständen (persönliche Präferenzen, Uhrzeit, Kultur usw.) sind.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, eine Anwendung für die ortsgebundene Anzeige von relevanten Objekten und Raumelementen mit der HoloLens zu Entwickeln und Evaluieren. Dazu gehört die Lokalisierung des Nutzers, die Platzierung der Hologramme und die korrekte Darstellung dieser. Der Nutzer soll außerdem dazu in der Lage sein, mit der virtuellen Umgebung zu interagieren, um die Position und die angezeigte Information der Hologramme zu verändern. Dabei wird auf die aktuell gängigen Steuerungselemente für AR zurückgegriffen. So sollen personifizierte Informationen angezeigt werden können, ohne dass der Nutzer das Gefühl hat, dass die reale Welt mit unechten Objekten angereichert ist. Dies steigert die Immersion für den Nutzer und das umgesetzte Konzept soll damit genauso unauffällig und allgegenwärtig werden, wie bereits vorhandene Konzepte für Fernsehen beziehungsweise statische Anzeigen in der Öffentlichkeit. Dieses Konzept würde sich dann auf viele verschiedene Bereiche anwenden lassen und nicht nur für personifizierte Werbung Verwendung finden.

1.3 Gliederung

Im folgenden werden die fünf Kapitel dieser Arbeit vorgestellt.

In der Einleitung (Kapitel 1) wird eine allgemeine Übersicht über das Arbeitsthema gegeben und anschließend die Motivation und Zielsetzung für diese Arbeit erläutert. Die darauffolgenden Kapitel zwei bis vier bilden den Hauptteil dieser Arbeit. In der Analyse (Kapitel 2) wird eine Übersicht der aktuellen Bestandteile des *Reality-Virtuality-Continuums* gegeben und die Augmented Reality in diesen Kontext eingeordnet. Danach wird die AR näher Untersucht und auf ihre drei Hauptbestandteile – Ausgabegeräte, Eingabegeräte und Tracking – analysiert. Daraus ergeben sich dann die Anforderungen, welche an die zu entwickelnde Anwendung gestellt werden. In Kapitel 3 werden die Eigenschaften der HoloLens vorgestellt und in Bezug auf die Hauptbestandteile der AR

bewertet. Zudem wird ein kleiner Überblick über die verwendete Software gegeben, welche zur Unterstützung der Entwicklung der Anwendung benutzt wurde. Als nächstes folgt dann das Design und die Umsetzung der Anwendung in Kapitel 4. Hier wird sich mit Hilfe der herausgearbeiteten Anforderungen für eine Softwarearchitektur entschieden und im Anschluss ihre Umsetzung erläutert und getestet. Der Test wird unterteilt in zwei Schritte – Testen mit dem Emulator und Testen auf der tatsächlichen Hardware der HoloLens. Anschließend wird der Test evaluiert. Im letzten Kapitel wird eine Zusammenfassung über diese Arbeit und der damit verbundenen Erkenntnisse gegeben und ein Ausblick über mögliche weitere Einsatzgebiete der Anwendung und allgemeine Zukunftsaussichten der HoloLens präsentiert.

2 Analyse

Dörner beschreibt [26, S.2-4] wie unterschiedlich die Menschen die Welt mit Hilfe der Sinne wahrnehmen. Zum Beispiel werden Sinneszellen auf der Netzhaut durch das von Objekten reflektierte Licht stimuliert und als Nervenimpulse ins Gehirn geleitet. Dort werden von jedem Menschen diese Impulse unterschiedlich interpretiert und führen damit entweder zur gleichen oder unterschiedlichen Wahrnehmung. Daraus ergibt sich, dass der Zusammenhang zwischen den Reizen der realen Welt und der Wahrnehmung des Menschen auf die Realität nicht eindeutig existiert. Dies ermöglicht einen gewissen Spielraum die Realität zu manipulieren. Somit wäre es möglich, die Wahrnehmung des Menschen zu beeinflussen, indem der Realität virtuelle Reize hinzugefügt werden, die auf eine Person wirken. Wenn hierbei beispielsweise das visuelle System des Menschen durch einen künstlich hinzugefügten Reiz überlistet wird, würde es die gleichen Impulse wahrnehmen wie bei einem realen Objekt und somit einem Trugschluss unterliegen. Dieser Trugschluss führt dazu, dass die Person annimmt das Hologramm wäre tatsächlich ein reales Objekt in der echten Welt. Um die HoloLens in diesen Kontext einordnen zu können, müssen zunächst alle wichtigen Begriffe erläutert werden, welche bei der Klassifizierung von realer bis virtueller Umgebung helfen. Die wichtigsten Begriffe sind hier Reality-Virtuality Continuum (RVC), Mixed-Reality(MR), Virtual-Reality (VR), Augmented-Virtuality (AV) und Augmented-Reality (AR).

2.1 Reality-Virtuality Continuum

Um Technologien, welche die Realität mit virtuellen Reizen anreichern, besser zu klassifizieren, haben Paul Milgrim u.a. im Jahr 1994 das Reality-Virtuality Continuum definiert [[24]]. Abbildung 2.1 stellt das RVC als eindimensionale Linie dar. Auf der linken Seite befindet sich die reale Welt und auf der rechten die virtuelle Welt. Dazwischen liegen verschiedene Konzepte, welche in die Klassifizierung der Mixed Reality fallen. Die Augmented Reality und die Augmented Virtuality sind also Teil der MR. Ob es sich um ein

AR- oder VR-System handelt, hängt demnach davon ab ob die echte Realität mit virtuellen Elementen angereichert wird oder ob reale Objekte in die virtuelle Welt eingebunden werden.

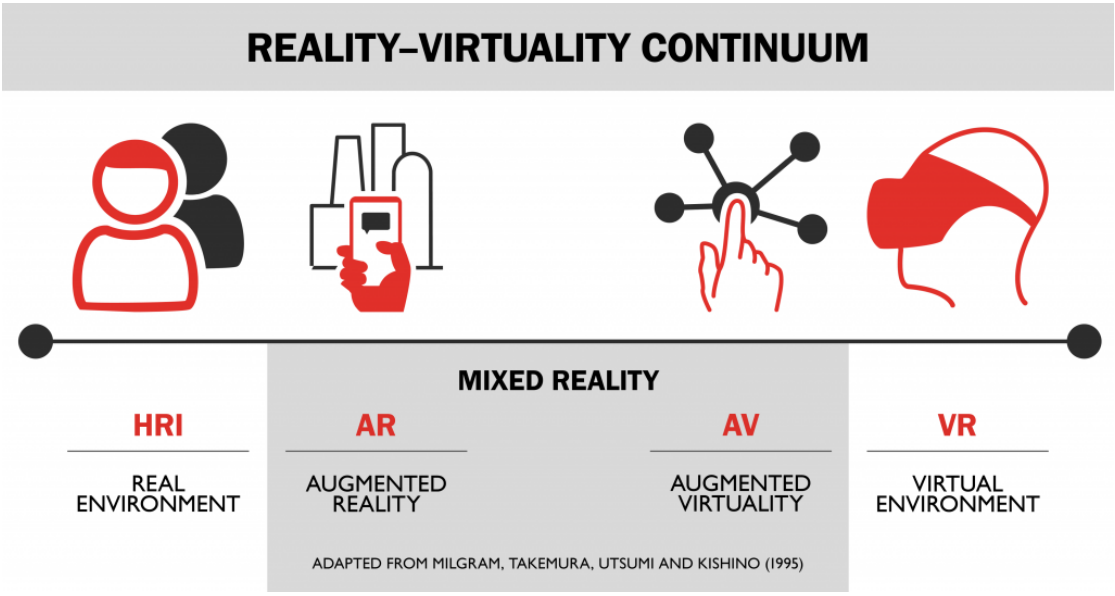

Abbildung 2.1: Vereinfachte Darstellung des RVC nach Milgram et al. ([13])

Mixed Reality

In Kapitel 2.1 wurde die Mixed Reality bereits in das RVC eingeordnet. Im RVC stellt die MR ein großen Teil dar, welcher genau zwischen der richtigen Welt und einer komplett virtuellen Welt liegt. Eine MR-Umgebung wird von Azuma und Kishino [23] als eine Mischung aus Gegenständen der virtuellen und realen Welt, die auf einem Display vereint werden, beschrieben. Diese Definition ist von 1994 und seitdem hat sich der Begriff der MR weiter verändert. In [10] wird beschrieben, dass mittlerweile nicht nur Displays für die MR Verwendung finden, sondern darüber hinaus auch Geräusche, die Position im Raum oder anderer Input der Umgebung oder des Menschen. In Abbildung 2.2 kann man erkennen, dass die Mixed Reality von den drei Faktoren Mensch, Computer und Umgebung abhängig ist. Hierbei ist besonders die Human Computer Interaction wichtig, da die Immersion der Hologramme auch davon abhängig ist, ob man mit ihnen auch interagieren und sie manipulieren kann.

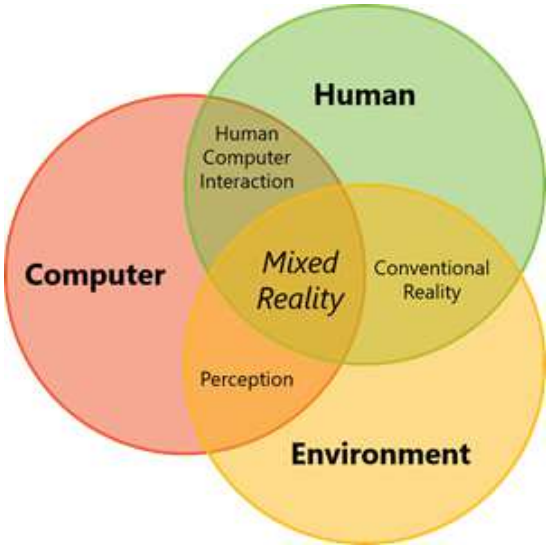

Abbildung 2.2: Darstellung der relevanten Komponenten für MR ([10])

Im Vergleich zu den Anfängen der AR, bei der nur einfache Darstellungen über das Bild der Kamera gelegt wurden, ist es mittlerweile möglich eine täuschend echte Verschmelzung der Realität mit Hologrammen zu schaffen. Dies geschieht, indem nicht nur der virtuelle Input auf die reale Umgebung wirkt, sondern auch die reale Umgebung Input auf Hologramme liefert. Die Position des Menschen zu den Hologrammen bestimmt den Blickwinkel, mit dem der Mensch das Hologramm wahrnimmt. Dieser Blinkwinkel ist entscheidend, da die Hologramme hinter realen Objekten verschwinden können oder bei Bewegung dieser Kollisionen untereinander entstehen könnten.

Abbildung 2.3: Darstellung der Unterteilung der MR ([10])

In Abbildung 2.3 wird deutlich, wie groß der Bereich der MR wirklich ist. Links wird die physikalische Realität dargestellt und rechts die digitale Realität. Der blaue Bereich links soll die AR darstellen und der grüne Bereich rechts die VR. Der Zwischenbereich

umfasst sehr viele unterschiedliche Möglichkeiten, die beiden Realitäten zu verschmelzen und geben den Grad an, inwieweit es in eine der Extreme geht. Auch wird dargestellt, dass die Geräte der MR grob in zwei Kategorien eingeteilt werden – die *Holographic Devices* und die *Immersive Devices*. Das Hauptaugenmerk bei den *Holographic Devices* liegt auf digitalen Objekten, welche in die reale Umgebung eingebettet werden, und darauf, sie möglichst real wirken zu lassen. Dies geschieht mit Hilfe von *see-through Displays* mit denen man die reale Welt noch sehen kann. Die HoloLens ist solch ein Gerät, welches in diese Kategorie einzuordnen ist. Die *Immersive Devices* versuchen die reale Welt meist vollständig auszublenden und diese durch eine virtuelle Welt zu ersetzen. Dabei wird die komplette Sicht mit Hilfe von Opaque Displays verdeckt, welche keine Möglichkeit mehr bieten die reale Umgebung zu sehen.

Momentan gibt es noch keine Geräte, welche auf beiden Seiten des Spektrums arbeiten können und so muss sich der Entwickler oder Nutzer im Vorfeld im Klaren darüber sein, welchen Effekt er erzielen beziehungsweise nutzen möchte. Im Falle der HoloLens sieht man in der Abbildung 2.4, dass dieses Gerät zwar noch im Bereich AR eingeordnet ist, es aber durch die Möglichkeiten der Verschmelzung und Erkennung der virtuellen mit der realen Welt zwischen AR und VR eingeordnet werden kann.

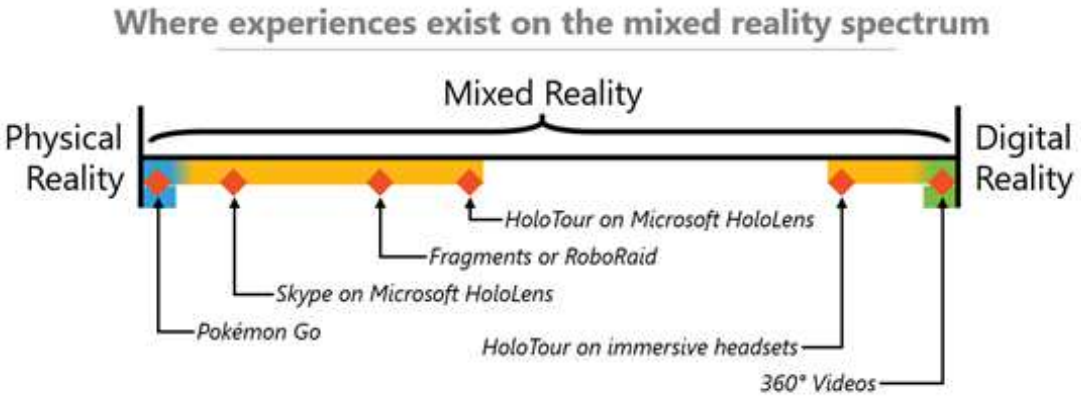

Abbildung 2.4: Einordnung der HoloLens in den MR Bereich ([10])

Augmented Virtuality

Augmented Virtuality liegt auf dem Spektrum des RVC in der MR und ist eher im Grenzbereich zu VR angesiedelt. Bei AV wird eine komplette virtuelle Welt durch das anreichern realer Objekte ergänzt. Somit ist nur ein kleiner Teil dieser Welt real beziehungsweise durch reale Objekte dargestellt. Ein Beispiel, welches schon lange existiert, wäre die Wettervorhersage im Fernsehen. Hier steht der Moderator (reales Objekt), meist vor virtuell erzeugten Wetterkarten und Animationen. Dabei wird das Bild des Moderators nicht verändert, sondern nur über die virtuelle Welt gelegt. Ein weiteres Beispiel, welches mehr immersiv ist, ist die Leap Motion. Hierbei werden mit Hilfe eines Sensors die Hände in die virtuelle Umgebung übertragen und der Nutzer kann somit ganz normale Handbewegungen als Interaktionswerkzeuge in der virtuellen Umgebung benutzen. Die *Leap Motion* kann mittlerweile mit vielen VR-Systemen kombiniert werden und verändert so VR-Umgebungen zu AV-Umgebungen [17].

Augmented Reality

Da die HoloLens der Augmented Reality zugeordnet ist, muss erst einmal geklärt werden, wo die AR anfängt und wo genau sie aufhört. In den vorherigen Kapiteln wurde erläutert, dass die Übergänge innerhalb des RVC meist fließend sind und nicht genau bestimmt werden können. Die Definition von Ronald Azuma hilft die AR besser einordnen zu können:

> Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. [28]

Folglich ist die AR der Gegenpart zu der AV, denn hier wird die reale Welt nur um einzelne Hologramme ergänzt. Zudem werden von Ronald Azuma noch drei Eigenschaften definiert, welche ein System aufweisen muss, um als AR-System klassifiziert zu werden:

1) Combines real and virtual

2) Interactive in real time

3) Registered in 3-D

Somit sind die wichtigsten Eigenschaften, dass in einer räumlichen Dimension in Echtzeit, reale und virtuelle Objekte kombiniert werden. Ob dies mit Hilfe von HMD geschieht oder andere Methoden benötigt werden, ist für die Definition eines AR-Systems irrelevant. Zudem wird hier nicht eingeschränkt, ob es sich nur um visuelle oder auch andere Sinne handeln könnte mit der die reale Welt angereichert wird. Damit allerdings die Immersion perfekt sein kann, ist der Punkt des *real time* besonders wichtig. Genauso wie bei der AV muss in der AR das Zusammenspiel zwischen realer und virtueller Welt perfekt sein, damit dem Nutzer nicht auffällt, dass virtuell erzeugte Objekte verwendet werden. Bekannte Beispiele in diesem Bereich sind mittlerweile schon sehr lange Bestandteile im normalen Umfeld. Dazu zählen zum Beispiel die Abseitslinie beim Fußball oder die Anzeige der Wartezeit an Bahnhöfen oder Flughäfen. In diesen Bereichen ist die AR schon so lange Teil des normalen Lebens, dass einem nicht mehr bewusst ist, dass dies überhaupt AR ist.

2.2 Geschichte von Augmented Reality

Von Augmented Reality ist das erste mal die Rede, als Morton Heilig 1955 das Kino der Zukunft namens Sensorama beschreibt, welches er 1962 dann auch umgesetzt hat. Dieser Prototyp hat mit Hilfe von Geräuschen, Vibration, Luftzug, Gerüchen und Bildern gearbeitet und so das erste mal die virtuelle mit der realen Welt verschmolzen. Das erste HMD erfanden Ivan Sutherland und Thomas A. Furnes III in den Jahren 1965-1968. Genannt wurde es das *Damoklesschwert*, da es so schwer war, dass es über dem Nutzer an der Decke angebracht werden musste und so über ihm schwebte. Es hing an beweglichen Stangen und konnte so die Position des Kopfes ermitteln, mit der dann die virtuelle Welt gesteuert werden konnte ([11]).

Abbildung 2.5: "The sword of damocles"HMD [12]

Durch die noch nicht ausgereifte Technologie konnte das Damokolesschwert nur einfache Wireframes (Drahtgittermodell) darstellen und selbst dort waren die Hardwareanforderungen so hoch, dass es bei einem Prototyp blieb und nie in Serie produziert wurde. 1975 wurde dann von Myron Krueger ein ganzer Raum geschaffen, in welchem die Nutzer mit virtuellen Objekten interagieren konnten. Dies geschah nur mit Hilfe von Monitoren und Kameras und so konnte der Nutzer durch seine Bewegungen die virtuelle Welt steuern. Bis Heute gab es viele Weiterentwicklungen und Prototypen im Bereich der AR, allerdings sind die beiden erwähnten die ersten richtigen Systeme als Beispiel für HMD und reine Display AR [12]. Heutzutage sind fast alle AR-Systeme Weiterentwicklungen dieser beiden Ursprungssysteme. Für Display AR-Systeme kam der große Durchbruch im privaten Bereich, als die ersten Smartphones auf den Markt kamen, welche integrierte

Kameras besaßen. Hier konnte das Smartphone eine Verschmelzung realer Bilder von der Kamera mit virtuellen Grafiken erzeugen [6]. Das wohl bekannteste Beispiel der letzten Jahre ist *Pokemon GO*. Dies ist ein Spiel für Smartphones und funktioniert mit Hilfe der Kamera und GPS. Der Nutzer muss hierbei durch die reale Welt gehen und dort nach Monstern auf seiner Kamera suchen und diese fangen. Dies war der erste Schritt, AR-Spiele der breiten Masse vorzustellen. Ein Vorreiter für die ersten privat verbreiteten HMD war die *Google Glass*. 2012 wurde dieses Projekt vorgestellt, wobei es sich um eine Brille mit integrierten See-through Displays handelte und welche über *touch* Sensoren und Spracheingaben gesteuert werden konnte. Die Technologie war das erste mal soweit, ein komfortables HMD zu entwickeln, welches eine gute Immersion erzeugte und hatte dadurch großen Einfluss auf die Weiterentwicklung neuerer MR Geräte. Heutzutage sind die wohl bekanntesten HMD die *Oculus Rift*, welche auch 2012 hauptsächlich für Spiele vorgestellt wurde, die *HTC Vive*, ebenfalls für Spiele, und die HoloLens. Im Gegensatz zu der *Oculus Rift* und *HTC Vive*, welches reine VR-HMD sind, ist die HoloLens für AR gedacht. Zudem ist die HoloLens ein alleinstehender Computer und benötigt daher keine externe Hardware und kann Kabellos genutzt werden. Der Hauptaugenmerk liegt auch nicht auf Spielen, sondern auf Anwendungen in allen gesellschaftlichen Bereichen [6].

2.3 Aufbau eines Augmented Reality Systems

Ein Augmented Reality System besteht aus drei essentiellen Teilen: Den Ausgabegeräten, den Eingabegeräten und dem Tracking. Im Folgenden werden diese Teile vorgestellt und Bezug auf die HoloLens genommen.

2.3.1 Ausgabegeräte

Wie schon in Kapitel 2.1 erwähnt, sind alle heutigen AR-Systeme im Ursprung in zwei verschiedene Arten von Ausgabegeräte einzuordnen. Hier gibt es die Unterscheidung der *optical* und *video technology* und beide Arten haben ihre Vor- und Nachteile.

Variante 1: Optical see-through Display

Beim optical see-through Display sieht der Nutzer durch halb spiegelnde Gläser hindurch und kann so die reale Welt sehen. Auf den Gläsern werden dann die virtuell erzeugten

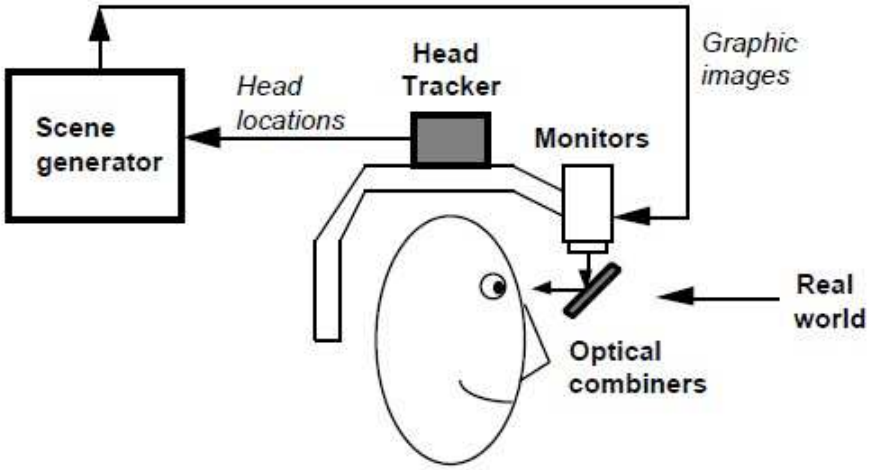

Abbildung 2.6: *Optical see-through* HMD Konzept Diagramm ([28])

Informationen gespiegelt und so sieht es für den Nutzer so aus, als wären diese inmitten des Raumes. Dies ermöglicht eine Verschmelzung der Realität mit virtuellen Daten, ohne dass die Sicht des Nutzers weitestgehend blockiert wird. Der Nachteil war, dass sich die Einstellung, wie viel Licht durch die Gläser gelassen wird, als schwierig herausstellte. Um die spiegelnden Hologramme gut zu sehen, darf nicht das komplette Licht der realen Welt durchgelassen werden. Zwar gibt es auch andere Möglichkeiten mit diesem Problem umzugehen, allerdings benutzen die meisten optical see-through HMDs diese Lichtreduzierung immernoch [28]. Abbildung 2.6 zeigt ein Konzeptbeispiel eines *optical see-through* Displays. Der Vorteil von optical see-through displays ist, dass das Blickfeld zwar teilweise eingeschränkt ist, aber nicht so weit, wie beim Schauen auf einen Monitor. Die optimale Lösung wäre ein Gerät, wie die *Google Glass*, welches den Nutzer so wenig im Sichtfeld einschränkt, wie es eine normale Brille tun würde [31].

Variante 2: Video see-through Display

Im Gegensatz zu *optical see-through* HMD hat der Nutzer bei *video see-through* HMD kleine Monitore vor den Augen und keine durchschaubaren Gläser. Mit Hilfe von Kameras die auf dem HMD angebracht sind, wird ein Bild aufgenommen welches die Sicht des Nutzers darstellen soll. Dann werden virtuell erzeugte Grafiken mit dem aufgenommenen Bild kombiniert, um die reale mit der virtuellen Welt zu vereinen. Das Ergebnis wird auf

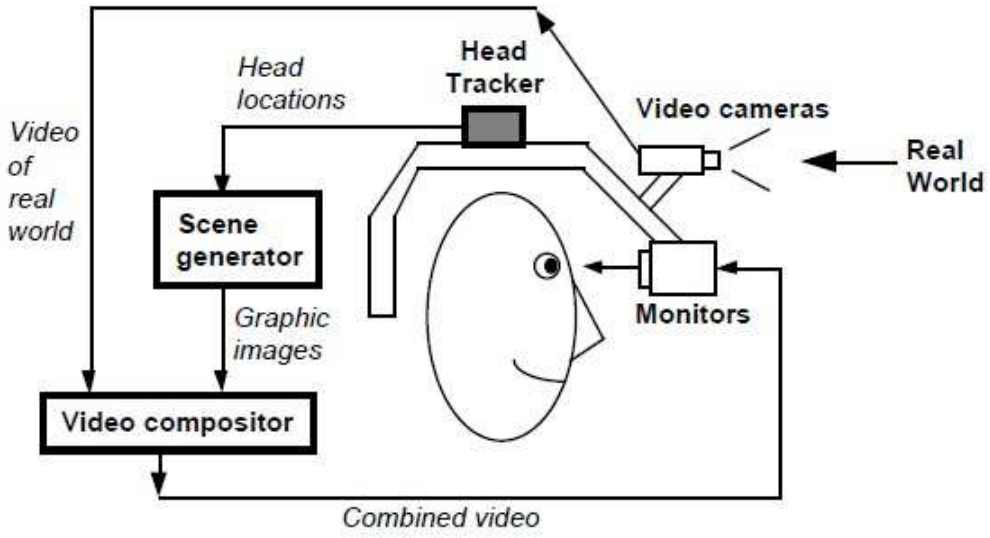

Abbildung 2.7: *Video see-through* HMD Konzept Diagramm ([28])

den kleinen Monitoren vor den Augen angezeigt. Somit hat der Nutzer keinen direkten Blick mehr zur realen Welt. Die Abbildung 2.7 zeigt ein Konzeptbeispiel eines Video see-through Displays. Für die Verschmelzung der realen Bilder mit den virtuellen Bildern gibt es verschiedene Möglichkeiten. Das bekannteste ist der sogenannte Green-Screen, bei welchem alle Bereiche des realen Bildes, welche einen bestimmten Grünton haben, mit virtuellen Bildern ausgetauscht werden. Wenn das System über Sensoren oder Kameras verfügt, welche eine Tiefenwahrnehmung besitzen, kann dies auch dazu benutzt werden die Hologramme möglichst real in das Bild einzuarbeiten [28].

Monitor-based configurations

Der Unterschied zwischen *monitor-based configurations* und *video see-through* HMD liegt darin, dass der Nutzer kein Headset trägt, sondern die Monitore statisch oder mobil eigenständig sind. Die Funktionsweise ist dieselbe wie bei video see-through HMD, nur dass der Winkel nicht aus der Sicht des Nutzers ist [28]. Beispiele hierfür wurden bereits in Kapitel 2.1 mit Fußball im Fernsehen oder *Pokemon Go* auf dem Smartphone erläutert. Abbildung 2.8 verdeutlicht solch ein Konzept.

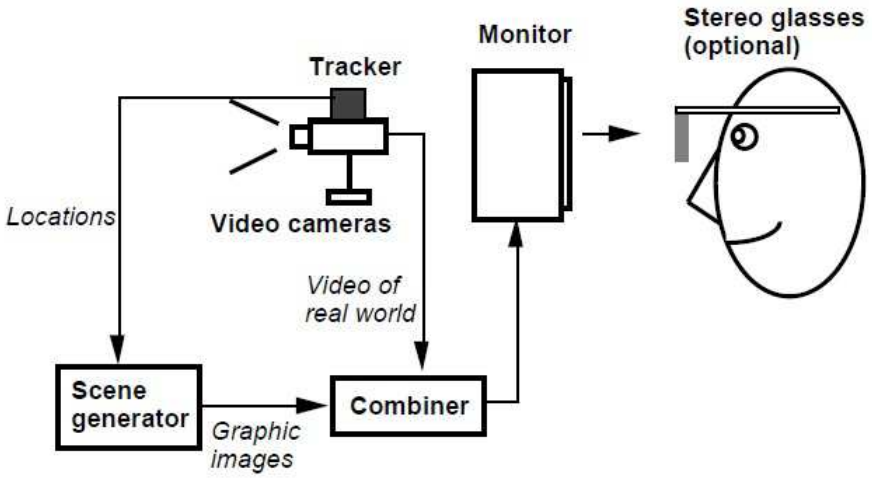

Abbildung 2.8: *Monitor-based* AR Konzept Diagramm ([28])

Verdeckungsberechnung

Die Verdeckungsberechnung ist ein sehr wichtiger Teil von AR-Systemen. Wenn Hologramme nicht richtig in den 3D-Raum eingearbeitet werden, verdecken diese meist reale Objekte, obwohl sie räumlich hinter ihnen liegen müssten. Dies zerstört die räumliche Wahrnehmung für den Nutzer eines solchen Systems. Gut zu sehen ist dies in Abbildung 2.9. Hier müsste die virtuelle Kaffeetasse hinter der Hand liegen, um eine gute Immersion zu schaffen. Wenn die Verdeckungsberechnung allerdings Fehlerhaft ist verdeckt die Tasse die Hand und es wirkt somit unreal. Es ist sehr herausfordernd eine gute Verdeckung zu kreieren und meist nur möglich mittels mehrerer Kameras, beziehungsweise Trackingsystemen [34]. Es gibt verschiedene Ansätze mit diesem Problem umzugehen. Die beste Lösung ist tiefensensitive Hardware oder vordefinierte statische Räume zu nutzen. Da dies nicht immer möglich ist, gibt es Versuche andere Wege zu gehen. Mit Hilfe von Geolokations Daten, werden im Freien geometrische Modelle der Gebäude erstellt und welche für die Verdeckungsberechnung auf Smartphones genutzt werden. Da Smartphones noch nicht über Tiefenkameras verfügen, ist dies eine passable Möglichkeit das gewünschte Ergebnis dennoch zu erreichen [16]. Eine gute Verdeckung kann nur erreicht werden, wenn die Registration in AR-Systemen gut funktioniert. In Kapitel 2.3.3 wird erläutert, welche Probleme es bei der Registration geben kann und wodurch diese hervorgerufen werden.

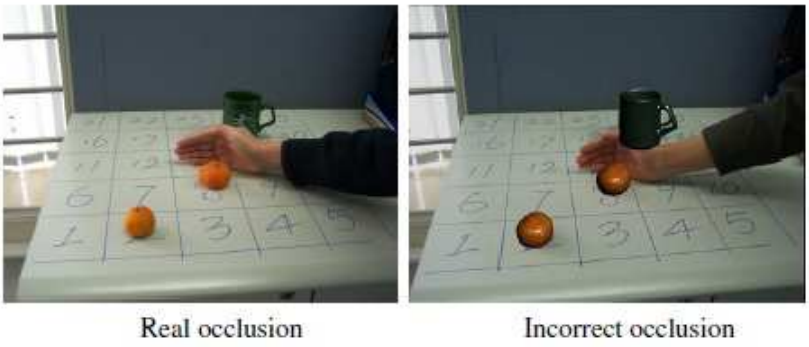

Abbildung 2.9: Beispiel einer inkorrekten Verdeckung ([34])

2.3.2 Eingabegeräte

Damit der Nutzer mit den virtuellen Gegenständen interagieren kann, bedarf es Möglichkeiten der Eingabe für ein solches System. Drei wichtige Arten der Eingabe sind die Eingabe per Blick, die Eingabe per Hand beziehungsweise Gesten und die Spracheingabe.

Eingabe per Blick

Die Blickerfassung kann bei allen Arten von AR-Systemen erfolgen. Mit Hilfe von Kameras wird die Blickrichtung mit einem Raypicking Algorithmus errechnet und kann dann genutzt werden, um zu ermitteln, ob der Nutzer etwas direkt anblickt beziehungsweise fokussiert. Wenn der Blick auf einem Objekt verharrt, kann nach einiger Zeit angenommen werden, dass der Nutzer das Objekt selektieren möchte, was dann ein Beispiel einer Eingabe wäre [30].

Handeingabe

Mit den Händen lässt sich gestikulieren, dies kann als Eingabe verwendet werden. Die älteste Möglichkeit sind Datenhandschuhe, welche der Nutzer anziehen kann. Dieser Handschuh kann dann zum Beispiel mit Hilfe von Glasfasern, bei denen eine Leuchtdiode Licht einstrahlt, messen wie stark das Licht ist, welches am anderen Ende der Faser ankommt.

Wenn also der Finger, und somit die Glasfaser, sehr doll gekrümmt ist, kommt am anderen Ende vergleichsweise wenig Licht an. Somit lassen sich zwar eine kleine Anzahl von Gesten messen, allerdings sind diese sehr banal. Wie in Kapitel 2.1 erwähnt, gibt es Systeme wie die *Leap Motion*, welche die Hände ohne Handschuhe scannen und die Gesten interpretieren können. Somit lassen sich mehr und unterschiedlichere Gesten (auch mit beiden Händen gleichzeitig) darstellen. Dies hat Vorteile gegenüber klassischen Controllern, wie eine Maus, welche aus der Hand fallen könnte oder Handschuhen, welche nicht passen könnten. Zudem ist die Interaktion mit dem AR-System somit intuitiv und es muss nicht erst eine komplizierte Steuerung erlernt werden [31].

Sprachsteuerung

Eine weitere Art der Eingabe ist es, Befehle an das System per Sprache zu übermitteln. Hierbei ist es wichtig, dass der Nutzer sich nicht zu sehr auf die Befehle konzentrieren muss. Wenn die Spracheingabe zu komplex ist oder in Kombination mit dem Blick erfolgen muss, kann es zu vermeidbaren Fehlern kommen. Am besten sind einfache Eingaben, die sogenannten Kommandos. Im Vergleich zur freien Sprache werden bei Kommandos Wörterbücher hinterlegt, die dann per Sprachmuster zugeordnet werden können. Wenn freie Sprache als Eingabe genutzt werden soll, zieht das einen sehr großen analytischen Teil der Interpretation mit sich [30]. Es gibt schon viele weit entwickelte Sprachsysteme, die aktuell benutzt und stetig weiterentwickelt werden. Die bekanntesten sind *Cortana* von Microsoft oder der *Google Assistent*.

Interaktion

Vorangehend wurden verschiedene Eingabemöglichkeiten vorgestellt, damit im folgenden aufgezeigt werden kann, welche Interaktionen sich damit bewerkstelligen lassen, wird im folgenden gezeigt. Die Interaktion muss intuitiv erfolgen können, um eine möglichst bequeme Bedienung und somit Immersion zu erzeugen. Es gibt verschiedene Interaktionstechniken, die wie folgt unterteilt werden: Selektion, Manipulation, Bewegung, Systemeinstellungen und Symbolische Eingaben [30].

Selektion:
Die Selektion ist eigentlich Teil der Manipulation. Wenn eines oder mehrere Hologramme selektiert werden sollen, kann dies mit Hilfe eines Cursors geschehen, welcher mit dem Blick oder der Hand gesteuert wird. Eine andere Art der Selektion ist, direkt mit der

Hand die Hologramme zu ergreifen, sofern diese in Reichweite sind. Es eigenen sich also die Eingabe per Blick und per Hand, um ein bestimmtes Objekt zu selektieren [30].

Manipulation:
Um ein Objekt zu manipulieren, muss es zunächst selektiert werden. Dann kann es in seiner Position und Orientierung verändert werden. Je nachdem mit welcher Methode das Objekt selektiert wurde, kann es durch eben diese Methode manipuliert werden. Wurde es zum Beispiel durch eine Handgeste gegriffen, kann der Nutzer das Objekt mit Hilfe seiner Hand an die gewünschte Position bewegen oder rotieren [30].

Bewegung:
Im Gegensatz zur VR ist die Bewegung in der AR relativ simpel. Da der Nutzer sich mit dem AR-System meist frei bewegen kann, benötigt es keine externe Eingabe durch Controller, um die aktuelle Position des Nutzers zu verändern. Meist sind also die Bereiche, in denen der Nutzer sich bewegen kann, nur durch die realen physikalischen Gegebenheiten beschränkt [30].

Systemeinstellungen:
Systemeinstellungen können über Menüs oder Spracheingabe erfolgen. Da sich bei klassischen Menüs die Frage stellt, wo diese zu platzieren sind, damit der Nutzer in seiner Sicht nicht eingeschränkt ist, liegt der Vorteil hier offensichtlich in der Spracheingabe [30]. Es gibt allerdings auch Möglichkeiten die Menüs sauber in die Umgebung einzubauen und wie bei der HoloLens ganze virtuelle Fenster wie Plakate an die Wand zu bringen oder diese mitten im Raum lichtdurchlässig zu gestalten.

Symbolische Eingaben:
Für symbolische Eingaben werden am klassischen Computer normalerweise Tastatur und Maus benutzt. Da dies bei AR-Systemen nicht so einfach zu bewerkstelligen ist, gibt es auch hier Alternativen. Eine virtuelle Tastatur, wie bei Smartphones, lässt sich einblenden oder es lassen sich direkt mit dem Finger Zeichen in den virtuellen Raum zeichnen [30]. Am effektivsten ist hier die Spracheingabe, bei der vorher ein gewisses Vokabular definiert wurde, mit Kombination von Gesten, die mit der Hand geformt werden können. Zwar müssen die Gesten im Vorfeld erlernt werden, ähnlich wie bei Gebärdensprache, aber meist sind diese Gesten intuitiv und somit sehr einfach gestaltet [7].

2.3.3 Tracking

Als Tracking wird die Lagebestimmung des Nutzer eines AR-Systems, sowie die von wichtigen Gegenständen bezeichnet. Dies ist ein wichtiger Bestandteil von AR-Systemen, denn ohne dies wüsste das System nicht, was es anzeigen soll beziehungsweise wo bestimmte Hologramme erscheinen und wieder verschwinden sollen. Es gibt eine Vielzahl von unterschiedlichen Trackingsystemen, die bei AR-Systemen zum Einsatz kommen können. Die am häufigsten eingesetzten Technologien sind *inertial Tracking* und das *optische Tracking* [30].

Inertial Tracking

Bei elektromechanischem Tracking werden kleine Sensoreinheiten angewendet, welche dann die Neigung beziehungsweise Rotation und die Beschleunigung messen und so eine Position liefern, die relativ zu ihrem Startpunkt liegt. Bei einem Beschleunigungsmesser wird die wirkende Trägheitskraft auf eine Testmasse bestimmt und so berechnet ob eine Geschwindigkeitszuname oder -abnahme stattfand. Die Beschleunigung wird dann zu einer Verschiebung, auch *Translation* genannt, umgerechnet und auf die aktuelle Position angewendet. Bei einem Neigungsmesser, zum Beispiel einem Gyroskop, wird das Drehmoment beziehungsweise die Kippbewegung gemessen. Wenn der Messer aus seiner Ursprungsposition bewegt wird, kann dann der Winkel ermittelt werden und auf die aktuelle Position angewendet werden. Es gibt noch weitere Arten von Messsensoren, welche aber für AR-Systeme ungeeignet sein können, denn wie zum Beispiel Lasergyroskope oder Drehbeschleunigungsmesser, sind diese zu groß oder zu teuer [30]. Hingegen sind die bereits erwähnten Sensoren mittlerweile in ihrer Form sehr klein und Preiswert und werden aktuell schon standardmäßig in Smartphones verbaut. Zudem sind diese Sensoren unabhängig von äußeren Einflüssen, wie Lichtverhältnisse, Magnetfeldern oder bestimmten Bewegungen, welche Markierungspunkte abdecken könnten [31]. Diese Art von Sensoren sind frei von Bezugspunkten in der normalen Welt. Der Vorteil von externen Systemen, für die Lagebestimmung unabhängig zu sein, ist gleichzeitig auch ein Nachteil. Ohne Ursprungspunkt, auf den dann die Sensormessungen angewendet werden können, gibt es keine Relation der Position im Raum. Zudem könnte nach längerer Laufzeit ein sogenannter Drift (also eine Verschiebung) entstehen, denn diese Systeme liefern immer ein Delta zum vorherigen Wert und sind somit nicht absolut genau [30].

Optisches Tracking

Optisches Tracking funktioniert mit Hilfe von Licht, welches sowohl im sichtbaren Bereich, als auch im Infrarotbereich liegen kann. Die am häufigsten vorkommende Umsetzung ist mit Infrarotlicht, denn es hat den Vorteil, dass es für den Nutzer nicht sichtbar ist und somit normale Lichtverhältnisse nicht beeinflusst werden. Zudem ist Infrarotlicht fast unabhängig vom Umgebungslicht und es entstehen somit weniger Störungen. Zwar muss zur Überprüfung, ob das Infrarotsystem funktioniert, immer ein Kamerabild verwendet werden, aber die Vorteile von Infrarotlicht zu Lichtblitzen überwiegen. Optisches Tracking wird unterteilt in Tracking mit und ohne Marker [30].

Markertracking

Beim Tracking mit Markern, werden Marker an dem Nutzer angebracht, welche entweder ein Muster aufweisen oder reflektierend sind. Bei beiden Möglichkeiten erkennt das System wo genau die Marker liegen und ermittelt daraus die Position und Orientierung.

Reflektierende Marker

Diese Marker bestehen aus retroreflektierendem Material und werfen so das Licht in die Richtung zurück, aus dem es kommt. Wenn von einem Marker, meist am Nutzer angebrachte Kugeln, die 3D-Position ermittelt werden soll, benötigt es mindestens 2 Kameras. Dann werden Blitz erzeugende Geräte verwendet, welche an den Kameras angebracht und mit diesen synchronisiert sind. Durch eine Raumkalibrierung der Kameras wird dann mit Hilfe des aufgenommenen Bildes und dem hellsten Punkt innerhalb des Bildes die Position des Markers bestimmt. Wenn mehrere Marker benutzt werden, kann im Vorfeld definiert werden, wo der Ursprung des Koordinatensystem liegt, in welchem sich die Marker befinden und wie die Marker dort angeordnet sind. Dadurch lassen sich alle Positionen der Marker neu zuordnen und es entstehen Rotation und Bewegung. Systeme mit reflektierenden Markern haben die Vorteile, dass der Aufbau meist simpel und frei von elektromagnetischen Feldern ist. Die Nachteile liegen darin, dass Marker durch Bewegungen verdeckt sein könnten oder es schwierig sein kann, diese Marker am Nutzer oder Objekt anzubringen.

Marker mit Mustern

Im Gegensatz zu reflektierenden Markern, werden meist flache Marker verwendet, welche ein bestimmtes Muster aufweisen. Wenn der Marker viereckig ist, kann ein Kamerabild die Abgrenzung und die Entfernung der Eckpunkte erkennen und somit nicht nur die Position, sondern auch die Rotation des Markers bestimmen. Ein Marker mit Mustern könnte zum Beispiel ein QR-Code sein, welcher an einem Objekt befestigt wird. Durch ihre flache Struktur können sie meist nicht aus allen Winkeln sichtbar sein. Um dies zu umgehen können 6 dieser Marker zu einem Würfel zusammengefügt werden und somit aus jeder Lage erkannt werden. Die Nachteile wie bei reflektierenden Markern bleiben aber bestehen und zudem kommt noch, dass diese Art stark von der Beleuchtung abhängig ist und nicht abgedeckt sein dürfen. Der einzige Vorteil dieser Marker liegt darin, dass sie kostengünstig und einfach zu reproduzieren sind [30].

Markerloses Tracking

Bei markerlosem Tracking werden natürliche Merkmale der zu trackenden Objekte verwendet, um diese zu erkennen. Bei dieser Art des Trackings gibt es sehr viele verschiedene Techniken. Bei einigen Systemen kommen vorher eingescannte 3D-Modelle von Objekten zum Einsatz, welche dann als Modell verwendet werden können, um im Kamerabild Vergleichspunkte zu finden. Andere Systeme versuchen mit Hilfe von Farb- und Kontrastunterschieden oder Ecken und Kanten bestimmte Muster zu finden, welche die gleichen Merkmale des Objekts sind, welches gefunden werden soll. Anhand dieser Merkmale lässt sich dann die Position des Objektes relativ zur Kamera feststellen. Auch bei diesem System bedarf es einer sehr guten Beleuchtung und auch die zu erkennenden Merkmale müssen dynamisch durch immer neu zu bestimmende Grenzwerte evaluiert werden. Wenn es also zu wenig Anhaltspunkte für dynamisches Erkennen gibt, zum Beispiel bei einer rein weißen Wand oder wenn sich das Objekt verformt, entstehen Probleme bei dieser Art des Tracking [30].

Zusätzliche Trackingsysteme

Neben den bereits erwähnten Systemen gibt es noch weitere, welche allerdings nicht in dieser Häufigkeit eingesetzt werden. Für AR-Systeme nicht relevante Systeme wären das magnetische Tracking und das mechanische Tracking. Das mechanische Tracking

wurde in Kapitel 2.2 beim Damoclesschwert erwähnt. Bei diesem System wurden Stangen eingesetzt um die Position zu ermitteln. Dies waren die Anfänge von AR-Systemen, diese Art ist heutzutage allerdings obsolet. Selbst Laufzeitbasierte Trackingssysteme und die bekannteste Art davon, das GPS-Tracking, kommen bei aktuellen AR-Systemen eher weniger zum Einsatz [30].

Registration

Wenn die Registration nicht perfekt funktioniert, indem zum Beispiel die Präzision nicht stimmt oder es zu Synchronisationsproblemen kommt, wird der Bezug von Hologrammen zur realen Welt verzerrt. Dies bedeutet, dass Hologramme verschwinden oder an willkürlicher Stelle wieder auftauchen könnten oder dass die Ausrichtung nicht stimmt. Somit wären Interaktionen zu diesem Objekt nicht möglich worunter die Immersion leidet. Diese Fehler werden unter dem Begriff des *Registration Problems* zusammengefasst [28]. Es gibt vier Ursachen, die einen *Registration Error* auslösen können:

1) Die Ursprungspunkte der Koordinatensysteme der realen und virtuellen Welt stimmen nicht überein. Dies hat zur Folge, dass alle Objekte leicht versetzt erscheinen können.

2) Der virtuelle Ursprungspunkt eines Objektes ist nicht der gleiche wie der reale Ursprungspunkt. Dadurch können einzelne Objekte von ihrer eigentlichen Position verrutscht sein.

3) Die virtuelle Kameraposition stimmt nicht mit der realen Kameraposition überein. Dies kann von Anfang an passieren oder während des Bewegens des Nutzers. Dadurch stimmt die Sicht auf Hologramme nicht mehr.

4) Das virtuelle Bild wird nicht korrekt auf das Kamerabild abgebildet. Somit wäre die Bildposition und damit auch alle Hologramme an der falschen Position. [8]

Im Gegensatz zu VR sind die Auswirkungen von *Registration Errors* im AR-System für den Nutzer deutlicher sichtbar. Wenn eine virtuelle Hand im VR-System nicht mit der Position der realen Hand übereinstimmt, kann dies dem Nutzer nicht auffallen, denn optisch sieht er keinen Unterschied. Hingegen beim AR-System sieht der Nutzer, dass die Position der realen und virtuellen Hand nicht übereinstimmen indem er sie quasi doppelt sieht. Unter anderem kann eine sogenannte *Motion Sickness* auftreten, wenn das Bild verzögert dargestellt wird und nicht mit den Bewegungen des Nutzers übereinstimmt.

Dies ist vergleichbar mit der Seekrankheit, welche eine Person auf einem Schiff erleiden kann. Es ist schwierig *Registration Errors* ausreichend zu verhindern, da es eine Vielzahl von Fehlerquellen geben kann, die diese hervorrufen können und man eine hohe Präzision der Sensoren benötigt, diese zu beheben [28].

2.4 Recommender System

Ein *Recommender System* benutzt gesammelte Daten über den Nutzer und seine Umgebung, um eine Empfehlung vorzuschlagen, dessen Wichtigkeit für ihn die höchste Präferenz aufweist. Seit ein paar Jahren gibt es solche Systeme immer häufiger, da immer mehr personifizierte Daten gesammelt werden können. Allerdings unterscheiden sich Recommender Systems für AR leicht von den mittlerweile gebräuchlichen bekannten Systemen. Folgende Eigenschaften sind besonders wichtig:

- Location: Der Vorschlag aus dem *Recommender System* muss zur aktuellen Position beziehungsweise dem Standort passen.
- Timing: Der Zeitpunkt für die Vorschläge ist entscheidend, denn am Sonntagmorgen sind Vorschläge für Restaurants zum Abendessen sinnlos.
- Cold start: Neue Nutzer sind für solche Systeme ein Problem, da ohne genügend gesammelte Daten meist keine guten Vorschläge generiert werden können.
- Immediate feedback: Recommender Systems müssen schnell auf das Verhalten des Nutzers reagieren können. Diese können uninteressante Vorschläge sofort ausblenden und so muss die Liste an Vorschlägen dauernd aktualisiert werden. [33]

Ein gutes Recommender System für AR muss also sowohl Daten über seinen Nutzer sammeln, als auch alle Sensoren des AR-Systems benutzen, wie zum Beispiel das GPS Signal eines Smartphones, um den Standort zu ermitteln.

2.5 Anwendungsszenarien

Im folgenden Abschnitt werden zwei Szenarien beschrieben, bei denen die positionsbezogene Informationsanzeige in der AR Anwendung finden könnte. Danach wird entschieden, welches Szenario die Vorgabe für den zu entwickelnden Prototyp sein wird.

Student beim orientieren in der Universität

Ein Student geht durch das Universitätsgebäude und versucht sich zurechtzufinden. Er versucht einen Raum zu finden in dem er ungestört lernen kann. Als Orientierungshilfe nimmt er seine Augmented-Reality Brille und läuft einen Flur hinunter. Normalerweise würde ihm die Brille anzeigen, wo der nächstgelegene Standort einer neuen Information ist, aber in diesem Fall ist eine optische Unterstützung, die dies anzeigen würde, nicht von Nöten. Aufgrund dessen, da der Student weiß, dass sich bei jeder Tür eine solche Information befindet, muss er einfach nur den Flur entlang und von Tür zur Tür gehen. Er sieht sofort ob ein Raum belegt ist, da die Tür farblich umrahmt wird, wenn er davor steht. Rot bedeutet, der Raum ist belegt und so kann er direkt zum nächsten Raum weitergehen. Er könnte sich allerdings auch genauere Informationen zu diesem Raum anzeigen lassen. Mit Hilfe der Gestensteuerung blättert er, wie bei einer Zeitschrift, durch eine virtuelle Anzeige, die neben der Tür an der Wand hängt. Dort werden ihm Informationen zur Raumbelegung des gesamten Tages oder der Woche angezeigt und auch, um was für ein Raum es sich handelt. Da der Student gerne an einem Computer lernt, weiß er direkt, dass er einen Raum sucht, bei dem über der Tür ein Laptop abgebildet ist. Wenn er vorher einstellt, dass er nur solche Räume sucht, werden ihm auch nicht belegte Räume rot markiert, wenn in ihnen keine Computer vorhanden sind. So kann er direkt zu einem Raum gehen, welcher grün markiert und somit passend für ihn ist, ohne Vorlesungen zu stören oder vor verschlossenen Türen zu stehen.

Recommender Systems im Einkaufszentrum

Ein Ladenbesitzer eines Kiosk in einem Einkaufszentrum kommt morgens in seinen Laden und überlegt welche Produkte er heute günstiger anbieten will. Er entscheidet sich für Kaffee, Orangensaft und belegte Brötchen und gibt dies in seiner App am Smartphone ein. Zudem stellt er ein, dass sich das Angebot ab 12 Uhr ändert, sodass statt den Brötchen, Hotdogs im Angebot sind. Nun weiß er, dass diese Produkte vor dem Laden als Hologramme in der AR für Kunden dargestellt werden, wenn diese am Laden vorbeilaufen. Dies setzt natürlich voraus, dass potenzielle Kunden mit einem AR fähigem Gerät, wie zum Beispiel einem Head-Mounted-Display durch das Einkaufszentrum gehen. Wenn also ein Kunde eigentlich nur an dem Laden vorbeigehen wollte, dann aber ein sich drehendes Hologramm einer dampfenden Kaffeetasse sieht, könnte dieser sich umentscheiden, weil er nicht nur durch diesen anregenden Anblick eine Lust auf Kaffee

bekommt, sondern auch direkt weiß, dass dieser günstiger zu bekommen ist. Zudem ist diese Art der Werbung individualisierbar, denn die AR-Geräte sammeln Informationen über ihren Träger. Dies reicht von Alter und Geschlecht, bis hin zu Vorlieben in Bezug auf Essen und Trinken. Die Möglichkeit für Recommender Systems ist hier fast grenzenlos. Wenn also zum Beispiel ein Pärchen an diesem Kiosk vorbeigeht, sieht sie das Hologramm für den Orangensaft und er den für Kaffee. So bekommen beide personalisierte Werbung und einen besseren individuellen Anreiz zum Kaufen.

2.6 Anwendungsfälle

Im Abbildung 2.10 werden die sich aus den beiden Szenarien ergebenden Anwendungsfälle beschrieben. Aufgelistet werden die Bezeichnung des Anwendungsfalls, eine kurze Beschreibung, die Abhängigkeiten zu anderen Anwendungsfällen, sowie Vor- und Nachbedingungen. Da die Anwendung nur für einen Nutzer ausgelegt ist, gibt es keine Spezifikation über diesen, da es sich dadurch immer auf dieselbe Person bezieht.

Name	Beschreibung	Verwendete Anwendungsfälle	Vorbedingung	Nachbedingung
Starten der App	Die App wird gestartet und alle benötigten Informationen werden geladen		Das AR-System ist betriebsbereit	Die App ist geladen und die ersten Informationen werden angezeigt
Ankerpunkte werden geladen	Wenn die App gestartet ist, werden die Ankerpunkte aller Objekte geladen und dementsprechend platziert	Starten der App	Gültige Ankerpunkte müssen vorliegen	Alle Objekte werden so im Raum platziert, wie die Ankerpunkte es vorgeben
Fokussierung eines Objekt	Ein Objekt welches angeschaut wird, wird fokussiert		Ein Objekt muss angeschaut werden	Das Objekt wird als aktuell manipulierbares Objekt intern markiert
Klicken des angezeigten Objekts	Wird ein fokussiertes Objekt angeklickt, verändern sich seine Eigenschaften	Fokussierung eines Objekts	Ein Objekt wird fokussiert und eine Tap-Geste wird ausgeführt	Das Objekt verändert sich entsprechend der hinterlegten Aktion bei einer Tap-Geste
Spracheingabe für ein angezeigtes Objekt	Wird ein Objekt fokussiert und per Spracheingabe ein Befehl gegeben, wird dieses Objekt entsprechend des Befehls manipuliert	Fokussierung eines Objekts	Ein Objekt wird fokussiert und ein Sprachbefehl wird gesprochen	Das Objekt wird entsprechend des Befehls manipuliert
Wechsel des angezeigten Objektes	Wenn sich das zum Benutzer am nächsten gelegene Objekt ändert, wird dieses ausgeblendet und das neue angezeigt		Die Position des Benutzers zu den Objekten muss verfügbar sein	Ausblendung des alten und Einblenden des neuen Objekts
Konfigurations Modus	Durch eine Spracheingabe kann der Konfigurationsmodus aufgerufen werden, bei dem alle Objekte angezeigt und manipuliert werden können		Der Befehl muss per Spracheingabe erfolgen	Alle Objekte werden angezeigt und befinden sich im Manipulations-modus
Selektion eines Objekts	Ein Objekt wird mit der Selektion-Hand-Geste selektiert	Fokussierung eines Objekts, Konfigurations-Modus	Ein Objekt muss fokussiert sein und die entsprechende Geste muss durchgeführt werden	Das Objekt wird intern selektiert
Drag and Drop eines Objekts	Durch eine Bewegung der Hand kann das Objekt im Raum verschoben werden	Selektion eines Objekts	Der Konfigurationsmodus muss aktiviert sein und das entsprechende Objekt muss selektiert sein	Das Objekt wird durch den Raum bewegt und dort platziert, wo die Drag and Drop Aktion aufhört
Manipulation des Objekts	Durch das Selektieren der Eckpunkte des Objektes im Konfigurations-Modus, kann das Objekt rotiert und skaliert werden	Konfigurations-Modus	Der Konfigurationsmodus muss aktiviert sein und eine der Ecken des Objekts muss selektiert sein	Das Objekt verändert sich in seiner Rotation und Größe entsprechend der Bewegung

Abbildung 2.10: Die benötigten Anwendungsfälle (Quelle: Eigene Darstellung)

2.7 Anforderungsanalyse

Im folgenden Abschnitt werden die Anforderungen an die Anwendung betrachtet. Diese Betrachtung wird in funktionale und nicht funktionale Anforderungen unterteilt. Funktionale Anforderungen beschreiben was die Anwendung für Funktionalitäten haben muss. Dies hilft bei der späteren Entwicklung der Anwendung, auf diese Funktionalitäten hinzuarbeiten. Nicht funktionale Anforderungen sind Anforderungen, die bestimmen, in welcher Qualität die geforderten Funktionalitäten in der Anwendung vorhanden sein sollten.

Funktionale Anforderungen

Welche funktionalen Anforderungen an die Anwendung gestellt sind, werden durch die Anwendungsfälle spezifiziert, welche in Kapitel 2.6 schon aufgelistet wurden. Jeder dieser Anwendungsfälle muss von dem Nutzer durchführbar sein. Hologramme im virtuellen Raum sollen alle manipulierbar sein und mit dem Raum interagieren (Verdeckung, Kollision). Zudem sollen alle Gesten und Sprachbefehle erkannt werden können, unabhängig davon, wer der Nutzer der Anwendung ist. Die Leistung, sprich die Anzahl an Hologrammen und virtuelle Raumgröße, sind durch die Hardware limitiert.

Nicht funktionale Anforderungen

Nicht funktionale Anforderungen umfassen die Qualität der Anwendung. Wichtig hierbei sind die Bedienbarkeit, Performance, Fehlertoleranz, Skalierbarkeit, Wartbarkeit und Optik.

- *Bedienbarkeit*:
  Die Anwendung muss möglichst intuitiv bedienbar sein. Alle Sprachbefehle und Gesten müssen eindeutig und leicht anwendbar sein. Wenn der Nutzer erst lange ausprobieren oder Anleitungen lesen muss, leidet das Erlebnis. Zudem muss eine weit verbreitete Sprache benutzt werden. Hierbei eignet sich Englisch am besten. Somit wird gewährleistet, dass die Anwendung auch international genutzt werden kann. Ähnlich sollten auch die Gesten möglichst global verständlich sein. Zudem

sollten alle Befehle, auf welche die Anwendung reagiert, deutlich voneinander unterscheidbar sein. Dies sichert, dass der Nutzer nicht aus Versehen eine Falscheingabe tätigt, nur weil die Befehle sich ähnlich sind.

- *Performance*:
  Eine gute Performance ist entscheidend für die Qualität eines AR-Systems. Wenn eine zu hohe Latenz auftritt, können Hologramme nicht richtig manipuliert werden oder erscheinen an Positionen, an denen sie eigentlich nicht mehr sein sollten. Wie in Kapitel 2.3.3 erwähnt, könnte dadurch *Motion Sickness* auftreten. Zudem könnte die Immersion leiden, wenn sich Hologramme nur ruckartig bewegen oder schlecht manipulieren lassen. Auch wenn die in Kapitel 2.3.1 erwähnte Verdeckungsberechnung nicht richtig von der Anwendung berechnet wird, leidet die Immersion.

- *Fehlertoleranz*:
  Alle Eingaben, die der Nutzer tätigt, sollten mit einer gewissen Fehlertoleranz behandelt werden. Wenn ein Sprachbefehl oder eine Geste nicht genau ausgeführt werden kann, sollte die Anwendung nicht darauf reagieren. Falls die Anwendung erraten würde, um welchen Befehl es sich handelt, würden wiederum nur weitere Fehler entstehen. Da die Gesten aber möglichst simpel gehalten sind, sollte dies keine große Fehlerquelle darstellen. Zudem gibt es aktuell kein perfekt funktionierendes AR-System, denn wie in Kapitel 2.3 erwähnt, gibt es in den Bereichen Eingabegeräte, Ausgabegeräte und Tracking bekannte Probleme, welche bis heute nicht komplett gelöst werden konnten. Wenn zum Beispiel die Umgebung nicht zu hundert Prozent richtig erkannt werden kann, ist es möglich, dass die Verdeckung unvollständig ist.

- *Skalierbarkeit und Wartbarkeit*:
  Die Anwendung sollte ohne großen Aufwand erweiterbar sein. Die Möglichkeit, neue Gesten und Sprachbefehle einzubauen oder zu bearbeiten, soll genauso einfach sein, wie die Anwendung zu benutzen. Auch neue Hologramme und Texte sollen mit wenig Aufwand ergänzt beziehungsweise verändert werden können.

- *Optik*:
  Da die Anwendung keine großartigen Menüs zur Verfügung stellen wird, bezieht sich der Punkt Optik nur auf die angezeigten Hologramme. Hierbei sollte es möglich sein, Hologramme und Animationen zu verwenden, welche möglichst real aussehen, ohne dass die Performance leidet. Zudem sollten sich die Hologramme wie echte Objekte in der Welt verhalten.

Zusammenfassung

Die zu entwickelnde Anwendung muss über jegliche aus den Anforderungen ermittelte Funktion verfügen. Es soll eine möglichst gute Immersion entstehen, damit der Nutzer nicht das Gefühl hat, die angezeigten Hologramme sind nur über die reale Welt gelegt. Dies bedeutet, die Anwendung muss performant sein, mit möglichst wenig Fehlern ausgeführt werden und leicht zu bedienen sein. Ansonsten würde die Benutzung dieser Anwendung nur stören und nicht den gewünschten Effekt erzielen, eine Unterstützung in Bereichen zu sein, in denen zusätzliche Informationen hilfreich wären. Zudem muss die Anwendung für die Wiederverwendbarkeit leicht skalierbar und wartbar sein. Die Software wird nur für die HoloLens entwickelt. Für die Entwicklung werden ein Computer mit *Windows 10* und *Visual Studio* für die eigentliche Software, *Unity* für die Anordnung der virtuellen Welt, *Blender* zum Erzeugen der Hologramme und die HoloLens zum Ausführen der Software benötigt.

3 Microsoft HoloLens

Da sich diese Arbeit mit einer lokationsbezogenen Objektanzeige im AR Raum mit der HoloLens beschäftigt, wird in diesem Kapitel aufgezeigt, was die technischen Gegebenheiten dieses HMD sind und warum sich für die HoloLens entschieden wurde. Die HoloLens ist eines der derzeit weit verbreitetesten HMD, welches häufig im professionellem Umfeld genutzt wird. Im Gegensatz zu anderen AR-Systemen, die eher im Bereich des Entertainment eingesetzt werden, möchte die HoloLens in allen Bereichen Fuß fassen. Die erste Generation dieses Gerätes, welches im Zuge dieser Arbeit benutzt wurde, kam 2016 für einen Preis von 3000 US-Dollar auf den Markt. Anfang 2019 erschien die zweite Generation der HoloLens [15]. Im Zuge dieser Arbeit wurde die erste Generation der HoloLens genutzt.

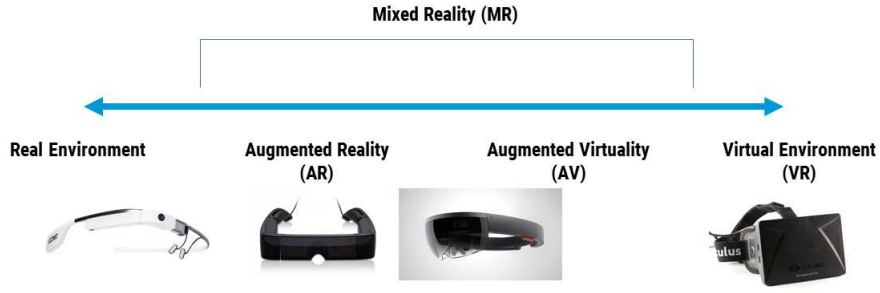

Abbildung 3.1: Einordnung der HoloLens in das RVC [4]

3.1 Hardware

Die HoloLens ist ein autarker Computer und um zu verstehen, wie sie arbeitet und mit bestimmten Problemen der AR umgeht, wird zunächst gezeigt, welche Hardware in ihr verbaut ist.

3.1.1 Optik

Die HoloLens verfügt über zwei durchsichtige holografische Linsen (Wellenleiter), 2 HD 16:9 Lichtgeneratoren und einer automatischen Pupillendistanzkalibrierung. Die Auflösung beträgt 2,3 Millionen Lichtpunkte mit einer holografischen Dichte von 2500 Radianten (Lichtpunkte pro Radiant).

3.1.2 Sensoren

Eine Inertiale Messeinheit, vier umgebungserfassende Kameras, eine Tiefenkamera, eine 2-Megapixel Foto / HD Videokamera, ein MR-Erfassungssensor, vier Mikrofone und ein Umgebungslichtsensor sind an der HoloLens angebracht, um so bestmöglich die Umgebung zu erfassen. Somit besitzt die HoloLens alle wichtigen Sensoren für ein AR-System. Zudem besitzt sie eingebaute Lautsprecher, diverse Knöpfe als Bedienelemente, Wi-Fi-Adapter, Bluetooth-Adapter, 64 Gigabyte internen Speicher, 2 Gigabyte Arbeitsspeicher und ist passiv gekühlt.

Somit kann die HoloLens Umgebungsgeräusche erkennen, hat eine Blickerfassung (Gaze), kann Gesten erkennen und hat eine Spracherkennung [32].

3.2 Blick/Gaze

Die einfachste Form der Eingabe bei der HoloLens ist der Blick. Dieser geschieht automatisch, sofern man das HMD trägt und seine Augen geöffnet hält. Durch die Position des Kopfes wird festgelegt, in welchem Fenster sich der Blick bewegen kann. Vergleichbar ist dies mit dem Mauszeiger bei einem Computer, denn dort wo der Blick hinfällt, wird ein kleiner Cursor angezeigt. Die Blickrichtung der Augen wird erkannt und eine imaginäre

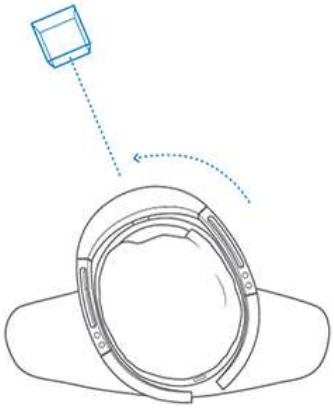

Abbildung 3.2: Darstellung des *Gaze* Prinzips [22]

Linie in den Raum geworfen (siehe Abbildung 3.2). Dort wo die Linie mit einem virtuellen oder realen Objekt kollidiert, kann der Cursor angezeigt werden und das Hologramm kann mit Gesten selektiert werden [22].

3.3 Gesten/Gestures

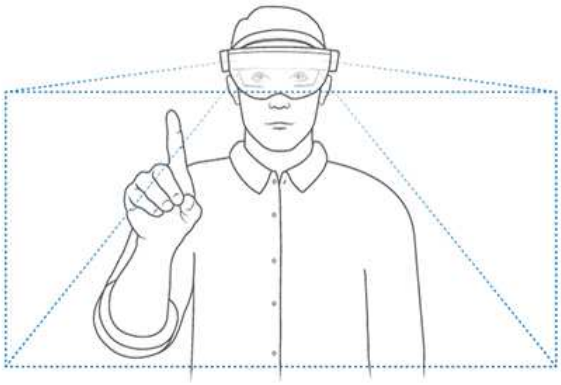

Abbildung 3.3: Darstellung des *Gesture Frame* [22]

Damit die HoloLens Gesten erkennt, müssen diese im sogenannten *Gesture Frame* ausgeführt werden. In Abbildung 3.3 ist das *Gesture Frame* dargestellt und wenn die HoloLens die Hände innerhalb dieses Fensters registriert, wird der Cursor auf dem Hologramm zu einem Ring und symbolisiert so, dass die HoloLens bereit ist, Gesten zu erkennen. Dieses

Fenster liegt immer direkt vor dem HMD, was bedeutet, wenn der Kopf gedreht wird, dreht sich das Fenster mit, auch wenn es der Oberkörper des Nutzers nicht tut. Die wichtigsten Gesten, welche die HoloLens erkennt sind:

- *Bloom*:
  Die *Bloom* Geste öffnet das Startmenü, pausiert somit auch Programme und schließt das Startmenü wieder. Um diese Geste auszuführen hält man die Hand mit der Handfläche nach oben und nimmt die Fingerspitzen zusammen. Dann öffnet man die Finger und simuliert so das aufblühen einer Blume (Daher der Name *Bloom*).

- *Air Tap*:
  In Kombination mit dem Blick kann mit dieser Geste ein Hologramm oder Menüpunkt ausgewählt werden. Man guckt das entsprechende Hologramm an und bewegt den Zeigefinger nach unten, als würde man das Hologramm antippen wollen.

- *Tap and Hold*:
  Beim *Tap and Hold* verfährt man wie bei einem *Air Tap*, nur dass man den Zeigefinger nicht wieder hoch nimmt, sondern die Bewegung in der Mitte endet und der Finger in dieser Position verweilt. Dadurch wird das Hologramm selektiert und man kann verschiedene Aktionen ausführen: Bewegen, Rotieren, Skalieren, Zoomen, und Scrollen.

Mit diesen drei Gesten lässt sich die komplette HoloLens steuern. Eine Beispielgeste ist in Abbildung 3.4 dargestellt, es handelt sich hierbei um den *Air Tap* [22].

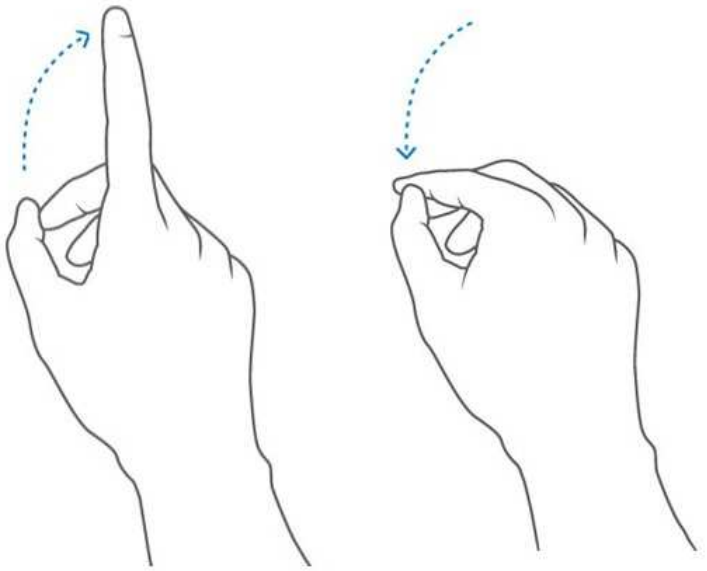

Abbildung 3.4: Beispiel einer *Air-Tap* Geste [22]

3.4 Spracheingabe

Die dritte Eingabeform für die HoloLens ist die Spracheingabe. Somit lassen sich ohne Gesten direkt Befehle an die HoloLens beziehungsweise die Hologramme leiten. Als Beispiel kann in Kombination mit dem Blick ein Hologramm angeguckt und der Befehl **select** gesagt werden. Select ist ein Standardbefehl, welcher in der HoloLens ohne explizite Implementation in einer Anwendung benutzt werden kann. Dabei fungiert der Befehl in gleicher Weise wie ein *Air Tap*. Wie aus *Windows* Smartphones oder Betriebssystemen bekannt, existiert hier ebenfalls der Befehl **Hey Cortana** um Cortana zu öffnen. Zusätzlich sind mehrere Grundfunktionen über die Spracheingabe, nach dem Motto **ßee it, say it"**, implementiert: Der Nutzer muss also nur den Namen eines Menüpunktes aussprechen, um diesen zu aktivieren (Adjust, Remove, Close...) [21].

3.5 Spatial Sound

Da das Sichtfeld des Nutzers eingeschränkt ist und der Mensch keine 360° sehen kann, bietet die HoloLens den sogenannte *Spatial Sound*. Hologramme, welche außerhalb des Sicht-

feldes des Nutzers liegen, können weiterhin Geräusche verursachen. Anhand der Position dieser Hologramme und die des Kopfes inklusive der Blickrichtung werden Geräusche in unterschiedlicher Lautstärke abgespielt, um Entfernung und Richtung zu simulieren. So kann der Nutzer ohne visuellen Einfluss Hologramme in seiner Nähe ausmachen, wodurch eine größere Immersion entsteht. Mit dem *Head Related Transfer Function* Prinzip wird analysiert, wie die Geräusche die jeweiligen Ohren erreichen würden. Dadurch wird das menschliche Gehirn überlistet und es kann ein Geräusch simuliert werden, bei dem der Nutzer denkt, dass es aus einer bestimmten Richtung im Raum kommt [20].

3.6 Spatial Mapping

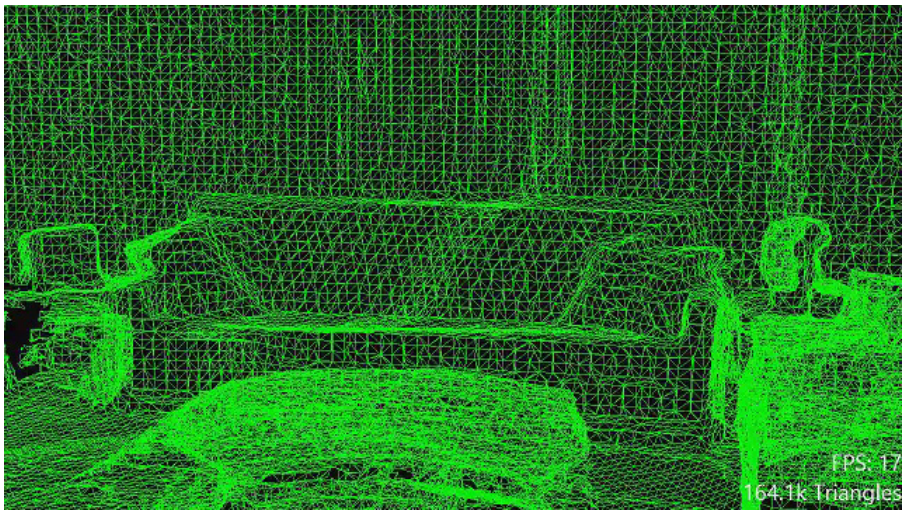

Abbildung 3.5: Beispiel einer *Spatial Mapping Mesh* eines Raumes [19]

Beim *Spatial Mapping* wird die reale Umgebung und ihre Oberfläche in einer detailreichen Repräsentation virtuell gespeichert. Somit können Hologramme innerhalb dieser virtuellen Karte platziert und eine Verschmelzung zwischen realer und virtueller Welt erreicht werden. Die zwei wesentlichen Bestandteile einer *Spatial Map* sind der *Spatial Surface Observer* und die *Spatial Surface*. Der *Spatial Surface Observer* besteht aus einem oder mehreren *Bounding Volumes*. Diese *Bounding Volumes* sind entweder statisch an einer bestimmten Position im realen Raum oder an die Position der HoloLens verankert. Diese werden benutzt um die Dimension darzustellen, in welcher die *Spatial Map* angelegt wird. Wenn die HoloLens benutzt wird, scannt sie permanent die reale Welt und speichert die

Ergebnisse in den *Bounding Volumes* als *Spatial Surface*. Eine *Spatial Surface* besteht aus einer Vielzahl aneinandergereihter Dreiecke, welche als *Mesh* bezeichnet werden. Wie so eine *Mesh* intern aussieht, wird in Abbildung 3.5 gezeigt. Hierbei ist wichtig, dass die HoloLens die reale Welt immer weiter scannt und dies nicht nur einmalig geschieht. Dadurch verändert sich die *Mesh* immer wieder und Veränderungen in der realen Welt werden erkannt. Diese *Mesh* wird dann in einem an die reale Welt gebundenes *Spatial Coordinate System* verankert und hat somit ein absolutes Koordinatensystem, in dem Hologramme platziert werden können. Räume, welche die HoloLens bereits erkundet hat, werden durch Charakteristika des Raumes wiedererkannt und alle gesammelten Daten für diesen Raum können dadurch geladen werden. Dazu zählen schon gescannte Bereiche, platzierte Hologramme usw. Mit Hilfe der *Spatial Map* lassen sich viele bekannte Probleme von AR-Systemen lösen. Zum einen können Hologramme möglichst real in der Umgebung platziert werden, was beinhaltet, dass sie reale Objekte verdecken, aber auch dass die reale Welt Hologramme verdecken kann. Zum anderen können physikalische Regeln auf Hologramme angewendet werden, wenn zum Beispiel ein virtueller Ball durch simulierte Schwerkraft von einem Tisch rollt und vom Boden wieder zurückspringt. Generell wird die *Spatial Map* für den Nutzer nicht angezeigt, da diese nur irritieren würde und auch zu viel Rechenleistung benötigen würde. Allerdings ist dies gerade für die Entwicklung ein sehr wichtiges Feature, um zu kontrollieren, ob die Verdeckung funktioniert oder bestimmte Hologramme sich so verhalten, wie sie sollten [19].

Spatial Understanding:
Da das reine Scannen des *Spatial Mappings* zu Fehlern führen kann, existiert bei der HoloLens das *Spatial Understanding*. Wenn die *Mesh* zum Großteil fertig geladen ist, kann mit Hilfe des *Spatial Understandings* die Karte um eine logische Komponente erweitert werden. Dadurch erkennt die HoloLens, ob es sich bei Flächen um Wände, Decken oder Böden handelt und auch kleine Scanfehler in denen in einer *Mesh* Löcher sein können, wo keine sein sollten, werden behoben. Ebenso wie *Spatial Mapping* evaluiert *Spatial Understanding* konstant seine Umgebung und versucht logische Muster zu erkennen und Fehler auszugleichen. Wie häufig und mit welcher Intensität dies geschieht, kann vom Entwickler manuell gesetzt werden. Dies erleichtert die Entwicklung, denn so kann deklariert werden, ob ein Hologramm zum Beispiel an einer Wand hängen soll oder an einer Decke [29].

3.7 Zusammenfassung HoloLens

Wie auf der Abbildung 3.1 zu sehen, befindet sich die HoloLens in der Mitte der Mixed Reality zwischen Augmented Reality und Augmented Virtuality. Dies ist durch die Vielzahl an Anwendungsmöglichkeiten zu erklären. Abbildung 2.4 aus Kapitel 2.1 verdeutlicht dies und gibt verschiedene Beispiele von Anwendungen an. Für die Interaktion stellt die HoloLens diverse unterschiedliche Möglichkeiten bereit. Diese reichen von Spracheingabe bis Gestenerkennung und so kann eine optimale Lösung für die jeweilige Anwendung getroffen werden. Dadurch, dass die HoloLens ein autarker, auf sich abgestimmter Computer ist, arbeiten die Komponenten Kamera, Display und Recheneinheit sehr gut zusammen. Daher ist die Performance für das *Tracking* der realen und virtuellen Objekte so gut, dass es kaum zu Registrationsproblemen der AR-Anwendung kommt. Mit Hilfe der Sensoren, insbesondere der Tiefenkamera und der daraus berechneten *Spatial Map*, können die Hologramme sowie reale Objekte im 3D-Raum platziert werden, was die Verdeckung ermöglicht und somit eine gute Immersion. Da die HoloLens über eine Hardware verfügt, welche mit den in Kapitel 2.3 bekannten Problemen der Interaktion, Registration und Verdeckungsberechnung für AR gut umgeht, erfüllt es die für diese Arbeit relevanten Kriterien.

Ausblick

Mit der Entwicklung der HoloLens hat *Microsoft* einen großen Schritt getan, um die AR durch HMD in ein professionelles Umfeld zu bringen. Anstatt Monitore als Anzeige für zusätzliche Informationen zu benutzen, wird in einigen Bereichen bereits die HoloLens verwendet. Dies hat den Vorteil, dass das Display immer am Nutzer des Systems und somit immer sichtbar ist und dass kein Gerät getragen werden muss und somit die Hände frei sind. In näherer Zukunft wird es die *HoloLens 2* geben, bei der sich die Größe des Sichtfeldes verdoppelt und die Steuerung für Hologramme noch weiter verbessert sein soll.

3.8 Verwendete Software

Damit eine Anwendung, welche die Kriterien dieser Arbeit erfüllt, für die HoloLens entwickelt werden kann, werden neben der eigentlichen Hardware auch Softwarelösungen benötigt. Im folgenden werden diese Softwaretools vorgestellt und kurz erläutert warum diese sich für die Umsetzung eignen.

3.8.1 Blender

Blender ist eine open source 3D-Grafiksuite, mit dessen Hilfe man alle nötigen Schritte zur Erstellung von 3D-Objekten durchführen kann. Die wichtigsten Schritte sind das Modellieren, Texturieren und Animieren von Objekten. Für diese Arbeit wurde Blender genutzt, um die Hologramme zu erzeugen, welche in der Anwendung dargestellt werden sollen [2].

3.8.2 Unity

Mit *Unity* lassen sich Computerspiele und andere interaktive 3D-Grafik-Anwendungen, für Computer, Spielekonsolen, mobile Geräte und Webbrowser entwickeln. Es handelt sich hierbei um eine eigenständige Laufzeit- und Entwicklungsumgebung (IDE) für *Windows*, *Linux* und *macOS*. Wie bei jeder IDE besteht die grafische Oberfläche (GUI) aus mehreren Fenstern, die verschoben, verändert, hinzugefügt oder entfernt werden können. Die wichtigsten Bestandteile der GUI sind das *Project Window*, die *Scene View*, das *Hierarchy Window*, das *Inspector Window* und die *Toolbar*.

Das Project Window:
Im *Project Window* werden die *Assets* angezeigt, welche für ein Projekt zur Verfügung stehen. Wenn neue *Assets* importiert werden, kann man diese dort finden. Wichtige *Assets* für dieses Projekt sind importierte Hologramme, Scripte, Texturen und das *Mixed-Reality-Toolkit*.

Die Scene View:
In der *Scene View* wird die aktuelle Szene dargestellt und dort kann man durch diese navigieren oder editieren. Die Szene kann in 2D oder in 3D angezeigt werden, wobei für dieses Projekt nur 3D relevant ist.

Das Hierarchy Window:
Im *Hierarchy Window* werden die Objekte der Szene in hierarchischer Repräsentation dargestellt. Hier können Objekte hinzugefügt oder entfernt werden und man kann sehen, wie die Objekte miteinander verknüpft sind. Dies bestimmt dann eine Parent-Child Struktur, welche für das spätere Design entscheidend sein wird. Alle Objekte die im *Hierarchy Window* angezeigt werden, existieren auch im *Scene Window*.

Das Inspector Window:
Ein weiteres wichtiges Fenster ist das *Inspector Window*. Wenn ein Objekt ausgewählt ist, werden dort seine Eigenschaften und Komponenten angezeigt. Eigenschaften können dort bearbeitet werden und Komponenten können hinzugefügt, bearbeitet oder entfernt werden. Scripte, welche auf ein Objekt wirken sollen, werden dort als Komponente hinzugefügt.

Die Toolbar:
In der *Toolbar* kann man die Werkzeuge auswählen, die zum Bearbeiten der Objekte innerhalb der Szene und der Szene selbst vorhanden sind. Zudem findet man dort die Steuerung, mit der eine Szene abgespielt und gestoppt werden kann. Für dieses Projekt ist *Toolbar* wichtig, um den Emulator starten zu können um die Anwendung zu testen. Hauptsächlich werden aber nur die Manipulationswerkzeuge benötigt, um die Objekte der Szene zu bearbeiten [1].

Wichtige Begriffe:
Für dieses Projekt relevante Begriffe für *Unity* sind *Scene*, *GameObject*, *Camera* und *Script*. Die *Scene* beinhaltet die komplette aktuelle Umwelt, in der die Anordnung der *GameObjects* deklariert ist. *GameObjects* sind die fundamentalen Objekte einer Anwendung für *Unity*. Alle wichtigen Komponenten sind im Grunde *GameObjects* mit unterschiedlichen Eigenschaften. So sind *GameObjects* bei der Neuerzeugung erst einmal leere Objekte und können dann mit Komponenten befüllt werden, damit sie ihre spezifischen Eigenschaften bekommen. Dadurch erhalten *GameObjects* dann zum Beispiel ihr Äußeres oder die Logik, mit der bestimmt wird, wie sich das *GameObject* in der Anwendung verhalten soll. Damit die in *Unity* erzeugte *Scene* mit der HoloLens betrachtet werden kann, braucht es eine *Camera*. Dieses wird fertig voreingestellt vom *Mixed-Reality-Toolkit* geliefert und ist auf die Eigenschaften der HoloLens zugeschnitten. Damit ist gesichert, dass wenn der Nutzer sich mit der HoloLens durch den Raum bewegt, die *Camera* dem aktuellen Sichtfeld folgt und die Hologramme anzeigt, die im aktuellen Sichtfeld liegen.

Mit seinen Eigenschaften bietet *Unity* die besten Grundlagen, das in Kapitel 4.1 vorgestellte Design, also eine immersive Anwendung für die HoloLens, zu entwickeln.

3.8.3 Mixed-Reality-Toolkit

Das *Mixed-Reality-Toolkit* (MRT) ist ein sehr wichtiger Bestandteil dieser Anwendung. Es fungiert als *Framework* und stellt viele essentiell wichtige Scripte und allgemeine *Assets* zur Verfügung, um eine optimale Anwendung für die HoloLens zu entwickeln. Welche dies sind und welche hier verwendet wurden, wird in Kapitel 4.2 näher erläutert. Damit die in *Unity* entwickelte Anwendung so auf der HoloLens funktioniert, wie sie funktionieren soll, müssen gewisse Einstellungen in *Unity* getätigt werden. Diese betreffen verschiedene Einstellungen für das compilen des Quellcodes auf die Chip-Architektur der HoloLens-HoloLens. Das *MRT* nimmt einem diese Arbeit ab und kann diese automatisieren. So lassen sich MR Anwendungen schneller und fehlerfreier entwickeln [3].

4 Anwendung

Im folgenden Kapitel werden Designentscheidungen getroffen, welche die Architektur der Anwendung bestimmen und die in Kapitel 2.7 herausgearbeiteten Anforderungen erfüllen. Dabei wird Rücksicht auf die Eigenschaften der HoloLens genommen. Diese Architektur wird dann in der Implementierungsphase umgesetzt, um anschließend Evaluiert zu werden.

4.1 Design

Zuerst werden allgemeine Designentscheidungen getroffen, die auf die Grundprobleme der AR zurückzuführen sind. Für die jeweiligen drei Problemgruppen der Interaktion, Registration und Verdeckungsberechnung, wird erläutert, welche Lösungsansätze es gibt, welches die Vor- und Nachteile sind und für welches sich für die Umsetzung entschieden wurde.

4.1.1 Interaktion

Damit der Nutzer die Hologramme auswählen, benutzen und platzieren kann, muss ihm die Möglichkeit gegeben werden, mit der virtuellen Welt interagieren zu können. In Kapitel 3 wurden bereits Eingabemöglichkeiten vorgestellt, welche mit der HoloLens möglich sind. Dazu zählen der Blick (Gaze), die Bewegung mit den Händen (Gestures) und die Spracheingabe. Der Blick ist die einfachste Möglichkeit Hologramme zu selektieren. Mit der Kopfposition und der Blickrichtung wird also bestimmt, ob und welches Hologramm angeguckt wird. Anstatt direkt an den Hologrammen stehen zu müssen und sie eventuell mit den Händen direkt berühren zu müssen, kann der Blick aus der Entfernung genutzt werden. Mit der Kombination aus *Gaze* und *Gesture* lassen sich Hologramme gut manipulieren und die Spracheingabe wird für allgemeine Befehle genutzt. Dies hat den Vorteil, dass zum Beispiel bei unsauberer Spracheingabe die Hologramme nicht fälschlicherweise

manipuliert werden. Um ein Hologramm in seiner Größe zu verändern, sind die Eingaben der Hand genauer als der Blick oder die Ansage in Zentimetern. Allerdings lassen sich mit der Sprache mehr Befehle durchführen als mit Gesten, da diese in ihrer Anzahl standardmäßig begrenzt sind. Da jede der drei Alternativen ihre Vorteile hat, werden bei der Anwendung alle diese Möglichkeiten zum Einsatz kommen.

Gesten

Die Gesten sollten möglichst simpel und natürlich sein. Es gibt zwar auch die Möglichkeit, weitere Gesten zu entwickeln, allerdings sind für diese Arbeit die beiden Gesten *Air Tap* und *Drag and Hold* ausreichend. Beide Gesten wurden in Kapitel 3.4 bereits vorgestellt und dienen zur Steuerung der Hologramme. Mit dem *Air Tap* lassen sich die Animationen der Hologramme starten und wieder stoppen. Wenn es erlaubt ist die Hologramme in ihrer Größe zu verändern, zu Rotieren oder im Raum zu verschieben, dann kann die *Drag and Hold* Geste verwendet werden.

Sprachbefehle

Die Sprachbefehle werden genutzt, um die den Hologrammen zugehörigen Infotexte anzeigen zu lassen. Wenn ein Hologramm mit dem Blick fokussiert wird und der Befehl **show text** gesprochen wird, wird der zu dem Hologramm dazugehörige Text eingeblendet. Um den nächsten Textteil anzuzeigen, während der Text eingeblendet ist, kann der Sprachbefehl **next** genutzt werden. Dies soll das Prinzip des *Recommender System* darstellen, bei dem an der selben Position im Raum unterschiedliche Informationen angezeigt werden können. Zum ausblenden des Textes kann wiederum **hide text** gesprochen werden. Um anzeigen zu lassen, welche Befehle die Anwendung im allgemeinen unterstützt, kann mit den beiden Sprachbefehlen **show commands** und **hide commands** die Liste der Befehle eingeblendet beziehungsweise wieder ausgeblendet werden. Als letztes fehlt noch der Befehl **admin mode**, dieser aktiviert beziehungsweise deaktiviert den Platzierungsmodus, welcher in Abschnitt 4.1.2 erläutert wird.

4.1.2 Registration

Damit die Interaktion zwischen Nutzer und Hologrammen funktioniert, müssen ihre Position zueinander und relativ im Raum bekannt sein. Dies wird dann dazu verwendet, nur

das Hologramm anzuzeigen, welches dem Nutzer am nächsten ist. Falls der Nutzer in eine Richtung blickt, in der ein Hologramm sein könnte, dieses aber aufgrund der Entfernung nicht angezeigt wird, dann wird ein Indikator eingeblendet. Indikatoren haben keine besonderen Eigenschaften und keine Animation, sie dienen lediglich der Orientierung. Die Bestimmung der Position der Hologramme kann auf verschiedene Arten geschehen. Die Hologramme relational an die Position des HMD zu heften, würde für diese Arbeit nicht funktionieren, da sich die Entfernungen zwischen Hologrammen und Nutzer dadurch nicht verändern würden. Optisches Tracking und genauer spezifiziert Markertracking, wäre eine Option, die benutzt werden könnte. Für AR und *Unity* gibt es *Vuforia*, welches vorher definierte *Tags* erkennt (zum Beispiel ausgedruckte QR-Codes) und an deren Position dann die Hologramme einblendet. Dies hat den Vorteil, dass die Umgebung nicht vorher eingescannt werden muss, um die Hologramme an ihre endgültige Position zu verschieben. Der Nachteil liegt deutlich in der Anzahl der möglichen Hologramme. Vor Nutzung der Anwendung müssen diese *Tags* ausgedruckt und verteilt werden, ein dynamisches hinzufügen von Hologrammen ist auf Softwareebene dann nicht möglich. Wenn also ein weiteres Hologramm der Umgebung hinzugefügt werden soll, muss zuerst immer ein weiterer *Tag* physisch erzeugt und verteilt werden. Aufgrund dieser Nachteile wird in dieser Arbeit mit Ankern gearbeitet, mit denen die HoloLens die Position der Hologramme anhand der *Spatial Map* bestimmen kann.

Anker

Falls die Anwendung zum ersten mal gestartet wird, werden die Hologramme zur absoluten Position des HMD im Raum platziert. Anschließend werden Ankerpunkte gesetzt. Wenn nun im Platzierungsmodus die Hologramme per *Drag and Hold* durch den Raum bewegt werden, wird jedes mal, wenn das Hologramm abgelegt wird, der Anker neu gesetzt. Gespeichert wird dann nicht mehr die Position in Relation zur *Camera*, sondern in Bezug auf die *Spatial Map*. Wird nun die Anwendung zu einem späteren Zeitpunkt wieder geöffnet und die HoloLens erkennt mit Hilfe des *Spatial Mapping* den Raum wieder, werden die Hologramme an die Punkte verankert, an denen sie beim letzten Verwenden der Anwendung verblieben sind. So kann die Anwendung verfolgen, wo sich welche Hologramme befinden, auch wenn diese nicht im Sichtfeld liegen.

Verdeckungsberechnung

Wie in Kapitel 3.6 beschrieben, verfügt die HoloLens über eine Tiefenkamera, mit dessen Hilfe eine *Spatial Map* erzeugt wird. Da diese eine Abbildung der realen Umgebung ist, können die Hologramme anhand dieser Karte im Raum verteilt werden. Diese *Spatial Map* wird dann für die Verdeckung genutzt. Wenn also ein Hologramm teils oder komplett hinter der *Mesh* der *Spatial Map* ist, wird dieser Teil ausgeblendet. Dies wäre beim optischen Tracking nicht möglich, denn wenn der Marker auch nur zum Teil verdeckt sein würde, könnte das Hologramm komplett nicht mehr dargestellt werden. Durch das *Spatial Mapping* und *Spatial Understanding* nimmt uns die HoloLens die Designentscheidung ab und es wird sich hier für diese Tracking-Methode entschieden.

4.1.3 Funktionalität

Die beiden Abbildungen 4.1 und 4.2 verdeutlichen den Ablauf, der während der Nutzung der Anwendung entstehen kann. Damit werden die funktionalen Anforderungen aus Kapitel 2.7 abgedeckt.

Abbildung 4.1 zeigt, was geschieht, wenn ein Nutzer im normalen Modus die Anwendung benutzt und bei bestimmten Events die Gesten oder Sprachbefehle ausführt.

Abbildung 4.2 zeigt die zusätzlichen Möglichkeiten, die entstehen, wenn der *Platzierungsmodus* aktiviert ist.

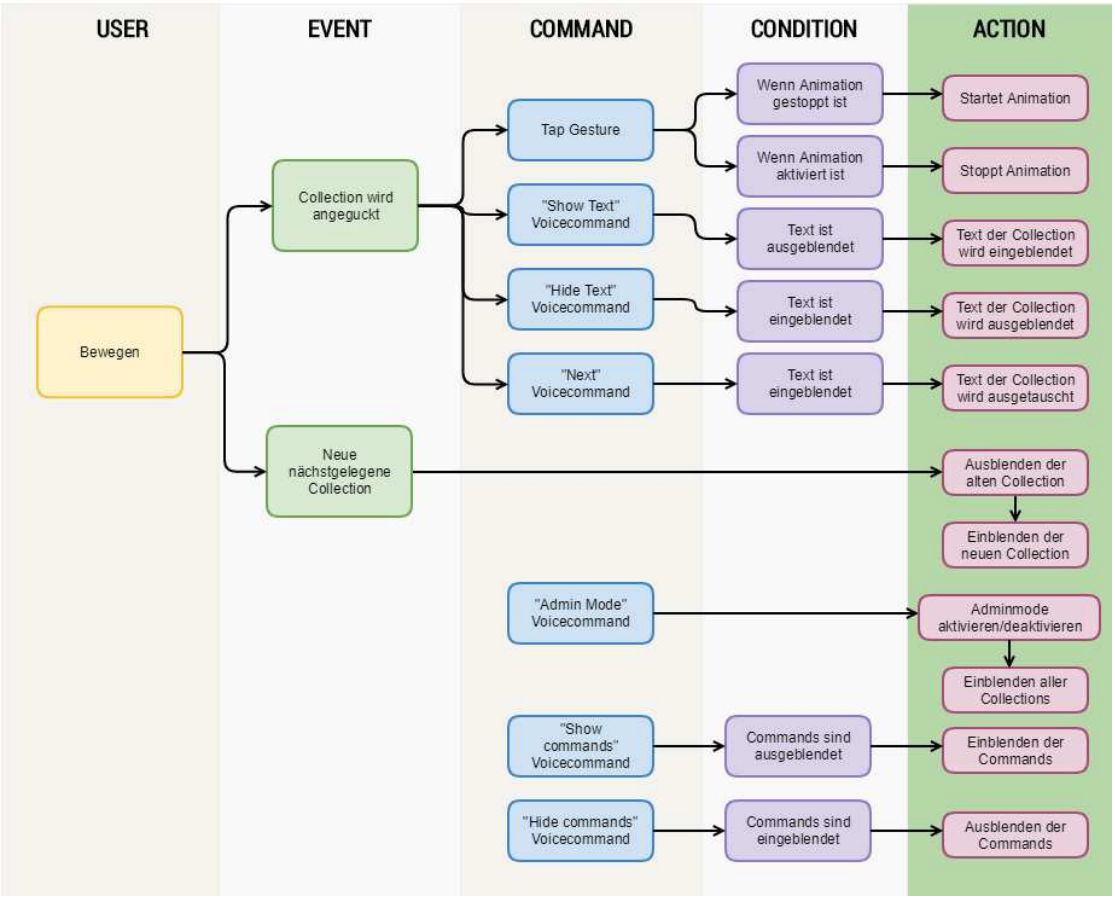

Abbildung 4.1: Übersicht des Anwendungsablauf (Quelle: Eigene Darstellung)

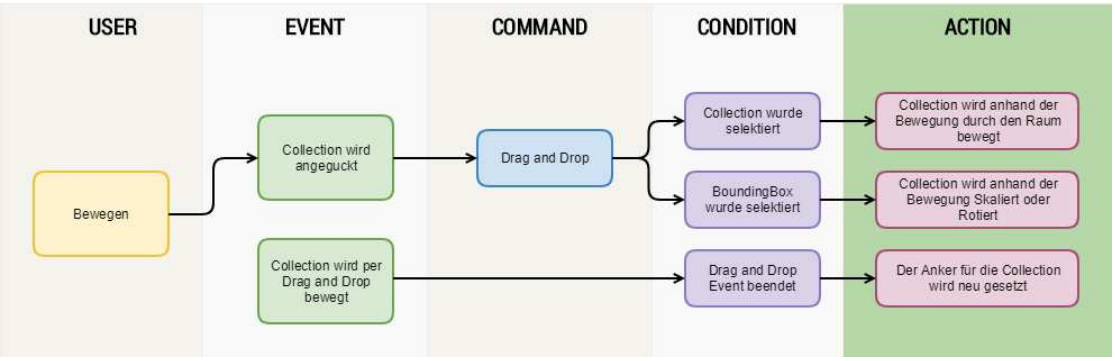

Abbildung 4.2: Übersicht des Anwendungsablauf im Platzierungsmodus (Quelle: Eigene Darstellung)

4.1.4 Komponenten

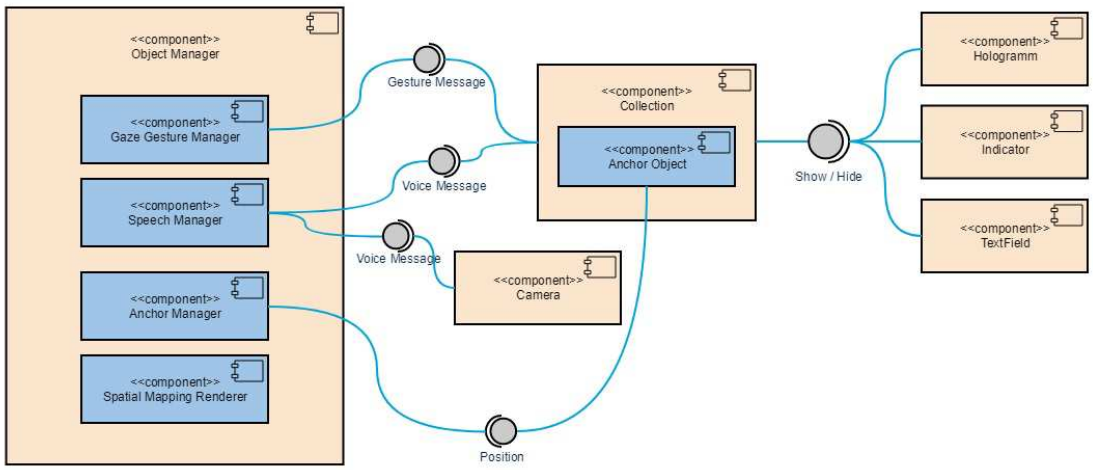

Abbildung 4.3: Aufbau der einzelnen Systemkomponenten (Quelle: Eigene Darstellung)

Das Diagramm in Abbildung 4.3 veranschaulicht die wichtigsten Komponenten der Anwendung und zeigt die Abhängigkeiten dieser. Der Vorteil dieser Aufteilung der einzelnen Komponenten, liegt in der Erweiterbarkeit der Anwendung. Die Anzahl und Art der Hologramme in einer *Collection* kann variiert werden, wie es bei den Textfeldern der Fall ist. Ebenso können beliebig viele *Collections* erstellt und hinzugefügt werden, welche der *ObjectManager* verwaltet. Falls für spezielle *Collections* beziehungsweise deren Hologramme eigene spezifische Befehle genutzt werden sollen, können diese mit *Tags* versehen werden. Der *ObjectManager* wird um Befehle erweitert, die nur für *Child-Objekte* gelten, welche diesen *Tag* besitzen. Damit können auch Gesten wiederverwendet werden, falls diese bei unterschiedlichen *Collections* unterschiedliche Aktionen hervorrufen sollen. Die Hierarchie wird in Abbildung 4.4 veranschaulicht.

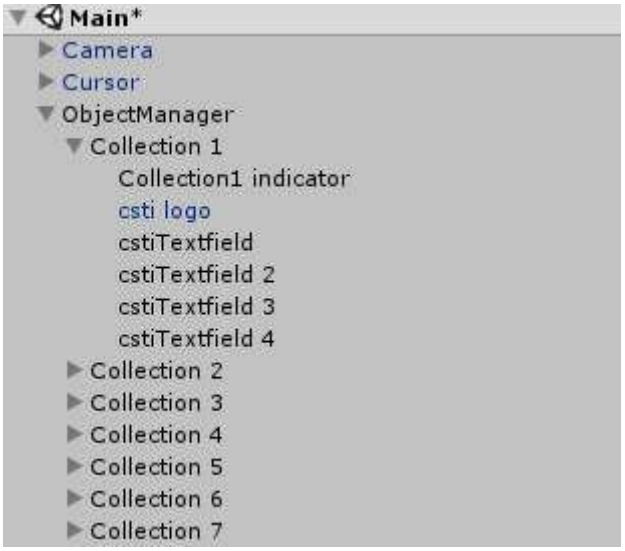

Abbildung 4.4: Die Hierarchie der Komponenten in der Anwendung (Quelle: Eigene Darstellung)

4.2 Umsetzung

Im folgenden Kapitel werden die verwendeten Softwarelösungen vorgestellt, welche benutzt wurden, um das Design umzusetzen. Dann werden die einzelnen Komponenten vorgestellt, aus denen die Anwendung besteht.

4.2.1 Toolkit

Vom *Mixed-Reality-Toolkit*, welches in Kapitel 3.8.3 vorgestellt wurde, wurden folgende Komponenten verwendet:

- **General Settings:** Allgemeine Grundeinstellungen in *Unity*, welche getätigt werden müssen, damit die Anwendung auf der HoloLens ausführbar ist.
- **MixedRealityCameraParent:** Die Hauptkamera für die MR-Anwendung. Diese ist so voreingestellt, dass sie optimal mit der Hardware der HoloLens funktioniert. Zusätzlich bietet sie Unterstützung für den Blick, Audio, Mikrofone und Motion Controller. Das abspielen von Audio und die Verwendung von Motion Controllern werden in dieser Arbeit allerdings nicht eingesetzt.

- **DefaultCursor:** Liefert den *Cursor* für die MR-Anwendung. Da bei normalen Anwendungen von *Unity* der *Cursor* zum Beispiel mit einer normalen Computer-Maus gesteuert wird, benötigt man spezielle Einstellungen damit der *Cursor* der HoloLens, welcher über den Blick gesteuert wird, benutzt werden kann. Zusätzlich verändert sich der *DefaultCursor* in seiner Erscheinung, wenn ein Hologramm anvisiert wird, mit welchem man interagieren kann.

- **InputManager:** Stellt die Schnittstelle aller möglichen Eingabegeräte für die HoloLens bereit. Für diese Arbeit wird nur der *GestureInput* benötigt, um mit Gesten arbeiten zu können, welche die HoloLens erkennt.

- **Bounding Box:** Diese wird benötigt, um die *Transform* Komponente eines Hologramms innerhalb der Anwendung zu manipulieren. Um das Hologramm entsteht eine Box mit Fixpunkten, welche ausgewählt und bewegt werden können, um die Größe oder Rotation zu verändern.

Somit wurden besonders wichtige Komponenten ohne große Probleme bereits vom MRT bereitgestellt. Gerade die Komponenten *Camera* und die *Spatial Map* sind für eine AR Anwendung besonders wichtig, denn wenn diese aufgrund von Synchronisationsproblemen oder fehlerhaften Einstellungen nicht perfekt miteinander funktionieren, entstehen visuelle Fehler, worunter die Immersion leiden würde.

4.2.2 GameObject

Wie in Kapitel 3.8.2 bereits erwähnt, sind die *GameObjects* die grundlegenden Bestandteile einer *Unity* Anwendung. Alle Komponenten, die in der Szene eine Funktion erfüllen sollen, sind im Grunde *GameObjects*, welche durch *Assets* um bestimmte Eigenschaften erweitert wurden. Hologramme sind in diesem Fall also auch *GameObjects*, die für diese Arbeit mit den speziellen relevanten Eigenschaften definiert wurden. Damit ein Hologramm für die Anwendung dieser Arbeit genutzt werden kann, benötigt es folgende Komponenten:

- **Transform:** Die Transformkomponente bestimmt die Position, Rotation und die Skalierung des Hologramms.

- **Mesh Filter:** Der *Mesh Filter* gibt die Form, also das Aussehen des Hologramms vor, indem es die *Mesh* an den *Mesh Renderer* weiterleitet.

- **Mesh Renderer:** Der *Mesh Renderer* sorgt dafür, dass die *Mesh* berechnet wird und somit in der Szene dargestellt werden kann.

- **Collider:** Der *Collider* sorgt dafür, dass das Hologramm physikalisch wie ein echtes Objekt wirkt. Wenn also zwei *GameObjects* mit *Collidern* aufeinander treffen, wird eine Intersektion erkannt. Wichtig ist dies, da der *Cursor* der Anwendung auch ein *GameObject* ist. Mit Hilfe des *Gaze* kann die virtuelle Linie, die dabei gezogen wird, auf das *GameObject* treffen und es dadurch selektieren. Ansonsten würde der *Gaze* durch das andere *GameObject* hindurch gehen und es würde keine Selektierung registriert werden. Der *Collider* kann für komplexe Hologramme die *Mesh* als Vorlage für die Form nehmen oder aus Performancegründen aus primitiveren Formen, wie Rechtecken oder Kugeln bestehen.

- **Scripte:** Ohne Scripte würden die *GameObjects* genau in dem Zustand verweilen, in dem sie bei Beginn der Anwendung deklariert wurden. Ein Script definiert also die Logik, mit der *GameObjects* reagieren und interagieren. Diese Scripte können mit **C#** und **UnityScript** (ist ähnlich zu JavaScript) geschrieben werden. Für diese Arbeit wurde **C#** verwendet.

- **Animator:** Der Animator ist für den Animationsablauf zuständig, falls das Hologramm animiert werden soll.

- **Spatial Mapping Renderer:** Der *Spatial Mapping Renderer* übernimmt die Aufgabe der *Verdeckungsberechnung*. Es sorgt dafür, dass die Hologramme durch reale Objekte der *Spatial Map* verdeckt werden, um eine bessere Immersion zu erzeugen.

Um alle Komponenten der Anwendung in *Unity* unterscheiden zu können, werden diese mit *Tags* versehen. So können Scripte gruppierte Komponenten finden und auf diese zugreifen. Verwendete *Tags* sind die in Kapitel 4.1 definierten Gruppen: *Collection*, *Indicator*, *Hologram*, *Text* und *MainCamera*.

4.2.3 Camera

Die *Camera* ist die wichtigste Komponente der AR-Anwendung. Sie wird mit allen Positions- und Rotationssensoren der HoloLens synchronisiert und stellt so die Blickrichtung des Nutzers dar. Nur Hologramme, welche im Blickfeld des Nutzers liegen, werden dargestellt. Dies steigert die Performance der Anwendung, denn so spart sich die HoloLens unnütze Berechnungen für die Darstellung dieser Hologramme. Die *Camera* liefert auch

die aktuelle Position des Nutzers im Koordinatensystem des virtuellen Raumes, welche dann für die Berechnung der Distanz zum Hologramm genutzt werden kann. Zusätzlich wurden dem *Camera-Object* in *Unity* noch zwei Textfelder hinzugefügt. Diese können durch Befehle eingeblendet werden und zeigen die Liste der verfügbaren Befehle und einen Indikatortext. Der Indikatortext gibt Aufschluss darüber, ob der *Platzierungsmodus* aktiviert ist. Das *Camera-Object* in *Unity* wird vom MRT zur Verfügung gestellt und ist auf die Sensoren der HoloLens zugeschnitten, wodurch das Zusammenspiel zwischen Software und Hardware verbessert wird.

4.2.4 Cursor

Der *Cursor* erscheint, wenn der Blick des Nutzers auf ein Hologramm fällt. Somit hat der Nutzer einen Indikator, der ihm verrät, ob er ein Hologramm anvisiert oder nicht und kann so sehen welches Hologramm von seinem Befehl betroffen sein wird. Visualisiert ist der *Cursor* durch einen dreidimensionalen Kreis, welcher auf der Oberfläche der Hologramme haftet.

4.2.5 Object Manager

Der *Object Manager* fungiert als Herzstück der *GameObjects*, welche in der Hierarchie unter ihm angesiedelt sind. Es ist also wie ein *Master GameObject*, welches die Information und somit die Übersicht über alle verfügbaren *GameObjects* der Anwendung hält. Der *Object Manager* besteht aus seiner eigenen *Logik*, dem *Gaze Gesture Manager*, dem *Speech Manager* und dem *World Anchor Manager*. In Abbildung 4.4 erkennt man, dass dem *Object Manager* beliebig viele *Collections* untergeordnet sein können. Beim starten der Anwendung erstellt der *Object Manager* eine Liste und befüllt diese mit allen *Collections*, die für ihn hierarchisch verfügbar sind. Danach blendet er alle diese aus und nur die nächstliegendste *Collection* wird eingeblendet. Dann startet eine Routine, in der jede Sekunde überprüft wird, ob nach einer Bewegung des Nutzers eine *Collection* näher am Nutzer ist, als die aktuell eingeblendete. Wenn dies der Fall ist, wird diese ein- und die alte ausgeblendet.

Gaze Gesture Manager:
Der *Gaze Gesture Manager* ist für den Blick und die Gesten zuständig. Es wird eine Routine gestartet, welche jede Sekunde überprüft, ob der Nutzer ein *GameObject* fokussiert. Falls dies der Fall ist, dieses *GameObject* in einer Referenz, für spätere Benutzung, speichert. Falls bereits eine Geste auf ein *GameObject* angewendet wird, aber ein neues *GameObject* fokussiert wird, sorgt die Routine dafür, dass die Geste abgebrochen wird. Zudem beinhaltet der *Gaze Gesture Manager* einen *Gesture Recognizer*, welcher durchgehend überprüft, ob der Nutzer eine Geste ausgeführt hat. Dann sendet der Manager eine Nachricht an das *GameObject*, um zu signalisieren, dass die bestimmte Geste auf ihn angewendet werden soll.

Speech Manager:
Der *Speech Manager* ist für die Erkennung der Sprachbefehle zuständig. Hier wird das *Dictionary* definiert, welches die Wörter beinhaltet, die als Befehle vorhanden sein sollen. Auch die auszuführende Aktion zu einem Befehl, wird in diesem definiert. Dann wird das *Dictionary* mit einem *Speech Recognizer* verknüpft, welches auf Sprache hört, die Wörter erkennt und mit dem *Dictionary* abgleicht und bei einem bekannten Befehl die dazugehörige Aktion ausführt.

World Anchor Manager:
Der *World Anchor Manager* sorgt für die Persistenz der Position der Hologramme. Wenn die Anwendung gestartet wird, wird auf der HoloLens überprüft, ob es einen gültigen *Anchor Store* gibt. Falls dies der Fall ist, wird dieser geladen und die Position von allen Anker-Objekten wiederhergestellt. Auch nachdem ein Anker-Objekten bewegt wurde, wird die neue Position im *Anchor Store* gespeichert. Da der *Anchor Store* abhängig vom *globalen Anchor Store* der HoloLens ist, werden beim Löschen der temporären Dateien und eben auch dem *globalen Anchor Store* auch die Speicherpunkte für die Anwendung gelöscht.

4.2.6 Collection

Eine *Collection* fungiert als Sammelobjekt für Indikatoren, Hologramme und Textfelder. Bei einem ausgeführtem Befehl für eine *Collection* sorgt sie dafür, dass der Befehl an das aktive darunterliegende *GameObject* weitergereicht wird. Im Falle von Textfeldern sorgt sie dafür, dass der nächste Textbereich angezeigt wird, sofern dies geschehen soll.

Bei Hologrammen und Indikatoren werden diese dementsprechend ein- oder ausgeblendet. Zudem ist eine *Collection* ein *Anker-Objekt* und so wird seine Position im *Anchor Store* gespeichert. Da die *Collection* eine Sammlung für alle sich darunter befindenden *GameObjects* ist, werden alle diese gleichzeitig bei einer Manipulation verschoben. Dies führt auch dazu, dass die Positionen aller *Child-GameObjects* der *Collection* durch einen Ankerpunkt wiederhergestellt werden können.

Indikator:
Der Indikator hat keine eigene Logik und dient nur als Informationsobjekt. Wenn durch die Position des Nutzers, die *Collection* und alle dazugehörigen Hologramme und Textfelder ausgeblendet sind, wird der Indikator eingeblendet, um die Position erkennbar zu machen.

Textfeld:
Ein *Textfeld* beinhaltet eine bestimmte Information, die für den Nutzer sichtbar gemacht werden kann. Wenn der Befehl des Einblendens erfolgt, wird zunächst das zuerst definierte *Textfeld* eingeblendet. Beim Sprachbefehl **next** wird dieses dann wieder ausgeblendet und das nächste in der Reihe angezeigt. So lassen sich beliebig viele *Textfelder* einer *Collection* hinzufügen, welche mit der Position in der Hierarchie im *Editor* die Reihenfolge der Einblendung bestimmen. Dies ist auch das erste *GameObject*, welches vorgestellt wurde, das einen *Collider* enthält. Dieser *Collider* sorgt dafür, dass der Blick des Nutzers auf das *Textfeld* treffen kann, um es zu selektieren, und nicht durch ihn hindurch geht. Dadurch wird dann auch der *Cursor* auf dem *Textfeld* dargestellt.

4.2.7 Hologramm

Zuerst muss das äußere Erscheinungsbild eines Hologramms definiert werden. Dazu wurde *Blender* verwendet, um die dreidimensionale Form zu erzeugen und anschließend zu exportieren. Das exportierte Design kann wiederum in *Unity* als *Asset* importiert werden und bei einem *GameObject* als Komponente hinzugefügt werden. Im Gegensatz zu Indikatoren und *Textfeldern* haben Hologramme eine eigene Logik. Diese reagiert auf Nachrichten, die vom *Gaze Gesture Manager* kommen. Falls also der Nutzer einen *Air Tap* auf das Hologramm angewendet hat, wird die vorher definierte Aktion des Hologramms aufgerufen. Diese sorgt dann für die Art und Weise wie das Hologramm animiert werden soll. Ansonsten bestehen Hologramme aus einer *Mesh*, welche die Form vorgibt und einem *Mesh Renderer*, welcher für die Darstellung zuständig ist. Genauso wie das *Textfeld*

besitzt ein Hologramm auch ein *Collider* für die Möglichkeit der Selektierung durch den Blick des Nutzers.

4.3 Testen der Anwendung

In diesem Abschnitt wird die Anwendung zuerst getestet und anschließend auf die in Kapitel 2.7 erarbeiteten Anforderungen überprüft. Zum Testen der Anwendung gibt es drei Möglichkeiten, die alle ihren Einsatz während der Entwicklung dieser Anwendung gefunden haben. Da die HoloLens nicht immer verfügbar war, wurde, wenn etwas an der Anwendung geändert wurde, zur groben Überprüfung der Änderung der *Unity Emulator* beziehungsweise der *HoloLens Emulator* verwendet. Selbst wenn die HoloLens verfügbar war, wurde manchmal auf diese Mittel zurückgegriffen, denn wenn nur ein kleiner Teil des Designs oder Codes geändert wurde, musste die Anwendung nicht komplett neu compiliert und auf die Hardware exportiert werden.

4.3.1 Emulatoren

Unity Emulator:
Der *Unity Editor* bietet die Möglichkeit, die Anwendung direkt in der *Unity Engine* zu starten. Es wird auf das *Game Window* gewechselt und mit dem Start-Knopf die Anwendung gestartet. Dann hat man verschiedene Möglichkeiten der Bedienung. Falls kein Gamepad vorhanden ist, können auch die Tastatur und die Maus benutzt werden, um sich in der aktuellen *Scene* zu Bewegen. Wenn die rechte Maustaste gedrückt bleibt und die Maus bewegt wird, wird die Kopfbewegung in der *Scene* simuliert. Mittels der Tastatur bewegt man sich im virtuellen Raum. Der Nachteil ist, dass keine Gesten mit den Händen getätigt werden können. Damit Gesten überhaupt möglich sind, muss der Player auf *Holographic Emulation* gestellt werden. Unter den **Window -> XR** Einstellungen, findet man die *Holographic Emulation* Einstellungen. Dort wird der Mode auf *Simulate in Editor* gestellt und es können ein vordefinierter Raum und die Hand, welche die Gesten ausführen soll, gewählt werden. Dann lassen sich mit einem Gamepad die *Air Tap* Geste und *Drag and Hold* ausführen. Die Sprachsteuerung funktioniert über ein an den Computer angeschlossenes Mikrofon.

HoloLens Emulator:
Von *Microsoft* gibt es für *Visual Studio* den *HoloLens Emulator*. Mit Hilfe dieses Emulators lassen sich durch *Unity* erstellte *Universal Windows Platform* Applikationen für die HoloLens ausführen. Mit Hyper-V, einer Hypervisor-basierten Virtualisierungstechnik von *Microsoft*, lässt sich die Hardware der HoloLens nachstellen. So erhält man einen Eindruck, ob die Anwendung auf der HoloLens performant arbeiten würde. Dabei wird das komplette Betriebssystem mit allen Applikationen, welche die HoloLens im Normalfall enthält, emuliert. Auch wie beim *Unity Emulator* lässt sich die Anwendung allerdings nur mit Maus und Tastatur bedienen und die Gesten *Air Tap* und *Drag and Hold* funktionieren mit Hilfe der Maus. Der *HoloLens Emulator* bietet außerdem ein *Simulation Control Panel* (Abbildung 4.5), mit dem man verschiedene Einstellungen für die Verwendung des Emulators tätigen kann. Diese reichen über die aktuelle Position, welche Gesten mit welcher Hand ausgeführt werden sollen inklusive deren Ausführung, das Laden eines vordefinierten virtuellen Raumes, bis zur Einstellung, welche Eingabegeräte genutzt werden sollen.

Wird der Emulator ohne Raum gestartet, bleibt der Hintergrund schwarz. Auf Abbildung 4.6 ist das erste Hologramm dargestellt, welches ein Nutzer sieht, wenn die Anwendung das erste mal gestartet wird. Hierbei sind alle anderen Hologramme ausgeblendet und das eingeblendete Hologramm könnte mit der *Air Tap* Geste animiert werden.

Bewegt sich der Nutzer durch den virtuellen Raum, kann es dazu kommen, dass ein anderes Hologramm für ihn eingeblendet und das aktuelle ausgeblendet wird. Abbildung 4.7 zeigt, wie hierbei nicht mehr der Wolf, sondern ein Logo eingeblendet ist. Es wurde zudem dieses Hologramm fokussiert und der Sprachbefehl **show text** gesprochen, welches den zum Hologramm gehörigen Text einblendet. Ausgeblendete Hologramme werden mit den gelben Indikatoren markiert, damit der Nutzer sehen kann, wo das nächste Hologramm zu finden wäre.

Abbildung 4.8 zeigt, wie der Sprachbefehl **show commands** ausgeführt wurde und im oberen linken Sichtfeld des Nutzers eine Liste mit möglichen Befehlen dargestellt wird. Zudem zeigt es den *Platzierungsmodus*, der mit dem Sprachbefehl **admin mode** aktiviert wurde. Dabei werden alle Hologramme eingeblendet und ein kleiner Indikatortext erscheint im oberen rechten Sichtfeld. In diesem Modus lassen sich die Hologramme per *Drag and Hold* Geste im virtuellen Raum bewegen. Zusätzlich können für Hologramme, über die Menüleiste unter diesen, Fixpunkte eingeblendet werden, mit denen das Hologramm in Skalierung und Rotation verändert werden kann. Dies geschieht, indem die

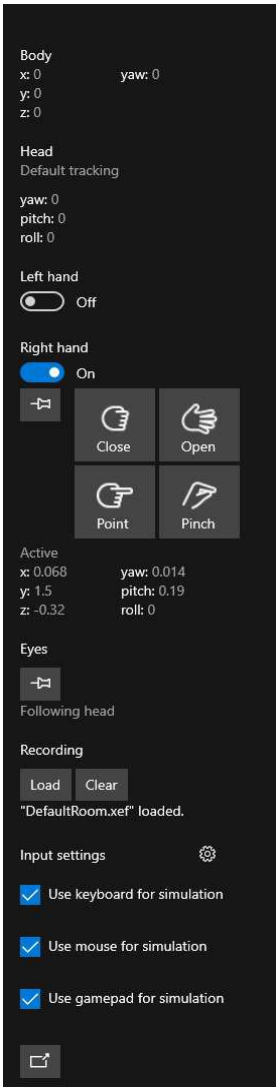

Abbildung 4.5: Darstellung des *Simulation Control Panel* [9]

Drag and Hold Geste auf einen der blauen Fixpunkte ausgeführt wird.

Vorteile und Nachteile:
Die Vorteile durch Emulatoren zeigen sich deutlich darin, dass die Anwendung nicht erst compiliert und auf die HoloLens transferiert werden muss. Dies ermöglicht ein schnelleres Testen, wenn es nur darum geht die Auswirkung durch eine kleine Änderung des Codes oder Struktur der Hologramme zu überprüfen. Zusätzlich lässt sich mit dem *HoloLens Emulator* die Anwendung auf einer emulierten Abbildung der Hardware ausführen. So

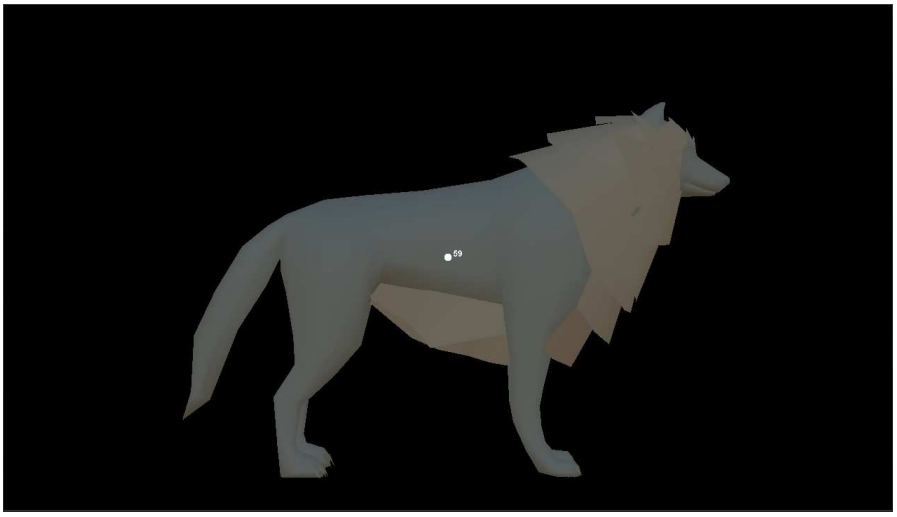

Abbildung 4.6: Darstellung eines Hologramms im Emulator (Quelle: Eigene Darstellung)

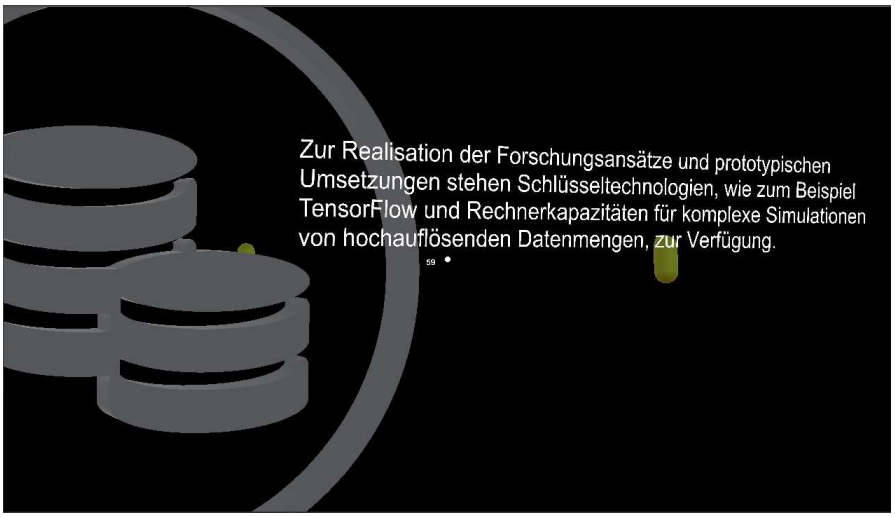

Abbildung 4.7: Darstellung des Textfeldes eines Hologramms im Emulator (Quelle: Eigene Darstellung)

kann getestet werden, ob bestimmte Teile des Codes oder die Hologramme und ihr Zusammenspiel performant sind oder es allgemein zu Problemen bestimmter Code-Teile kommen kann.

Das Benutzen von Emulatoren hat allerdings auch deutliche Nachteile. Durch die Emulatoren hat man keine Intuitive Bedienung der Anwendung. Die Immersion leidet stark,

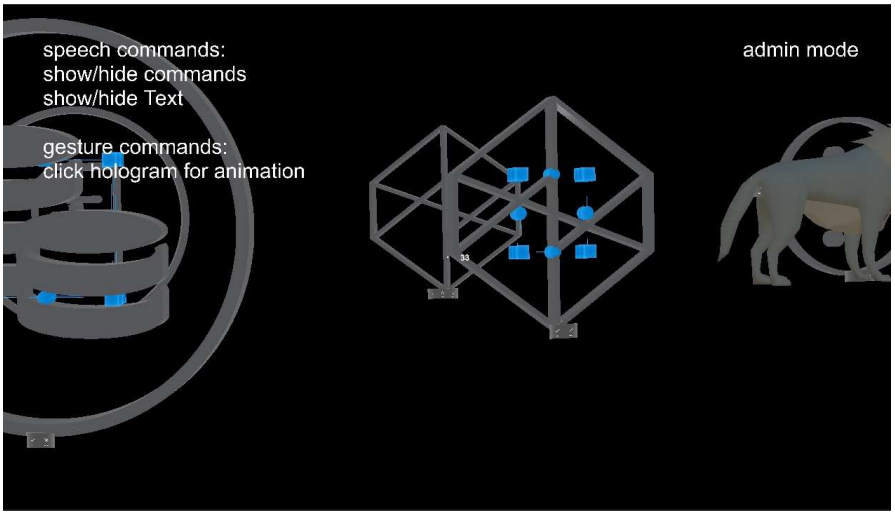

Abbildung 4.8: Darstellung des Platzierungsmodus im Emulator (Quelle: Eigene Darstellung)

da sowohl der Blick auf einen Monitor, als auch die Bedienung durch Maus und Tastatur nicht das Gefühl einer echten Verwendung der HoloLens vermitteln kann. Zudem kann es dazu kommen, dass Code-spezifische Teile für die HoloLens nicht funktionieren können. Ein Beispiel für so ein Problem sind die Ankerpunkte der Hologramme. Da der Emulator keinen eigenen persistenten internen Speicher besitzt und die Ankerpunkte im *Anchor Store* und somit auf dem lokalen Speicher der HoloLens gespeichert werden, kann der Teil der Anwendung, der die Ankerpunkte betrifft, nicht mit Emulatoren getestet werden. Zwar konnte während der Entwicklung viel Zeit mit Emulatoren eingespart werden, allerdings gab es im Vorfeld viele Hürden, die bewältigt werden mussten. Wohingegen der *Unity Emulator* auf Anhieb funktioniert hat, gab es beim *HoloLens Emulator* mehr Probleme. Das System, auf dem der *HoloLens Emulator* ausgeführt werden soll, muss Hyper-V fähig sein. Auch wenn die Hardware dies erlaubte (ein 64-Bit-Prozessor), war auf dem für die Entwicklung zu nutzenden Computer *Windows 10 Home* installiert. Bei dieser Version lässt sich Hyper-V allerdings nicht aktivieren und so musste *Windows 10* als *Education Version* installiert werden. Und auch nach dem Installieren von der neuesten Version von *Visual Studio* gab es einige Hindernisse mit fehlenden .dll Dateien, welche das System nicht finden konnte. Auch die Visualisierung eines mit der HoloLens eingescannten virtuellen Raumes, hat nicht funktioniert. Allerdings hat der Emulator vordefinierte Räume, welche stattdessen benutzt werden konnten.

4.3.2 Microsoft HoloLens

Um die Anwendung auf der HoloLens auszuführen, muss zunächst das erstellte Projekt aus *Unity* exportiert und in *Visual Studio* importiert werden. Ähnlich wie bei 4.3.1 wird also eine *Universal Windows Platform* Applikation erstellt und durch *Visual Studio* auf die HoloLens, statt einem Emulator geladen. Dies kann geschehen, indem die HoloLens per USB-Kabel direkt an den Computer angeschlossen wird und der Datentransfer darüber geschieht oder indem die HoloLens im gleichen Netzwerk (WLAN) ist. Das übermitteln der Anwendung über das Netzwerk dauert dementsprechend länger, dafür kann man aber das Remote-Debugging benutzen. Hingehen bei der Übertragung mit dem USB-Kabel würde das Debugging nur funktionieren, solange das Kabel angeschlossen bleibt, was wiederum die Bewegung des Nutzers einschränken würde. Zusätzlich kann man auf das *Device Portal* zugreifen, wenn die HoloLens mit dem selben Netzwerk verbunden ist. Hier können zum Beispiel die Einstellungen, Hardware Auslastung, die Sicht des Nutzers und die Position im virtuellen Raum angezeigt werden. So kann eine weitere Person die HoloLens benutzen und der Entwickler am Computer überprüfen, wie sich die HoloLens intern verhält beziehungsweise sich die Sicht des Nutzers anzeigen lassen. Mit diesem Verfahren wurden auch die Abbildungen dieser Arbeit, wie zum Beispiel Abbildung 4.9, erstellt.

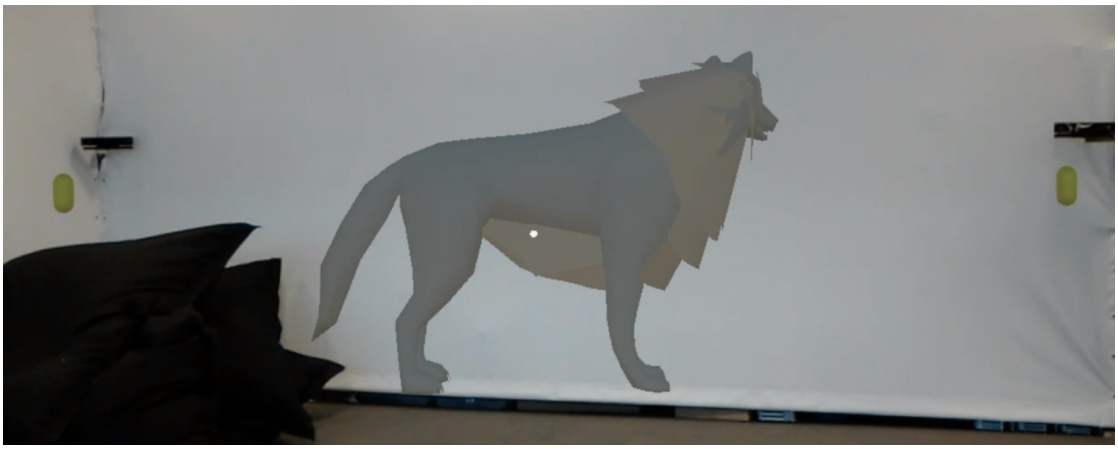

Abbildung 4.9: Darstellung eines Hologramms bei Benutzung der HoloLens (Quelle: Eigene Darstellung)

4.4 Evaluation

In diesem Abschnitt wird die aus dem Design entstandene Anwendung für die HoloLens überprüft. Dazu werden die aus der Analyse entstandenen Anforderungen betrachtet und erläutert, ob diese erfüllt wurden. Dabei wird unterschieden, falls eine Anforderung nur zum Teil oder gar nicht erfüllt wurde, ob dies ein generelles Problem der Hardware oder ein Problem der Software ist. Zudem werden kurz alternative Möglichkeiten der Umsetzung der Anwendung erläutert und eine Einschätzung über die zukünftige Verwendung und Erweiterbarkeit der Anwendung gegeben.

Anforderungen

Alle aus der Anforderungsanalyse definierten funktionalen Anforderungen konnten in der Anwendung umgesetzt werden. Die im virtuellen Raum platzierten Hologramme können fokussiert und angeklickt werden. Bei der Bedienung sind keine negativen Eigenschaften aufgefallen. Die Erkennung vom *Gaze* und den *Gesten* funktioniert mit der HoloLens tadellos. Bei der Spracherkennung wurden selbst definierte Wörter benutzt und diese werden bei sauberer Aussprache immer erkannt. Zudem wird dem Nutzer immer nur das Hologramm angezeigt, welches seiner Position am nächsten ist. Bei dem Platzierungsmodus, welcher per Spracheingabe aktiviert wird, lassen sich alle Hologramme mit den definierten Gesten verschieben und skalieren. Auch die Persistenz der Position der Hologramme konnten über den internen Speicher der Anwendung umgesetzt werden. Dadurch konnte auf eine externe Datenbank, welche diese Information speichern müsste, verzichtet werden.

Nicht funktionale Anforderungen

Bedienbarkeit:
Bei der Anforderung der Bedienbarkeit war die intuitive Bedienung der Anwendung das wichtigste Kriterium. Sowohl die Sprachbefehle als auch die Gesten sind simpel gewählt und funktionieren auf diese Weise. Bei den Sprachbefehlen muss lediglich das Wissen vorhanden sein, welche Befehle es gibt. Der wichtigste Befehl hier ist **show commands**, da sich mit diesem alle Möglichkeiten der Bedienung anzeigen lassen können. Da Sprache generell sehr intuitiv funktioniert gab es keine Einschränkungen in Bezug auf Befehle,

welche ohne den *Gaze* funktionieren. Lediglich wenn die Kombination *Gaze* und Sprache benutzt werden musste, kam es zu kleinen Anwendungsschwierigkeiten. Wenn der Text zu einem Hologramm durchgelesen wurde und der nächste Text angezeigt werden sollte, konnte es passieren, dass der Nutzer beim Aussprechen des Befehls **next** nicht gleichzeitig auf das Hologramm oder den Text schaute. Somit wurde nichts fokussiert und die *Collection* konnte den Befehl nicht erhalten. Dies könnte in Zukunft verhindert werden, wenn nicht die *Collection* zuständig für das Einblenden der Texte wäre, sondern der *ObjectManager* diesen Befehl global erkennt und diesen dann an die aktiv angezeigte *Collection* durchreicht. Dies ist demnach ein Problem der Anwendung und nicht der Erkennungshardware der HoloLens. Bei den Gesten wurden keine komplizierten Bewegungen genutzt, was diese Art der Bedienung sehr intuitiv macht. Die Geste, dass ein Hologramm angeschaut wird und mit dem Finger angetippt werden kann, ist vergleichbar mit der Benutzung von Smartphones. Auch hierbei ist es keine Seltenheit mehr, einzelne Elemente auf einem Bildschirm mit dem Finger antippen zu können, beziehungsweise diese mit dem Finger zu verschieben. Diese Bewegung wurde in den dreidimensionalen Raum übertragen und aus dem wischen des Fingers wurde *Drag and Hold*.

Performance:
Die Hardware, welche die HoloLens zur Verfügung stellt, haben für die Umsetzung der Anwendung ausgereicht. Während der Entwicklung wurde mit Hilfe einer FPS (Frames per Second) Anzeige durchgehend überprüft, wie performant die aktuelle *Scene* ist. Dabei stellte sich heraus, dass die Performance durch verschiedene Teile der Anwendung beeinträchtigt werden konnte. Als erstes muss hier deutlich die Anzahl der Hologramme genannt werden. Der Vorteil der HoloLens besteht darin, dass nur Hologramme gerendert werden, welche im aktuellen Sichtfeld des Nutzers liegen. Bei der normalen Benutzung der Anwendung würde also dieses Problem nie auftauchen, da immer nur ein Hologramm, nämlich das von der Position am nächsten liegende, dargestellt wird. Anders sieht es da beim Platzierungsmodus aus, denn dort werden alle im Sichtfeld liegenden Hologramme gerendert. Da die Anwendung auf verteilte Hologramme basiert, ist es hier im Endeffekt aber zu keiner Einschränkung gekommen, lediglich zu Testzwecken wurde der virtuelle Raum mit Hologrammen überfüllt, um nachvollziehen zu können, wie sehr die Leistung beeinträchtigt wird. Ein weiterer Punkt, welcher in Verbindung mit der Performance genannt werden muss, ist die Umsetzung der ständigen Überprüfung der Positionen des Nutzers und der Hologramme. Zuerst wurde hier mit der Funktion **Update()** gearbeitet, was zur Folge hatte, dass die Abfrage der Positionen zu häufig geschehen ist. Diese wird pro Frame einmal ausgerufen und da die Aufrufhäufigkeit abhängig von den FPS

ist, bedeutet dies, je höher die FPS desto mehr Aufrufe pro Sekunde. Dadurch gingen im Allgemeinen die FPS runter und zudem schwankten diese erheblich, denn bei hoher FPS gab es mehr Aufrufe, was die FPS kurzzeitig verringerte, was wiederum dazu führte, dass es weniger Aufrufe gab, was die FPS wieder erhöhte. Das erzeugte einen Kreislauf der die Darstellung erheblich einschränkte. Somit schwankten die FPS bei normaler Benutzung zwischen 15-25 FPS, statt auf stabilen 60 FPS zu bleiben. Dies wurde behoben, indem die Funktion **Update()** mit der Funktion **InvokeRepeating()** ersetzt wurde. Bei dieser kann man die Häufigkeit des Aufrufes, unabhängig von den FPS, durch ein Zeitintervall festlegen, wobei hier einmal pro Sekunde gewählt wurde. Dies verringerte die Anzahl der Aufrufe also von 60 (bei konstanten 60 FPS) auf einen FPS. Als letztes müssen noch die Eigenschaften der Hologramme und der *Scene* erwähnt werden. Die *Scene* enthält im Normalfall ein Beleuchtungsobjekt, damit Hologramme Schattierungen und auch Spiegelungen haben können. Wenn ein Material als Komponente einem Hologramm hinzugefügt wird, welches zum Beispiel Metall nachempfunden ist, spiegelt das Hologramm dieses virtuelle Licht. Diese Berechnung der Materialeigenschaften ist sehr aufwändig und deswegen muss bei der Auswahl und Umsetzung der Materialien für Hologramme sehr genau gearbeitet werden. Das MRT liefert hier vordefinierte Materialien, von denen bei dieser Arbeit eine ausgewählt und benutzt wurde.

Optik:
Der Nutzer sollte bei der Verwendung der HoloLens nicht das Gefühl haben, dass die Hologramme sich optisch nicht in die Welt einfügen. Da sich bei den Hologrammen auf dreidimensionale Symbole und keine Aufwendigen farbenfrohen Objekte beschränkt wurde, ist die Optik schlicht aber dafür Fehlerfrei. Die Animationen sind flüssig und die Performance wird durch sie nicht beeinflusst, was eventuelles Ruckeln oder Auflösungsfehler verhindert. Da die Hologramme in der *Spatial Map* platziert werden, entsteht eine Verdeckung, was die Hologramme wie echte Objekte erscheinen lässt. Im nächsten Punkt der Fehlertoleranz wird darauf noch weiter eingegangen.

Fehlertoleranz:
In Bezug auf die Eingabemöglichkeiten wurden auf sehr simple Gesten und Sprachbefehle gesetzt. Diese konnten so umgesetzt werden, dass es nur sehr selten zu Fehlern kommt. Die Steuerung per Gesten funktioniert einwandfrei. Wenn der *Cursor* auf dem Hologramm angezeigt wird, werden die Eingabegesten des *Air Tap* und *Drag and Hold* gut erkannt und es kam dort zu keinen Fehleingaben beziehungsweise ungewollten Bewegungen. Lediglich bei der Spracheingabe kommt es zu kleinen Fehlerquellen. Zwar werden die benutzerdefinierten Befehle gut erkannt, wenn man diese sauber und laut

Abbildung 4.10: Darstellung der Anwendung bei falscher Kantenerkennung (Quelle: Eigene Darstellung)

ausspricht, allerdings kann es während der Benutzung der Anwendung dazu kommen, dass der Nutzer darauf nicht genügend Wert legt und Wortsilben verschluckt oder undeutlich ausspricht. So können diese nicht immer von der Anwendung erkannt werden, wobei erwähnt werden muss, dass die Erkennung im generellen doch häufiger funktionierte und das Nichterkennen eher eine Seltenheit darstellt. In Bezug auf die Registration und die Verdeckungsberechnung ist aufgefallen, dass die Fehlerquellen sich dort mehr bemerkbar gemacht haben. Zwar funktioniert die Positionierung der Hologramme und die Erkennung der Position des Nutzers mit der HoloLens ohne Fehler, aber gerade die Verdeckungsberechnung funktioniert nicht perfekt. In Abbildung 4.10 kann man sehen, dass die Kante des Monitors nicht korrekt erkannt wurde und so mehr vom Hologramm verdeckt wird als es sollte. Dies kann allerdings auch dadurch passieren, dass der Monitor erst kurz vorher bewegt wurde, und die *Spatial Map* sich in dem Zeitraum noch nicht aktualisiert hat und sich so in der virtuellen Umgebung noch ein Gegenstand befindet, welcher die Sicht blockieren würde. Eine weitere Fehlerquelle sind unvollständige *Spatial Map* Daten durch Nichterkennung. Wenn das Scannen des Raumes noch nicht vollständig abgeschlossen ist, kann es dazu kommen, dass die *Mesh* der *Spatial Map* Lücken beziehungsweise Löcher aufweist. Dies kann in Abbildung 4.11 betrachtet werden. Dort wird das Hologramm des Wolfes durch die beiden Bildschirme hindurch angezeigt, so als wäre

das Hologramm teilweise vor den Monitoren. Dies kann aber verhindert werden, wenn dementsprechend eine statische Umgebung benutzt wird, welche im Vorfeld ausgiebig und lange genug eingescannt wurde.

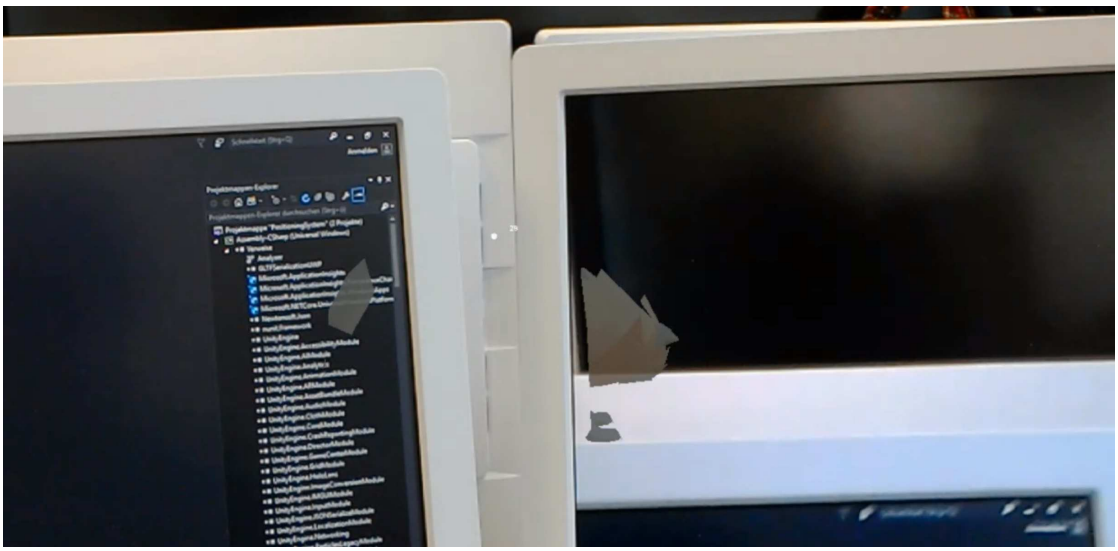

Abbildung 4.11: Darstellung der Anwendung bei lückenhafter *Spatial Map* (Quelle: Eigene Darstellung)

Skalierbarkeit und Wartbarkeit:
Die Anwendung ist so gestaltet, dass diese sich einfach Skalieren lässt. Neue Hologramme können als *Collection* dem *ObjectManager* hinzugefügt werden. Dabei ist wichtig, dass es keine Begrenzung in der Anzahl der *Collections* gibt. Zudem können beliebig viele Textfelder mit Informationen hinzugefügt werden, wobei darauf zu achten ist, dass das Durchklicken durch zu viele Texte eines Hologramms zu müßig werden könnte. Falls neue Gesten oder Sprachbefehle eingebaut werden sollen, müssen diese im zugehörigen Manager registriert werden und dem Hologramm eine weitere Funktion hinzufügt werden, die bestimmt, was bei diesem Befehl geschehen soll. Da man dies in den allgemeinen Scripten in den *Assets* deklariert, bekommen automatisch alle Manager und Hologramme diese Logik. Da also die Logik des Verhaltens der Hologramme durch die Eingabefunktionen zentral gesteuert wird, lässt sich dies gut Warten. Falls ein bestimmtes Verhalten verändert werden soll, muss dies nur einmal in der Anwendung geändert werden. Auch neu hinzugefügte Hologramme benötigen nur die Zuteilung dieser Scripte als Komponente und erhalten somit die gesamte Logik.

Zukunft

Diese Arbeit hat das Design als eine simple Anwendung umgesetzt um lediglich die Durchführbarkeit der Grundprinzipien zu testen. Mit weiterem Arbeitsaufwand würden sich gegebenenfalls einige Fehlerquellen minimieren oder ganz eliminieren lassen. Auch die Performance dieser Anwendung könnte sich mit besserer Hardware steigern. Weitere Features der *Recommender Systems*, welche hier durch das durchblättern der Texte dargestellt wurde, könnten implementiert werden. Dadurch würden sich für unterschiedliche Nutzer unterschiedliche Hologramme anzeigen lassen. Auch das exportieren der *Spatial Map* und der Ankerpunkte könnte automatisiert werden, um anderen Nutzern mit einer weiteren HoloLens zu ermöglichen, diese zu importieren. So könnte die Anwendung in einer komplett neuen Umgebung direkt inklusive allen Einstellungen verwendet werden.

5 Fazit und Ausblick

5.1 Fazit

In dieser Arbeit wurde ein Prototyp für die HoloLens, zur ortsbasierenden Anzeige von Hologrammen in der Augmented Reality entwickelt. Diese wurde anschließend evaluiert, um festzustellen, ob das Konzept umgesetzt werden konnte und inwieweit die vorhandene Technik dafür fortgeschritten ist. Durch den Entwicklungsprozess der Anwendung konnte aufgezeigt werden, dass ein leichter Einstieg durch vorhandene Tools wie *Unity* und besonders dem *Mixed Reality Toolkit* möglich ist. Dadurch lassen sich alle Funktionen, welche die HoloLens bietet, schnell integrieren und verwenden. Die vorhandenen Interaktionsmöglichkeiten funktionieren zwar alle so wie gedacht, allerdings sind diese momentan noch sehr ungewohnt. Trotz der einfach gewählten Steuerung durch Sprache und simple Gesten, ist die Bedienung nicht intuitiv. Dies liegt unter anderem daran, dass man nicht gewohnt ist, Objekte in der realen Welt aus der Entfernung zu steuern beziehungsweise mit gesprochenen Worten zu manipulieren. Hingegen überzeugend ist die grafische Darstellung der Hologramme und ihre Platzierung, wodurch sie wie echte Objekte wirken. Im Indoorbereich funktioniert die HoloLens sehr gut, denn durch die Sensoren wird die Umgebung fast makellos erkannt und kann so durch Anwendungen gut verarbeitet werden. Dabei muss aber betont werden, dass das Sichtfeld, in dem die Hologramme angezeigt werden können, ein wenig zu klein ist und die HoloLens ausschließlich für die Benutzung im Indoor Bereich ausgelegt ist. Für den Outdoorbereich müsste die Raumerkennung angepasst werden, wobei hier der Mangel an Speicherkapazität und die sich schnell verändernde Umgebung die wichtigsten Hindernisse darstellen. Wenn die Platzierung der Hologramme allerdings durch Markertracking oder hinzufügen von GPS-Daten externer Hardware ersetzt werden würde, könnte das erarbeitete Konzept dieser Arbeit auch für den Outdoorbereich verwendet werden. Dadurch, dass die Anzeige von Objekten ortsgebunden ist, kann dieses Konzept überall eingesetzt werden, wo an festen Punkten unterschiedliche Informationen virtuell dargestellt werden sollen.

Wenn also die Hardware der HoloLens durch externe Datenbanken, welche Karten- und Objektinformationen enthalten und weitere Sensoren wie GPS, erweitert werden würde, könnte die HoloLens in vielen weiteren Einsatzgebieten Verwendung finden. Die HoloLens verfügt sogar über geeignete Signalschnittstellen wie Wi-Fi und Bluetooth, um dies umzusetzen.

5.2 Ausblick

Obwohl es das Konzept der Augmented Reality schon sehr lange gibt, befinden wir uns in einer frühen Phase der Hardwareentwicklung. Die Hardware wird stetig verbessert und so wird die Augmented Reality immer mehr Teil des aktuellen Lebens. Das Interesse an solchen Systemen wird auch vorerst nicht abklingen, denn es werden immer mehr Informationen bereitgestellt, welche für den Menschen von Bedeutung sind und für diesen auf angenehme Art und Weise dargestellt werden müssen. Je nachdem, was von so einem System gefordert wird, entscheidet über die Hardware und Software, welche verwendet werden muss. Allerdings ist durch die zunehmende Anzahl an Smartphones auch die Verfügbarkeit von AR fähigen Geräten gestiegen, welche hierfür verwendet werden können. Zudem erscheinen immer mehr Konkurrenzprodukte im Bereich der Head-Mounted-Displays, welche auch für den privaten Gebrauch und nicht nur für den professionellen Bereich geeignet sind. Die Bereiche, in denen die Augmented Reality zukünftig Anwendung finden kann, sind dementsprechend sehr weitläufig. Der Bereich der ortsbasierenden Anzeige von Informationen erstreckt sich von platzierter individualisierter Werbung, also dem wirtschaftlichen Bereich, bis hin zum Entertainment, also dem privaten Bereich. In der Vergangenheit versuchte Projekte, die Head-Mounted-Displays in den Alltag der Menschen zu integrieren, sind nach aktuellem Wissen zwar gescheitert (*Google Glass*), allerdings bleibt abzuwarten, ob dies auch in Zukunft so bleibt. Denn durch verbesserte Hardware, welche auch immer kleiner wird und somit besseren Tragekomfort bietet, lässt sich vermuten, dass die Vielfalt der Einsatzgebiete von AR in Kombination mit HMD stark zunehmen wird.

Literaturverzeichnis

[1] *Unity*. November 2018. – URL https://unity.com/de. – Zugriffsdatum: 17.01.2019

[2] *Blender*. 2019. – URL https://www.blender.org/. – Zugriffsdatum: 23.11.2018

[3] *Mixed Reality Toolkit*. 2019. – URL https://microsoft.github.io/MixedRealityToolkit-Unity/README.html. – Zugriffsdatum: 05.05.2019

[4] AG iTiZZiMO: *Die Wahrheit über die HoloLens*. Mai 2015. – URL https://www.itizzimo.com/die-wahrheit-ueber-hololens/. – Zugriffsdatum: 23.11.2018

[5] ARCore: *ARCore overview*. Februar 2019. – URL https://developers.google.com/ar/discover/. – Zugriffsdatum: 22.05.2019

[6] Arth, Clemens ; Grasset, Raphaël ; Gruber, Lukas ; Langlotz, Tobias ; Mulloni, Alessandro ; Wagner, Daniel: The History of Mobile Augmented Reality. In: *CoRR* abs/1505.01319 (2015). – URL http://arxiv.org/abs/1505.01319

[7] Bai, Huidong ; Lee, Gun ; Billinghurst, Mark: Using 3D Hand Gestures and Touch Input for Wearable AR Interaction. In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2014 (CHI EA '14), S. 1321–1326. – URL http://doi.acm.org/10.1145/2559206.2581371. – ISBN 978-1-4503-2474-8

[8] Bajura, M. ; Neumann, U.: Dynamic Registration Correction in Augmented-reality Systems. In: *Proceedings of the Virtual Reality Annual International Symposium (VRAIS'95)*. Washington, DC, USA : IEEE Computer Society, 1995 (VRAIS '95), S. 189–. – URL http://dl.acm.org/citation.cfm?id=527216.836008. – ISBN 0-8186-7084-3

[9] Barnett, Paul: *Using the HoloLens Emulator*. April 2019. – URL https://docs.microsoft.com/de-de/windows/mixed-reality/using-the-hololens-emulator. – Zugriffsdatum: 02.05.2019

[10] Brandon Bray, Matt Z.: *What is mixed reality?* März 2018. – URL https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality. – Zugriffsdatum: 05.01.2019

[11] Brill, M.: *Virtuelle Realität*. Springer Berlin Heidelberg, 2008 (Informatik im Fokus). – URL https://books.google.de/books?id=NQEOmAEACAAJ. – ISBN 9783540851189

[12] Carmigniani, Julie ; Furht, Borko: *Augmented Reality: An Overview*. 07 2011. – 3–46 S

[13] Casa, Emiliano D.: *Milgram's Continuum: Best pick-up line ever*. Januar 2018. – URL https://hackernoon.com/milgrams-continuum-best-pick-up-line-ever-cf9e335ea813. – Zugriffsdatum: 12.01.2019

[14] Erickson, Scott: *Microsoft HoloLens enables thyssenkrupp to transform the global elevator industry*. September 2016. – URL https://blogs.windows.com/devices/2016/09/15/microsoft-hololens-enables-thyssenkrupp-to-transform-the-global-elevator-#GCeCmcm9t3vGby6E.97. – Zugriffsdatum: 23.05.2019

[15] Grabham, Dan: *Microsoft HoloLens 2 features, news and release date: Everything you need to know*. Februar 2019. – URL https://www.pocket-lint.com/ar-vr/news/microsoft/146803-microsoft-hololens-2-specs-release-date-news-features. – Zugriffsdatum: 12.04.2019

[16] Kasperi, Johan ; Edwardsson, Malin P. ; Romero, Mario: Occlusion in Outdoor Augmented Reality Using Geospatial Building Data. In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. New York, NY, USA : ACM, 2017 (VRST '17), S. 30:1–30:10. – URL http://doi.acm.org/10.1145/3139131.3139159. – ISBN 978-1-4503-5548-3

[17] LEAP MOTION, INC: *Leap Motion's software and hardware platform brings your bare hands directly into virtual and augmented reality.* 2019. – URL https://www.leapmotion.com/technology/. – Zugriffsdatum: 17.01.2019

[18] LIBRARY, ACM D.: *ACM Digital Library.* Mai 2019. – URL https://dl.acm.org/results.cfm?query=augmented%20reality&filtered=&within=owners%2Eowner%3DHOSTED&dte=1990&bfr=&srt=%5Fscore. – Zugriffsdatum: 12.05.2019

[19] MATT ZELLER, et a.: *Spatial mapping.* März 2018. – URL https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping. – Zugriffsdatum: 27.02.2019

[20] MATT ZELLER, et a.: *Spatial sound.* März 2018. – URL https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-sound. – Zugriffsdatum: 27.02.2019

[21] MATT ZELLER, et a.: *Voice input.* Februar 2019. – URL https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input. – Zugriffsdatum: 26.02.2019

[22] MICROSOFT: *Use gestures.* Mai 2018. – URL https://support.microsoft.com/de-de/help/12644/hololens-use-gestures. – Zugriffsdatum: 27.02.2019

[23] MILGRAM, Paul ; KISHINO, Fumio: A Taxonomy of Mixed Reality Visual Displays. vol. E77-D, no. 12 (1994), 12, S. 1321–1329

[24] MILGRAM, Paul ; TAKEMURA, Haruo ; UTSUMI, Akira ; KISHINO, Fumio: Augmented reality: A class of displays on the reality-virtuality continuum. 2351 (1994), 01

[25] PAAVILAINEN, Janne ; KORHONEN, Hannu ; ALHA, Kati ; STENROS, Jaakko ; KOSKINEN, Elina ; MAYRA, Frans: The PokéMon GO Experience: A Location-Based Augmented Reality Mobile Game Goes Mainstream. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* New York, NY, USA : ACM, 2017 (CHI '17), S. 2493–2498. – URL http://doi.acm.org/10.1145/3025453.3025871. – ISBN 978-1-4503-4655-9

[26] RALF, Dörner ; WOLFGANG, Broll ; PAUL, Grimm ; BERNHARD, Jung: *Virtual und Augmented Reality (VR / AR) Grundlagen und Methoden der Virtuellen und*

Augmentierten Realität. 2013. – URL http://www.worldcat.org/search?qt=worldcat_org_all&q=9783642289026

[27] Seidl, Maik: *Pokémon GO: The Pokémon Company nennt Downloadzahl*. Mai 2018. – URL https://www.playm.de/2018/05/pokemon-go-35-403807/. – Zugriffsdatum: 16.04.2019

[28] T. Azuma, Ronald: A Survey of Augmented Reality. 6 (1996), 02

[29] Team, Windows A.: *Getting Started with a Mixed Reality Platformer Using Microsoft HoloLens*. Juni 2018. – URL https://blogs.windows.com/buildingapps/2017/02/27/getting-started-mixed-reality-platformer-using-microsoft-hololens/#H2goowzQwzJBACCI.97. – Zugriffsdatum: 13.12.2018

[30] Tönnis, Marcus: *Augmented Reality: Einblicke in die Erweiterte Realität*. Heidelberg : Springer, 2010 (Informatik im Fokus). – ISBN 978-3-642-14178-2

[31] Ulbricht, Christiane ; Schmalstieg, Dieter: Tangible Augmented Reality for Computer Games / Institute of Computer Graphics and Algorithms, Vienna University of Technology. Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Februar 2003 (TR-186-2-03-02). – Forschungsbericht. – URL https://www.cg.tuwien.ac.at/research/publications/2003/Ulbricht-2003-TanX/. human contact: technical-report@cg.tuwien.ac.at

[32] Zeller, Matt: *HoloLens (1. Generation) Hardwaredetails*. März 2018. – URL https://docs.microsoft.com/de-de/windows/mixed-reality/hololens-hardware-details. – Zugriffsdatum: 23.11.2018

[33] Zhang, Zhuo ; Shang, Shang ; Kulkarni, Sanjeev R. ; Hui, Pan: Improving Augmented Reality Using Recommender Systems. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. New York, NY, USA : ACM, 2013 (RecSys '13), S. 173–176. – URL http://doi.acm.org/10.1145/2507157.2507211. – ISBN 978-1-4503-2409-0

[34] Zhu, Jiejie ; Pan, Zhigeng: Occlusion registration in video-based augmented reality, 01 2008

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:

Augmented Reality mit der Microsoft HoloLens im Indoor Bereich

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