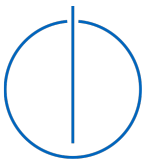
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik: Games Engineering

Konzepte zur Realisierung von umgebungsabhängigen Benutzeroberflächen für die Anwendung in Mixed-Reality-Szenarios

Oliver Jung



FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik: Games Engineering

Konzepte zur Realisierung von umgebungsabhängigen Benutzeroberflächen für die Anwendung in Mixed-Reality-Szenarios

Concepts for realizing environment dependent user interfaces in mixed reality scenarios

Autor:	Oliver Jung
Aufgabenstellerin:	Prof. Gudrun Klinker, Ph.D.
Betreuer:	Sven Liedtke, M.Sc.
Abgabedatum:	15. März, 2019

Ich versichere, dass ich diese Bachelorarbeit in Informatik: Games Engineering selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 15. März, 2019 Oliver Jung

Danksagungen

I want to acknowledge

- . . . acknowledgement 01,
- . . . acknowledgement 02,
- . . . acknowledgement 03.

Zusammenfassung

English abstract text. Eingrenzung des Forschungsbereichs Beschreibung des Problems Mängel an existierenden Arbeiten bzgl. des Problems x Eigener Lösungsansatz x Art der Validierung + Ergebnisse x

German abstract text.

Durch die Entwicklungen der letzten Jahre in den Gebieten der virtuellen und erweiterten Realität, welche eine Vielzahl an verschiedenen Umsetzungen von sogenannten Head-Mounted-Displays mit sich brachten, wird klar, dass diese Bereiche in Zukunft eine immer größere Rolle spielen könnten. Diese Arbeit beschäftigt sich mit dem Thema der Platzierung von Benutzungsoberflächen an Körperpositionen, welche durch das Programm erfasst werden können, und genauer mit der Behandlung des Wegfallens solcher Positionen.

Hier etwas einfügen

cross platform?

Inhaltsverzeichnis

1 Einleitung

Alle Bildpositionen überprüfen!!!!

Die Begriffe virtuelle Realität und erweiterte Realität kennen heutzutage viele Menschen. Allerdings wissen einige davon nicht wirklich was sich genau dahinter verbirgt. Dabei handelt es sich hierbei um zwei Forschungsbereiche, in welchen viele Firmen ein großes Potential sehen. Noch sind sie zwar unwichtig für die meisten privaten Nutzer, da die Anschaffung der bekanntesten notwendigen Geräte relativ kostspielig ist, aber In Zukunft werden diese Bereiche voraussichtlich auch dort noch weiter an Wichtigkeit gewinnen. Dies lässt sich bereits an den starken Entwicklungen auf dem Markt der sogenannten VR- und AR-Brillen in den letzten Jahren erkennen. Zwar gab es laut dem Marktforschungsunternehmen IDC einen Rückgang der Verkaufszahlen im Jahr 2018, die Prognosen für die kommenden Jahre sehen allerdings vielversprechend für diese Technologien aus, wie Abbildung 1.1 zeigt [1].

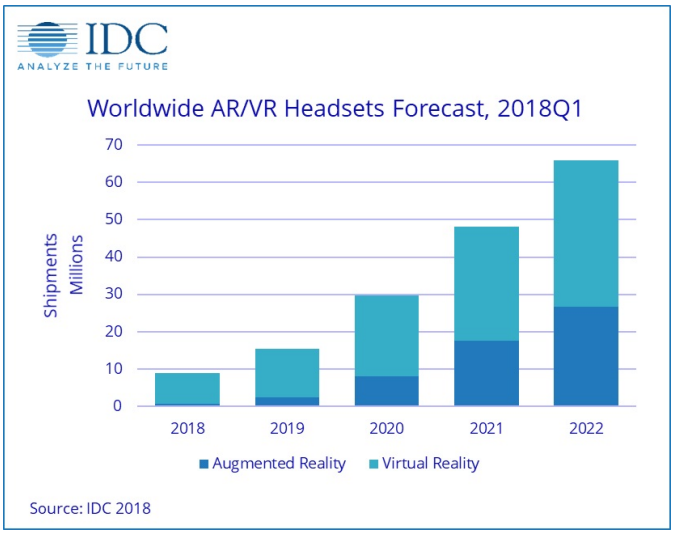


Abbildung 1.1: Vorhersage der Verkaufszahlen nach IDC 2018 [genommen von 1].

Diese speziellen Brillen dienen als Medium, um Benutzern einen Einblick in eine teilweise oder gänzlich veränderte Welt zu geben. Neben bereits bekannteren Exemplaren wie der HTC Vive von den Firmen HTC und Valve, der Oculus Rift von Oculus VR oder der Hololens von Microsoft, gibt es nun auch billigere Umsetzungen wie die Daydream

View von Google, welche das eigene Smartphone als Display verwendet. Durch diese billigeren Varianten ist es auch mehr privaten Nutzern möglich ein eigenes VR-Gerät zu erwerben und auch zuhause einen Blick in virtuelle Welten zu werfen. Zudem bieten die teureren Modelle viele Möglichkeiten für die Anwendung in Unternehmen. Es wird also durch verschiedene Preiskategorien ein Angebot für jedermann erzeugt. Es wird allerdings davon ausgegangen, dass die Verkäufe der Varianten von VR und AR, welche mit Hilfe von Smartphones funktionieren, durch ihre geringe Leistungsfähigkeit in Zukunft wieder zurückgehen werden [2]. Für andere Modelle wird eine Steigerung der Verkaufszahlen erwartet [1][2]. Zudem werden weiterhin verbesserte Versionen älterer Geräte veröffentlicht. Ein Beispiel dafür ist die HTC Vive Pro, welche sowohl in der virtuellen als auch, im Gegensatz zu ihrem Vorgängermodell, in der erweiterten Realität angewendet werden kann.

Natürlich ist die Hardwareseite nicht der einzige Punkt, an dem weitergeforscht wird. Auf der Seite der Software für diese Forschungsgebiete hat sich ebenfalls viel weiterentwickelt. Sowohl im Bereich der Unterhaltung als auch im Bereich der produktiven Nutzung, wie zum Beispiel in Form von Lernsimulationen. Allerdings sind die meisten dieser Programme nur für ein bestimmtes Gerät entwickelt worden und sind somit auf anderen Modellen nicht verwendbar. Es gibt aber auch schon einige wenige Umsetzungen von Spielen für sogenanntes *Crossplay* in der virtuellen Realität [3]. Dies bedeutet, dass diese Spiele auf mehreren Plattformen spielbar und nicht nur auf eine einzelne beschränkt sind. Bei deren genauerer Betrachtung fällt allerdings auf, dass dabei innerhalb des tatsächlichen Spiels fast komplett auf Benutzeroberflächen verzichtet wird. Somit werden zwar ein paar Schwierigkeiten umgangen, die sonst auftreten könnten, aber gleichzeitig wird das Potential von erfassten Controller- oder Handpositionen zum Anzeigen von Informationen nicht genutzt. Deshalb setzt sich diese Arbeit mit diesem Thema und möglichen Umsetzungen auseinander.

Das Ziel dieser Arbeit ist es, ein Konzept zu entwickelt, welches die Verwendung eines Programms auf mehreren Plattformen des VR- und AR-Bereichs erleichtert. Der Fokus liegt dabei auf der Platzierung der Nutzeroberfläche an verfügbaren Ankerpositionen aus der realen Welt und der Behandlung von Veränderung der Anzahl solcher Ankerpunkte.

1.1 Definitionen von virtueller und erweiterter Realität

Bei der Definition von virtueller Realität (VR) und erweiterter Realität (AR, engl. Augmented Reality) und deren Abgrenzung zueinander gibt es relativ große Meinungsunterschiede. Der Designer *Tidjane Tall* zum Beispiel beschreibt diese beiden Begriffe in seinem Artikel *Augmented Reality vs. Virtual Reality vs. Mixed Reality – An Introductory*

Guide als zwei verschiedene Technologien, welche in Mixed Reality vereint werden. VR ist dabei eine Art Simulation, in der es dem Nutzer ermöglicht wird eine andere Umgebung zu sehen, als die, in der er sich tatsächlich befindet. Die erweiterte Realität hingegen definiert er durch die Projizierung computergenerierter Objekte in die reale Umgebung in Form einer Maske, welche die Realität überlagert. Dieses Prinzip wird beispielsweise in AR-Anwendungen für Smartphones verwendet.[4]

Als Alternative dazu definiert *R. Azuma* die erweiterte Realität mit drei klaren Eigenschaften, welche den Bereich von AR weiter einschränken. Sein erster Punkt ist, dass in einem Verfahren eine Vermischung von virtuellen Bausteinen mit der Realität stattfinden müsse, damit es sich bei dessen Produkt um eine erweiterte Realität handeln kann. Um dabei allerdings Randfälle wie Bildfilter oder die simple Einblendung von Inhalten über der realen Welt auszuschließen, hat er zudem noch hinzugefügt, dass die virtuellen Elemente innerhalb der dreidimensionalen Welt platziert sein müssen. Zuletzt entfernt er noch Fälle, wie computergenerierte Inhalte innerhalb von Filmen, aus seiner Definition der erweiterten Realität, indem er nach der Möglichkeit zur Echtzeitinteraktion mit den virtuellen Inhalten verlangt.[5]

Eine der verbreitetsten Definitionen stammt allerdings von *Milgram et al.* aus deren Arbeit *Augmented Reality: A class of displays on the reality-virtuality continuum*. Dabei wird der Bereich der Mixed-Reality als der Übergang zwischen der realen Welt und der virtuellen Welt definiert [6]. Die beiden Enden zählen allerdings jeweils nicht zu diesem sogenannten Reality-Virtuality Continuum [6]. Dieses Modell veranschaulicht Abbildung 1.2. In dieser Darstellung sieht man allerdings ebenfalls, dass die erweiterte Realität innerhalb des Kontinuums nicht weiter eingegrenzt wurde, weshalb im Nachfolgenden für diesen Bereich die Definition von Azuma verwendet wird. Ansonsten gilt das Modell von *Milgram et al.* .

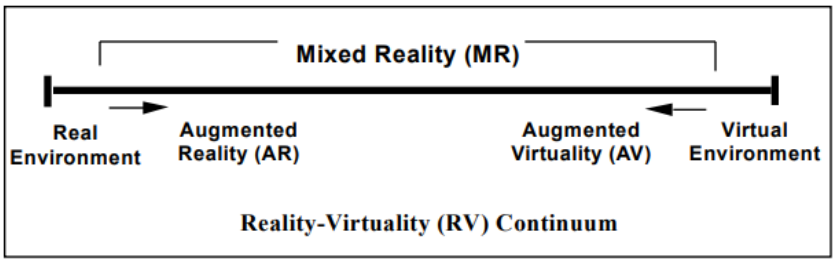


Abbildung 1.2: Das Reality-Virtuality Continuum nach *Milgram et al.* [genommen von 6].

Zur Umsetzung von VR und AR gab es in der Vergangenheit bereits mehrere Ideen, welche auf unterschiedliche Szenarien zugeschnitten sind. Ein Beispiel dafür ist das HUD (Head-Up Display), welches teilweise im Flugverkehr Anwendung findet [5].

Die Variante mit einem HMD (Head Mounted Display) hat sich bisher allerdings in vielen Bereichen durchgesetzt, da sie durch die vollständige Kontrolle des Sehsinns ein hohes Maß an Immersion bietet und dabei nicht zwingend viel Platz verbraucht. Die Verwendung von Systemen zur Positionserkennung von Controllern, dem HMD oder von Händen in der realen Welt revidiert letzteren Vorteil aber meist wieder. Dafür kann man sich mit einem HMD relativ frei im Raum bewegen, auch wenn einige Modelle aus Gründen der Leistungsfähigkeit eine Verbindung zu einem Rechner benötigen und somit nur in der Nähe von diesem verwendet werden können. Dies ist ebenso bei der Verwendung von System zur Positionserfassung des HMDs oder der verwendeten Controller der Fall, da diese Systeme normalerweise statisch in der Welt platziert sind. Im Bereich der erweiterten Realität ist diese Bindung allerdings oft störend. Sie lebt schließlich von der realen Welt, in welcher sich der Benutzer bewegen können soll. Deshalb gibt es Bemühungen die Notwendigkeit eines stationären Computers zu beheben, ohne dabei die Performanz zu mindern. Ein Durchbruch hierbei ist die Hololens von Microsoft, welche ohne zusätzlichen Computer funktioniert. Sie hat dementsprechend aber weniger Leistung, weshalb ein kontinuierliches Erkennen der Handpositionen nur im Sichtfeld ermöglicht wird.

1.2 Benutzeroberflächen

Im Bereich der Mensch-Maschine-Interaktion, kurz HCI (Human-Computer-Interaction) wird im Groben zwischen drei Bereichen unterschieden: Mensch, Maschine und deren Schnittstelle.

Die Benutzeroberfläche (UI, engl. User Interface) stellt eben jenen Übergang zwischen Maschine und Mensch dar und ermöglicht die Interaktion beider Seiten miteinander [7]. Sie lässt sich weiter in physische Steuerelemente und graphische Bestandteile (das GUI, engl. Graphical User Interface) aufteilen. Zu dem ersten Bereich zählen alle Arten von Knöpfen, Hebeln, Reglern und Sensoren, welche Eingaben durch den Nutzer ermöglichen, ebenso wie visuelle, haptische oder akustische Ausgabegeräte, welche dem Nutzer eine Rückmeldung zu dessen Aktionen geben und den Status des ausgeführten Programms an den Benutzer weiterleiten.

Der graphische Anteil, der in dieser Arbeit betrachtet wird, stellt dem Nutzer über Texte, Bilder und Farben Informationen zur Verfügung und kann deren Manipulation ermöglichen. Damit soll Benutzern die Erfassung und Veränderung der virtuellen Umgebung erleichtert werden. Die graphische Benutzeroberfläche hat die früher übliche Kommandozeile abgelöst um die Darstellung von Informationen und Funktionen des Systems auf eine verständlichere und ansprechende Art zu ermöglichen. Sie liegt in Form einer Maske vor, welche auf einem Bildschirm angezeigt werden kann. Im Fall

einer Interaktion führt sie vordefinierte Befehle aus und wird dabei gegebenenfalls verändert. Dadurch ist es nicht mehr nötig die einzelnen Befehle und deren Auswirkungen zu kennen, was die Benutzung eines Programms besonders für Neulinge erleichtert. Die Inhalte der Oberfläche werden meist innerhalb von Menüs und Untermenüs geordnet, um sie übersichtlicher darzustellen.[7]

1.2.1 Element

Die graphische Benutzeroberfläche setzt sich aus einzelnen Elementen zusammen. Diese können beispielsweise Texte, Bilder oder Schaltflächen sein, welche normalerweise innerhalb einer Menüstruktur untereinander oder nebeneinander angeordnet sind. Sie enthalten Informationen und Funktionen, welche durch das Auswählen des jeweiligen Elements aufgerufen werden können.

1.2.2 Anker

Ein Anker bezeichnet eine Position, zu welcher die Elemente der Benutzeroberfläche relativ platziert werden. In Desktop-Programmen werden dafür die Bildschirmkanten beziehungsweise Ecken verwendet.

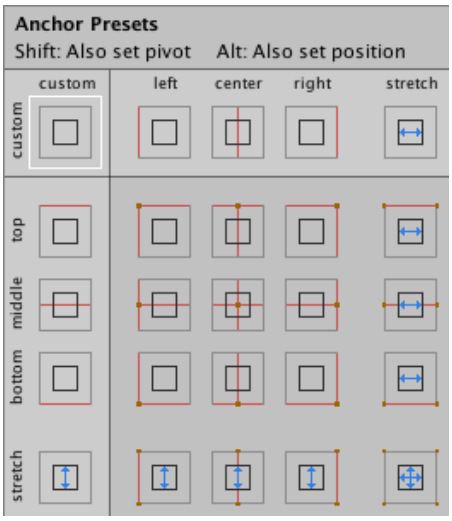


Abbildung 1.3: Anker Beispiel aus Unity[genommen von 8].

Am Beispiel aus der Spiel-Engine Unity (siehe Abbildung 1.3), sieht man eine Auswahl an möglichen Ankerpositionen innerhalb des Anwendungsfensters. Zudem sind verschiedene Kombinationen von Verbindungen zu zwei oder mehr Ankern angege-

ben, wodurch eine Vergrößerung oder Verkleinerung mit dem Anwendungsfenster durchgeführt werden kann.

1.3 Definition der Umgebungsabhängigkeit

Als umgebungsabhängig werden in dieser Arbeit Benutzeroberflächen bezeichnet, welche sich, im Gegensatz zum herkömmlichen zweidimensionalen Äquivalent, im dreidimensionalen Raum befinden und somit auch von Objekten aus der virtuellen oder realen Umgebung verdeckt werden können. Sie sind somit nicht wie eine Maske, die zwischen Kamera und der Umgebung angezeigt wird, sondern wie Bestandteile der, den Nutzer umgebenden, Welt. Zudem müssen sie sich nicht statisch zu, beziehungsweise abhängig von, der Kamera bewegen, sondern können auch an einen anderen Gegenstand aus der Umgebung gebunden werden, mit dem sie sich mitbewegen und gegebenenfalls mitrotieren. Diese Gegenstände werden in dieser Arbeit *Ankerobjekt* genannt. In 3D-Programmen (siehe Kapitel 2.3) wird solche UI häufig an Objekten positioniert und ist zu diesen statisch. In Spielen handelt sich dabei meist um eine diegetische Benutzeroberfläche und gilt als Teil der Umgebung [9]. An einem Beispiel aus *Metro: Last Light* (siehe Abbildung 1.4) sieht man, wie ein zweidimensionales Textobjekt an einem Blatt Papier in der dreidimensionalen Umgebung platziert wurde.

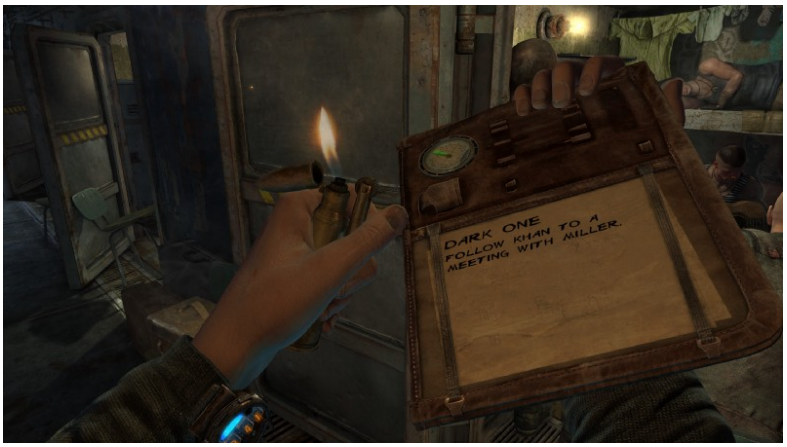


Abbildung 1.4: Diegetisches Element im Spiel *Metro: Last Light* [genommen von 9].

2 Verwandte Arbeiten

Vor dem Behandeln eines Problems ist es wichtig sich ebenso darüber zu informieren, was in diesem Bereich bereits in anderen Arbeiten behandelt wurde und welche Ergebnisse dabei erzielt wurden. Für diese Arbeit sind beispielsweise die Forschungen im Bereich der Platzierung von Benutzeroberflächen in 2D- und 3D-Anwendungen relevant, ebenso wie das Themenfeld der Priorisierung von anzuzeigenden Informationen. Während Forschungen im Bereich der Positionierung und des Designs von Nutzeroberflächen in Desktop-Anwendungen bereits auf jahrelange Erfahrung zurückblicken können, steht man dort bei HMD-Anwendungen noch relativ am Anfang. Auch bei dem Thema plattformübergreifender Programme in der erweiterten Realität gibt es schon einige Ansätze, wie ein Projekt von der Organisation Mozilla und eine Reihe von VR- und AR-Spielen zeigt [10][3]. Bei näherer Betrachtung dieser Programme fällt jedoch auf, dass hierbei nicht das ganze Potential zur Anzeige von Informationen ausgeschöpft wurde.

2.1 Priorisieren von Benutzeroberflächen

Das Umstrukturieren von Menüs ist nicht trivial, denn jede Veränderung in der Benutzeroberfläche kann das Nutzererlebnis drastisch verändern, falls der Nutzer dadurch irritiert wird. Dies ist sowohl in zweidimensionalen Programmen der Fall, wie auch in der virtuellen dreidimensionalen Welt. Mit diesem Nutzererlebnis setzt sich seit einigen Jahren der Bereich des UX Designs (User Experience Design) auseinander. Ein wichtiger Aspekt davon ist es, dem Nutzer sowohl nicht zu viel als auch nicht zu wenig Informationen auf einmal zu geben. Dies könnte sonst zu einer Überfordert führen, beziehungsweise durch mangelnde Daten die Nutzung erschweren. Es gibt für dieses Problem jedoch keine klare Lösung, da es sich dabei um ein subjektives Erlebnis handelt. Um aber das Risiko einer Verschlechterung des Nutzungserlebnisses gering zu halten, hilft es die vorhandenen Informationen zu priorisieren und abhängig davon zu entscheiden, wann und wie diese dem Nutzer angezeigt werden. Dazu verwendet man normalerweise Untermenüs, Tabs oder Ähnliches.[11]

Dies lässt bereits eine klare Priorisierung erkennen. Ausgehend von der Darstellung eines Elements der Benutzeroberfläche in einem Desktop-Programm, kann dessen Wichtigkeit in vier grobe Kategorien einteilt werden.

In die Kategorie mit der höchsten Priorität fallen jene Objekte, die immer angezeigt werden, egal wie das Programmfenster verformt wird. Notfalls wird dafür sogar die Skalierung eingeschränkt. Typische Beispiele für diese Kategorie sind die *Speichern*- und die *Programm Schließen*-Schaltflächen.

Etwas weniger wichtige Informationen werden zwar auch immer versucht anzuzeigen, aber falls der Platz dafür nicht ausreichen sollte, werden sie etwa in ein temporäres Untermenü verschoben oder auf andere Weise minimiert. Sobald wieder genug Platz vorhanden ist, tauchen sie erneut an der gewohnten Stelle auf. Dazu zählen unter anderem Hilfsfunktionen, wie das *Einfügen* oder *Ausschneiden* in Texteditoren.

Untermenüs beherbergen die nächst tiefere Kategorie. Deren Inhalt kann manuell angezeigt werden, aber ist sonst nie sichtbar. Es handelt sich meist um sekundäre Funktionen, die nur gelegentlich benötigt werden.

Die unwichtigsten Elemente befinden sich in Menüs innerhalb von Untermenüs und sind somit nur durch mehrere Interaktionen auffindbar. Im Normalfall sind dieses Einstellungen, welche nur selten verwendet werden.

Tabs erzeugen jedoch Ausnahmen bei dieser Einordnung. Sie stellen sozusagen verschiedene Kontexte dar, in welchen die Prioritäten anders verteilt sein können. So existieren möglicherweise in einem Kontext bestimmte Informationen nicht, aber in einem anderen werden sie durchgehend angezeigt. Aus diesem Grund wird zwischen kontextabhängigen und -unabhängigen Elementen unterschieden.

2.2 Positionierung in 2D-Anwendungen

Das verbreitetste Konzept bei der Erstellung von Nutzeroberflächen ist das WIMP-Konzept. Die Abkürzung steht für *Windows*, *Icons*, *Menus* und *Pointers*. Damit deutet sie bereits darauf hin, wie man sich die Umsetzung dieses Konzepts vorstellen kann. Programme werden bei dieser nämlich als Fenster dargestellt. Diese können weitere Fenster enthalten oder auch Menüs, in welchen Inhalte angeordnet sind. Um Funktionen möglichst kompakt anzeigen zu können, werden diese teilweise in Form von einem Bild oder Zeichen, einem sogenannten *Icon* dargestellt. Als Anker zum Positionieren des Fensterinhalts dienen jeweils die Eckpunkte des Fensters. Die einzelnen Elemente werden in den Fenstern also immer relativ zu mindestens einer Ecke des jeweiligen Fensters platziert. Dadurch kann der Inhalt eines Programmfensters bei dessen Verschiebung oder Skalierung mitbeweget werden. Damit ein Objekt sich zudem mit dem enthaltenden Bildausschnitt vergrößert, verkleinert oder darin zentriert bleibt, muss es hingegen an zwei oder mehr Ecken gebunden werden. Dies wird in der Abbildung 1.3 aus der Spiel-Engine Unity am rechten und unteren Rand dargestellt.

2.3 Positionierung in 3D-Anwendungen

In Programmen, die eine dreidimensionale virtuelle Welt anzeigen (im Folgenden *3D-Programm* genannt), wird eine spezielle Form der graphischen Benutzeroberfläche verwendet. Diese liegt nicht in einer zweidimensionalen Ebene vor der Kamera, sondern befindet sich im dreidimensionalen Raum. Dies ermöglicht auch nicht-planare Formen, was dieser Abwandlung den Namen *3D User Interface* gab.

Diese dreidimensionalen Elemente werden in 3D-Programmen für Desktop-Geräte gewöhnlich an festen Orten angebracht und bewegt sich gegebenenfalls an einem Objekt der Umgebung mit. Dies wird auch als diegetische Benutzeroberfläche bezeichnet, da sie tatsächlich Teil der virtuellen Welt ist.

In VR- und AR-Anwendungen steht teilweise zusätzlich die Option zur Verfügung, einen virtuellen Gegenstand oder eine Anzeige an den Handpositionen des Nutzers zu befestigen, oder ihn mit der Nutzeroberfläche auf andere Arten räumlich zu umgeben. Dies bietet dem Benutzer eine natürliche Art der Interaktion. Um etwas auszuwählen könnte er einfach auf das gewünschte Element zugehen und nach ihm die Hand ausstrecken. Denkbar ist eine solche Umsetzung unter anderem in Form eines Zylinders, wie in Abbildung 2.1 dargestellt.

In dieser Arbeit wurden als Ankerpunkte für solche Elemente die beiden Handpositionen des Benutzers und die Kopfposition verwendet, da diese Stellen im Fall der Verwendung von Erfassungssystemen durch Controller, das HMD oder die reale Hand gegeben sind.

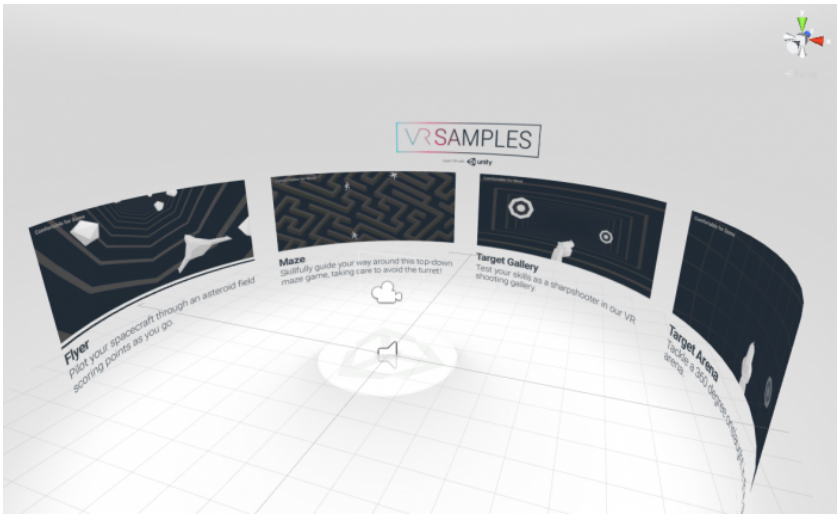


Abbildung 2.1: Example picture that was taken from an external source [genommen von 12].

3 Problemdiskussion

3.1 Problembeschreibung

Da es viele verschiedene Umsetzungen der virtuellen und erweiterten Realität gibt, steht man als Entwickler eines Mixed-Reality Programms über Kurz oder Lang vor einigen Hindernissen. Eines davon ist die Platzierung der Benutzeroberfläche, da hierbei im Gegensatz zu gewöhnlichen Desktop-Anwendungen keine Bildschirmränder oder vergleichbare Begrenzungen vorhanden sind. Zudem existiert keine konkrete Vorgabe wie die Benutzeroberfläche in der dreidimensionalen Welt platziert werden sollte. In bestehenden Programmen wird diese aber oft an Controllern platziert, wessen Position durch ein Ortungssystem erfasst wird. Eine Alternative bildet die Platzierung der Oberfläche rings um den Nutzer in Form eines Zylinders oder einer Kugel. Besondere Schwierigkeiten entstehen aber noch zusätzlich, wenn man das Programm auf verschiedenen Geräten nutzen möchte. So werden zum Beispiel bei vielen Varianten der erweiterten Realität die Handpositionen nie oder zumindest nicht kontinuierlich verfolgt und ermöglichen somit kein konstantes Anzeigen von virtuellen Elementen an diesen Stellen. Sie können daher schlecht als Ankerpositionen für die Benutzeroberfläche verwendet werden. Andere Geräte stellen durch die Positionserfassung der Controller und der VR- oder AR-Brille die Kopf- und Handpositionen zu jeder Zeit zur Verfügung. Das Wegfallen von Controllern durch niedrigen Akkustand oder andere Umstände erzeugt dabei allerdings ein ähnliches Problem wie bei den Varianten ohne solche Controller. Da die Positionierung der Benutzeroberfläche an den Handpositionen deren Elemente näher zum Nutzer bringt und eine intuitive Art zum Verstecken von Informationen bietet, sollten diese auch genutzt werden, solange diese Positionen vorhanden sind. Für den Fall ihrer Abwesenheit ist es dann allerdings nötig eine Rückfallmöglichkeit bereitzustellen, damit keine Elemente der Benutzeroberfläche verloren gehen und mit ihnen deren Funktion und Information. Diese Verluste könnten sonst eine produktive Nutzung erschweren oder sie sogar gänzlich verhindern.

3.2 Ansatz

Um die Verwendung von allen Elementen der Benutzeroberfläche auch beim Verlust einer oder mehrerer möglicher Ankerpositionen zu gewährleisten, wurden in dieser Arbeit mehrere Möglichkeiten erarbeitet, um eine Umverteilung der enthaltenen Informationen von einem nicht verwendeten Anker auf einen oder mehrere aktive Anker durchführen zu können. Dafür wird betrachtet wie wichtig die einzelnen Inhalte sind, welche aktiven Ankerpositionen zur Verfügung stehen und was für eine Rückfallart bei der Verschiebung verwendet werden soll. Letztere entscheidet darüber wie der Rückfall aufgefangen wird. Die Neuplatzierung der umverteilten Bausteine übernimmt jeweils der neue Anker.

3.2.1 Umsetzung der Prioritätsstufen

Um herauszufinden wie wichtig ein Element ungefähr ist, werden erst klare Klassen benötigt, welche unterschiedliche Prioritäten darstellen. Diese entscheiden darüber wie und wo eine Information dargestellt wird und ebenso wohin sie, im Falle des Verlusts des zugehörigen Ankers, verschoben wird. Als Vorbild für das in dieser Arbeit verwendete Modell dienen die in Kapitel 2.1 genannten Kategorien. Es gliedert sich in die vier Stufen *high*, *medium*, *low* und *none*. Diese werden abhängig von dem aktuellen Kontext und dem Typ des zugehörigen Ankerposition wie folgt umgesetzt:

An einer Handposition werden Informationen mit den Prioritäten *high* und *medium* gleich behandelt. Sie sind, im Gegensatz zu den Prioritäten *low* und *none*, nicht manuell ausschaltbar, aber können dennoch durch das Bewegen der Hand aus dem Sichtfeld entfernt werden. Beim Wegfall der Handposition werden alle Elemente der Stufen *high*, *medium* und *low* einem anderen Anker zugewiesen. Ein Inhalt aus der Kategorie *none* wird nur an andere Hand-Anker weitergeleitet, nicht jedoch an Kopf-Anker.

An einem Kopf-Anker findet kein Zusammenschluss von diesen vier Stufen statt. Dort wird ein Exemplar der Kategorie *high* immer im Sichtfeld des Nutzers angezeigt. *medium* und *low* werden wie bei einem Hand-Anker behandelt und ein Objekt mit Priorität *none* sollte es dort nicht geben.

3.2.2 Rückfallarten

Für die Umverteilung der Ankerinhalte wurden drei Methoden entworfen, welche diese Aufgabe auf verschiedene Art verwirklichen. Dabei wurde jeweils unterschiedlich viel Wert auf individueller Platzierung, Automatisierung und Nutzbarkeit gelegt.

Eine Variante, welche manuelles Platzieren erlaubt, ist die Verschiebung über eine Identifikationsnummer. Dabei werden, gegebenenfalls veränderte, Kopien des zu ver-

schiebenden Inhalts bereits vorher auf den Rückfall-Ankern platziert. Beim Wegfallen des Ankers wird dann lediglich auf diesen Ankern nach einem Element mit der gleichen Identifikationsnummer gesucht und dieses aktiviert. Der Nachteil dieser Option ist, dass für jedes Element manuell alle Ausweichpositionen definiert werden müssen. Dies bietet aber auch den Vorteil, dass eine Information nie an eine ungewollte Stelle rückt, sondern genau so angezeigt wird, wie es durch den Ersteller vorgegeben wurde. Es lässt also ein individuelles Abfangen zu.

Eine andere Option, welche ein automatisiertes geordnetes Platzieren umsetzt, benötigt eine Art Container für Inhalte. Ein Beispiel für einen solchen Container sind die Anker selbst, da sie ebenso Elemente enthalten. Allerdings ordnen diese von sich aus keine neuen Informationen auf bestimmte Weise neben den Bestehenden an, sondern übergeben diese Aufgabe an andere Container, welche in ihnen enthalten sind. Die Art und Weise wie ein Container neue Inhalte handhabt muss manuell festgelegt werden und wird dann zur Laufzeit automatisiert ausgeführt. Somit fällt, im Vergleich zu dem Rückfall mit ID, der Aufwand des manuellen Platzierens weg. Dafür muss hingegen eine Logik für den Container entworfen werden und die Umsetzung wird weniger individuell, wodurch es zu ungewollten Anordnungen von Elementen kommen kann.

Zuletzt gibt es noch eine Art der Verschiebung, bei der die betroffenen Elemente innerhalb ihres Behälters bleiben und dieser dann mitsamt seinem Inhalt als ein Objekt verschoben wird. Dabei muss der Zielanker auf eine, in Kapitel 3.2.3 beschriebene, Weise erweitert werden, da dem Anker die genaue Lage des Inhalts nicht bekannt ist. Diese Variante ist auf eine Nutzeroberfläche im Stil eines Gitters mit rechteckigen Elementen ausgelegt. Eine Verwendung mit anderen Stilen oder in Verbindung mit weiteren Rückfallarten kann Lücken zwischen den Elementen erzeugen. Sie erfordert dazu aber keine weitere Arbeit seitens des Erstellers.

Bei allen Varianten wird geprüft ob die Priorität des verschobenen Objekts höher oder gleich wie die Mindestpriorität des Zielankers ist, um jederzeit möglichst nur die benötigte Information anzuzeigen und unwichtige Inhalte gegebenenfalls zu verbergen. Falls die Priorität nicht zu dem Anker passen sollte, wird das Verfahren auf einem anderen Anker weitergeführt, solange noch ein weiterer zur Verfügung steht.

3.2.3 Erweiterung

Wie bereits in Kapitel 3.2.2 erwähnt, kann es vorkommen, dass ein Anker durch das Hinzufügen von Informationen gegebenenfalls erweitert werden muss, um die zusätzlichen Elemente platzieren zu können. Dazu wurden zwei Varianten erarbeitet. Welche davon verwendet werden soll, muss in jedem Anker manuell angegeben werden.

Die erste Art erweitert den Bereich, in dem der Inhalt platziert werden kann, in eine vorgegebene Richtung. Daraufhin werden darin neuer und alter Inhalt untereinander

beziehungsweise nebeneinander platziert. Somit wird ein Überschneiden von Elementen verhindert.

Die andere Variante erzeugt eine Schaltfläche, mit deren Hilfe zwischen der Anzeige von den alten und den neuen Informationen gewechselt werden kann. Somit wird durch neuen Inhalt nicht mehr Platz eingenommen. Allerdings kann man dadurch die verschiedenen Informationen nicht alle gleichzeitig sehen.

3.3 Umsetzung

Um diesen Ansatz umzusetzen wurden im Programm *Visual Studio 2015* mit der Programmiersprache C# die Klassen *UIAnchor*, *AnchoredUI*, *AnchorManager* und das Interface *UIContainer* erstellt. Sie stellen die Funktionen zur Manipulation der Benutzeroberfläche zur Verfügung. Deren Funktionsweise wird in den nachfolgenden Abschnitten beschrieben.

3.3.1 AnchoredUI

Die einzelnen Elemente der Benutzeroberfläche werden durch Instanzen dieser Klasse repräsentiert. Sie beinhaltet Informationen für deren Verschiebung auf andere Anker. Dazu zählen ihre Priorität und die Rückfall-Variante. Bei dem Verlust ihres Ankers wird in der *AnchoredUI*-Instanz eine Funktion aufgerufen, welche über die Art ihres Neuplatzierens bestimmt und dieses daraufhin startet.

3.3.2 UIContainer

Dieses Interface dient als Vorlage für Behälter von Instanzen der Klasse *AnchoredUI*. Diese Container enthalten eine Logik zum systematischen Platzieren von neuen Elementen innerhalb des bereits vorhandenen Inhalts und dienen dazu, eine automatische Verschiebung und geordnete Neuplatzierung von Ankerinhalten individueller gestalten zu können.

3.3.3 UIAnchor

Die Klasse *UIAnchor* stellt einen Anker für Bestandteile der Benutzeroberfläche dar. Sie beinhaltet unter anderem die Attribute *style* und *type*, welche angeben, inwiefern der Inhalt des Ankers manipuliert werden soll. Der Wert *type* steht dafür, ob der Anker am Kopf, der rechten oder der linken Hand angebracht ist. Die Variable *style* definiert, ob enthaltene Elemente in einer Ebene (Wert: Rectangle) oder auf einem Zylinder (Wert: Cylinder), mit dem angegebenen Radius *distance*, um den Anker herum

angeordnet werden sollen. Darüber wird dann gemeinsam mit der Priorität des Ankers die Positionierung der enthaltenen *AnchoredUI*-Instanzen durchgeführt. Als Vorbild für die Umsetzung des Stils *Cylinder* an der Kopfposition fungierte eine Abwandlung des Prinzips, das unter Anderem in Planetarien genutzt wird (siehe Abbildung 2.1).

Für einen Kopf-Anker mit dem Stil *Rectangle* und der Priorität *medium* wurde das Standardmenü der Hololens (siehe Abbildung 3.1) als Vorlage genommen. Das Menü wird in diesem Fall also nicht immer an der gleichen Stelle des Sichtfelds angezeigt, sondern bleibt so lange unbewegt, bis es aus dem Blickfeld verschwinden würde. Dann bewegt es sich wieder in die Sicht des Nutzers. Somit blockiert der Inhalt des Menüs nicht so sehr das Blickfeld, obwohl er durchgehend angezeigt wird.

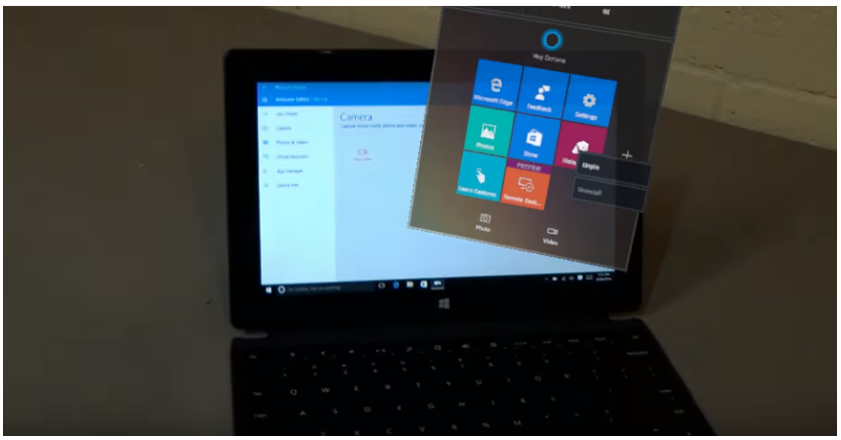


Abbildung 3.1: Menü der Hololens [genommen von 13].

Außerdem hat er noch vier Attribute, welche seine Bindung zum zugehörigen Ankerobjekt darstellen. Eins gibt die relative Position des Ankermittelpunkts zur Mitte des Ankerobjekts an. Ein weiteres steht für die relative Rotation zu diesem Objekt. Die anderen beiden Variablen legen fest, ob und wie sich der Anker mit dem Ankerobjekt rotiert. Zudem besitzt die Klasse eine optionale Referenz auf eine weitere Ankerinstanz, welche im Nachfolgenden *Kindanker* genannt wird. Diese sollte im Normalfall eine niedrigere Mindestpriorität besitzen und vom gleichen Typ sein, denn sie dient als weitere Möglichkeit für die Platzierung von Inhalten, falls diese in dem Anker mit höherer Mindestpriorität keinen Platz finden. Außerdem wird durch die Kindanker das Platzieren mehrerer Anker an einem einzelnen Objekt ermöglicht. Bei der Deaktivierung eines Ankers wird ebenso dessen Kindanker deaktiviert. Beim Nichtvorhandensein des Ankerobjekts, an der ein Anker platziert werden sollte, sucht dieser über die statische Klasse *AnchorManager* nach einem alternativen Ort, um seine Informationen weiterhin anzeigen zu können. Falls kein Anker dafür gefunden wurde, wird die Information

verworfen und eine Warnmeldung angezeigt. Ansonsten wird versucht diesem den Inhalt zu übergeben. Der Zielanker berechnet dann mittels dessen Priorität, ob er die Informationen bei sich platziert oder sie an einen Kindanker weiterleitet.

3.3.4 AnchorManager

Die statische Klasse *AnchorManager* wird beim Start der Anwendung über die Verfügbarkeit von Ankerobjekten informiert und übermittelt diese Informationen an das zugehörigen *UIAnchor*-Objekt. Sie startet damit auch das Wegfallen der Anker, falls deren Ankerobjekt nicht vorhanden sein sollte. Die Anker vermitteln die Informationen weiter an deren Kindanker und fragen im gegebenen Fall beim *AnchorManager* eine Ausweichoption ab. Bei der Auswahl einer Option werden Handanker in dieser Umsetzung immer bevorzugt.

3.4 Evaluierung

Um die beschriebene Umsetzung zu evaluieren wurde eine Nutzerstudie durchgeführt, an der dreizehn Freiwillige im Alter zwischen 16 und 26 Jahren teilnahmen. Der Fragebogen zu dieser Studie wurde mit dem Programm LimeSurvey erstellt und befindet sich im Anhang dieser Arbeit (siehe ...). Dieses Kapitel beschreibt deren Aufbau und Ablauf.

Referenz

3.4.1 Studienablauf

In der Studie mussten die Probanden in vier verschiedenen Szenarien jeweils eine Reihe von Aufgaben durchführen. Die Reihenfolge der vier Abschnitte wurde dabei jedes Mal variiert und war für jeden Teilnehmer unterschiedlich, um einen Einfluss der Bearbeitungsreihenfolge auf das Gesamtergebnis möglichst gering zu halten. Die Aufgabenstellung blieb in allen Szenarien die Gleiche.

Zu Anfang sollten die Tester ein paar einleitende Fragen beantworten, dann führten sie die Aufgaben ein erstes Mal in einem der Szenarien durch und beantworteten danach Fragen zu diesem Testfall und einzelnen Elementen darin. Dies beinhaltete zwei Fragen nach der gefühlten Schwierigkeit der Aufgabe innerhalb des Testfalls. Einmal sollten sie angeben ob sie gefühlt viel Zeit für die Aufgabe gebraucht haben, beim zweiten Mal mussten sie deren Schwierigkeit auf einer Skala von eins bis fünf bewerten, wobei eins für sehr leicht und fünf für sehr schwierig steht. Zusätzlich sollten sie angeben, ob sie die Positionierung der Benutzeroberfläche übersichtlich oder willkürlich fanden.

Danach folgten jeweils drei Fragen zu sieben Elementen der Benutzeroberfläche aus der Testszene. Bei diesen konnten sie jeweils aus einer gegebenen Liste auswählen, ob und warum sie nach dem Element suchen mussten, es anstrengen anzusehen fanden oder es störend fanden. Eine nicht gelistete Antwort konnte ebenfalls angegeben werden.

Die zu bewertenden Elemente wurden so ausgewählt, dass jeweils mindestens zwei Beispiele aus den Prioritätsklassen *high*, *medium*, *low* und zwei kontextabhängige Exemplare bewertet wurden. Die Auswahl dieser Elemente war ebenfalls in jedem Szenario gleich.

Dies wurde viermal wiederholt. Zum Abschluss sollten sie noch angeben, wie schwierig sie die Aufgabenstellung und die Steuerung unabhängig von den Szenarien fanden und als wie wichtig sie die ausgewählten Elemente auf einer Skala von eins bis fünf einordnen würden. Die genauen Definitionen der verschiedenen Stufen wird in Kapitel 4.1 beschrieben.

3.4.2 Szenario-Beschreibung

In jedem der Testfälle hatten die Tester die Aufgabe mit einem offen sichtbaren Gegenstand zu interagieren. Dazu musste der *Cursor* auf diesen ausgerichtet werden. Der *Cursor* ist in der Mitte des Sichtfeldes platziert und bewegt sich beim Drehen und Neigen des Kopfes mit. Man wählt also im Groben ein Objekt aus indem man darauf schaut. Nach dem Interagieren mit dem Gegenstand öffnet sich ein verborgenes Menü, dessen Inhalt zylinderförmig um die Position des Nutzers aufgereiht ist. Dieses heißt im Folgenden *Handelsfenster*. Es beinhaltet zwölf Elemente mit je einem farbigen Würfel und einer Preisangabe, welche alle ausgewählt werden können.

Als nächstes sollten die Tester herausfinden, wie viel Geld ihnen zur Verfügung steht und dann einen Würfel auswählen, dessen Preis die verfügbare Geldmenge nicht überschreitet. Diese Menge wird im Infofeld *Vermögen* angezeigt, welches nur bei geöffnetem Handelsfenster verfügbar ist. Daraufhin wird der ausgewählte Würfel gekauft und in das Feld *Ausgewähltes Objekt* verschoben, welches zu jeder Zeit angezeigt wird. Es informiert darüber, ob etwas gerade ausgewählt ist und um was es sich dabei handelt.

Die letzten drei Schritte beinhalten das Legen des gekauften Würfels in eines der *Inventar*-Felder, das erneute Auswählen des Würfels und das Ablegen in ein Feld der *Ausrüstung*. Sowohl das *Inventar* als auch die *Ausrüstung* können manuell ein- und ausgeblendet werden. Wie dies funktioniert und wie die beiden Elemente erkannt werden können wurde den Teilnehmern vor dem Ausführen der Aufgabe erklärt. Danach gilt die Aufgabe als abgeschlossen.

Die vier Szenarien unterscheiden sich nur in der Platzierung der genannten Elemente

der Nutzeroberfläche. Sie simulieren jeweils das Vorhanden- oder Nichtvorhandensein der beiden realen Handpositionen im virtuellen Raum.

Referenz auf Bilder im Anhang

In der Simulation von zwei Handpositionen werden die genannten UI-Elemente auf beide Hände verteilt. An der linken Seite befindet sich das *Inventar*, welches manuell verborgen und wieder sichtbar gemacht werden kann. Außerdem ist das *Vermögen* dort platziert, das nur bei geöffnetem Handelsfenster angezeigt wird. Auf der rechten Seite ist das *Ausgewählte Objekt* zu sehen und daneben die *Ausrüstung*, welche wie das *Inventar* aus- und eingeklappt werden kann.

Die beiden Fälle mit nur einer Hand wurden so gelöst, dass die Oberflächen der einen Hand auf die andere übertragen wurde. Dabei unterschieden sich die Fälle darin, dass jeweils andere Erweiterungsarten (siehe Kapitel 3.2.3) verwendet wurden. Auf der rechten Seite war dies die Umsetzung mit der Schaltfläche.

Bei der vierten Version mussten die Nutzer ohne Hände auskommen. *Inventar* und *Ausrüstung* waren dafür ähnlich zum *Handelsfenster* auf einer Zylinderoberfläche um den Nutzer herum angeordnet und konnten wie in allen anderen Fällen ein- und ausgeblendet werden. Das *Ausgewählte Objekt* und das *Vermögen* waren nach dem Vorbild des Standardmenüs der Hololens (siehe Abbildung 3.1) im Blickfeld des Testers befestigt.

4 Ergebnisse

Dieses Kapitel gibt die Ergebnisse der Studie wieder und vergleicht diese mit den ursprünglichen Annahmen. Dazu werden zuerst die angenommenen Prioritäten mit den Bewertung durch die Tester gegenübergestellt. Danach werden die Zusammenhänge zwischen den Umsetzungen der betrachteten Elemente und deren Nutzerbewertung betrachtet.

4.1 Prioritäten

Die nachfolgende Tabelle (siehe Tabelle 4.1) stellt die Nutzerbewertung der, in Kapitel 3.4.2 erwähnten, Elemente der Benutzeroberfläche dar. Die einzelnen Abstufungen werden darunter definiert.

Tabelle 4.1: Bewertung der Wichtigkeit von Elementen aus den Szenarien

Elementbezeichnung	unnötig (1)	unwichtig (2)	neutral (3)	wichtig (4)	sehr wichtig (5)
Inventar	1	0	9	2	1
Ausrüstung	0	0	5	4	4
Cursor	0	0	1	0	12
Ausgewähltes Objekt	0	1	2	6	4
Switch	1	1	7	3	1
Handelsobjekt	0	7	2	3	1
Schließen	0	5	2	3	3
Vermögen	0	3	5	3	2

Ein Objekt ist *sehr wichtig*, wenn es zu jedem Zeitpunkt angezeigt wird und sich ebenso immer im Sichtfeld des Nutzers befindet. Es kann sich somit weder verbergen lassen, noch sich aus dem Sichtfeld bewegen. Diese Stufe ist äquivalent mit der Priorität *high*, welche in Kapitel erläutert wird. Kontextabhängige Varianten davon bilden eine Ausnahme, da sie nur in bestimmten Situationen als *sehr wichtig* gelten und deshalb nur dort auf die beschriebene Weise angezeigt werden. In anderen Situationen sind diese Informationen versteckt.

Referenz

Als *wichtig* gilt ein Element allerdings bereits, wenn es lediglich zu jeder Zeit angezeigt wird, aber nicht immer sichtbar ist. Es kann sich also auch aus dem Sichtfeld hinaus bewegen, aber lässt sich auf keine Weise ausschalten. Dies entspricht der Priorität *medium* Wie bei der Stufe *sehr wichtig* bilden auch hierbei kontextabhängige Objekte die selbe Ausnahme.

Neutral sind Informationen, die der Benutzer jederzeit manuell ein- und ausblenden kann. Dazu zählen alle kontextunabhängigen Inhalte mit Priorität *low*. Die Objekte hingegen, welche vom aktuellen Kontext abhängen, werden als eigene Stufe gehandhabt. Sie werden als *unwichtig* bezeichnet.

Die letzte Stufe ist äquivalent zu der Priorität *none*. Bei diesen, als *unnötig* erachteten, Elementen macht es keinen Unterschied für die Bewältigung der Aufgabe, ob diese Elemente vorhanden sind oder nicht.

Im Vergleich mit der angenommenen Priorität der UI-Elemente fallen ein paar Abweichungen in der Nutzerbewertung auf. Die *Ausrüstung* wird zum Beispiel von etwa zwei Drittel der Befragten als *wichtig* oder sogar *sehr wichtig* angesehen und nicht als *neutral*. Auch das *Vermögen* wird von mehr als drei Viertel der Tester wichtiger eingestuft als angenommen (*unwichtig*). Beim *Handelsfenster* und dessen Elementen gehen hingegen die Meinungen stark auseinander. Die durchschnittliche Bewertung der anderen Objekte stimmen mit der Annahme überein.

4.2 Bewertung der Szenarien

Die Aufgabenstellung wurde insgesamt von allen Nutzern als einfach empfunden. Zwei Drittel bewerteten sie mit Schwierigkeit eins von fünf und das andere Drittel mit zwei von fünf. Innerhalb der Szenarien wurde sie meist als schwieriger empfunden. Dabei gab es häufig kleine und vereinzelte große Schwankungen zwischen den Bewertungen innerhalb der verschiedenen Fälle. Eine klare Verbindung zwischen Fall und diesen Schwankungen lässt sich dabei allerdings nicht erkennen.

Mit dem *Curor* waren alle Tester zufrieden, lediglich die Geschwindigkeit, mit der ein Objekt ausgewählt wurde, war ihnen vereinzelt zu schnell. Dies war auf die Dauer einer Sekunde festgelegt. Ebenso positiv wurde das *Handelsfenster* bewertet. Einzelne Tester merkten aber an, dass es auf Dauer stören könnte, da sich dessen Elemente auf der Augenhöhe befinden und somit der Blick gesenkt oder gehoben werden muss, um sich umzusehen ohne etwas auszuwählen. Dies liegt an der gewählten Steuerungsform, die insgesamt zwar als eher verständlich aber nicht als intuitiv empfunden wurde.

Deutlich negativ wurde die Umsetzung der Schaltfläche im Szenario mit nur der rechten Hand bewertet. Aufgrund dieser haben drei Viertel der Nutzer nicht das *Inventar* gefunden. Die Gründe dafür waren bei einem Drittel, dass die Schaltfläche

nicht bemerkt wurde. Die anderen fanden dessen Funktion und Benennung (Switch) nicht intuitiv und somit verwirrend.

Die Bewertungen im Szenario ohne Hände fiel, wie erwartet, an einigen Stellen schlechter aus, als in den anderen Fällen. Dies hing einerseits damit zusammen, dass etwa die Hälfte der Tester die Umsetzung von *Vermögen* und *Ausgewähltes Objekt* als störend angaben. Sie waren ihnen zu nah und verdeckten einen zu großen Teil des Sichtbereiches. Andererseits bemängelte ebenfalls ungefähr die Hälfte, dass die einzelnen UI-Elemente sich die meiste Zeit lang gegenseitig überlappten. Dadurch würden sie gestaffelt wirken und die Nutzer bei der Ausführung der Aufgabe behindern.

Bei der Verwendung von nur der linken Hand fanden zwei Drittel der Teilnehmer die Menge an Elementen zu umfangreich. Durch die Platzierung von *Ausrüstung*, *Vermögen* und *Ausgewähltes Objekt* in der Verlängerung des Inventars wurde es ihnen zu sperrig und die *Ausrüstung* war ihnen zu weit entfernt und damit zu klein.

5 Fazit

Im Nachfolgenden werden die wichtigsten Ergebnisse dieser Arbeit kurz zusammengefasst und es wird ein Ausblick darüber gegeben, was man in Zukunft potentiell an dem Konzept verbessern könnte, beziehungsweise inwiefern man es erweitern kann. Dies beinhaltet ebenso die Grenzen und Schwächen dieser Arbeit.

5.1 Zusammenfassung

- Relativ wenige Tester - Steuerung war für einige Nutzer nicht intuitiv aber mit Übung ging es Durchschnittlich Bewertung knapp über 3 - Umsetzung fanden die Tester im Groben gut?

- mitbewegende DauerUI wurde negativ bewertet, zu groß, zu nah - Schwierigkeiten mit verborgenen Elementen

5.2 Ausblick

In der Zukunft wäre eine umfangreichere Studie nützlich, um die Wirkung einzelner Elemente auf die Nutzer besser zu erfassen und ein genaueres Abbild der durchschnittlichen Nutzermeinung zu erhalten. Dazu ist es nötig weitere Altersgruppen miteinzubeziehen, wie zum Beispiel mehr Tester, welche zwischen 13 und 18 oder 25 und 40 Jahren alt sind. Die bisherige Studie umfasst nämlich lediglich den Altersbereich 16 bis 26. Außerdem muss die Anzahl der Teilnehmer erhöht werden.

Des weiteren kann die Umsetzung einiger Elemente des, über diese Arbeit erstellten, Programms verbessert werden. So wäre es zum Beispiel sinnvoll, die einzelnen zweidimensionalen Objekte eines Zylinderankers so zu verformen, dass sie tatsächlich auf der Oberfläche des Zylinders angezeigt werden und somit selbst unterschiedlich große Elemente nahtlos nebeneinander platziert werden können. Bisher sind sie noch planar und nur ihr Mittelpunkt liegt auf dem Zylinder, was besonders bei großen Elementen dazu führt, dass deren seitliche Kanten deutlich weiter vom Zylindermittelpunkt und somit vom Standpunkt des Nutzers entfernt sind. Denkbar wäre es außerdem das Prinzip des Zylinders ebenso auf eine Kugel umzuformen

Graphisch darstellen

Zudem fehlen bisher noch alternative Implementierungen von Containern neben den Ankern, welche eine sinnvolle und einfache Umlagerung von Bausteinen zwischen zwei verschiedenen Ankern ermöglichen. Möglich wären auch Container für andere Container.

Ein weiteres wichtiges Thema, welches in dieser Arbeit nicht behandelt wurde, aber in diesem Bereich in Zukunft eine große Rolle spielen wird, ist die Begrenzung des Informationsumfangs an einem einzelnen Anker. Dies ist nämlich ein wichtiger Punkt im Hinblick auf das Nutzererlebnis. Zu viel Information kann den Nutzer überfordern beziehungsweise stören, wie man bei der Auswertung der Umsetzung mit nur der linken Hand in Kapitel 4.2 sieht.

Zur Erleichterung der Interaktion wäre die Entwicklung verschiedener Interaktionsformen für die einzelnen Szenarien praktisch. Die in dieser Arbeit verwendete Form war nämlich nicht sonderlich intuitiv und nutzt das Potential der Controller nicht.

Die Probleme des umschaltbaren Menüs könnte man durch eine klare Beschriftung eventuell verbessern. Das Anbringen eines kleinen Hinweises, welcher am Rand des Sichtfelds auftaucht, könnte das Finden verborgener Elemente ebenfalls weiter vereinfachen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] T. Grohganz. *IDC: Verkaufszahlen von AR- und VR-Brillen um 30 % gesunken, großer Verkaufsboom steht bevor*. Zuletzt eingesehen am 10.03.2019. Juni 2018. URL: https://www.vrnerds.de/idc-verkaufszahlen-von-ar-und-vr-brillen-um-30-gesunken-grosser-verkaufsboom-steht-bevor/.

[2] M. Bastian. *Bericht: Verkaufszahlen von VR- und AR-Brillen fallen um 30 Prozent*. Zuletzt eingesehen am 7.03.2019. Juni 2018. URL: https://mixed.de/bericht-verkaufszahlen-von-vr-und-ar-brillen-fallen-um-30-prozent/.

[3] M. Bauer. *Eve Valkyrie: Crossplay für Rift, Vive und PS VR bestätigt*. Zuletzt eingesehen am 10.03.2019. Apr. 2016. URL: https://www.computerbild.de/artikel/cbs-News-PC-Eve-Valkyrie-Crossplay-Rift-Vive-PlayStation-VR-15463857.html.

[4] T. Tall. *Augmented Reality vs. Virtual Reality vs. Mixed Reality – An Introductory Guide*. Eingesehen am 10.03.2019. 2018. URL: https://www.toptal.com/designers/ui/augmented-reality-vs-virtual-reality-vs-mixed-reality.

[5] R. T. Azuma. „A Survey of Augmented Reality“. In: *Presence: Teleoperators and Virtual Environments* 6.4 (Aug. 1997), S. 355–385. ISSN: 1054-7460.

[6] P. Milgram, H. Takemura, A. Utsumi und F. Kishino. „Augmented Reality: A class of displays on the reality-virtuality continuum“. In: *Telemanipulator and telepresence technologies* 2351 (1995), S. 282–292.

[7] J. Reimer. *A History of the GUI*. Zuletzt eingesehen am 11.03.2019. Mai 2005. URL: https://arstechnica.com/features/2005/05/gui/.

[8] *Unity User Manual (2018.3)*. Eingesehen am 10.03.2019. 2018. URL: https://docs.unity3d.com/Manual/UIBasicLayout.html.

[9] A. Stonehouse. *Considering the narrative in user interface design for video games*. Zuletzt eingesehen am 11.03.2019. Dez. 2016. URL: https://medium.com/@thewanderlust/considering-the-narrative-in-user-interface-design-for-video-games-c45953c22760.

[10] J. Siegle. *Virtual Reality plattformübergreifend: Firefox bringt VR-Browser*. Zuletzt eingesehen am 10.03.2019. Jan. 2019. URL: http://www.techfieber.de/2019/01/25/virtual-reality-plattformuebergreifend-firefox-bringt-vr-browser/.

[11] J. Natoli. *The Power of Progressive Disclosure*. Zuletzt eingesehen am 12.03.2019. Feb. 2017. URL: https://www.givegoodux.com/the-power-of-progressive-disclosure/.

[12] *User Interfaces for VR*. Eingesehen am 9.03.2019. 2016. URL: https://unity3d.com/de/learn/tutorials/topics/virtual-reality/user-interfaces-vr.

[13] C. Pietschmann. *HoloLens Start Menu UI*. Eingesehen am 10.03.2019. Apr. 2016. URL: https://www.youtube.com/watch?v=ES_ntyV7ITI.