

Bachelorarbeit

Interaktive Visualisierung von Requirements mit Augmented-Reality: Eine Analyse der Usability und Effektivität

im Studiengang Softwaretechnik und Medieninformatik (SWB)
der Fakultät Informationstechnik
Sommersemester 2024

Kyle Mezger
Matrikelnummer: 765838

Zeitraum: 01.03.2024 bis 31.08.2024

Erstprüfer: Prof. Dr. -Ing. Andreas Rößler

Zweitprüfer: Prof. Dr. rer. nat. Dieter Morgenroth

Firma: IT-Designers GmbH

Betreuer: Stefan Kaufmann

Eidesstattliche Erklärung

Hiermit versichere ich, Kyle Mezger, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Weiterhin erkläre ich, dass ich für die Umformulierung einzelner Textpassagen und die Korrektur von Rechtschreib- und Grammatikfehlern die Hilfe von digitalen Tools, speziell ChatGPT und DeepL Write, in Anspruch genommen habe. Diese Unterstützung betraf ausschließlich die Formulierungshilfe und die Korrektur von Grammatik und Rechtschreibung, ohne dass die inhaltliche Eigenständigkeit und Urheberschaft meiner Arbeit dadurch berührt wurden.

Esslingen, den 8. Juli 2024

Unterschrift

Inhaltsverzeichnis

1 Kurzfassung	5
2 Einleitung	6
2.1 Motivation	6
2.2 Zielsetzung	7
3 Grundlagen	9
3.1 Requirements-Engineering	9
3.2 Virtual-Reality	12
3.3 Augmented-Reality	12
3.3.1 Head-Mounted Displays	14
3.3.2 Handheld-Devices	17
3.3.3 Spatial-Displays	19
3.4 Gamification von UX	21
4 Technologien	22
4.1 Meta Quest 3	22
4.2 WebGL	24
4.3 WebXR	24
5 Konzept	27
5.1 Explodierende Bauteile	27
5.2 Wolken von Requirements	30
6 Implementierung	32
6.1 Entwicklungsumgebung für WebXR und Meta Quest 3	32
6.2 Implementierung der Interaktionskonzepte	35
6.2.1 Explodierende Bauteile	36
6.2.2 Wolken von Requirements	40
6.2.3 2D-Mockup für Wolken aus Requirements	44
7 Zusammenfassung	47
7.1 Ergebnisse	47
7.2 Fazit	49
7.3 Ausblick	50

Abbildungsverzeichnis

1	Realitäts-Virtualitäts-Kontinuum nach Milgram	13
2	Meta Quest 3 VR- und AR-Headset	14
3	Screenshots aus dem AR-Spiel Pokémon Go	18
4	Innenraum des Mercedes-Benz Fahrsimulators	20
5	Frontal ausgerichtete Kameras und Tiefenprojektor der Meta Quest 3	23
6	Die beiden nach unten gerichteten Kameras der Meta Quest 3	23
7	Application Flow von WebXR-Anwendungen	25
8	Szenen-Explorer und -Inspektor von Babylon.js	26
9	Beispiel einer Explosionsanimation anhand eines Rubiks Cube	27
10	Ablaufdiagramm des Interaktionskonzepts der Explodierenden Bauteile	28
11	Beispiel einer Punktewolke	30
12	Screenshots der Immersive Web Emulator Erweiterung in Chrome	33
13	Ablaufdiagramm des normalen Entwicklungsablaufs	34
14	Schema der Bedienelemente des Quest Controllers mit den im Rahmen dieser Arbeit genutzten Namen	35
15	Screenshots des explodierenden Tetris-Blocks mit Requirements in AR	37
16	Screenshot des in der Anwendung genutzten Porsche Modells	38
17	Bildsequenz der Animation des Porsche Modells	39
18	Funktionsweise des Punktewolken Algorithmus	40
19	Wolke aus 5 Requirementpanels	41
20	Funktionsweise des angepassten Punktewolken Algorithmus für flachere Wolken	41
21	Wolke aus 9 Requirements mit undurchsichtigen Panels und größerem Abstand	42
22	Extremversion eines abgeflachten Algorithmus zur Erstellung einer Punktewolke	43
23	Mockup zur Übersicht über eine große Anzahl an Requirements	44
24	Mockup der hereingezoomten Ansicht	46
25	Bildsequenz aus dem Prototyp der Explodierenden Bauteile in AR	48
26	Ablaufdiagramm der nächsten Schritte zur weiteren Untersuchung der beiden Interaktionskonzepte	51

1 Kurzfassung

Die vorliegende Bachelorarbeit befasst sich mit der Visualisierung von Requirements mithilfe von Augmented-Reality.

Unter Requirements werden Anforderungen verstanden, die vor der Entwicklung eines Systems an dieses gestellt werden. Sie stellen einen essenziellen Bestandteil der Entwicklung von Produkten und Systemen dar, da sie die Basis für die Entwicklung bilden. Die Visualisierung von Requirements zielt darauf ab, im Verlauf des Requirements-Engineering den Überblick zu bewahren und die Requirements dadurch besser zu verstehen. Dazu werden zwei verschiedene Interaktionskonzepte ausgearbeitet und jeweils in einem WebXR-Prototyp mit Babylon.js umgesetzt.

Das erste Konzept umfasst die Visualisierung von Requirements in Kombination mit einem 3D-Modell des zu entwickelnden Produkts. In diesem Zusammenhang wird das 3D-Modell, für welches in dieser Arbeit beispielhaft ein Fahrzeug verwendet wird, in einer Augmented-Reality-Umgebung dargestellt. Das 3D-Modell kann dann in einer Animation in seine Einzelteile zerlegt werden, um die Requirements als Textpanels, welche an den einzelnen Bauteilen angehängt sind, zu visualisieren.

Der Prototyp dieses Konzepts demonstriert den Mehrwert des Interaktionskonzepts durch AR, welcher sich aus der direkten räumlichen Verknüpfung von Requirements und Bauteilen ergibt. Des Weiteren können in dem Interaktionskonzept theoretisch noch zahlreiche Interaktionsmöglichkeiten des Nutzers mit den Requirements und dem 3D-Modell implementiert werden. Allerdings ist die Implementierung des Konzepts in Anwendungen aufwendig, da für manche Schritte, wie bspw. die Erstellung der Animation, viel Handarbeit erforderlich ist. Ohne weitere Automatisierung der Implementierung ist daher die wirtschaftliche Effektivität des Konzepts fraglich.

Das zweite Interaktionskonzept basiert auf einer Visualisierung von Requirements in Wolken aus Textpanels, die in der Augmented-Reality-Umgebung schweben. Jedes Panel beinhaltet dabei ein Requirement. Die Wolken fungieren als Cluster von Requirements, die thematisch zusammengehören. Durch Anklicken eines Panels sollen die zugehörigen Requirements angezeigt werden.

Dieses Konzept führt im Prototyp schon bei kleinen Wolken zu Überlappungen der Requirementpanels. Daher ist die Weiterentwicklung des Konzepts in 3D, bzw. AR, nicht sinnvoll. Allerdings wäre es denkbar, das Interaktionskonzept in 2D als zusätzliche Ansicht in klassischen Requirement-Engineering-Tools wie bspw. Jira zu integrieren. Aus diesem Grund wurde die Implementierung des Konzepts als zweidimensionale Anwendung im Rahmen dieser Bachelorarbeit als Mockup ausgearbeitet und untersucht.

2 Einleitung

In diesem Kapitel wird eine thematische Einführung in die Motivation und die Ziele der vorliegenden Arbeit gegeben. Im Rahmen der Arbeit wird der Begriff „Requirements-Engineering“ anstatt des deutschen Begriffs „Anforderungsanalyse“ und „Requirements“ anstatt „Anforderungen“ verwendet.

2.1 Motivation

Das Requirements-Engineering stellt einen wesentlichen Bestandteil der Entwicklung von Systemen dar und dient der Erfassung sowie Spezifikation der Requirements an ein System. Requirements, die gut formuliert sind, definieren die gewünschten Funktionalitäten eines Systems und beschreiben dessen Eigenschaften. Eine unzureichende Spezifikation der Requirements eines Systems birgt das Risiko von Missverständnissen und Fehlinterpretationen in der Entwicklung, die bis in das Endprodukt hineinreichen können. Da diese Fehler meist erst spät im Entwicklungsprozess identifiziert werden, gehen mit ihnen in der Regel hohe Kosten einher. Daher wird in der Industrie viel Wert auf ein genaues und strukturiertes Requirements-Engineering gelegt. Ein höherer Aufwand im Requirements-Engineering kann dazu beitragen, Fehler frühzeitig zu erkennen und zu vermeiden, wodurch letztlich Kosten, die durch Fehler und Missverständnisse entstehen, reduziert werden können.

Daher ist es von essenzieller Bedeutung, dass die Requirements klar und verständlich formuliert sind, um potenzielle Missverständnisse und Fehlinterpretationen zu vermeiden. Einen Ansatz zur Verbesserung der Qualität der Requirements stellt reQlab dar, ein Projekt der IT-Designers GmbH in dessen Rahmen diese Bachelorarbeit entstanden ist. Das Tool unterstützt Nutzer bei der korrekten Formulierung und Strukturierung von Requirements. Dazu untersucht reQlab die einzelnen Requirements auf ihre Qualität und gibt eine Bewertung ab, wie gut sie formuliert sind. Auf diese Weise kann das Tool die Qualität der einzelnen Requirements signifikant verbessern und somit auch die Qualität des gesamten Systems steigern.

Der von reQlab verfolgte Ansatz zielt jedoch lediglich auf die Optimierung eines Teils des Requirements-Engineerings ab. Ein weiterer wesentlicher Aspekt ist die kontinuierliche Kommunikation der Requirements zwischen Entwicklern und Auftraggebern. Da trotz eines umfangreichen Aufwands im Requirements-Engineering eine vollständige Vermeidung von Missverständnissen und Fehlinterpretationen nicht möglich ist, ist es unerlässlich, die Requirements einer regelmäßigen Überprüfung und Diskussion zu unterziehen. Es ist von

essenzieller Bedeutung, dass alle Beteiligten über alle Requirements, die sie betreffen, informiert sind, diese verstehen und nachvollziehen können. Allerdings erweist es sich in der Praxis häufig als schwierig – insbesondere für Auftraggeber, die nicht täglich in den Entwicklungsprozess involviert sind – einen umfassenden Überblick über alle relevanten Requirements zu gewinnen.

Gleichzeitig eröffnet der Fortschritt der Technik eine Vielzahl neuer Möglichkeiten, um Daten zu visualisieren und zu präsentieren. Die Verfügbarkeit neuer Endgeräte wie der Meta Quest 3 oder der Microsoft HoloLens führt zu einer zunehmenden Zugänglichkeit von Augmented-Reality-Technologien, wodurch sich potenziell neue Möglichkeiten zur Visualisierung und Präsentation von Requirements eröffnen.

In der vorliegenden Bachelorarbeit soll daher untersucht werden, ob sich Requirements in einer AR-Umgebung darstellen lassen und ob dadurch ein Mehrwert gegenüber herkömmlichen Darstellungsmethoden entsteht.

2.2 Zielsetzung

Im Rahmen der Bachelorarbeit erfolgt eine Untersuchung diverser Interaktionskonzepte für die Anzeige von Requirements in einer AR-Umgebung. Ziel ist es, die Vor- und Nachteile der einzelnen Ansätze sowie ihre Eignung für den Einsatz in einem realen Projekt zu untersuchen.

Im Rahmen der Untersuchung sollen möglichst mehrere Konzepte entwickelt und prototypisch umgesetzt werden, um diese anschließend einer Evaluierung und einem Vergleich zu unterziehen. Bei der Entwicklung der Konzepte soll darauf geachtet werden, dass möglichst unterschiedliche Ansätze hinsichtlich Darstellung und Interaktion verfolgt werden, um auf diese Weise die jeweiligen Vor- und Nachteile der verschiedenen Konzepte untersuchen zu können.

Die zu entwickelnden Prototypen sollen in der Lage sein, die Interaktionskonzepte anhand von Beispielen zu veranschaulichen. Es ist nicht erforderlich, sämtliche Features der ausgearbeiteten Interaktionskonzepte umzusetzen. Vielmehr sollen die Prototypen die Interaktionsmöglichkeiten und den Mehrwert der Darstellung in AR gegenüber herkömmlichen Methoden veranschaulichen.

Daher soll jeder Prototyp mindestens zwei verschiedene Interaktionsmöglichkeiten mit den Requirements bieten, um einerseits einen hohen Grad an Interaktivität zu gewährleisten und andererseits den Aufwand der Implementierung der Prototypen angemessen zu halten. Im Rahmen der Entwicklung ist zu untersuchen, inwiefern die Konzepte technisch umsetzbar sind, um eine Einschätzung des Aufwands einer echten Implementierung zu erlangen. Ein wesentlicher Aspekt bei der Umsetzbarkeit ist die Frage, ob eine automatisierte Umsetzung möglich ist, um eine realistische Integration in einer realen Anwendung

zu gewährleisten. Konzepte, die einen hohen manuellen Aufwand bei der Implementierung erfordern, müssen einen entsprechend höheren Mehrwert bieten, um die tatsächliche Umsetzung zu rechtfertigen.

Bei der Evaluation der Prototypen sollen insbesondere die folgenden Kriterien berücksichtigt werden:

- Usability: Wie einfach und intuitiv ist die Bedienung der Prototypen?
 - Mehrwert: Bieten die Prototypen einen Mehrwert der Usability?
 - AR vs 2D: Ist die Darstellung in AR sinnvoll und bietet sie einen Mehrwert gegenüber herkömmlicher zweidimensionaler Darstellungsmethoden? Wären die Interaktionskonzepte auch ohne AR sinnvoll?
- Umsetzbarkeit: Wie aufwändig ist die Implementierung der Prototypen und wie gut lassen sie sich in bestehende Systeme integrieren?
 - Automatisierbarkeit: Wie einfach könnte die Umsetzung der Prototypen automatisiert werden?
 - Aufwand: Wie hoch wäre der Aufwand für die Realisierung der Interaktionskonzepte in einer realen Anwendung?

Die beiden Kriterien und ihre Unterkriterien dienen der Identifizierung und Bewertung der Vor- und Nachteile der verschiedenen Konzepte. Auf Basis der Evaluationsergebnisse erfolgt eine Einschätzung hinsichtlich der Eignung der Konzepte für die Implementierung in einer realen Anwendung für die Verwaltung echter Projekte. Der resultierende Mehrwert der Darstellung gegenüber den herkömmlichen Methoden ist gegen den Aufwand der Implementierung aufzuwiegen.

3 Grundlagen

Im folgenden Kapitel sollen konzeptionelle Grundlagen erläutert werden, welche für das Verständnis der Bachelorarbeit notwendig sind. Im Rahmen dessen erfolgt eine vertiefende Auseinandersetzung mit den Themengebieten Requirements-Engineering und Augmented-Reality.

3.1 Requirements-Engineering

Die Bachelorarbeit hat zum Ziel, sogenannte Requirements, also Anforderungen an ein System, zu visualisieren. Für das Verständnis der Arbeit ist es daher von entscheidender Bedeutung, sich mit den Grundlagen des Requirements-Engineering vertraut zu machen.

System

Der Begriff „System“ wird gemäß der Definition des International Requirements Engineering Board (IREB) als „eine kohärente, abgrenzbare Menge von Elementen, die durch koordiniertes Handeln einen bestimmten Zweck erfüllen“ definiert [1]. Der Begriff „System“ ist also ein Überbegriff für Produkte, Services, Geräte, Prozeduren und Werkzeuge und kann sowohl physisch als auch virtuell sein. Ein System kann folglich beispielsweise ein Software-Produkt, ein Fahrzeug oder sogar eine Dienstleistung sein. Daher wird auch in dieser Bachelorarbeit das Wort „System“ als Überbegriff für alle Arten von Systemen genutzt.

Requirements

Grundlegend sind Requirements Anforderungen, die an ein System gestellt werden. Das IREB definiert sie in ihrem Glossar mit drei Eigenschaften [1, Def. Requirement]:

- Ein Bedürfnis eines Interesseneigners (Stakeholder)
- Eine Eigenschaft oder Fähigkeit, die ein System haben soll
- Eine dokumentierte Repräsentation eines Bedürfnisses, einer Fähigkeit oder einer Eigenschaft

Die Aufgabe der Requirements besteht demnach in der Repräsentation und Dokumentation der Bedürfnisse der Stakeholder in Bezug auf das System

Die Gestaltung von Requirements kann je nach System und sogar je nach Requirement unterschiedlich sein. In ihrem Buch „Requirements-Engineering und -Management“ präsentiert Chris Rupp eine Reihe von Beispielen für unterschiedliche Formen von Requirements [2, S. 19]:

- User-Stories
- Use-Cases
- Stories
- Formalisierte natürlichsprachliche Requirements
- Requirements in Form von Diagrammen (semiformales Modell)

Natürlichsprachliche Requirements können mit geringem Aufwand selbst formuliert werden. Allerdings birgt das auch das Risiko von Missverständnissen und Unklarheiten.

In der Motivation in Kapitel 2.1 wurde das Tool reQlab der IT-Designers GmbH angesprochen, in dessen Rahmen diese Bachelorarbeit entstanden ist. Das Ziel von reQlab besteht in der Erkennung von Missverständnissen und Unklarheiten in natürlichsprachlichen Requirements, um somit die Qualität der Requirements zu verbessern. Daher werden im Umfang dieser Bachelorarbeit nur natürlichsprachliche Requirements genutzt.

Des Weiteren werden Requirements in funktionale und nicht-funktionale Requirements unterteilt. Funktionale Requirements beschreiben „die Funktionen, die das System leisten soll, die Informationen die es verarbeiten soll; das gewünschte Verhalten, welches das System an den Tag legen soll“ [3, S. 12].

Der Begriff „nicht-funktionale Requirements“ ist dabei jedoch etwas irreführend, da auch nicht-funktionale Requirements gewissermaßen Funktionen des Systems beschreiben. Peter Hruschka beschreibt in seinem Buch „Funktionale Requirements“ mit der Frage: „Was soll das System/Produkt tun?“ Nicht-funktionale Requirements hingegen unterteilt er in zwei Kategorien [3, S. 13]:

- Qualitätsanforderungen: „Wie gut? Wie schnell? Wie zuverlässig? ...“
- Randbedingungen: „Ressourcen, Wiederverwendung, Zukauf, geforderte Technologie ...“

Diese Unterteilung ist hilfreich zur Strukturierung der Requirements und könnte im User-Interface der Visualisierung genutzt werden, um die Requirements zu kategorisieren.

Stakeholder

Der Begriff Stakeholder umfasst laut dem IREB Glossar „Personen oder Organisationen, welche die Requirements eines Systems beeinflussen oder die von dem System beeinflusst werden“ [1]. Als Stakeholder können beispielsweise die Endnutzer eines Systems bezeichnet werden, die durch das System beeinflusst werden. Sie haben ein Bedürfnis an das System, können dieses jedoch nicht selbst umsetzen. Im Gegensatz dazu stehen die Auftraggeber

bzw. der Produkteigner (Product Owner), welche ebenfalls Stakeholder sind, die aber selbst über die Systementwicklung entscheiden und die Requirements festlegen.

Eine Vielzahl an Stakeholdern, wie beispielsweise der Product Owner, ist nicht unmittelbar in den täglichen Entwicklungsprozess des Systems involviert und verfügt daher nur über eine begrenzte Übersicht über den aktuellen Stand des Systems. Dennoch nehmen sie durch die gestellten Requirements einen maßgeblichen Einfluss auf das System. Für eine effiziente Zusammenarbeit zwischen Stakeholdern und Entwicklern ist es unabdinglich, dass die Requirements sowohl den Stakeholdern als auch den Entwicklern klar und verständlich sind.

Das Verständnis aller Requirements kann insbesondere bei großen und komplexen Systemen herausfordernd sein, da die Zahl der Requirements mit der Komplexität des Systems signifikant ansteigt. Die große Menge an Requirements kann in solchen Projekten schnell zu einer gewissen Unübersichtlichkeit führen. Daher ist es von entscheidender Bedeutung, die Requirements klar und übersichtlich zu dokumentieren und zu verwalten.

Requirements-Engineering

Requirements-Engineering ist der Prozess, in dem Requirements an ein System erhoben, dokumentiert, analysiert, spezifiziert und validiert werden. Laut Chris Rupp besteht Requirements-Engineering dabei aus vier Haupttätigkeiten [2, S.20]:

- Wissen vermitteln
- Gute Requirements herleiten
- Requirements vermitteln
- Requirements verwalten

Requirements-Engineering ist der erste Schritt bei der Entwicklung eines Systems und kann daher große Auswirkungen auf die Qualität und den Erfolg des Systems haben. Ein sauberes Requirements-Engineering verhindert Missverständnisse und Unklarheiten und somit Fehler, welche sich durch die Systementwicklung ziehen und dann mit einem hohen finanziellen und zeitlichen Aufwand behoben werden müssen, wenn sie erkannt sind [2, S.20].

Die vorliegende Bachelorarbeit verfolgt das Ziel, einen innovativen Ansatz zur Vermittlung und Verwaltung von Requirements zu entwickeln, um auf diese Weise die Qualität und Nützlichkeit der Requirements langfristig zu optimieren. Des Weiteren soll die Visualisierung der Requirements dazu beitragen, bei umfangreichen Projekten mit einer Vielzahl an Requirements eine bessere Übersicht über diese zu erlangen, um die Verwaltung der Requirements zu erleichtern. Die Visualisierung kann den Stakeholdern durch eine verbesserte Übersichtlichkeit möglicherweise helfen, die Requirements besser zu verstehen und so potenzielle Missverständnisse und Unklarheiten zu vermeiden. Auf diese Weise können kostspielige Fehler in der Systementwicklung bereits in einem frühen Stadium identifiziert und somit vermieden werden.

3.2 Virtual-Reality

Die Kenntnis und das Verständnis der Begriffe der Virtual-Reality (VR) sind für das Verständnis von Augmented-Reality von entscheidender Bedeutung. Virtual-Reality beschreibt eine Technologie, die es ermöglicht, eine virtuelle Welt zu erschaffen, in der der Nutzer interagieren kann. Dafür wird der Nutzer über verschiedenste audiovisuelle Technologien in diese virtuelle Welt versetzt, die durch Computer generiert wird. Im Gegensatz zu Augmented-Reality ist die gesamte Umgebung in Virtual-Reality digital. [vgl. 4, S.15].

Der Begriff der Virtual-Reality lässt sich noch genauer in die Begriffe immersive und nicht-immersive Virtual-Reality unterteilen. Die folgenden Eigenschaften, die sich auf die Nutzer beziehen, lassen sich als Indikatoren für nicht-immersive Virtual-Reality betrachten:

- Der Nutzer steht nicht im Mittelpunkt.
- Der Nutzer ist nicht vollständig von digitalen Inhalten umgeben.
- Der Nutzer erfährt die virtuelle Realität als Beobachter anstatt als Teilnehmer.

Im Gegensatz dazu sind bei immersiver Virtual-Reality alle äußeren Einflüsse so weit wie möglich reduziert und alle Indikatoren sollten auf immersive Virtual-Reality hinweisen. Der Nutzer sollte bei immersiver VR das Gefühl haben, sich selbst in der virtuellen Umgebung zu befinden [vgl. 5, S.23-24]. Daher geht es bei immersiver virtueller Realität oft vor allem um den visuellen Sinn, da dieser am meisten zur Immersion in die virtuelle Welt beiträgt. In den meisten immersiven VR-Anwendungen wird jedoch auch der auditive Sinn über Kopfhörer oder Lautsprecher angesprochen, um die Immersion zu steigern. Auch der Tastsinn spielt heutzutage eine Rolle, da viele VR-Controller, wie beispielsweise die Meta Quest Touch Plus-Controller, dem Nutzer haptisches Feedback für Interaktionen geben.

3.3 Augmented-Reality

Augmented-Reality (AR) ist eine Technologie, die die reale Welt mit digitalen Informationen erweitert. Der Ansatz bei AR ist ähnlich wie bei Virtual-Reality, allerdings wird die reale Welt nicht komplett ersetzt, sondern nur erweitert. Demnach sieht der Nutzer also weiterhin seine reale Umgebung, diese wird aber durch digitale Informationen ergänzt.

Auf dem Realitäts-Virtualitäts-Kontinuum von Milgram, welches, wie in Abbildung 1 dargestellt, einen fließenden Übergang zwischen Realität und Virtualität beschreibt, liegt Augmented-Reality zwischen der realen Welt und der virtuellen Welt [vgl. 6, S.9].

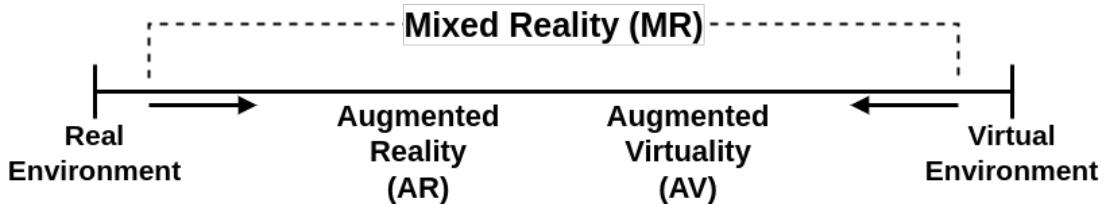


Abbildung 1: Realitäts-Virtualitäts-Kontinuum nach Milgram

Quelle: [6, S.9]

Daher wird Augmented-Reality dem Überbegriff der Mixed-Reality zugeordnet, da die reale Welt mit der digitalen Welt vermischt wird. Infolgedessen ist die Immersion bei Augmented-Reality im Vergleich zu Virtual-Reality oft von geringerer Bedeutung, da die reale Welt stets als Referenzpunkt wahrnehmbar ist. Die Intention von AR-Technologien ist nicht die Versetzung des Nutzers in eine andere Welt, sondern die Erweiterung der realen Welt.

Für diese Erweiterung der Realität müssen die Anzeigegeräte auch Informationen über die echte Umgebung sammeln können. Will man beispielsweise dreidimensionale virtuelle Objekte in die reale Welt einfügen, so muss das Anzeigegerät die eigene Position kontinuierlich bestimmen können um die Position und Rotation des virtuellen Objekts anhand der Bewegungen des Nutzers anzupassen.

Anzeigegeräte für Augmented-Reality haben viele Gemeinsamkeiten mit Anzeigegeräten für Virtual-Reality. Die Besonderheit von AR-Anzeigegeräten besteht jedoch darin, dass sie die reale Welt mit digitalen Informationen erweitern. Das impliziert, dass die Anzeigegeräte dem Nutzer auch eine Sicht auf die reale Welt ermöglichen und in diese Informationen einblenden müssen. Beispielsweise kann das Display eines Anzeigegeräts transparent sein, sodass der Nutzer durch das Display hindurch sehen kann während auf dem Display Daten angezeigt werden können. Alternativ kann das Gerät eine oder mehrere Kameras besitzen, dessen Aufnahmen auf undurchsichtige Bildschirme projiziert werden. Diese Vorgehensweise wird auch als Image-Passthrough bezeichnet.

Es gibt verschiedene Arten von Anzeigegeräten, die für die Darstellung von Augmented-Reality genutzt werden können. Grundsätzlich lassen sich drei Hauptkategorien unterscheiden: Head-Mounted-Displays, Handheld-Geräte und Spatial-Displays [7, S. 346]. Im Folgenden werden diese drei Kategorien genauer erläutert und anhand von Beispielen veranschaulicht.

3.3.1 Head-Mounted Displays

Head-Mounted-Displays (HMDs) sind grundsätzlich Bildschirme, die direkt auf dem Kopf des Nutzers getragen werden. Ihre Nähe zu den Augen des Nutzers ermöglicht die Abdeckung eines großen Sichtfelds, ohne dass dafür eine große Display-Fläche benötigt wird. Folglich sind sie für eine vollständige Immersion im Normalfall kosteneffektiver als großflächige Displays, wie beispielsweise eine Leinwand. Allerdings führt die Befestigung am Kopf automatisch zu einer höheren Gewichtsbelastung für den Nutzer, was bei einer längeren Nutzung als unangenehm empfunden werden kann.

VR- und AR-Headsets

Klassische VR-Headsets kann man sich vorstellen wie eine Skibrille mit zwei Bildschirmen, die auf dem Kopf getragen wird und üblicherweise ein Display hinter jeweils einer Linse pro Auge hat. In Abbildung 2 ist die Meta Quest 3 als Beispiel eines solchen VR- und AR-Headsets dargestellt. Durch die zwei Bildschirme – für welche auch oft ein Bildschirm virtuell in zwei aufgeteilt wird – wird für jedes Auge ein eigenes, leicht verschobenes Bild erzeugt, wodurch Inhalte in 3D dargestellt werden können. Außerdem können sie durch ihre Nähe zu den Augen des Nutzers das volle Sichtfeld des Nutzers abdecken und so eine immersive Erfahrung schaffen.



Abbildung 2: Meta Quest 3 VR- und AR-Headset

Quelle: [8]

Die Nähe des Nutzers zu den Displays führt jedoch auch zu visuellen technischen Problemen, wie beispielsweise dem sogenannten Screen-Door-Effekt, bei dem die Zwischenräume zwischen den Pixeln sichtbar sind. Die entscheidende Kennzahl in diesem Zusammenhang sind die Pixel per Degree (PPD), welche angibt, wie viele Pixel auf ein Grad des Sichtfelds des Nutzers kommen. Eine Möglichkeit um die PPD zu erhöhen und somit den Screen-Door-Effekt zu minimieren, besteht im Einsatz von Displays mit einer hohen Pixeldichte (Pixels per Inch, PPI). Beispielsweise hat die Meta Quest 3 eine PPI von 1218 und erreicht damit einen PPD Wert von 25 Pixeln pro Grad [9]. Vergleichsweise hat das iPhone 15 Pro eine Pixeldichte von lediglich 460 PPI [10]. Trotz der hohen Pixeldichte des Displays der Meta Quest wirkt das Display des iPhone bei normaler Nutzung aufgrund der Entfernung zum Bildschirm schärfer als das der Meta Quest, auch wenn es eine nicht einmal halb so hohe Pixeldichte aufweist.

Da die Displays jedoch das gesamte Sichtfeld des Nutzers einnehmen, kann es bei diesen Anzeigegeräten schnell zu Übelkeitsgefühlen kommen, wenn die Bewegungen des Nutzers nicht mit den Bewegungen in der virtuellen Welt übereinstimmen. Dem wird so weit wie möglich durch eine hohe Bildwiederholrate der Displays entgegengewirkt, um die Bewegungen des Nutzers möglichst flüssig darzustellen. Bei schlecht optimierten Anwendungen kann es jedoch durch Ruckler trotzdem zu Übelkeitsgefühlen kommen.

Um die Bewegung der Nutzer zu verfolgen und so die virtuelle Welt auf den Bildschirmen anpassen zu können ist eine kontinuierliche Bestimmung der Position des Headsets erforderlich. Für diese Bestimmung wird generell zwischen 2 Methoden unterschieden [11]:

- Inside-Out-Tracking: Das Headset trackt seine eigene Position mithilfe von Kameras und Sensoren.
- Outside-In-Tracking: Die Position des Headsets wird durch externe Kameras oder Sensoren bestimmt.

Eine Möglichkeit zur Realisierung von Outside-In-Tracking stellt die Verwendung von Kameras dar, welche im Raum verteilt werden und visuelle Marker am Headset sowie den Steuergeräten tracken. Diese Methode ermöglicht eine präzise Positionsbestimmung, allerdings ist der Nutzer durch die festgelegten Trackingstationen in einem bestimmten Raum und den erforderlichen Sichtkontakt zu den Stationen eingeschränkt. Des Weiteren ist zu berücksichtigen, dass die Installation der Kameras einen erhöhten Aufwand erfordert, da diese zunächst aufgestellt und kalibriert werden müssen.

Eine Methode des Inside-Out-Trackings ist das Berechnen der Position durch Daten aus Kameras und 3D-Sensoren im Headset. Die Umgebung des Nutzers wird aufgenommen und aus den Kamera- und Sensordaten wird ein 3D-Modell der Umgebung erstellt, um die Position des Headsets zu bestimmen. Auf diese Weise wird kontinuierlich eine neue 3D-Repräsentation der Kamera- und Sensordaten erstellt, um die Position des Headsets über die Zeit zu tracken. Für diese Methode des Trackings ist jedoch eine hohe interne Rechenleistung notwendig, um die Daten in Echtzeit auf dem Headset selbst verarbeiten zu können.

Die Interaktion mit der Umgebung erfolgt in der Regel mittels Controllern, deren Position und Rotation ebenfalls kontinuierlich durch Inside-Out- oder Outside-In-Tracking bestimmt werden muss. In AR-Headsets mit integrierten Kameras besteht zudem oft die Möglichkeit, die Hände des Nutzers mithilfe der internen Kameras direkt zu tracken und als Eingabegeräte zu nutzen. In diesem Zusammenhang können spezifische Gesten oder Handbewegungen als Eingaben identifiziert und zur Interaktion genutzt werden. Ein Beispiel für die Interpretation von Gesten ist die Meta Quest 3, welche das Zusammenführen des Zeigefingers und des Daumens als Klick interpretiert. Sofern sich die Finger jedoch nicht vollständig berühren, wird dies als Zeigen interpretiert und lässt einen Zeigestrahl an den Fingerspitzen erscheinen.

Ursprünglich war die Verbindung eines VR- und AR-Headsets mit einem Computer unabdingbar, um die für die Darstellung der Inhalte erforderliche Rechenleistung zu gewährleisten. Im Laufe der Zeit wurden jedoch auch Standalone-VR- und AR-Headsets entwickelt, wie beispielsweise das im Jahr 2024 auf den Markt gebrachte Apple Vision Pro. Diese verfügen über eine eigene Rechenleistung, die in der Regel in Form von stromsparenden ARM-Prozessoren realisiert ist, und können somit unabhängig von einem Computer genutzt werden. Da sie ohne Kabel auskommen, bieten sie eine größere Bewegungsfreiheit und sind daher besser für Anwendungen geeignet, bei denen sich der Nutzer viel bewegt.

Allerdings ist nur eine geringe Anzahl von VR-Headsets auch für AR geeignet. Die meisten VR-Headsets verfügen über keine Kamera, die für eine qualitativ gute Aufnahme und Wiedergabe der realen Welt geeignet sind. Nur Headsets wie beispielsweise die Meta Quest 3, welche in Abbildung 2 dargestellt ist, oder die Apple Vision Pro besitzen eine oder mehrere hochauflösende Kameras nach außen, um die reale Welt aufzunehmen und so AR durch eine Vermischung aus echter Welt und generierten Daten zu ermöglichen.

Zusammenfassend ermöglicht die neueste Generation von VR- und AR-Headsets durch ihre technische Ausstattung eine besonders hohe Immersion, die durch eine herausragende audiovisuelle Qualität für Anwendungen in AR gekennzeichnet ist. Sie sind besonders dafür geeignet, hochauflösende dreidimensionale Inhalte in die reale Welt zu projizieren und so eine immersive Erfahrung zu schaffen. Daher werden sie hauptsächlich für Anwendungen eingesetzt, bei denen eine hohe Immersion und eine hohe Qualität der Darstellung von entscheidender Bedeutung sind, wie beispielsweise bei Spielen oder Simulationen.

Die visuelle Qualität erreicht jedoch noch nicht die Standards moderner Bildschirme für zweidimensionale Betrachtung. Während auf modernen Desktop- und Handy-Displays praktisch keine Pixel mehr zu erkennen sind, ist der Pixel-Door-Effekt auch bei den neusten VR- und AR-Headsets noch klar zu sehen. Folglich haben sie trotz höheren Pixeldichten eine geringere darstellbare Informationsdichte als klassische zweidimensionale Displays. Des Weiteren sind sie für lange Anwendungszeiten und die Verwendung in der Öffentlichkeit in der Regel zu groß und zu schwer, da sie direkt auf dem Gesicht getragen werden müssen.

Aufgrund der Nähe zu den Augen des Nutzers sind sie mit relativ kleinen Bildschirmen ausgestattet, was sie im Vergleich zu vielen anderen dedizierten Anzeigegeräten für AR und VR zu einer kostengünstigen Alternative macht. Die niedrigen Kosten resultieren

auch daraus, dass VR-Headsets auch für Videospiele und andere Unterhaltungsanwendungen genutzt werden und daher eine große Nutzerbasis aufweisen. So existiert ein weiterer finanzieller Anreiz für die Entwicklung von günstigen VR- und AR-Headsets, da die Hersteller von den hohen Stückzahlen profitieren können.

Smart-Glasses

Smart-Glasses sind Brillen, die digitale Informationen in das Sichtfeld des Nutzers einblenden. Sie zeichnen sich durch die Transparenz bzw. Semi-Transparenz der Displays aus, sodass der Nutzer auch ohne Kameras durch die Displays hindurch sehen kann. So kann auch ohne digitales Image-Passthrough ein AR-Effekt erzielt werden. Des Weiteren ist die Durchsicht durch die Displays meist schärfer als bei Image-Passthrough in VR-Headsets, da direkt die echte reale Welt betrachtet wird und nicht eine imperfekte Kameraaufnahme die auf einem imperfekten Bildschirm wiedergegeben wird.

Außerdem sind Smart-Glasses meist leichter und kleiner als VR- und AR-Headsets, da sie in der Regel nicht das volle Sichtfeld des Nutzers abdecken müssen und die Technik für Image-Passthrough nicht erforderlich ist. Folglich eignen sie sich besser für den Alltag und für längere Anwendungszeiten als konventionelle VR- oder AR-Headsets. Zudem verursachen sie durch ihre permanente Transparenz weniger Übelkeitsgefühle bei den Nutzern, da sie sich stets auf die reale Welt als Referenzpunkt beziehen können.

Aufgrund dieser Eigenschaften werden sie viel als potenzielle Anzeigegeräte für AR in der Arbeitswelt untersucht. So haben beispielsweise Forscher der Hochschule Osnabrück und der Universität Osnabrück in nur zwei Logistikunternehmen insgesamt 36 Anwendungsfälle für Smart-Glasses identifiziert [12].

3.3.2 Handheld-Devices

Der Begriff „Handheld-Device“ bezeichnet ein Gerät, welches vom Nutzer in der Hand gehalten wird. Die Kategorie umfasst heutzutage hauptsächlich Smartphones und Tablets. Da die meisten dieser Geräte über eine Kamera und einen Bildschirm verfügen, können sie fast alle für die Darstellung von AR-Inhalten genutzt werden. Dazu wird die Kamera sowie diverse Sensoren genutzt, um die Umgebung des Nutzers aufzunehmen und digitale Informationen in das Kamerabild einzublenden. Das Smartphone oder Tablet fungiert so als eine Art Fenster in die digitale Welt, durch das der Nutzer die erweiterte Realität wahrnehmen kann.

Die Immersion ist bei diesem Anzeigegerät jedoch relativ gering, da der Nutzer stets die reale Welt sieht und das Smartphone oder Tablet lediglich ein kleines Fenster in die digitale Welt darstellt. Allein durch die Entfernung der Geräte von den Augen der Nutzer können sie nur ein vergleichsweise kleines Sichtfeld abdecken.

Die Nutzung von Smartphones für AR ist jedoch weit verbreitet, da nahezu jeder ein Smartphone besitzt und somit keine zusätzliche Hardware benötigt wird. Beispielsweise hatte das AR-Spiel Pokémon Go, welches 2016 veröffentlicht wurde, bereits Anfang 2019 über 1 Milliarde Downloads [13]. Bei dem Spiel werden, wie in Darstellung 3 gezeigt ist, sogenannte Pokémons über die Smartphone-Kamera in die reale Welt projiziert, sodass der Nutzer sie fangen kann. Der Erfolg dieses simplen Konzepts demonstriert das Potenzial der riesigen Nutzergruppe von Smartphones für AR-Anwendungen.

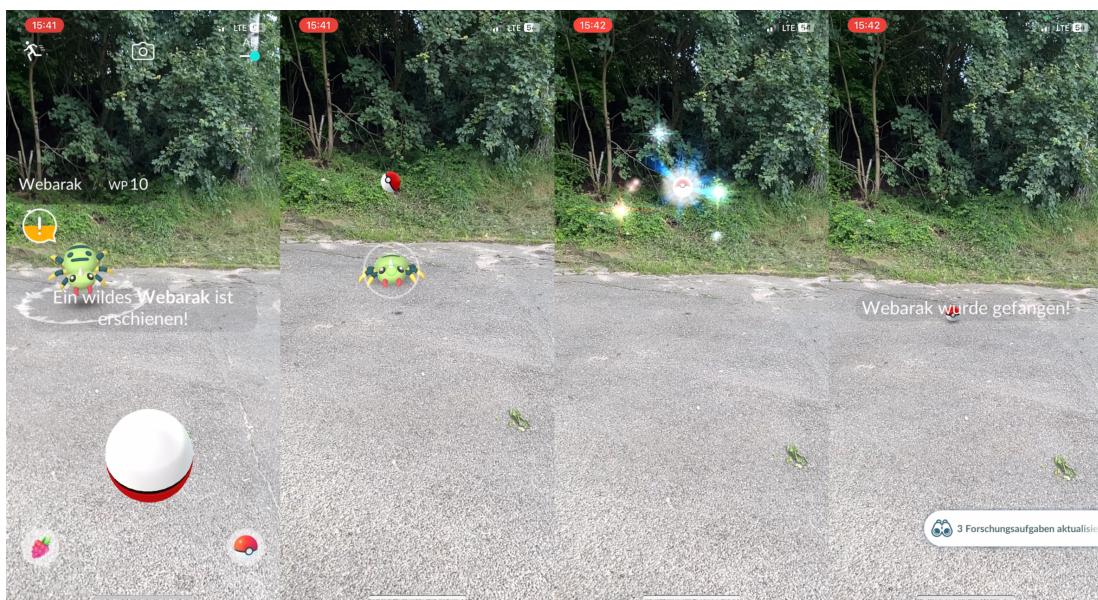


Abbildung 3: Screenshots aus dem AR-Spiel Pokémon Go

Die genannten Technologien können auch in praktischen Anwendungen zum Einsatz kommen, beispielsweise zur Navigation in Städten oder zur Längenmessung mithilfe von Kameras und Sensoren.

Des Weiteren besteht die Möglichkeit, Smartphones als Displays für HMDs zu nutzen. Allerdings ist in den meisten Fällen die Kamera des Smartphones nicht mehr nutzbar, sodass lediglich die Wiedergabe von VR-Inhalten, jedoch keine AR möglich ist. Zudem ist die Pixeldichte von Smartphones meist geringer als bei speziellen VR- und AR-Anzeigegeräten, was die Immersion und Nutzererfahrung verschlechtern kann, da das Display sehr nah an den Augen des Nutzers positioniert ist, wodurch daher starke visuelle Fehler auftreten können.

3.3.3 Spatial-Displays

Die bisher vorgestellten Konzepte für AR-Anzeigegeräte basieren auf Displays die direkt am Nutzer selbst befestigt sind. Im Gegensatz dazu ist die meiste Technik bei Spatial-Displays in der Umgebung des Nutzers verbaut. Der Nutzer selbst muss bei Spatial-Displays meist keine spezielle Hardware tragen, um die AR-Inhalte zu sehen [14]. Jedoch können beispielsweise zur Interaktion mit Inhalten Eingabegeräte genutzt werden, die der Nutzer in der Hand hält.

In den meisten Fällen muss keine spezielle Hardware am Nutzer selbst befestigt werden, was zu einer hohen Immersion durch Spatial-Displays führen kann, wenn die audiovisuelle Immersion auch gegeben ist. Auch das ergonomische Nutzererlebnis über lange Zeiträume kann durch Spatial-Displays verbessert werden, da der Nutzer nicht durch das Tragen von schwerer Hardware belastet wird.

Ein wesentlicher Nachteil von Spatial-Displays ist jedoch der Abstand der Displays zum Nutzer. Da sie weiter vom Nutzer entfernt sind als HMDs oder Handheld-Devices können sie mit der gleichen Displaygröße nur ein kleineres Sichtfeld abdecken. Daher werden bei Spatial-Displays oft sehr große Displays wie bspw. Beamer genutzt, um trotz der größeren Distanz zum Nutzer ein großes Sichtfeld abzudecken. Folglich sind Spatial-Displays mit höheren Kosten und einem größeren Aufwand bei der Installation verbunden als HMDs oder Handheld-Devices. Daher werden sie in der Regel in anspruchsvollen Industrieanwendungen eingesetzt und eignen sich meist nicht für den privaten Gebrauch.

Eine solche industrielle Anwendung ist ein Fahrimulator von Mercedes – dargestellt in Abbildung 4 – bei der die kompletten Wände von 14 Projektoren bespielt werden, um eine immersive Umgebung für den Fahrimulator zu schaffen. Die Anzeige der Inhalte ist auf die Fahrerposition des Fahrzeugs abgestimmt, sodass der Fahrer das Gefühl hat, sich tatsächlich in einem Fahrzeug zu befinden. Mit solchen hoch entwickelten Spatial-Displays können also sehr realistische und immersive Umgebungen geschaffen werden, die für viele Anwendungen, wie bspw. Fahrsimulatoren, sehr nützlich sind. Allerdings sind derartige Darstellungsmöglichkeiten mit hohen Kosten und einem erheblichen Aufwand verbunden, der im Vergleich zu anderen Lösungen deutlich höher ist.



Abbildung 4: Innenraum des Mercedes-Benz Fahrsimulators

Quelle: [15]

Betrachtet man Spatial-Displays jedoch nicht nur auf Basis der Immersion, die durch sie erreicht werden kann, ergeben sich auch viele Anwendungsmöglichkeiten für Spatial-Displays, die nicht auf Immersion abzielen. Beispielsweise können Spatial-Displays genutzt werden, um Informationen in die reale Welt zu projizieren, die für den Nutzer nützlich sind. So zählen auch Head-Up-Displays in Fahrzeugen zu der Kategorie der Spatial-Displays. Sie projizieren Informationen in das Sichtfeld des Fahrers, ohne dass sich dieser auf ein Display abseits der Straße vor ihm konzentrieren muss. Sie nutzen ähnliche Technologien wie Smart-Glasses, um Informationen auf semi-transparente Displays zu projizieren.

3.4 Gamification von UX

„Gamification ist die Nutzung von Spielmechanismen und -designprinzipien in nicht-spielerischen Kontexten um das Nutzerengagement zu erhöhen“ [16, S.63]. So beschreibt Cornel Hillmann, ein Experte für User-Experience-Design, die Bedeutung von Gamification von UX.

Dieses Prinzip lässt sich aufgrund der hohen Immersion und Interaktivität von VR und AR in XR-Anwendungen besonders gut und quasi standartmäßig nutzen, um das Nutzerengagement zu erhöhen. Auch durch den Neuheitsfaktor von AR und VR können Nutzer dazu motiviert werden, die Anwendung mehr zu nutzen.

Hillmann gibt drei Hauptbestandteile seines Ansatzes von Gamification von UX an, die in einer Anwendung vorhanden sein müssen, um das Nutzerengagement zu erhöhen [16, S.66]:

- Motivation: Die Anwendung muss Elemente enthalten, die den Nutzer zur weiteren Nutzung motivieren.
- Herausforderung: Der Nutzer muss eine Herausforderung haben, die er bewältigen kann.
- Trigger: Der Nutzer muss für die Herausforderung belohnt werden.

Der Ansatz von Hillmann basiert auf der Nutzung der Belohnungsmechanismen von Spielen, um die Motivation der Nutzer zu fördern. Eine von Hillmann beschriebene Möglichkeit ist die Nutzung von Punkten, um den Fortschritt des Nutzers zu visualisieren und ihn somit zu motivieren, das Arbeiten in der Anwendung fortzusetzen.

Hillmann beschreibt auch, wie Gamification von UX bei der Orientierung, Beruhigung und Onboarding-Problemen der neuen Technologie von XR helfen kann. So zeigt sich in der Praxis beispielsweise häufig, dass neue Nutzer nicht in die richtige Richtung schauen, um die Inhalte zu sehen, oder von der neuen Technologie überfordert sind. Daher kann durch eine intelligente Anordnung der Objekte und eine spielähnliche Zielführung die Orientierung sowie die User-Experience der Nutzer verbessert werden [16, S.67].

4 Technologien

In diesem Kapitel wird ein Überblick über die Technologien gegeben, die zum Implementieren der Prototypen zu den Interaktionskonzepten genutzt werden. Im Folgenden wird zunächst auf die Hardware eingegangen, wobei das verwendete Anzeigegerät für AR, die Meta Quest 3, näher erläutert wird. Im Anschluss erfolgt eine Vorstellung der Software-Frameworks, in welchen die Prototypen implementiert werden.

4.1 Meta Quest 3

Im Rahmen der Bachelorarbeit wird als Anzeigegerät für die AR-Anwendung die Meta Quest 3 verwendet. Hierbei handelt es sich um ein Standalone-AR- und VR-Headset, welches von Meta (ehemals Facebook) entwickelt wurde und Ende 2023 auf den Markt kam. Das HMD ist mit einem eigens für mobile XR-Geräte entwickelten Qualcomm Snapdragon XR2 Gen2 Prozessor ausgestattet, wodurch es selbst in der Lage ist, anspruchsvolle Inhalte zu rendern. Es muss nicht wie viele andere HMDs an einen Computer oder ein Smartphone angeschlossen werden, sondern kann direkt allein verwendet werden. Das Headset verfügt beispielsweise auch über einen eigenen Browser, wodurch AR-Anwendungen einfach übers Internet aufgerufen werden können. Dieses Nutzen von Anwendungen über den Browser wird auch für die Prototypen dieser Arbeit genutzt, worauf in Kapitel 4.3 genauer eingegangen wird.

Das Headset verfügt über zwei Displays mit einer Auflösung von jeweils 2064 x 2208 Pixeln und einer Bildwiederholrate von bis zu 120 Hz. Dadurch werden Inhalte mit einer hohen Bildqualität und einer flüssigen Darstellung angezeigt. Die Kombination der geringen Latenz aufgrund des On-Board-Prozessors mit der hohen Bildwiederholrate resultiert in einer hohen Immersion und einem angenehmen Nutzungserlebnis.

Darüber hinaus verfügt die Meta Quest 3, wie in Abbildung 5 dargestellt, über zwei RGB-Kameras (untere Kameras), zwei IR-Kameras (obere Kameras) sowie einen Tiefenprojektor (schwarze Fläche in der Mitte) auf der Vorderseite, welche gemeinsam die Umgebung des Nutzers erfassen und so AR-Anwendungen mittels Image-Passthrough ermöglichen.



Abbildung 5: Frontal ausgerichtete Kameras und Tiefenprojektor der Meta Quest 3

Zusätzlich zu den zwei RGB-Kameras sind insgesamt noch vier Infrarot-Kameras auf dem Headset verbaut, zwei davon auf der Vorderseite, direkt über den RGB-Kameras, und zwei auf der Unterseite. Über diese Kameras wird die Position des Nutzers im Raum kontinuierlich getrackt und angepasst, sodass auf externe Trackingstationen verzichtet werden kann.

Die Meta Quest 3 ermöglicht zudem die Erkennung und das Tracking der Hände des Nutzers mithilfe der vier Infrarotkameras und zwei RGB-Kameras als alternative Eingabegeräte. Dadurch kann der Nutzer seine eigenen Hände in VR- und AR-Anwendungen als Eingabegerät verwenden [9]. Dafür sind auch zwei der vier Infrarotkameras wie in Abbildung 6 zu sehen ist nach unten gerichtet, um die Hände des Nutzers jederzeit tracken zu können.



Abbildung 6: Die beiden nach unten gerichteten Kameras der Meta Quest 3

4.2 WebGL

WebGL ist eine JavaScript-API, die das Rendern von 3D-Grafiken in einem Webbrowser ermöglicht. Die WebGL-Spezifikation wird von der Khronos Group entwickelt, einer Non-Profit-Organisation, welche als ein Zusammenschluss aus über 180 Unternehmen aus der Tech-Industrie technologische Standards entwickelt. Zu den Unternehmen, die an der Entwicklung von WebGL beteiligt sind, gehören auch die Unternehmen, die die größten Browser der Industrie entwickeln: Google (Chrome), Mozilla (Firefox), Apple (Safari) und Microsoft (Edge) [17, 18]. Aufgrund dieser Mitarbeit verschiedener Unternehmen und der offenen Entwicklung der Spezifikation kann WebGL in fast allen modernen Webbrowsersn verwendet werden.

WebGL basiert auf der OpenGL ES Spezifikation, welche, ebenfalls von der Khronos Group, speziell für Embedded Systems und mobile Systeme, also zum Großteil Systeme ohne eigene Grafikkarte, entwickelt wurde [19]. Aufgrund dieser Ausrichtung auf mobile Geräte, können mit WebGL Anwendungen entwickelt werden, die auf einfachen Geräten wie Smartphones oder aber auch auf VR- und AR-Headsets ohne zusätzliche Rechenleistung laufen [20, S.3].

4.3 WebXR

Die WebXR-API ist eine standardisierte Schnittstelle für Webanwendungen, die die Erstellung immersiver VR- und AR-Anwendungen für das Internet ermöglicht. Das World Wide Web Consortium (W3C), welches die WebXR-Standards definiert beschreibt WebXR wie folgt: „Die WebXR Device API bietet die notwendigen Schnittstellen, damit Entwickler ansprechende, komfortable und sichere immersive Anwendungen im Web für eine Vielzahl von Hardware-Formfaktoren erstellen können“ [21, 1. Introduction]. Mit WebXR entwickelte Anwendungen können als Webanwendungen direkt von einem Webbrowser eines AR- oder VR-Geräts aus aufgerufen werden, ohne dass eine zusätzliche Installation der Anwendung notwendig ist.

Der Ablauf von WebXR-Anwendungen ist in Abbildung 7 dargestellt. Dabei wird zuerst beim Aufrufen der Webseite geprüft, ob die angegebene Art von XR-Inhalt von der Hardware und dem User Agent (UA) unterstützt werden. Der User Agent der Software bezeichnet die Software, mit welcher der Nutzer auf die Anwendung zugreift, also den Webbrowser des Nutzers. Ist die Art des XR-Inhalts von der Technik des Nutzers unterstützt, kann die XR-Session vom Nutzer manuell über einen Knopf gestartet werden. Nach erfolgreichem Start der Session wird der Frame Loop der Anwendung gestartet. Der Frame Loop ist eine Schleifenfunktion, die kontinuierlich während der XR-Session ausgeführt wird um die einzelnen Frames, bzw. Bilder, für das XR-Display zu rendern. Dieser Frame Loop läuft, bis die Session durch den UA beendet wird. Befindet sich der UA noch auf der Seite der WebXR-Anwendung, wird wieder der Knopf zum Starten der XR-Session angezeigt, falls der Nutzer direkt wieder eine neue Session starten möchte.

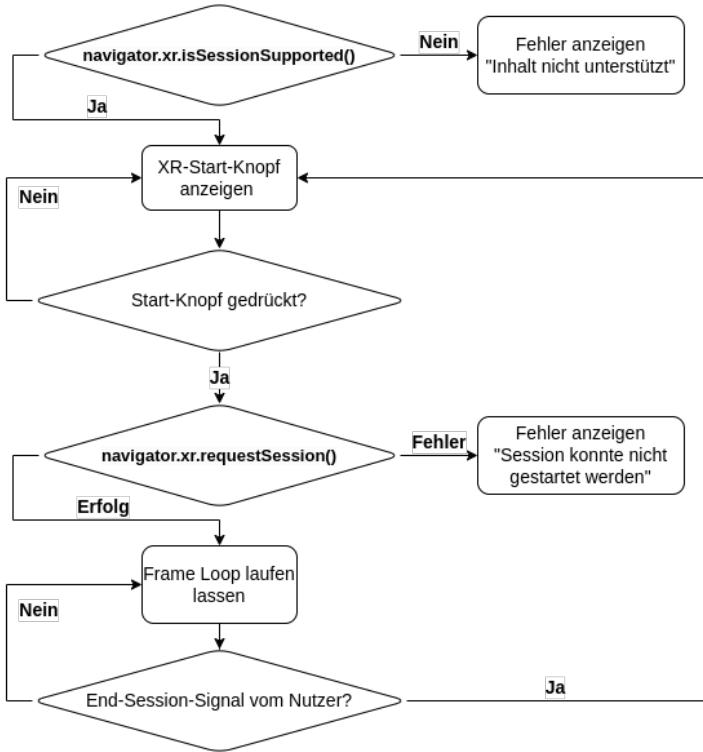


Abbildung 7: Application Flow von WebXR-Anwendungen

Quelle: nach [21, Kap. 1.2. Application Flow]

Three.js

Three.js ist eine Open-Source-JavaScript-Bibliothek, die unter der MIT-Lizenz steht und auf WebGL basiert. Sie vereinfacht die Erstellung und Darstellung von 3D-Grafiken in Webanwendungen. Die Bibliothek bietet eine Vielzahl von Funktionen, die es Entwicklern ermöglichen, detaillierte 3D-Modelle und -Szenen zu erstellen und zu rendern [22]. Dabei unterstützt Three.js auch die Erstellung von AR- und VR-Anwendungen mithilfe von WebXR.

Des Weiteren existieren Frameworks, die auf der Three.js API basieren und die Entwicklung von AR- und VR-Anwendungen sowie 3D-Anwendungen im Allgemeinen weiter vereinfachen. Ein Beispiel dafür ist das Framework A-Frame, welches auf die Entwicklung von AR- und VR-Anwendungen spezialisiert ist und Funktionen einer nahezu vollwertigen Game-Engine bietet [23].

Zudem wird Three.js von vielen Entwicklern und Unternehmen verwendet und hat dadurch eine große Community und viele Ressourcen und Tutorials, welche Entwicklern bei der Erstellung und Fehlerbehebung ihrer Anwendungen behilflich sein können.

Die Vielzahl an Funktionen sowie die große Nutzerbasis machen Three.js, insbesondere in Kombination mit A-Frame, zu einer vielversprechenden Option für die Entwicklung von

AR- und VR-Anwendungen. Jedoch ergaben sich in den ersten Tests der Funktionalität von Three.js und A-Frame im Browser der Meta Quest 3 Probleme mit der Erkennung der AR-Brille und der Darstellung von AR-Inhalten.

Babylon.js

Babylon.js ist eine Open-Source-Rendering- und Game-Engine, die in einer JavaScript-Bibliothek verpackt ist. Sie wird unter anderem von einem Team von Entwicklern bei Microsoft entwickelt. Die Bibliothek bietet Support für WebGL 1.0/2.0 sowie WebGPU und verfügt über eine Vielzahl von Funktionen, die es Entwicklern ermöglichen, detaillierte 3D-Modelle und -Szenen zu erstellen und zu rendern. Darüber hinaus bietet es native Unterstützung sowie eine Dokumentation für WebXR, wodurch die Entwicklung von VR- und AR-Anwendungen vereinfacht wird [24].

Die Bibliothek bietet auch viele Quality-of-Life-Features, wie beispielsweise einen Szenen-Explorer und -Inspektor, der sich für die Entwicklung und das Debugging von 3D-Szenen direkt im Browser eignet und mit einer Zeile Code in die Anwendung eingebunden werden kann. Über den Szenen-Explorer, welcher in Abbildung 8 links dargestellt ist, können aus dem Szenengraph der aktuellen Szene Objekte ausgewählt werden um im Szenen-Inspektor eine Detailansicht des Objekts und seinen Eigenschaften zu erhalten. Mithilfe des Szenen-Inspektors, welcher in Abbildung 8 rechts dargestellt ist, können dann in Echtzeit die Eigenschaften von ausgewählten Objekten betrachtet und verändert werden.

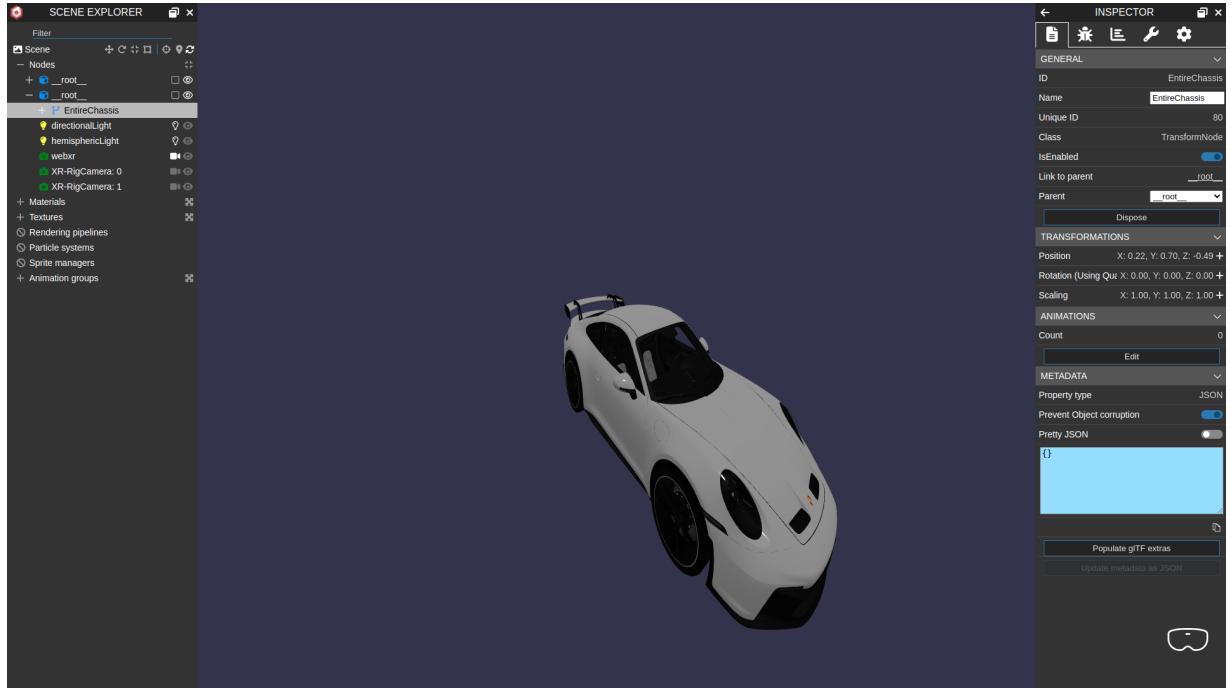


Abbildung 8: Szenen-Explorer und -Inspektor von Babylon.js

5 Konzept

Die vorliegende Bachelorarbeit beschäftigt sich mit der Darstellung von Requirements in Augmented-Reality. Im Rahmen dessen wird untersucht, wie Requirements in einer 3D-Umgebung dargestellt werden könnten. Dazu werden verschiedene Interaktionskonzepte entwickelt, welche als Basis für die Implementierung von Prototypen mit WebXR dienen sollen. Die erste Ausarbeitung der Konzepte dient dabei nicht unbedingt dazu, vollständig so implementiert zu werden, sondern soll einen Überblick über die grundlegende Idee und die möglichen Interaktionen geben.

5.1 Explodierende Bauteile

Das erste untersuchte Interaktionskonzept ist hauptsächlich für die Darstellung von Requirements von physischen Produkten gedacht. Die Idee ist, ein Produkt in einer Animation in seine einzelnen Bauteile zu zerlegen und die Requirements direkt an ihren zugehörigen Bauteilen darzustellen. In der Animation werden die Bauteile wie in der Bildsequenz in Abbildung 9 von einem Punkt in der Mitte des Produkts nach außen bewegt, sodass sie sich um den Ursprungspunkt des Produkts herum anordnen. Ein Fahrzeug könnte beispielsweise so zerlegt werden, dass sich bei der Animation die Räder, die Karosserie, der Motor und die Innenausstattung einzeln als eigene Objekte auseinanderbewegen. Auf diese Weise kann der Nutzer das gesamte Produkt betrachten und sich dann auf Wunsch einzelne Bauteile und deren Requirements genauer ansehen. Im Beispiel des Rubiks Cube aus Abbildung 9 wäre dann jeder Würfel ein Bauteil, der seine eigenen Requirements angehängt bekommen würde.



Abbildung 9: Beispiel einer Explosionsanimation anhand eines Rubiks Cube

Die Requirements sollen in Form von Text auf UI-Panels dargestellt werden, welche an den zugehörigen Bauteilen angebracht sind.

Bei komplexen Produkten mit einer Vielzahl an Bauteilen, wie beispielsweise einem Fahrzeug, existiert eine hohe Anzahl an Requirements. Eine gleichzeitige Animation aller Komponenten würde dann bei komplexen Produkten zu einer übermäßigen Zahl an Requirements führen, die gleichzeitig dargestellt werden müssten. Daher ist es bei umfangreichen Produkten eventuell sinnvoll, die Darstellung der Bauteile in mehreren Schritten zu realisieren. Dafür könnten Bauteile des Produkts, die aus mehreren anderen Bauteilen bestehen, als separate Produkte mit ihrer eigenen Explosionsanimation betrachtet werden. Ein Beispiel für eine Komplettübersicht ist die Darstellung eines kompletten Fahrzeugs in wenigen Bauteilen. Dabei wird ein Rad, das aus Reifen, Felge, Radmuttern, Bremsscheibe etc. besteht, als ein einzelnes zusammengehöriges Objekt behandelt. Bei Bedarf kann der Nutzer durch Auswahl eines Bauteils in eine Detailansicht wechseln, in der lediglich ein Rad mit all seinen Bauteilen animiert wird. Durch eine Verschachtelung von immer detaillierteren Ansichten kann eine hohe Komplexität der Darstellung erreicht werden, ohne dass die Übersichtlichkeit durch eine zu hohe Anzahl an Requirements gleichzeitig verloren geht. Dieses Prinzip der Verschachtelung ist auch im Ablaufdiagramm in Abbildung 10 anhand der beiden rechts und links abgehenden Schleifen des unteren Entscheidungsblocks zu erkennen.

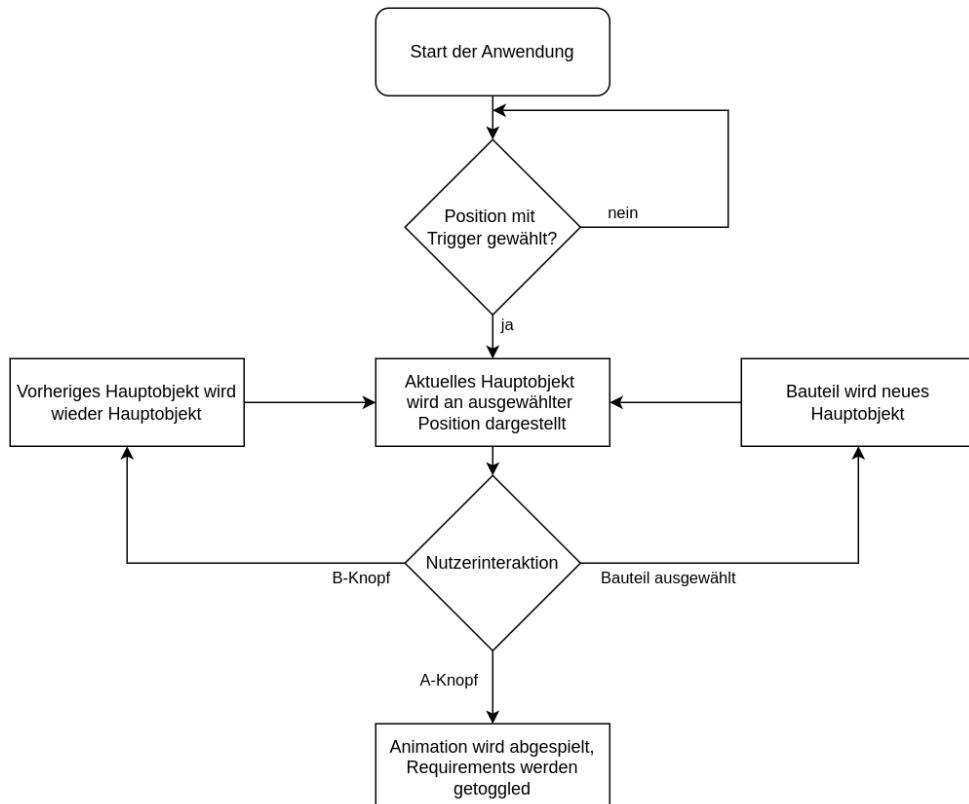


Abbildung 10: Ablaufdiagramm des Interaktionskonzepts der Explodierenden Bauteile

Für diese Darstellung soll der Nutzer, wie am ersten Entscheidungsblock im Diagramm in Abbildung 10 zu sehen ist, zunächst mithilfe eines Controllers einen Ort für die Darstellung im Raum auswählen. Der ausgewählte Punkt im Raum wird als Ursprungspunkt für die Darstellung der Bauteile verwendet und das 3D-Modell an diesem Punkt platziert. Anschließend hat der Nutzer die Möglichkeit, die Explosionsanimation des aktuell angezeigten Modells per Knopfdruck zu aktivieren. Beim Drücken des A-Knopfs wird dann die Animation des aktuellen 3D-Modells abgespielt und den Bauteilen werden ihre Requirement-Panels angehängt.

Alternativ besteht für den Anwender die Möglichkeit, mit dem Trigger ein anderes Bauteil auszuwählen, um dieses als neues Hauptmodell anzuzeigen und somit dessen Requirements zu betrachten. Durch Betätigen des B-Knopfes erfolgt eine Reduzierung des Detailgrads der Darstellung, sodass die vorherige Ansicht wiedergegeben wird.

Auch bei Software-Requirements ist es denkbar, diese in ihre Komponenten zu zerlegen und zu diesen Komponenten zugehörige Requirements darzustellen. Jedoch bietet die Darstellung in AR bzw. VR hierbei quasi keine Vorteile im Gegensatz zu einer 2D-Darstellung auf einem konventionellen Bildschirm. Bei Software-Anwendungen ist die einfache Darstellung auf einem Bildschirm näher an der tatsächlichen Laufumgebung der meisten Softwareprodukte, weshalb das Konzept eher für physische Produkte sinnvoll ist.

Bereits bei der Erstellung des Konzepts wird ersichtlich, dass einige der Implementierungsschritte, wie beispielsweise das Erstellen der Animation, vermutlich einen hohen Zeitaufwand erfordern werden. Im Rahmen dieses Interaktionskonzepts soll daher die Realisierbarkeit solcher Darstellungen für physische Produkte untersucht werden. In diesem Zusammenhang ist zu berücksichtigen, dass aufgrund des zu erwartenden hohen Zeitaufwands für die Implementierung eine kritische Betrachtung der Kosten-Nutzen-Relation erforderlich ist. Um dennoch eine sinnvolle Grundlage für eine tatsächliche Anwendung zu bieten, muss das Interaktionskonzept einen wesentlichen Mehrwert bei der Vermittlung der Requirements aufweisen.

5.2 Wolken von Requirements

Der Ansatz der explodierenden Bauteile ist aufgrund der individuellen Ausgestaltung des Interaktionskonzepts mit einem hohen Zeitaufwand und einer komplexen Umsetzung verbunden. Des Weiteren ist zu berücksichtigen, dass dieser Ansatz lediglich für die Darstellung von Produkten, also physischen Systemen, geeignet ist. Daher wird ein weiteres Interaktionskonzept entwickelt, welches sich theoretisch auch automatisiert generieren lassen soll und für alle Arten von Requirements geeignet ist.

Des Weiteren ist beabsichtigt, eine Übersicht über eine größere Anzahl von Requirements gleichzeitig zu erlangen, als im ersten Interaktionskonzept möglich war. Dadurch soll eine visuelle Darstellung vieler Requirements gleichzeitig ermöglicht werden.

Die grundlegende Idee für ein Interaktionskonzept, welches diese Anforderungen erfüllt ist, Requirements in Wolken von Texten darzustellen, also als wolkenförmige Gruppierungen von UI-Elementen im Raum. Die UI-Elemente sollen Panels sein, auf denen der Text der Requirements sowie weiterführende Informationen zu den Requirements dargestellt werden. Diese Panels sollen dann in einer wolkenartigen Anrichtung, ähnlich wie die Punktewolke in Abbildung 11, dargestellt werden.

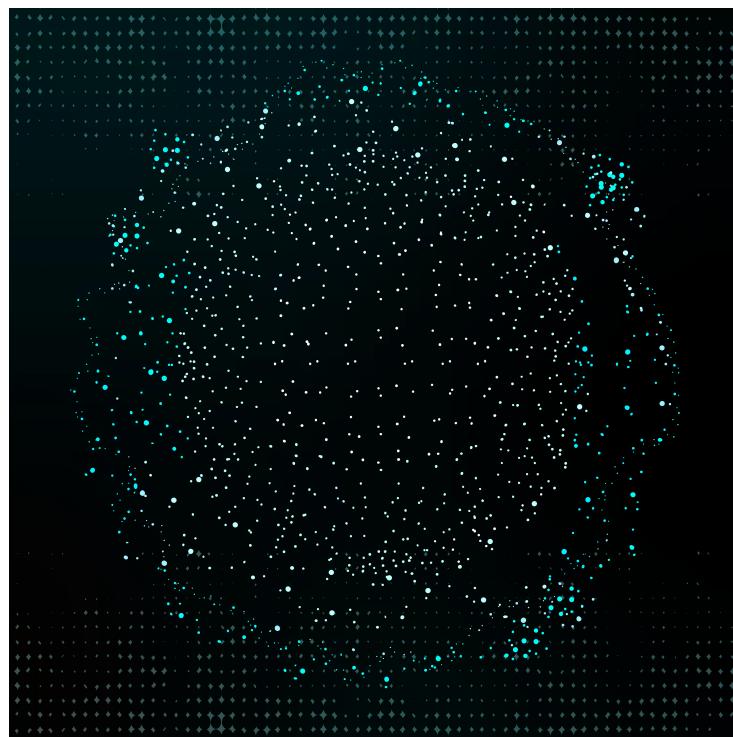


Abbildung 11: Beispiel einer Punktewolke

Quelle: [25]

Im Rahmen dessen soll eine räumliche Gruppierung der Requirements nach verschiedenen

Kriterien ermöglicht werden. So wäre es beispielsweise denkbar, Requirements, die zu einem bestimmten Feature gehören, in einer Wolke zu gruppieren, während Requirements, die zu einem anderen Feature gehören, in einer anderen Wolke gruppiert werden. Die räumliche Anordnung der Wolken soll es dem Nutzer ermöglichen, schnell zu erkennen, welche Requirements nach der aktuellen Filterung eine hohe Korrelation aufweisen und welche nicht.

Des Weiteren ist vorgesehen, dass eine Zoomfunktion implementiert wird, mittels derer es möglich ist, in die Wolke hinein- und herauszuzoomen, um die Granularität der angezeigten Requirements zu erhöhen. Ein Beispiel für eine mögliche Umsetzung wäre, dass in großen Wolken initial nur Punkte als Repräsentationen von Requirements angezeigt werden, in welche man hereinzoomen kann, um die Requirements zu lesen. In einem nächsten Schritt besteht die Option, wieder heraus zu zoomen und somit wieder eine Übersicht über eine größere Anzahl an Requirements der Wolke zu erlangen.

Eine weitere Möglichkeit wäre die Implementierung von Filtermöglichkeiten für verschiedene Beziehungen der Requirements auf den Requirementspanels, sodass bei einer Auswahl in die neue Requirementswolke gewechselt werden kann. Die Interaktion sowie die räumliche Anordnung der Wolken zielen darauf ab, dem Nutzer auch bei einer großen Anzahl von Requirements einen Überblick zu gewährleisten und eine schnelle Auffindbarkeit der gewünschten Requirements zu ermöglichen.

Bei diesem Konzept soll insbesondere der Vorteil gegenüber einer zweidimensionalen Darstellung kritisch analysiert werden. Denn die Darstellung der Requirements in Wolken ist prinzipiell auch in 2D denkbar, sogar mit der Interaktionsmöglichkeit des Hinein- und Herauszoomens. In diesem Zusammenhang ist zu untersuchen, ob die räumliche Anordnung der Requirements in AR tatsächlich einen Mehrwert gegenüber einer 2D-Darstellung bietet und ob die Interaktionen intuitiv und effizient sind. Zudem soll ein Fokus darauf liegen, auch große Zahlen von Requirements gleichzeitig darzustellen.

6 Implementierung

In diesem Kapitel wird der Prozess der Implementierung der in Kapitel 5 erarbeiteten Interaktionskonzepte in Prototypen dargelegt. Zunächst wird ein Überblick über die Entwicklungsumgebung gegeben, welche für die Entwicklung mit WebXR auf einer Meta Quest 3 eingerichtet wurde. Im Anschluss erfolgt eine Beschreibung der Implementierung der Prototypen für jedes Interaktionskonzept, wobei einige technische Details erläutert werden.

6.1 Entwicklungsumgebung für WebXR und Meta Quest 3

Die Entwicklung von WebXR-Anwendungen für die Meta Quest 3 erfordert eine spezielle Entwicklungsumgebung für einen effizienten Entwicklungsprozess. Diese muss verschiedene Technologien und Tools miteinander kombinieren, um eine reibungslose Entwicklung und ein möglichst schnelles Testen der Anwendung zu gewährleisten.

Der erste Schritt besteht in der Anzeige der WebXR-Anwendung auf der Meta Quest 3. Im Vergleich zu „normalen“ Webanwendungen ist dabei zu beachten, dass WebXR-Anwendungen ausschließlich über HTTPS aufgerufen werden können. Das bedeutet, dass die Anwendung über HTTPS gehostet werden muss, um direkt vom Browser der Meta Quest 3 aufgerufen werden zu können. Dafür existieren verschiedene Möglichkeiten, wie beispielsweise das Erstellen eines eigenen Zertifikats für den lokalen Entwicklungsrechner oder das Hosting der Anwendung auf einem Server mit HTTPS-Unterstützung. Im Rahmen der Entwicklung dieser Bachelorarbeit wird die Anwendung jedoch – wie auch in der Artikelserie des Taikonautenmagazins [26, Part 0/8] empfohlen – mit LocalTunnel gehostet, um die Anwendung direkt von der Meta Quest 3 aus testen zu können. LocalTunnel generiert einen temporären HTTPS-Link, über den die Anwendung aufgerufen werden kann, ohne dass ein eigenes Zertifikat oder ein eigener Server erforderlich ist. Dazu muss die Anwendung lediglich lokal auf dem Entwicklungsrechner ausgeführt werden und der LocalTunnel-Client gestartet sein, um einen temporären Link zu erstellen. Um eine zusätzliche Sicherheit zu gewährleisten, muss beim Aufrufen der Seite die IP-Adresse des Entwicklungsrechners angegeben werden. Auf diese Weise kann sichergestellt werden, dass ausschließlich der Entwickler die Anwendung testen kann. Zudem wird so verhindert, dass Betrüger LocalTunnels nutzen, um HTTPS-Websites nachzuahmen. Diese Eingabe muss in der Regel jedoch nur einmal nach jedem Neustart oder Crash gemacht werden, da der

Link für die Dauer der Sitzung gespeichert wird. Allerdings neigt der LocalTunnel dazu, regelmäßig abzustürzen, weshalb auch diese Lösung als suboptimal zu bewerten ist.

Der nächste Schritt, sofern ein Zugriff auf die WebXR-Anwendung mit der Meta Quest 3 besteht, umfasst die Anzeige von Entwickler-Tools und Debugging-Informationen der Meta Quest 3. Diese Arbeit sowie die Prototypen wurden nahezu vollständig unter Linux entwickelt. Für das Debuggen von Meta Geräten kann auf Windows die Meta Quest Developer Hub App (MQDH) genutzt werden, welche jedoch zum Zeitpunkt dieser Arbeit noch nicht auf Linux installiert werden kann. Daher mussten für das Debuggen der Meta Quest 3 Umwege gefunden werden.

Da die Meta Quest 3 auf Android basiert, besteht die Möglichkeit, auf dem Linux-Entwicklungsrechner die Android Debug Bridge (ADB) zu installieren, um über USB eine Verbindung zur Meta Quest 3 herzustellen. Die Verbindung muss zusätzlich noch in der AR-Brille bestätigt werden, um den Zugriff auf die Entwickleroptionen zu ermöglichen. Sobald dieser Vorgang abgeschlossen ist, wird die Quest 3 mit ihren geöffneten Websites in der Geräteliste der Chrome DevTools angezeigt, welche über die URL `chrome://inspect/#devices` aufgerufen werden kann. Von dort aus können dann die Entwickler-Tools der Meta Quest 3 geöffnet werden, um beispielsweise die Performance der Anwendung zu überwachen und Fehlermeldungen zu sehen.

Viele Aspekte der Entwicklung von XR-Anwendungen, wie beispielsweise einfache Tests von Interaktionen wie einzelnen Klicks können auch über einen WebXR-Emulator direkt am Entwicklungsrechner getestet werden. In dieser Arbeit wird dafür die Chrome-Erweiterung „Immersive Web Emulator“ von Meta verwendet, welche das Testen von WebXR-Anwendungen direkt im Browser ermöglicht [27]. Mit dieser Erweiterung wird für WebXR ein virtueller Raum mit einem Headset und den beiden Meta Quest Controllern simuliert. Die beiden Controller sowie das Headset können unabhängig voneinander über die Erweiterung positioniert und gesteuert werden. Das in Abbildung 12 links dargestellte Panel ermöglicht die Simulation verschiedener Interaktionen, wie beispielsweise Klicks oder das Bewegen der Controller, um die Anwendung zu testen. Der Emulator generiert in Echtzeit eine Vorschau der Anwendung, die direkt im Browser angezeigt wird.

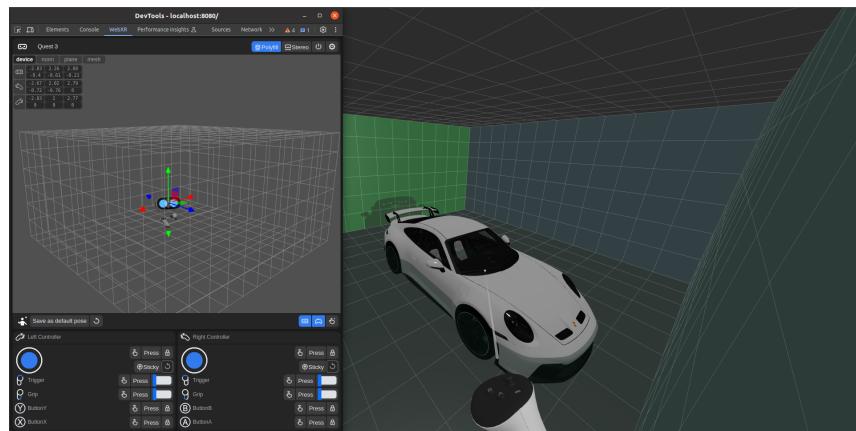


Abbildung 12: Screenshots der Immersive Web Emulator Erweiterung in Chrome

Die Anwendung kann mit dieser Erweiterung deutlich schneller und einfacher getestet werden als mit der Meta Quest 3 selbst, da keine Verbindung zu einem externen Gerät notwendig ist und die Anwendung direkt im Browser des Entwicklungsrechners getestet werden kann. Des Weiteren sind die Ladezeiten beim Neuladen der Anwendung deutlich kürzer, da die Anwendung nicht auf die Übertragung der Daten auf die Meta Quest 3 warten muss. Aufgrund der relativ großen Datenmengen, die für die Darstellung der 3D-Modelle erforderlich sind, ist der Unterschied der Ladezeiten zwischen Headset und Erweiterung signifikant. Daher wurde jede neue Änderung zunächst im Emulator getestet, bevor die Anwendung im Headset neu geladen wurde.

Dennoch ist es von essenzieller Bedeutung, die Anwendung auch regelmäßig auf der Meta Quest 3 zu testen, da die Performance und die Interaktionen auf dem Emulator nicht immer exakt der Realität entsprechen. So können beispielsweise leichte Ruckler auftreten, die durch eine unzureichende Optimierung der Anwendung bedingt sind. Diese Ruckler sind nur bei der höheren Bildwiederholrate des Headsets sichtbar und können daher im Emulator nicht erkannt werden. Daher wurde vor allem nach großen Änderungen des Prototyps stets auch mit dem AR-Headset selbst getestet. Für kleinere Anpassungen, wie beispielsweise das Positionieren von Objekten, die häufig getestet werden müssen, wurde teilweise nur der Web-Emulator genutzt, um die lange Ladezeit zu vermeiden. Die Vorgehensweise beim Testen ist im folgenden Diagramm in der Abbildung 13 zusammenfassend dargestellt.

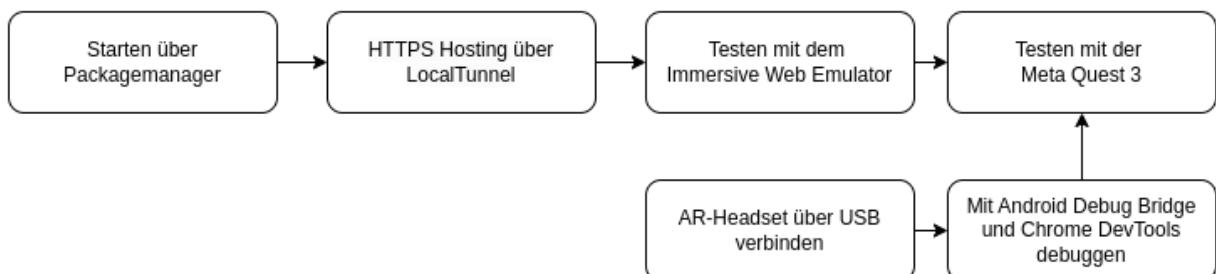


Abbildung 13: Ablaufdiagramm des normalen Entwicklungsablaufs

6.2 Implementierung der Interaktionskonzepte

Im Folgenden wird die Implementierung der zuvor beschriebenen Interaktionskonzepte erörtert. In diesem Kontext werden grundlegende Ideen sowie die für deren Implementierung relevanten Technologien vorgestellt und erläutert. Des Weiteren werden Screenshots der implementierten Konzepte gezeigt und die Funktionsweise der Interaktionen beschrieben. Schließlich werden auch die Herausforderungen und Probleme bei der Implementierung aufgezeigt und diskutiert.

Vor der Implementierung des Konzepts muss zunächst eine grundlegende WebXR-Anwendung erstellt werden, die die Interaktionen mit dem Controller ermöglicht. Als Basis hierfür dient das Skelett einer WebXR-Anwendung aus einer Artikelserie des Taikonauten-Magazins [26]. Die Anwendung nutzt bereits die vom Nutzer gescannten Umgebungen, um einen virtuellen Raum zu erstellen, in dem die Interaktionen stattfinden. Dabei werden Wände und Böden der Umgebung erkannt und als Mesh in die Szene eingefügt, um dem Nutzer eine Interaktion mit der realen Umgebung zu ermöglichen. Zudem wurden bereits einige Interaktionen des Controllers, wie das Auswählen von Objekten durch Raycasting, implementiert, die als Basis für die Implementierung der Interaktionskonzepte dienen.

Da im Rahmen der Implementierung auch auf spezifische Knopfbelegungen des Controllers eingegangen wird, ist in Abbildung 14 eine Darstellung eines rechten Meta Quest Touch Plus-Controllers mit den in dieser Arbeit genutzten Namen für die verschiedenen Bedienelemente für besseres Verständnis dargestellt.

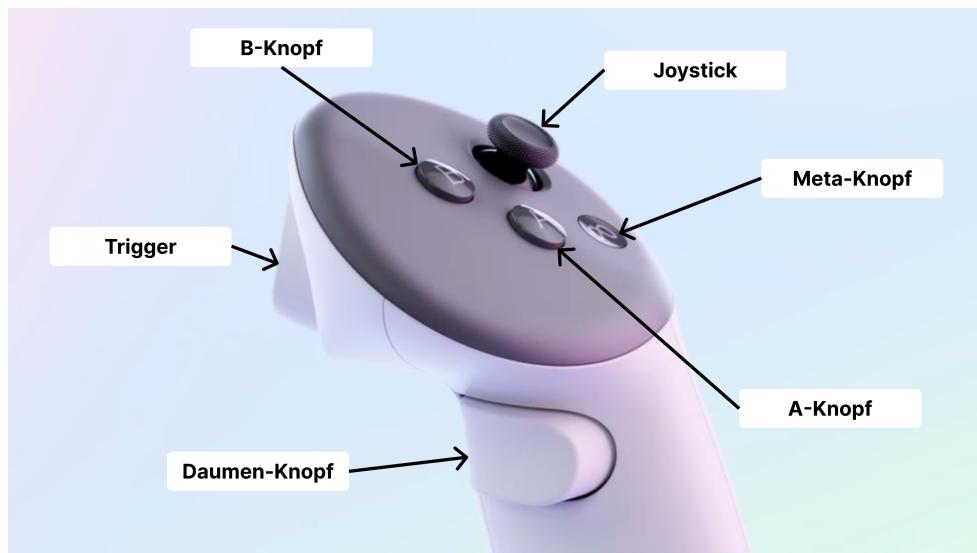


Abbildung 14: Schema der Bedienelemente des Quest Controllers mit den im Rahmen dieser Arbeit genutzten Namen

Quelle: [9]

6.2.1 Explodierende Bauteile

Der erste Schritt bei der Implementierung des Interaktionskonzepts der Explodierenden Bauteile ist die Erstellung eines 3D-Modells, welches die Bauteile des Produkts enthält. Jedes Bauteil muss dabei als eigenes Objekt im 3D-Modell vorhanden sein, damit es in der Animation separat dargestellt und referenziert werden kann. Für die erste Implementierung wird ein einfaches 3D-Modell eines Tetris-Blocks verwendet, welcher aus vier verschiedenenfarbigen Bauteilen besteht. Basierend auf der Implementierung des Tetris-Blocks soll dann das Konzept auf ein komplexeres Modell, wie beispielsweise ein Fahrzeug, übertragen werden.

Das Modell wurde zunächst in Blender erstellt und als 3D-Modell im glTF-Format, welches wie WebXR von der Khronos Group entwickelt wurde, in die Anwendung exportiert. Das glTF-Format ist ein offenes 3D-Dateiformat, welches für die effiziente Übertragung von 3D-Modellen im Web optimiert ist und die Dateigröße möglichst klein hält. Ein weiterer Vorteil des glTF-Formats besteht darin, dass es auch Animationen und Materialien unterstützt, die im 3D-Modell enthalten sind. So kann die Animation der Bauteile auch direkt in der Modellierungssoftware erstellt und in das glTF-Modell mit integriert bzw. eingebettet werden. Dies ist von essenzieller Bedeutung für die effektive technische Umsetzbarkeit des Konzepts, da die Animation der Bauteile auf diese Weise in speziellen Animationsprogrammen, wie beispielsweise Blender, erstellt werden kann, ohne dass hierfür die WebXR-Anwendung selbst erforderlich wäre. Dies ermöglicht die Erstellung komplexerer Animationen, die nur mit den Tools von Babylon.js selbst nicht oder nur mit einem zu hohen Aufwand realisierbar wären.

Der nächste Schritt ist das Platzieren des erstellten 3D-Modells in der WebXR-Umgebung. Das Prinzip des Raycastings wird verwendet, um dem Nutzer zu ermöglichen, mit dem Controller einen Punkt im Raum auszuwählen, an dem das 3D-Modell platziert werden soll. Der Raum, in dem sich der Anwender befindet, muss dafür vor Beginn einmalig in den Meta Quest Einstellungen als Umgebung eingescannt werden. Ist das geschehen, werden durch das Skelett der Anwendung des Taikonauten-Magazins bereits die Wände und Böden der Umgebung als Meshes erkannt und in die Szene eingefügt. Anschließend wird durch den Controller ein Strahl in die Szene geschossen, wobei bei Knopfdruck des Triggers der Punkt, an dem der Strahl ein Mesh trifft, als Event zurückgegeben wird.

An dieser Stelle erfolgt die Definition eines Ankerpunktes, welcher vom 3D-Modell als Referenzpunkt im AR-Raum genutzt wird. Der Ankerpunkt stellt ein Babylon.js-Objekt dar, welches in der Lage ist, einen Punkt im AR-Raum auch bei Bewegung des Nutzers am gleichen Ort zu halten. Bewegt sich der Ankerpunkt, so bewegt sich auch das 3D-Modell mit, sodass es stets an der gleichen Stelle im Raum bleibt.

Nachdem das 3D-Modell platziert wurde, kann die Animation des Produkts gestartet werden. Der A-Knopf des Controllers wird als Start-Button für die Animation genutzt, wobei die Wiedergabe vorwärts und rückwärts möglich ist. Für die Animation wird die aus Blender in das glTF-Modell eingebettete Animation verwendet. Dafür wird in Babylon.js eine

Animationsfunktion erstellt, welche die Steuerung der Animation des Modells über einige Parameter, wie beispielsweise den Start- und Endframe der Animation, ermöglicht. So kann für die Vorwärts- und Rückwärtsanimation die gleiche Funktion verwendet werden, indem die Start- und Endframe-Parameter basierend auf einem globalen Boolean, welcher den Animationsstatus speichert, gesetzt werden.

Beim Abspielen der Animation wird jedes Bauteil des Produkts in einer Schleife durchlaufen und dessen Animation gestartet. Diese Schleife wird dann auch genutzt, um den einzelnen Bauteilen ihre Requirements als Text-UI-Elemente zuzuweisen. Innerhalb der Schleife erfolgt zunächst eine Überprüfung, ob zu der jeweiligen ID des Bauteils Requirements existieren, die dargestellt werden sollen. Sofern dies der Fall ist, wird in Babylon.js eine Fläche erstellt, auf welcher der Text dargestellt wird, und diese Fläche an das Bauteil als Kindobjekt angehängt. Analog zum gesamten Modell, welches den Ankerpunkt als Referenzpunkt nutzt, nutzt jedes Requirement-Panel die Position seines Bauteils als Referenzpunkt für seine Position im AR-Raum. Die Fläche kann zudem als Knopf fungieren, sodass bei Betätigung beispielsweise das Bauteil vergrößert oder eine Detailansicht des Bauteils aufgerufen werden kann. Diese UI-Elemente werden jedoch, wie in Abbildung 15 zu sehen ist, nur angezeigt, wenn das Produkt gerade „explodiert“ ist und nicht, wenn das Produkt sich gerade im Normalzustand befindet.

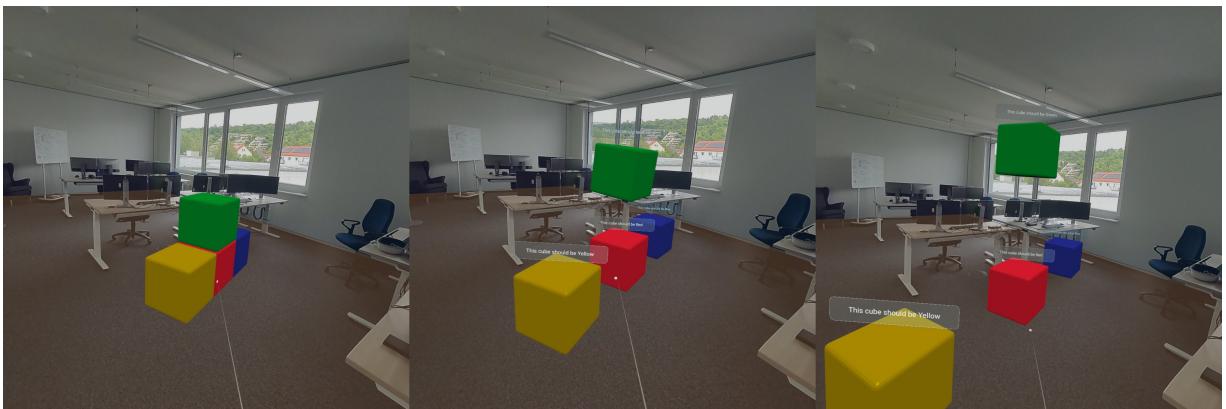


Abbildung 15: Screenshots des explodierenden Tetris-Blocks mit Requirements in AR

Implementierung an einem komplexeren Modell

Nachdem das Interaktionskonzept funktionell am einfachen Modell eines Tetris-Blocks implementiert wurde, ist die nächste Herausforderung die Implementierung an einem komplexeren Modell. Dafür wird ein relativ detailliertes 3D-Modell eines Porsche 911 von der 3D-Asset-Website Sketchfab verwendet, welches von dem Nutzer Abdul Azim Sharif erstellt und unter der CC BY 4.0 Lizenz veröffentlicht wurde [28].

Das Modell, welches in Abbildung 16 dargestellt ist, wurde ausgewählt, da es eine Vielzahl an unterschiedlichen Bauteilen beinhaltet, die in der Animation separat dargestellt werden können.



Abbildung 16: Screenshot des in der Anwendung genutzten Porsche Modells

Quelle: [28]

Bei dem in dieser Bachelorarbeit verwendeten Modell ist jedoch zu berücksichtigen, dass es sich nicht um ein vollständig akkurate und funktionales Modell eines Fahrzeugs handelt. Beispielsweise existieren keine Achsen, an denen die Räder hängen. Für die Zwecke dieser Bachelorarbeit ist das 3D-Modell jedoch ausreichend, da es eine hinreichende Anzahl an „realen“ Bauteilen enthält, um das Interaktionskonzept anschaulich zu demonstrieren. Würde das Interaktionskonzept in einer echten Anwendung für Kunden verwendet werden, sollte jedoch mit möglichst akkuren CAD-Modellen gearbeitet werden, um das Produkt möglichst genau darzustellen.

Für die Verwendung des 3D-Modells in der Anwendung muss als Nächstes eine Animation mit dem Modell erstellt werden, welche die Bauteile des Fahrzeugs in ihre Einzelteile zerlegt. Dafür wird in Blender eine Animation erstellt, welche einige Bauteile, wie beispielsweise die Verkleidung, die Räder und den Spoiler, nach außen wegbewegt. Beim Erstellen der Animation musste zudem beachtet werden, dass sich möglichst wenig Bauteile bei der Animation überschneiden. Infolgedessen haben einige Bauteile, wie beispielsweise die Räder, welche nur nach außen verschoben werden, eine sehr einfache Animation mit lediglich zwei Keyframes. Andere Bauteile, wie bspw. der Spoiler, hingegen erfordern eine komplexere Animation mit bis zu fünf Keyframes, um Überschneidungen zu vermeiden.

Zur Veranschaulichung der Animation ist in Abbildung 17 eine Bildsequenz der Animation des Porsche-Modells zu sehen.



Abbildung 17: Bildsequenz der Animation des Porsche Modells

Um den Bauteilen später ihre Requirements zuzuweisen, ist es erforderlich die relevanten Bauteile identifizierbar zu machen. Zu diesem Zweck wird in Blender jedem animierten Objekt zu Beginn des Objektnamens eine ID zugewiesen, beispielsweise „#1_Rad“ oder „#2_Lichter“. Diese ID dient im späteren Verlauf als Referenz für die Requirements.

Die IDs werden dabei zusätzlich in anzeigenende und nicht-anzeigenende IDs unterteilt, um bei Bauteilen welche aus mehreren kleinen Objekten bestehen in der großen Ansicht nur die Requirements des Hauptobjekts anzuzeigen. Beispielsweise besteht ein Rad aus Reifen, Felge, Bremsscheibe etc., wobei dann nur der Reifen als Hauptobjekt Requirements an das gesamte Rad anzeigt, während die Requirements der kleineren Einzelteile nur in der Detailansicht angezeigt werden. Um eine einfache Zuordnung der Objekte zu ihren Elternobjekten zu ermöglichen, beispielsweise für die Klick-Abfrage des Rads, werden die IDs der Elternobjekte in den IDs der Kindobjekte gespeichert. Dabei wird jedoch auf die Verwendung eines Unterstrichs „_-“ hinter der ID verzichtet, wodurch sie beim Erstellen der Requirementpanels herausgefiltert und ignoriert werden können.

Die detaillierte Radansicht stellt ein separates, ebenfalls in Blender erstelltes glTF-Modell dar, welches lediglich das Rad, seine Bauteile sowie die darin eingebetteten Animationen umfasst. Im Rahmen der Implementierung der detaillierten Radansicht wird eine Abfrage hinzugefügt, welche prüft, ob der Nutzer mit dem Trigger seines Controllers auf ein Rad klickt. Die Bestimmung des Punktes, an dem der Controller auf ein Objekt trifft, erfolgt wiederum gemäß dem Raycasting-Prinzip. Sofern das getroffene Objekt Teil des Rads ist, also die ID des Rads enthält, wird an der betreffenden Stelle ein neuer Ankerpunkt definiert, an dem die Detailansicht des Rads angezeigt wird. In der Folge werden der bisherige Ankerpunkt sowie die Bauteile des Fahrzeugs ausgeblendet, während die Bauteile des Rads an den neuen Ankerpunkt angehängt werden. Bei einem erneuten Wechsel zur Gesamtansicht des Fahrzeugs wird das Fahrzeugmodell wieder an den Ankerpunkt angehängt, während die Detailansicht des Rads ausgeblendet wird. Auf diese Weise kann der Nutzer die Requirements an das Rad in einer Detailansicht betrachten und bei Bedarf wieder zur Gesamtansicht des Fahrzeugs zurückkehren.

6.2.2 Wolken von Requirements

Das Interaktionskonzept der Wolken aus Requirementpanels ist im Vergleich zum Konzept der explodierenden Bauteile deutlich einfacher zu implementieren, da kein zugehöriges Modell oder eine individuelle Animation benötigt wird.

Im Rahmen der Implementierung wurde zunächst ein einfacher Algorithmus entwickelt, der einen Array von Requirements erhält und zu diesen eine 3D-Wolke von Punkten in einem gegebenen Bereich erstellt. Der ursprüngliche Algorithmus generiert, wie im Ablaufdiagramm in Abbildung 18 dargestellt, einen etwas zu großen Array aus 3D-Punkten in Form eines Würfels. In der Folge wird aus der Mitte des Punktarrays ein Teilarray in der Länge des Requirementarrays genutzt, um UI-Panels mit den Requirements an den jeweiligen Punkten zu platzieren. Dafür werden die Requirements eines Clusters in einer Schleife durchlaufen, wobei für jedes Requirement ein UI-Panel mit dem Text des Requirements an der Stelle des zugehörigen Punktes des Punktarrays erstellt wird.

Die Skalierung der Abstände der Punkte wurde durch das Testen verschiedener Abstände mit einfachen Beispielpanels bestimmt und kann auch über den Algorithmus selbst angegeben werden. Auf diese Weise sollte das Clipping, also das Kollidieren von nebeneinanderliegenden Panels, verhindert werden.



Abbildung 18: Funktionsweise des Punktewolken Algorithmus

Wie beim Konzept der explodierenden Bauteile wird auch hier ein Ursprungspunkt der Wolken durch Raycasting und Auslösen des Triggers des Controllers bestimmt. Die Wolken sollen dann über dem Punkt, der auf dem Boden ausgewählt wurde, schweben. Dafür werden die Textpanels beim Erstellen etwas nach oben verschoben.

Eine erste Betrachtung der Requirements in Wolken zeigte jedoch, dass sich die Textpanels bereits in kleinen Wolken stark überlappen und viele Requirements so nur schwer bis gar nicht lesbar sind. Wie in Abbildung 19 zu sehen ist, treten bereits bei Wolken von fünf Requirements starke Überlappungen auf, welche die Lesbarkeit der Requirements stark beeinträchtigen.

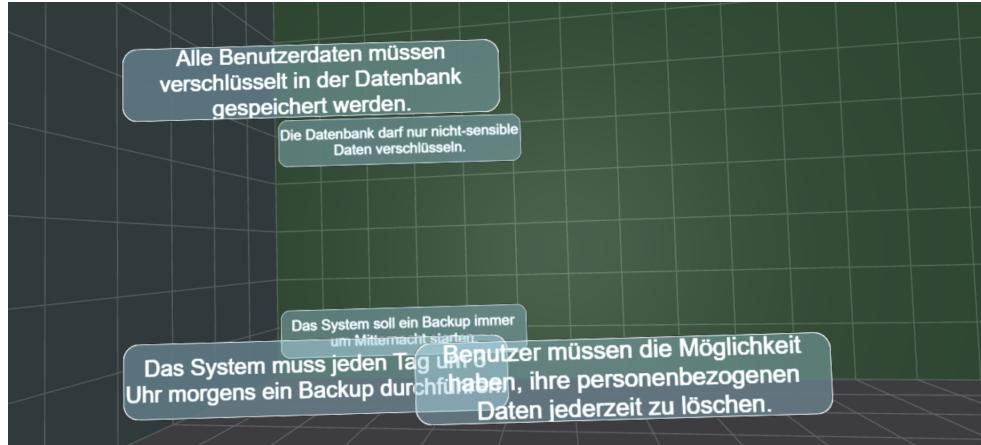


Abbildung 19: Wolke aus 5 Requirementpanels

In Abbildung 19 wird auch ersichtlich, dass die Transparenz der UI-Elemente dazu führt, dass Überlappungen auch die vordersten Requirements unlesbar machen. Daher war die erste Änderung, die UI-Elemente undurchsichtig zu machen, um zumindest die Lesbarkeit der vordersten Elemente zu gewährleisten. Des Weiteren wurde der Abstand der Requirement-Panels zueinander vergrößert und die Requirements wurden mittels einer Zufallsfunktion jeweils leicht von ihrem Ursprungspunkt verschoben, um die Überlappungen zu reduzieren. Der Algorithmus, der die Punktewolke generiert, wurde zusätzlich noch so wie in Abbildung 20 dargestellt angepasst, damit er keine würfelförmigen Punktewolken mehr erstellt, sondern diese Würfelform etwas abflacht, um Wolken in Form eines Quaders zu erhalten. Bei kleinen Requirementwolken resultieren die vorgenommenen Anpassungen in einer verbesserten Lesbarkeit der Requirements, wobei die meiste Zeit nahezu alle Requirements lesbar sind.



Abbildung 20: Funktionsweise des angepassten Punktewolken Algorithmus für flachere Wolken

Allerdings lässt sich bereits bei einer geringfügig größeren Requirementwolke, wie sie in Abbildung 21 dargestellt ist, erkennen, dass sich Überlappungen bei realistischen Wolkengrößen nicht gänzlich vermeiden lassen.

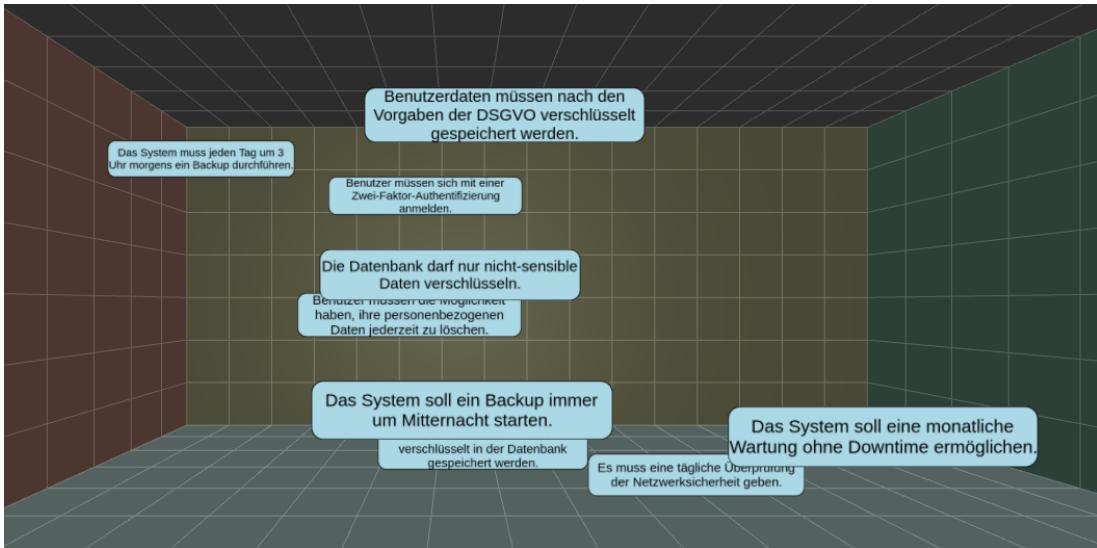


Abbildung 21: Wolke aus 9 Requirements mit undurchsichtigen Panels und größerem Abstand

Insofern lässt sich festhalten, dass das Konzept trotz initialer Anpassungen noch immer mit einer signifikanten Beeinträchtigung der Lesbarkeit und damit der Usability der Darstellung zu kämpfen hat. Das Interaktionskonzept der Wolken war insbesondere für die Betrachtung umfangreicher Mengen an Requirements konzipiert, weshalb die Usability der Darstellung auch bei großen Mengen von Requirements gewährleistet sein muss. Um die Lesbarkeit der Requirements auch in großen Wolken zu gewährleisten, wurden drei alternative Ansätze evaluiert, die im Folgenden erörtert werden.

Der erste Ansatz zielte darauf ab, die Requirements mithilfe des Raycasting des Controllers zu fokussieren. Auf diese Weise könnten Requirements, die gelesen werden sollen, in den Fokus gebracht und somit lesbar gemacht werden. Jedoch wird bei diesem Ansatz nicht der Überblick über alle Requirements gewährleistet, da nur ein Requirement gleichzeitig fokussiert werden kann. Zudem kann es bei großen Requirementwolken sein, dass gesuchte Requirements gar nicht gefunden und daher auch nicht fokussiert werden können, da sie durch andere Requirements komplett verdeckt sind. Bei Berücksichtigung der Tatsache, dass dieses Interaktionskonzept insbesondere für die gleichzeitige Darstellung einer Vielzahl von Requirements konzipiert wurde, erweist sich dieser Lösungsansatz als nicht geeignet.

Der zweite Ansatz zielte auf eine Verkleinerung der Requirementpanels ab, um eine größere Anzahl von Requirements in einer Wolke darstellen zu können. Allerdings wird bei dieser Vorgehensweise die Lesbarkeit der Requirements zum Problem. Aufgrund des in Kapitel

3.3.1 beschriebenen Screen-Door-Effekts und anderer Probleme bei der Schärfe von AR-Displays ist es schwierig, kleine Texte in AR gut lesbar darzustellen. Dieses Problem der Unlesbarkeit wird bei großen Requirementwolken, in denen viele Requirements dargestellt werden, noch verstärkt, da diese auch einen größeren Raum abdecken müssen und so einige Requirements weit vom Nutzer entfernt sind. Diese Probleme der Unschärfe durch Nähe zum Bildschirm treten jedoch bei modernen, konventionellen zweidimensionalen Bildschirmen aufgrund des Betrachtungsabstands nicht so stark auf. Dadurch ist eine Darstellung mit einer deutlich höheren Informationsdichte möglich. Folglich erweist sich auch dieser Lösungsansatz als ungeeignet, um eine Vielzahl von Requirements gleichzeitig in einer geeigneten Form darzustellen.

Der dritte Ansatz bestand in der weiteren Entzerrung der Requirements, wobei die Requirements möglichst breitflächig ausgerichtet wurden, um Überlappungen zu vermeiden. In der Folge wurde der Algorithmus dahingehend modifiziert, dass flachere, dafür aber breitere und höhere Punktewolken generiert werden. Umso mehr Requirements dabei existieren, desto flacher sollte die Wolke werden um alle Requirementpanels gleichzeitig sichtbar zu machen. Dieser Ansatz führt jedoch, wie in Abbildung 22 dargestellt, in seiner letztem Konsequenz bei großen Wolken zu einer Darstellung aller Requirements auf einer zweidimensionalen Ebene. Allerdings ist das Navigieren über große Flächen in AR unpraktisch im Vergleich mit einer ähnlichen Darstellung auf einem normalen Bildschirm. Daher wird durch diesen Lösungsansatz die Darstellung der Requirements im dreidimensionalen Raum und in AR weniger effizient als in einer 2D-Darstellung. Wenn die Wolken auch in AR in eine zweidimensionale Form gebracht werden, kann auch direkt eine zweidimensionale Darstellungsform gewählt werden.

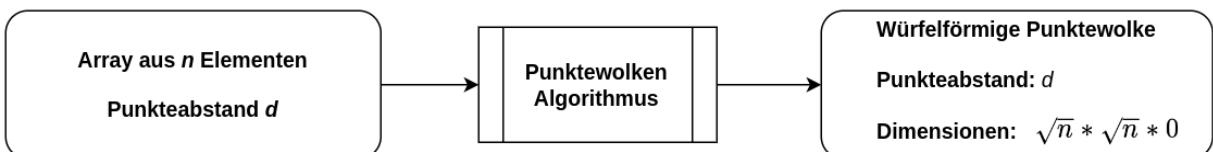


Abbildung 22: Extremversion eines abgeflachten Algorithmus zur Erstellung einer Punktewolke

Die Analyse der vorliegenden Ansätze zeigt, dass keiner der Ansätze eine Lösung für das Problem der Überlappung und Lesbarkeit der Requirements bietet, welches das Konzept nicht weniger sinnvoll macht als eine 2D-Darstellung. Daher wird das Konzept der Wolken von Requirements in AR nicht weiter verfolgt.

Im folgenden Kapitel 6.2.3 wird zur Untersuchung des Konzepts in 2D ein Mockup erstellt und analysiert. Die darauf basierende Möglichkeit der Implementierung des Konzepts als 2D-Anwendung wird zudem noch im Ausblick in Kapitel 7.3 erörtert.

6.2.3 2D-Mockup für Wolken aus Requirements

Zur Veranschaulichung der Darstellung des Interaktionskonzepts der Requirementwolken wurden im UI-Prototyping-Tool Figma Mockups erstellt, welche die Wolken in einer Desktop-Anwendung visualisieren. Im Folgenden werden anhand einiger Screenshots der Mockups die Unterschiede zur Darstellung in AR erörtert. Der Fokus liegt dabei auf dem Vergleich der Usability der beiden Darstellungsformen.

Im ersten Screenshot aus Abbildung 23 ist eine Übersicht über eine große Anzahl an Requirements dargestellt. Da auch auf einem herkömmlichen Display lediglich eine begrenzte Anzahl von Requirements gleichzeitig dargestellt werden kann, werden die Requirements in der Übersicht abstrakt vereinfacht als Punkte ohne Textinhalt präsentiert. Jeder Punkt stellt dabei ein Requirement dar, die durch den Nutzer gelesen werden kann, sobald dieser auf das entsprechende Element hereinzoomt.

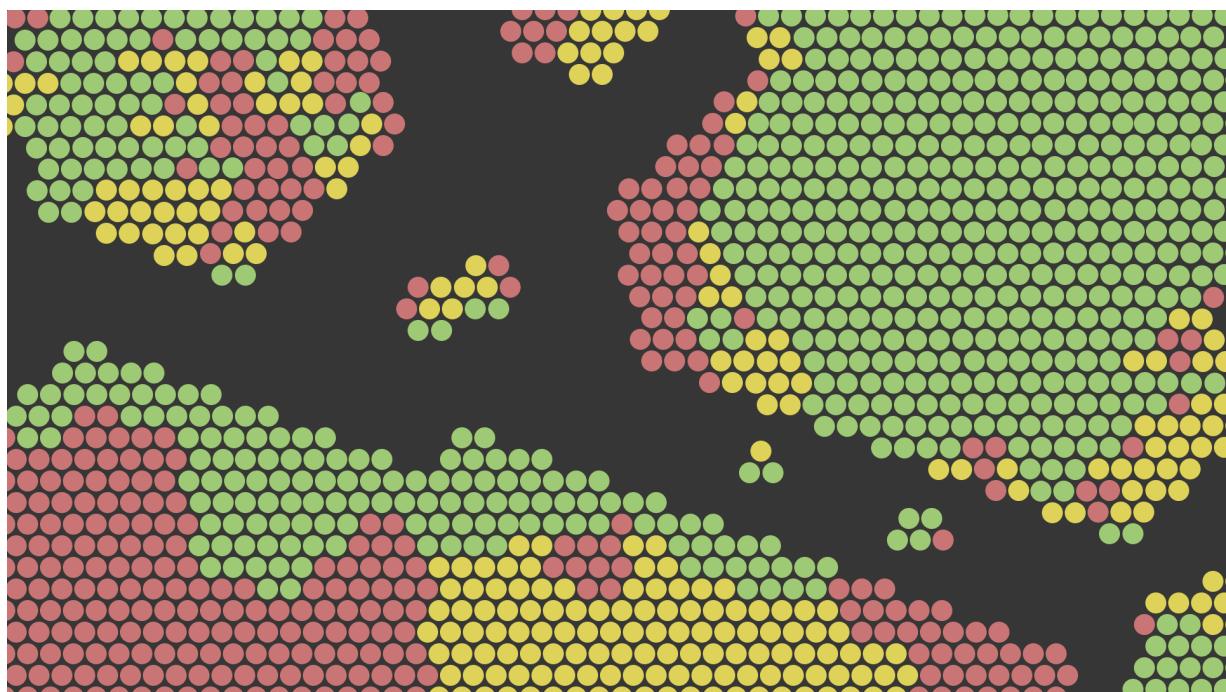


Abbildung 23: Mockup zur Übersicht über eine große Anzahl an Requirements

Die Farben der Requirements dienen der visuellen Darstellung des jeweiligen Bearbeitungsstatus. Dadurch wird eine unmittelbare Einordnung der Requirements ermöglicht.

- **Grüne Requirements:** Die Requirements dieser Farbe wurden bereits vollständig bearbeitet und überprüft, sodass eine weitere Bearbeitung aktuell nicht erforderlich ist.
- **Rote Requirements:** Rote Requirements wurden noch nicht bearbeitet, sie sind meist neu oder wurden lang ignoriert.
- **Gelbe Requirements:** Requirements in gelber Farbe wurden bereits einmal bearbeitet, bedürfen jedoch noch einer Überprüfung. Alternativ könnten in dieser Farbe alte Requirements mit neuen Änderungen dargestellt werden.

Es wäre auch denkbar, die Farben auf andere Filter anzupassen. Beispielsweise könnten Requirements mit der gleichen Farbe ein gleiches Bauteil betreffen. Die Implementierung farbbasierter Filter in der Anwendung könnte zu einer weiteren Steigerung der Übersichtlichkeit der Requirements beitragen.

Die dargestellte Farbenkorrelation erlaubt eine einfache Erkennung von Bereichen, in denen noch Verwaltungsarbeit zu erledigen ist. Zwar können offensichtlich nicht alle Requirements gleichzeitig mit ihrem Text betrachtet werden, doch durch die Vereinfachung der einzelnen Requirements als kleine Kreise ohne Text lässt sich ein gutes Gesamtbild erlangen.

Will man dann einzelne Requirements betrachten und mit diesen interagieren, kann in die Ansicht nahtlos hereingezoomt werden, um ab einer gewissen Größe die Requirement-Texte lesen zu können. Wollen Nutzer auch nach spezifischen Requirements direkt suchen, ließe sich auch eine Suchfunktion einbauen, mit welcher direkt nach einzelnen Requirements bspw. anhand ihres Textes gesucht werden kann. Die Größe, ab der die Texte als lesbar empfunden werden, ist dabei von den genutzten Bildschirmen sowie einigen weiteren nutzerspezifischen Eigenschaften abhängig. Sie kann jedoch flexibel und intuitiv durch das Zoomen durch den Nutzer selbst angepasst werden.

In Abbildung 24 ist ein Mockup einer hereingezoomten Perspektive dargestellt. Hier können dann alle Texte der Requirements selbst gelesen werden und es können einzelne Requirements angeklickt werden, um weiter mit ihnen zu interagieren. Denkbar wären auch Seitenmenüs, die sich beim Anklicken eines Requirements öffnen und detaillierte Informationen sowie eventuelle Interaktionsmöglichkeiten zu den Requirements anzeigen.



Abbildung 24: Mockup der hereingezoomten Ansicht

Bei der Erstellung eines Mockups des Wolken-Interaktionskonzepts in 2D wurde ersichtlich, dass das Konzept zumindest bei erster Betrachtung keine nennenswerten Nachteile durch den Wechsel auf eine zweidimensionale Darstellung erfährt. Tatsächlich lassen sich, wie in Abbildung 23 gezeigt, durch die höhere mögliche Informationsdichte in der zweidimensionalen Darstellung sogar größere Wolken aus Requirements darstellen als in AR. Obwohl durch die Abstraktion der Requirements als Punkte auch in AR eine deutlich größere Anzahl von Requirements dargestellt werden könnte, wäre dies nicht so effizient wie in 2D. Schon durch die höhere Schärfe der meisten konventionellen Displays durch den größeren Abstand des Nutzers im Vergleich zu HMDs können in einer klassischen, zweidimensionalen Anwendung deutlich kleinere Datenpunkte unterschieden werden.

In AR ist zu berücksichtigen, dass sehr kleine Elemente mit einem Controller schwieriger auszuwählen sind, als mit einer Maus. Dies liegt daran, dass eine Maus auf einer statischen Fläche abgelegt wird, während der Controller frei in der Luft gehalten werden muss. Aufgrund der genannten Eigenschaften ist sogar zu erwarten, dass das Interaktionskonzept in 2D eine Verbesserung der Usability ermöglicht, insbesondere im Hinblick auf das Zielen mit einer Maus auf kleine Elemente.

7 Zusammenfassung

Im Rahmen dieses Kapitels erfolgt zunächst ein Rückblick auf die im Rahmen der Arbeit geleisteten Tätigkeiten sowie eine Zusammenfassung der erzielten Ergebnisse. Des Weiteren erfolgt eine Evaluierung der Ergebnisse, wobei ein Fazit über die Effektivität und Usability der entwickelten Konzepte gezogen wird. In diesem Kontext werden ebenfalls Herausforderungen und Schwierigkeiten thematisiert, die bei der Umsetzung der Konzepte entstanden sind. Abschließend erfolgt ein Ausblick auf potenzielle Weiterentwicklungen der Konzepte.

7.1 Ergebnisse

Im Rahmen der Bachelorarbeit wurden zwei Interaktionskonzepte für die Visualisierung von Requirements in Augmented-Reality entwickelt und in jeweils einem Prototyp umgesetzt. Bei den Prototypen handelt es sich jeweils um eine WebXR-Anwendung, die mit dem Framework Babylon.js in TypeScript entwickelt wurde. Die Anwendung kann direkt von einem AR-fähigen Browser aufgerufen werden und ist somit plattformunabhängig.

Im ersten Konzept erfolgt die Visualisierung von Requirements in Kombination mit einem möglichst akkuraten 3D-Modell des zu entwickelnden Produkts. Das 3D-Modell wird vom Nutzer auf einer freien Fläche im Raum platziert und kann in einer Explosionsanimation, welche in Abbildung 25 anhand einer Bildsequenz dargestellt ist, in seine Bauteile zerlegt werden. Im Rahmen der Zerlegung des Modells werden die Requirements als Panels an den einzelnen Bauteilen visualisiert. Der Nutzer hat anschließend die Möglichkeit, die Animation wieder rückwärts abzuspielen, um das Modell wieder zusammenzusetzen. In der Konsequenz werden die Requirements wieder unsichtbar, sodass das Produkt in seinem originalen Gesamtzustand ohne Requirements betrachtet werden kann. Im Rahmen der Entwicklung des Prototyps wurde ein 3D-Modell eines Fahrzeugs verwendet, welches in einer Animation in seine Einzelteile zerlegt wird. Zur Veranschaulichung der geplanten Detailansicht besteht im Prototyp die Möglichkeit, die Räder des Modells anzuklicken, um eine Detailansicht der Requirements an die Räder zu erhalten. Das Interaktionskonzept offenbart ein beachtliches Potenzial, um eine Vielzahl von Interaktionen mit dem 3D-Modell und den Requirements zu ermöglichen. Ein entscheidender Vorteil der AR-Visualisierung ist die Möglichkeit, das 3D-Modell in Originalgröße darzustellen. Dadurch wird eine realistische Darstellung des Produkts gewährleistet.



Abbildung 25: Bildsequenz aus dem Prototyp der Explodierenden Bauteile in AR

Das zweite Konzept basiert auf der Visualisierung von Requirements in Form von Textpanels, die in der AR-Umgebung schweben. Die Wolken stellen Cluster aus Requirements dar, die thematisch zusammengehören. Des Weiteren besteht die Möglichkeit, die Requirements in den Wolken anzuklicken, um zugehörige Requirements anzuzeigen. Da die Requirements lediglich als Textpanels dargestellt werden, ist die Umsetzung dieses Konzepts deutlich einfacher automatisiert realisierbar, da weder Animationen noch 3D-Modelle erforderlich sind.

Allerdings ist auch der Mehrwert für den Nutzer in diesem Konzept geringer, da es weniger Möglichkeiten zur Interaktion in Augmented-Reality gibt als im ersten Konzept. Zudem ergeben sich bei der dreidimensionalen Darstellung der Requirementwolken Schwierigkeiten, die in einer zweidimensionalen Darstellung nicht auftreten würden. Beispielsweise werden Requirements oft hintereinander angezeigt, was dazu führt, dass der Nutzer die Requirements nur schwer oder gar nicht mehr lesen kann. Auch können nicht sehr viele Requirements in einer Wolke dargestellt werden, da die Requirements sonst zu klein werden und nicht mehr lesbar sind. In Anbetracht der limitierten AR-spezifischen Interaktionsmöglichkeiten sowie der Herausforderungen bei der Darstellung des Konzepts im dreidimensionalen Raum ist zu hinterfragen, ob das Konzept in AR einen Mehrwert gegenüber einer 2D-Darstellung eines ähnlichen Konzepts bietet.

7.2 Fazit

Die vorliegende Untersuchung belegt, dass die Visualisierung von Requirements in einer Augmented-Reality-Umgebung zumindest im Rahmen eines Prototyps realisierbar ist. Das erste Konzept demonstriert zudem, dass durch die Visualisierung in AR mittels diverser neuer Interaktionsmöglichkeiten ein Mehrwert in der Visualisierung von Requirements geschaffen werden kann.

In AR können, wie vor allem am Konzept der explodierten 3D-Modelle gezeigt wurde, viele neue Interaktionsmöglichkeiten geschaffen werden, die in einer 2D-Darstellung nicht möglich wären. Physische Produkte und ihre Bauteile können in Originalgröße dargestellt und interaktiv mit ihren Requirements verknüpft werden. So entsteht ein neuer Ansatz, um Requirements besser zu verstehen, zu kommunizieren und zu visualisieren, welcher in anderen Darstellungsformen nur schwer nachzuahmen ist.

Die Untersuchung verdeutlicht jedoch auch, dass die Umsetzung der untersuchten Konzepte noch mit zahlreichen Herausforderungen verbunden ist. Visuell beeindruckende Darstellungen erfordern in der Regel eine aufwendige und somit kostspielige Implementierung.

Das erste Konzept beispielsweise erfordert die Erstellung und Instandhaltung eines aktuellen 3D-Modells, inklusive Animation des Produkts, welches in der Anwendung dargestellt wird. Da dieses Konzept der explodierenden Bauteile, welches aber den meisten Mehrwert der Usability bietet, nur schwer automatisiert umzusetzen ist, ist die Implementierung in einer realen Anwendung zeitaufwändig und damit teuer.

Das Konzept der Requirementwolken hingegen ist zwar einfacher umzusetzen, bietet jedoch auch weniger Mehrwert für den Nutzer. Die Kosten-Nutzen-Effektivität beider Konzepte muss daher in weiteren Untersuchungen genauer betrachtet werden.

Der Vergleich des Konzepts der Requirementwolken in einer dreidimensionalen und zweidimensionalen Darstellung hat gezeigt, dass sich das Konzept der Requirementswolken mehr für eine konventionelle Darstellung in 2D eignet als für eine Implementierung in AR. Daher sollte das Konzept nicht weiter in AR verfolgt werden.

7.3 Ausblick

In weiteren Forschungsarbeiten könnte die praktische Umsetzbarkeit der beiden Konzepte in einer realen Anwendung weiter untersucht werden. Eine Untersuchung der professionellen Umsetzbarkeit des Konzepts der explodierenden Bauteile wäre dabei von besonderem Interesse, da dieses bereits im Prototypen den größten Mehrwert der Darstellung bietet und noch signifikant ausgebaut werden kann. Bei dieser Untersuchung sollte das Konzept vor allem auf Möglichkeiten zur Reduzierung des Aufwands der Implementierung analysiert werden, da die Wirtschaftlichkeit des Konzepts aufgrund der aufwendigen Implementierung noch fraglich ist.

Stellt sich das Konzept als wirtschaftlich sinnvoll heraus, bietet das Konzept vielfältige Möglichkeiten zur Erweiterung der Interaktionen. Im Folgenden werden einige mögliche Erweiterungen der Interaktionsmöglichkeiten aufgelistet:

- Herausgreifen von Bauteilen aus dem explodierten Modell zur genaueren Betrachtung
- Rotation ausgewählter Bauteile mit den Joysticks
- Manuelle Skalierung von ausgewählten Bauteilen zur genaueren Betrachtung

Die im Grundlagenkapitel 3.4 beschriebenen Konzepte der Gamification könnten ebenfalls in die Anwendung integriert werden, um die User Experience zu optimieren. Dafür könnten beispielsweise bereits betrachtete Bauteile oder Requirements ausgegraut werden, um dem Nutzer eine Übersicht über bereits betrachtete Requirements und ein Gefühl für den „Fortschritt“ in der Betrachtung zu geben.

Das Konzept der Requirementwolken sollte hingegen aufgrund der identifizierten Probleme in 3D als 2D-Darstellung weiter untersucht werden, da sich die Wolken von Requirements auch in einer zweidimensionalen Darstellung realisieren lassen. In einer 2D-Darstellung könnte das Interaktionskonzept von einer hohen Informationsdichte, wie im Mockup in Kapitel 6.2.3 gezeigt, und einer einfachen Navigation der Fläche an Requirements profitieren. Zudem kann es, wenn es als 2D-Darstellung umgesetzt wird, auch in anderen Anwendungen zur Visualisierung von Requirements eingesetzt werden. Beispielsweise könnten dafür Erweiterungen für Requirement-Management-Tools wie Jira oder Confluence entwickelt werden, um alternative Visualisierungen als Ergänzung zu den normalen Listen oder Tabellen von Requirements anzubieten. Dadurch könnte der Kosten-Nutzen-Faktor der Requirementwolken in einer 2D-Darstellung potenziell deutlich verbessert werden.

In Abbildung 26 sind die nächsten Schritte zur weiteren Untersuchung der beiden Interaktionskonzepte zusammenfassend dargestellt. Dafür wurden auch mögliche Ziele nächster Untersuchungen definiert, welche auf der hier vorliegenden Arbeit basieren könnten.

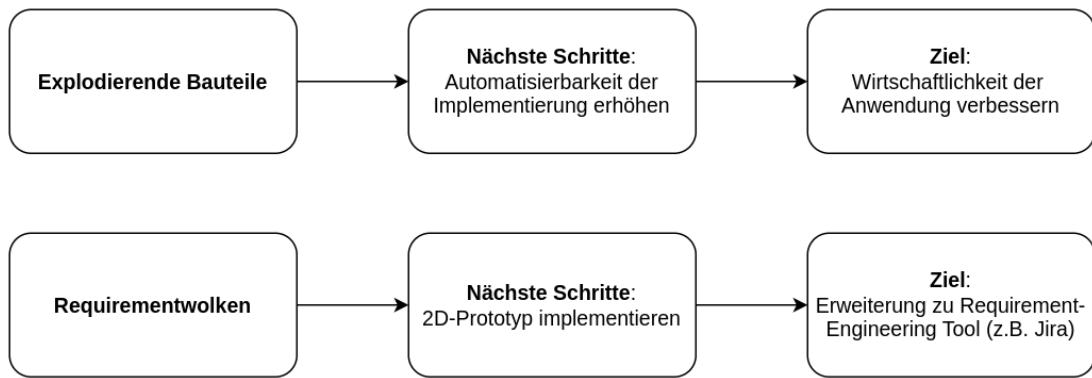


Abbildung 26: Ablaufdiagramm der nächsten Schritte zur weiteren Untersuchung der beiden Interaktionskonzepte

Literatur

1. INTERNATIONAL REQUIREMENTS ENGINEERING BOARD (IREB). *CPRE Glossary* [online]. 2024 [besucht am 2024-03-20]. Abgerufen unter: <https://www.ireb.org/de/cpre/glossary/>. Definition übersetzt von Englisch auf Deutsch.
2. RUPP, Christine; DIE SOPHISTEN. *Requirements-Engineering und -Management*. 7., aktualisierte und erweiterte Auflage. München: Carl Hanser Verlag GmbH & Co. KG, 2020. Abgerufen unter DOI: 10.3139/9783446464308.
3. HRUSCHKA, Peter. *Business Analysis und Requirements Engineering*. 3., updated edition. München: Carl Hanser Verlag GmbH & Co. KG, 2023. Abgerufen unter DOI: 10.3139/9783446478190.
4. DALTON, Jeremy; ACKER, Olaf. *Immersive Unternehmenswelten: Wie Augmented, Mixed und Virtual Reality die Wirtschaft transformieren*. Schäffer-Poeschel Stuttgart, 2023. Abgerufen unter DOI: 10.34156/978-3-7910-5689-0.
5. WÖLFEL, Matthias. *Immersive Virtuelle Realität*. Karlsruhe: Springer Vieweg Berlin, Heidelberg, 2023. Abgerufen unter DOI: 10.1007/978-3-662-66908-2.
6. MILGRAM, Paul; COLQUHOUN, Herman u. a. A taxonomy of real and virtual world display integration. *Mixed reality: Merging real and virtual worlds*. 1999, Jg. 1, Nr. 1999, S. 1–26.
7. CARMIGNIANI, Julie; FURHT, Borko; ANISETTI, Marco; CERAVOLO, Paolo; DAMIANI, Ernesto; IVKOVIC, Misa. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*. 2011, Jg. 51, Nr. 1, S. 341–377. ISSN 1573-7721. Abgerufen unter DOI: 10.1007/s11042-010-0660-6.
8. AZWEDO L.LC [online]. 2024 [besucht am 2024-06-30]. Abgerufen unter: <https://unsplash.com/de/fotos/ein-mann-der-auf-einem-stuhl-sitzt-und-eine-virtuelle-brille-tragt-AZ-ND5uJ4S4>. Bildquelle.
9. META PLATFORMS, INC. *Meta Quest 3 Homepage* [online]. 2024 [besucht am 2024-04-03]. Abgerufen unter: <https://www.meta.com/de/quest/quest-3/>.
10. APPLE INC. *iPhone 15 Pro - Technische Daten* [online]. 2023 [besucht am 2024-04-03]. Abgerufen unter: <https://support.apple.com/de-de/111829>.
11. GOURLAY, Michael J.; HELD, Robert T. Head-Mounted-Display Tracking for Augmented and Virtual Reality. *Information Display*. 2017, Jg. 33, Nr. 1, S. 6–10. Abgerufen unter DOI: <https://doi.org/10.1002/j.2637-496X.2017.tb00962.x>.

12. NIEMÖLLER, Christina; ZOBEL, Benedikt; BERKEMEIER, Lisa; METZGER, Dirk; WERNING, Sebastian; ADELMEYER, Thomas; ICKEROTT, Ingmar; THOMAS, Oliver. Sind Smart Glasses die Zukunft der Digitalisierung von Arbeitsprozessen? Explorative Fallstudien zukünftiger Einsatzszenarien in der Logistik. *13th International Conference on Wirtschaftsinformatik*. 2017.
13. LINDNER, Jannik. *Must-Know Pokemon Go Usage Statistics* [online]. 2023 [besucht am 2024-04-08]. Abgerufen unter: <https://gitnux.org/pokemon-go-usage-statistics/>.
14. BIMBER, Oliver; RASKAR, Ramesh. Modern approaches to augmented reality. In: *Acm siggraph 2006 courses*. 2006, 1–es.
15. MERCEDES-BENZ GROUP [online] [besucht am 2024-06-30]. Abgerufen unter: <https://group.mercedes-benz.com/innovation/produktinnovation/technologie/fahrsimulator.html>. Bildquelle.
16. HILLMANN, Cornel. *UX for XR: User Experience Design and Strategies for Immersive Technologies*. 1. Aufl. Berkeley, CA: Apress, 2021. Design Thinking. ISBN 978-1-4842-7019-6. Abgerufen unter DOI: 10.1007/978-1-4842-7020-2.
17. KHRONOS GROUP. *WebGL Overview* [online]. 2024 [besucht am 2024-04-08]. Abgerufen unter: <https://www.khronos.org/webgl/>.
18. KHRONOS GROUP. *About the Khronos Group* [online]. 2024 [besucht am 2024-04-08]. Abgerufen unter: <https://www.khronos.org/about>.
19. KHRONOS GROUP. *OpenGL ES Overview* [online]. 2024 [besucht am 2024-04-08]. Abgerufen unter: <https://www.khronos.org/opengles/>.
20. BARUAH, Rakesh. *AR and VR Using the WebXR API: Learn to Create Immersive Content with WebGL, Three.js, and A-Frame*. Entering VR Through WebXR. Berkeley, CA: Apress, 2021. ISBN 978-1-4842-6318-1. Abgerufen unter DOI: 10.1007/978-1-4842-6318-1_6.
21. WORLD WIDE WEB CONSORTIUM (W3C). *WebXR Device API* [online]. 2024 [besucht am 2024-04-02]. Abgerufen unter: <https://www.w3.org/TR/webxr/>.
22. THREE.JS. *three.js Documentation* [online]. 2024 [besucht am 2024-05-13]. Abgerufen unter: <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>.
23. A-FRAME. *A-Frame Introduction* [online]. 2024 [besucht am 2024-05-13]. Abgerufen unter: <https://aframe.io/docs/1.5.0/introduction/>.
24. BABYLON.JS. *Babylon.js Features* [online]. 2024 [besucht am 2024-04-03]. Abgerufen unter: <https://www.babylonjs.com/specifications/>.
25. RAWPIXEL.COM ON FREEPIX [online]. 2024 [besucht am 2024-06-30]. Abgerufen unter: <https://aframe.io/docs/1.5.0/introduction/>. Bildquelle.

26. TAIKONAUTEN. *WebXR with Babylon.js: Beginner's Guide* [online]. 2024 [besucht am 2024-03-15]. Abgerufen unter: <https://medium.com/taikonauten-magazine-english/welcome-to-the-exciting-world-of-webxr-and-babylon-js-e2dbd406dbcb>.
27. META PLATFORMS, INC. *Immersive Web Emulator Chrome Extension* [online]. 2024 [besucht am 2024-07-02]. Abgerufen unter: <https://chromewebstore.google.com/detail/immersive-web-emulator/cgffilbpcibhmcfbggfhhfolhkfbhmik>.
28. SHARIF, Abdul Azim. *Porsche 911 GT3* [<https://sketchfab.com/3d-models/porsche-911-gt3-very-high-quality-free-4eee314e142f4737872e08b0df0ff7f4>]. Published under the CC BY 4.0 license.