

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 Gruppe

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 1. Juli 2024

Unterschrift

Inhaltsverzeichnis

1 Kurzfassung	6
2 Einleitung	7
2.1 Motivation	7
2.2 Zielsetzung	8
3 Grundlagen	10
3.1 Requirements Engineering	10
3.2 Virtuelle Realität	12
3.3 Augmented Reality	13
3.3.1 Head-Mounted Displays	14
3.3.2 Hand-Held-Devices	17
3.3.3 Spatial Displays	19
3.4 Gamification von UX	20
4 Technologien	22
4.1 Meta Quest 3	22
4.2 WebGL	24
4.3 WebXR	24
4.3.1 Three.js	25
4.3.2 Babylon.js	26
5 Konzept	28
5.1 Explodierende Bauteile	28
5.2 Wolken von Anforderungen	30
6 Implementierung	32
6.1 Entwicklungsumgebung für WebXR und Meta Quest 3	32
6.2 Implementierung der Interaktionskonzepte	34
6.2.1 Explodierende Bauteile	34
6.2.2 Wolken von Anforderungen	38
6.2.3 2D-Mockup für Wolken aus Anforderungen	40
7 Zusammenfassung	43
7.1 Ergebnisse	43
7.2 Fazit	44

Inhaltsverzeichnis

7.3 Ausblick	45
------------------------	----

Abbildungsverzeichnis

1	Realitäts-Virtualitäts-Kontinuum nach Milgram	13
2	Meta Quest 3 VR- und AR-Headset	16
3	Screenshots aus dem AR-Spiel Pokémon Go	18
4	Innenraum des Mercedes-Benz Fahrsimulators	19
5	Frontal ausgerichtete Kameras und Tiefenprojektor der Oculus Quest 3 . .	23
6	Die beiden nach unten gerichteten Kameras der Oculus Quest 3	23
7	Application Flow von WebXR-Anwendungen	25
8	Szenen-Explorer und -Inspektor von Babylon.js	27
9	Beispiel einer Explosionsanimation anhand eines Rubiks Cube	28
10	Beispiel einer Punktewolke	30
11	Screenshots der Immersive Web Emulator Erweiterung in Chrome	33
12	Screenshots des explodierenden Tetris-Blocks mit Anforderungen in AR . .	36
13	Screenshot des in der Anwendung genutzten Porsche Modells	36
14	Bildsequenz der Animation des Porsche Modells	37
15	Wolke aus 5 Anforderungspanels	38
16	Wolke aus 9 Requirements mit undurchsichtigen Panels und größerem Abstand	39
17	Mockup zur Übersicht über eine große Anzahl an Requirements	41
18	Mockup der hereingezoomten Ansicht	42
19	Bildsequenz aus dem Prototyp der Explodierenden Bauteile in AR	44

1 Kurzfassung

Die Bachelorarbeit beschäftigt sich mit der Visualisierung von Requirements mithilfe von Augmented Reality.

Requirements sind dabei Anforderungen, die vor der Entwicklung eines Systems an dieses gestellt werden. Sie sind ein essentieller Bestandteil der Entwicklung von Produkten und Systemen, da sie die Basis für die Entwicklung bilden. Die Visualisierung von Requirements soll dabei helfen, im fortlaufenden Prozess des Requirements-Engineering den Überblick zu behalten und die Anforderungen besser zu verstehen. Dafür werden 2 verschiedene Interaktionskonzepte ausgearbeitet und in einem WebXR-Prototypen mit Babylon.js umgesetzt.

Das erste Konzept ist eine Visualisierung von Requirements in Kombination mit einem 3D-Modell des zu entwickelnden Produkts. Dabei wird das 3D-Modell, welches in dieser Bachelorarbeit ein Auto ist, in einer Augmented-Reality-Umgebung dargestellt. Dieses Modell kann in einer Animation in seine Einzelteile zerlegt werden, um dann die Anforderungen als Panels an den einzelnen Bauteilen zu visualisieren.

Das zweite Konzept ist eine Visualisierung von Requirements in Wolken aus Textpanels, die in der Augmented-Reality-Umgebung schweben. Die Wolken sind dabei Cluster von Anforderungen, die thematisch zusammengehören. Die Anforderungen sollen dann angeklickt werden können, um zugehörige Anforderungen anzuzeigen.

Im Rahmen der Bachelorarbeit wird ein Prototyp entwickelt, der die beiden Konzepte umsetzt.

2 Einleitung

2.1 Motivation

Die Anforderungsanalyse ist ein wichtiger Bestandteil der Softwareentwicklung und dient dazu, die Anforderungen an ein System zu erfassen und zu spezifizieren. Sie sind dazu da, um genau zu definieren, was von einem System erwünscht ist und welche Funktionalitäten es bieten soll. Sind die Anforderungen von Systemen nicht genau definiert, kann es zu Missverständnissen und Fehlinterpretationen in der Entwicklung und in schlimmen Fällen bis in das Endprodukt kommen. Da diese Fehler meist erst spät im Entwicklungsprozess erkannt werden, sind sie oft mit hohen Kosten verbunden. Daher wird in der Industrie viel Wert auf eine genaue und strukturierte Anforderungsanalyse gelegt. Ein höherer Aufwand in der Anforderungsanalyse kann dabei helfen, Fehler frühzeitig zu erkennen und zu vermeiden und so Kosten durch Fehler und Missverständnisse zu reduzieren.

Deshalb ist es wichtig, dass die Anforderungen klar und verständlich formuliert sind, um Missverständnisse und Fehlinterpretationen zu vermeiden. Einen Ansatz zur Verbesserung der Qualität der Anforderungen stellt reQlab dar, das bei der korrekten Formulierung und Strukturierung von Anforderungen unterstützt. reQlab untersucht die einzelnen Anforderungen auf ihre Qualität und gibt eine Bewertung ab, wie gut sie formuliert sind. Dadurch kann reQlab die Qualität der einzelnen Anforderungen stark verbessern und so die Qualität des gesamten Systems steigern.

Der durch reQlab verfolgte Ansatz zielt jedoch nur auf die Verbesserung eines Teils der Anforderungsanalyse. Ein weiterer wichtiger Aspekt ist die kontinuierliche Kommunikation der Anforderungen zwischen Entwicklern und Auftraggebern. Da, egal wie viel Aufwand in die Anforderungsanalyse gesteckt wird, es immer zu Missverständnissen und Fehlinterpretationen kommen kann, ist es wichtig, dass die Anforderungen regelmäßig überprüft und diskutiert werden. Dabei ist es wichtig, dass alle Beteiligten möglichst alle Anforderungen kennen, verstehen und nachvollziehen können. Jedoch ist es oft schwer, vor allem als Auftraggeber, der nicht täglich in den Entwicklungsprozess involviert ist, einen vollen Überblick über alle Anforderungen zu behalten und zu verstehen, was genau gefordert wird.

Gleichzeitig öffnen sich mit dem Fortschritt der Technik immer mehr Möglichkeiten, um Anforderungen zu visualisieren und zu präsentieren. Augmented Reality Technologien werden durch neue Endgeräte wie die Meta Quest 3 oder die Microsoft HoloLens immer zugänglicher und bieten neue Möglichkeiten, um Daten zu visualisieren und zu präsentieren.

Daher soll in dieser Bachelorarbeit untersucht werden, ob sich Anforderungen an Systeme in einer AR-Umgebung darstellen lassen und ob dadurch ein Mehrwert gegenüber herkömmlichen Darstellungsmethoden entstehen kann.

2.2 Zielsetzung

Im Laufe der Bachelorarbeit sollen verschiedene Interaktionskonzepte für die Anzeige von Anforderungen in einer AR-Umgebung untersucht und auf ihre Vor- und Nachteile, sowie auf ihre Eignung für den Einsatz in einem realen Projekt untersucht werden.

Dabei sollen möglichst mehrere Konzepte entwickelt und prototypisch umgesetzt werden, um diese anschließend zu evaluieren und zu vergleichen. Die Konzepte sollen dabei möglichst unterschiedliche Ansätze der Darstellung und Interaktion verfolgen, um so die Vor- und Nachteile der verschiedenen Ansätze zu untersuchen.

Die Prototypen sollen in der Lage sein die Interaktionskonzepte anhand von Beispielen darzustellen. Sie müssen dabei nicht jedes Feature der ausgearbeiteten Interaktionskonzepte umsetzen, sondern sollen vor allem die Interaktionsmöglichkeiten und den Mehrwert der Darstellung in AR gegenüber herkömmlichen Methoden zeigen. Daher soll jeder Prototyp mindestens 2 verschiedene Interaktionsmöglichkeiten mit den Anforderungen bieten, um einen hohen Grad an Interaktivität zu gewährleisten und gleichzeitig den Aufwand der Implementierung der Prototypen angemessen zu halten. Während der Entwicklung soll dabei untersucht werden, wie einfach die Konzepte technisch umsetzbar sind. Ein wichtiger Faktor bei der Umsetzbarkeit spielt dabei, ob die Umsetzung automatisiert werden könnte, um realistisch in einer realen Anwendung eingesetzt zu werden.

Der Fokus der Evaluation der Prototypen soll dann vor allem auf den folgenden Kriterien liegen:

- Usability: Wie einfach und intuitiv ist die Bedienung der Prototypen?
 - Mehrwert: Bieten die Prototypen einen Mehrwert der Usability?
 - AR: Ist die Darstellung in AR sinnvoll und bietet sie einen Mehrwert gegenüber herkömmlicher Methoden? 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?

2 Einleitung

Die beiden Kriterien und ihre Unterkriterien sollen dabei helfen, die Vor- und Nachteile der verschiedenen Konzepte zu identifizieren und zu bewerten. Anhand der Ergebnisse der Evaluation soll dann eine Einschätzung gegeben werden, ob sich die Konzepte für die Implementierung in einer realen Anwendung für reale Projekte eignen. Dabei soll der entstehende Mehrwert der Darstellung gegenüber herkömmlichen Methoden im Gegenzug zum Aufwand der Implementierung bewertet werden.

3 Grundlagen

Im folgenden Kapitel sollen konzeptionelle Grundlagen erläutert werden, welche für das Verständnis der Bachelorarbeit notwendig sind. Dabei wird auf die Themen Requirements Engineering, Augmented Reality und die Software reQlab eingegangen.

3.1 Requirements Engineering

Die Bachelorarbeit soll sogenannte Requirements, also Anforderungen, visualisieren. Daher ist es für das Verständnis der Arbeit wichtig, die Grundlagen des Requirements Engineering zu kennen.

Requirements

Grundlegend sind Requirements Anforderungen, die an ein System gestellt werden. Das International Requirements Engineering Board (IREEB) definiert sie in ihrem Glossar mit drei Eigenschaften [1, Def. Anforderung]

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

Sie sollen also die Bedürfnisse der Stakeholder an das System repräsentieren und dokumentieren.

Die Gestaltung von Requirements kann dabei je nach System und Anforderungen unterschiedlich sein. Chris Rupp nennt in ihrem Buch „Requirements-Engineering und -Management“ einige Beispiele für verschiedene Formen für Requirements [2, S. 19]:

- User-Stories
- Use-Cases
- Stories
- formalisierte natürlichsprachliche Anforderungen
- Anforderungen in Form von Diagrammen (semiformales Modell)

Natürlichsprachliche Anforderungen können sehr einfach selbst formuliert werden. Dadurch sind sie jedoch auch anfällig für Missverständnisse und Unklarheiten. Das Ziel von reQlab ist es, diese Missverständnisse und Unklarheiten in natürlichsprachlichen Anforderungen zu erkennen und so die Qualität der Anforderungen zu verbessern. Daher werden im Umfang dieser Bachelorarbeit nur natürlichsprachliche Anforderungen genutzt.

Zudem 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]. Nicht-funktionale Requirements hingegen beschreiben alle Requirements, die nicht funktionaler Natur sind, also beispielsweise Performance, Sicherheit oder Zuverlässigkeit. Der Begriff nicht-funktional ist dabei jedoch etwas irreführend, da auch nicht-funktionale Requirements gewissermaßen Funktionen des Systems beschreiben. Peter Hruschka beschreibt in seinem Buch Funktionale Anforderungen mit der Frage: „Was soll das System/Produkt tun?“. Auch unterteilt er nicht-funktionale Anforderungen 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 Anforderungen und könnte im User-Interface der Visualisierung genutzt werden, um die Anforderungen zu kategorisieren.

Stakeholder

Stakeholder können „Personen oder Organisationen sein, die die Anforderungen eines Systems beeinflussen oder die von dem System beeinflusst werden.“ [1]. Beispielsweise wären die Endnutzer eines Systems Stakeholder, welche durch das System beeinflusst werden. Sie haben also ein Bedürfnis an das System, können dieses jedoch nicht selbst umsetzen. Im Gegensatz dazu stehen die Auftraggeber, beziehungsweise der Produkteigner (Product Owner), welche das System entwickeln und die Anforderungen festlegen.

Viele Stakeholder, wie bspw. der Product Owner, sind dabei nicht direkt in den täglichen Entwicklungsprozess des Systems involviert und haben daher nur wenig Überblick über den aktuellen Stand des Systems. Sie nehmen durch die gestellten Anforderungen jedoch großen Einfluss auf das System. Für eine effiziente Zusammenarbeit zwischen Stakeholdern und Entwicklern ist es also wichtig, dass die Anforderungen sowohl den Stakeholdern als auch den Entwicklern klar und verständlich sind.

Das kann vor allem bei großen und komplexen Systemen schwierig sein, da die Zahl der Anforderungen mit der Komplexität des Systems stark wächst. Durch die große Menge an Anforderungen kann in solchen Projekten schnell die Übersicht verloren gehen, weshalb es wichtig ist, die Anforderungen klar und übersichtlich zu dokumentieren und zu verwalten.

System

Die IREB definiert ein System als „Eine kohärente, abgrenzbare Menge von Elementen, die durch koordiniertes Handeln einen bestimmten Zweck erfüllen.“ [1] Das Wort System ist dabei ein Überbegriff für Produkte, Services, Geräte, Prozeduren und Werkzeuge und kann sowohl physisch als auch virtuell sein. Daher wird auch in dieser Bachelorarbeit das Wort System als Überbegriff für alle Arten von Systemen genutzt.

Requirements Engineering

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

- Wissen vermitteln
- Gute Anforderungen herleiten
- Anforderungen vermitteln
- Anforderungen 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 verhindert so Fehler, welche sich durch die Systementwicklung ziehen und dann teuer und aufwendig behoben werden müssen, wenn sie erkannt sind. [2, S.20]

Diese Bachelorarbeit soll versuchen einen neuen Ansatz in der Vermittlung und Verwaltung von Anforderungen zu finden, um so langfristig die Qualität und Nützlichkeit der Anforderungen zu verbessern. Zudem soll die Visualisierung der Anforderungen helfen, bei großen Projekten, welche eine sehr große Zahl an Anforderungen besitzen, eine bessere Übersicht über die Anforderungen zu erhalten, um die Verwaltung der Anforderungen zu erleichtern. In der Vermittlung kann die Visualisierung den Stakeholdern eventuell helfen, die Anforderungen besser zu verstehen und so Missverständnisse und Unklarheiten zu vermeiden. So können teure Fehler in der Systementwicklung schon früher erkannt und vermieden werden.

3.2 Virtuelle Realität

Für das Verständnis von Augmented Reality ist es wichtig, die Begriffe der virtuellen Realität (VR) zu kennen und zu verstehen. Virtuelle Realität beschreibt eine Technologie, die es ermöglicht, eine virtuelle Welt zu erschaffen, in der der Nutzer interagieren kann. Dabei wird der Nutzer dann über verschiedenste audiovisuelle Technologien in diese virtuelle Welt versetzt, die durch Computer generiert wird. In virtueller Realität ist, im Gegensatz zu Augmented Reality, die gesamte Umgebung digital [vgl. 4, S.15].

Der Begriff der virtuellen Realität lässt sich noch genauer in die Begriffe immersive und nicht-immersive virtuelle Realität unterschieden. Folgende nutzerbezogene Eigenschaften sind dabei Indikatoren für nicht-immersive virtuelle Realität:

- 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 virtueller Realität alle äußeren Einflüsse so weit wie möglich reduziert und alle Indikatoren sollten auf immersive virtuelle Realität 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 vor allem um den visuellen Sinn, da dieser am meisten zur Immersion in die virtuelle Welt beiträgt. Jedoch wird in den meisten immersiven VR-Anwendungen 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 bspw. die Meta Quest Touch Plus-Controller, dem Nutzer auch haptisches Feedback für Interaktionen geben.

3.3 Augmented Reality

Augmented Reality (AR) ist eine Technologie, die die reale Welt mit digitalen Informationen erweitert. Dabei wird ein ähnlicher Ansatz wie bei Virtueller Realität verfolgt, jedoch wird die reale Welt nicht komplett ersetzt, sondern nur erweitert. Der Nutzer sieht 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 fällt Augmented Reality unter den Überbegriff der Mixed Reality, da die reale Welt mit der digitalen Welt gemischt wird. Daher ist bei AR die Immersion im Vergleich zu VR oft geringer und weniger wichtig, da der Nutzer immer die reale Welt als Referenzpunkt

hat. AR Technologien sind nicht darauf ausgelegt, den Nutzer in eine andere Welt zu versetzen, sondern die reale Welt zu erweitern.

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 Virtuelle Realität. Die Besonderheit von AR-Anzeigegeräten ist jedoch, dass sie die reale Welt mit digitalen Informationen erweitern. Das heißt sie müssen dem Nutzer auch eine Sicht auf die reale Welt ermöglichen und in diese Informationen einblenden. Beispielsweise kann das Display eines Anzeigegeräts transparent sein, sodass der Nutzer durch das Display hindurch sehen kann. Alternativ kann das Gerät eine oder mehrere Kameras besitzen, dessen Aufnahme auf undurchsichtige Bildschirme projiziert wird. Diese Methode nennt sich auch Image Passthrough. Es gibt verschiedene Arten von Anzeigegeräten, die für Augmented Reality genutzt werden können, wobei man allgemein zwischen 3 Hauptkategorien unterscheiden kann: Head-Mounted Displays, Hand-Held-Devices und Spatial Displays [7, S. 346]. Im Folgenden werden diese drei Kategorien genauer erläutert und einige Beispiele genannt.

3.3.1 Head-Mounted Displays

Head-Mounted Displays (HMDs) sind grundsätzlich Bildschirme, die direkt auf dem Kopf des Nutzers getragen werden. Durch ihre Nähe zu den Augen des Nutzers können sie ein großes Sichtfeld abdecken, ohne dabei selbst große Displays zu nutzen. Dadurch sind sie für volle Immersion im Normalfall kosteneffektiver als große Displays, wie bspw. eine Leinwand. Jedoch entsteht durch die Befestigung am Kopf auch automatisch ein höheres Gewicht am Kopf des Nutzers, was bei längerer Nutzung unangenehm werden kann.

VR Headsets

Klassische VR-Headsets kann man sich vorstellen wie eine Skibrille mit 2 Bildschirmen, die auf dem Kopf getragen wird und üblicherweise 2 Displays hinter 2 Linsen haben, bzw. ein Display virtuell in 2 Displays aufteilen. Durch die 2 Bildschirme wird für jedes Auge ein eigenes, leicht verschobenes Bild erzeugt, wodurch Inhalte in 3D dargestellt werden können. Außerdem können sie das volle Sichtfeld des Nutzers abdecken und so eine immersive Erfahrung schaffen.

Jedoch entstehen durch die Nähe des Nutzers zu den Displays auch visuelle technische Probleme, wie bspw. der Screen-Door-Effekt, bei welchem die Zwischenräume zwischen den Pixeln sichtbar sind. Entscheidend dabei ist die Kennzahl der Pixel pro Grad (Pixel

per Degree, PPD), welche angibt, wie viele Pixel auf einen Grad des Sichtfelds des Nutzers kommen. Eine Möglichkeit um die PPD zu erhöhen und so den Screen-Door-Effekt zu minimieren, ist das Einsetzen von Displays mit einer hohen Pixeldichte (Pixels per Inch, PPI). Beispielsweise hat die Oculus Quest 3 eine PPI von 1218 und erreicht damit einen PPD Wert von 25 Pixeln pro Grad [8]. Vergleichsweise hat das iPhone 15 Pro eine Pixeldichte von nur 460 PPI [9].

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 mit einer hohen Bildwiederholrate der Displays entgegengewirkt, um die Bewegungen des Nutzers möglichst flüssig darzustellen.

Die Position des Headsets muss kontinuierlich bestimmt werden, um die Bewegungen der Nutzer zu verfolgen und so die virtuelle Welt auf den Bildschirmen anpassen zu können. Dabei wird generell durch 2 Methoden unterschieden [10]:

- 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.

Outside-In-Tracking kann beispielsweise durch Kameras realisiert werden, welche im Raum verteilt werden und visuelle Marker am Headset und den Steuergeräten tracken. Diese Methode ist sehr genau in der Bestimmung der Position, jedoch ist der Nutzer dadurch an einen festen Raum gebunden, in dem die Trackingstationen aufgestellt sind und muss immer in deren Blickfeld sein. Zudem müssen die Kameras zuerst aufgestellt und kalibriert werden, was den Aufwand für die Installation erhöht. Eine Methode des Inside-Out-Trackings ist das Berechnen der Position durch Daten aus Kameras und 3D-Sensoren im Headset. Dabei wird die Umgebung des Nutzers aufgenommen und sich daraus ein 3D-Modell der Umgebung erstellt, um die Position des Headsets zu bestimmen. So wird kontinuierlich eine neue 3D-Repräsentation der Kamera- und Sensorendaten 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 zu verarbeiten.

Die Interaktion mit der Umgebung wird dann meist mit Controllern realisiert, deren Position und Rotation ebenfalls kontinuierlich durch Inside-Out- oder Outside-In-Tracking bestimmt werden muss. In AR-Headsets mit eigenen Kameras ist es auch möglich, die Hände des Nutzers zu tracken und als Eingabegeräte zu nutzen. Dabei können dann bestimmte Gesten oder Handbewegungen als Eingaben interpretiert und zur Interaktion genutzt werden. Beispielsweise interpretiert die Oculus Quest 3 ein Zusammenführen des Zeigefingers und Daumens als Klick. Berühren sich die Finger jedoch nicht ganz, wird dies als Zeigen interpretiert.

Ursprünglich mussten fast alle VR-Headsets mit einem Computer verbunden werden, um genug Rechenleistung für die Darstellung der Inhalte zu haben. Mit der Zeit wurden jedoch auch Standalone-VR-Headsets entwickelt, wie zum Beispiel das im Jahr 2024 auf den Markt gebrachte Apple Vision Pro. Diese verfügen über eine eigene Rechenleistung, meist in Form von ARM-Prozessoren, 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 der Nutzer sich viel bewegt.



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

Quelle: [11]

Jedoch sind nicht alle VR-Headsets für AR geeignet. Die meisten VR-Headsets besitzen keine Kamera, um die reale Welt aufzunehmen und dem Nutzer wiederzugeben. Nur Headsets wie bspw. die Oculus Quest 3, welche in Abbildung 2 dargestellt ist, oder die Apple Vision Pro besitzen eine oder mehrere Kameras nach außen, um die reale Welt aufzunehmen und so AR durch eine Vermischung aus echter Welt und generierten Daten zu ermöglichen.

Durch all diese Technik, bieten die neusten VR-Headsets eine sehr hohe Immersion durch herausragende audiovisuelle Qualität für Anwendungen in AR. Sie sind besonders dafür geeignet hochauflösende dreidimensionale Inhalte in die reale Welt zu projizieren und so eine immersive Erfahrung zu schaffen. Daher sind sie besonders für Anwendungen geeignet, bei denen eine hohe Immersion und eine hohe Qualität der Darstellung wichtig sind, wie bspw. bei Spielen oder Simulationen.

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

Da sie aufgrund der Nähe zu den Augen des Nutzers aber mit relativ kleinen Bildschirmen auskommen, sind sie im Vergleich zu vielen anderen dedizierten Anzeigegeräten für AR sind VR-Headsets auch relativ günstig. Die niedrigen Kosten entstehen auch, da VR-Headsets auch für Videospiele und andere Unterhaltungsanwendungen genutzt werden und daher eine große Nutzerbasis haben. So existiert ein weiterer finanzieller Anreiz für die Entwicklung von günstigen VR-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 durch die Displays hindurch sehen kann. So kann auch ohne digitales Image-Passthrough ein AR-Effekt erzielt werden. Die Durchsicht durch die Displays sind zudem meist schärfer als bei Image-Passthrough in VR-Headsets, da direkt die echte reale Welt gesehen wird und nicht eine Kameraaufnahme.

Smart-Glasses sind außerdem meist leichter und kleiner als VR-Headsets, da sie meist nicht das volle Sichtfeld des Nutzers abdecken müssen und die Technik für Image-Passthrough nicht benötigt wird. Dadurch sind sie besser für den Alltag und für längere Anwendungszeiten geeignet als konventionelle VR- oder AR-Headsets. Zudem verursachen sie durch ihre permanente Transparenz weniger Übelkeitsgefühle bei den Nutzern, da sie immer die reale Welt als Referenzpunkt haben.

Aufgrund dieser Eigenschaften werden sie viel als potenzielle Anzeigegeräte für AR in der Arbeitswelt untersucht. Beispielsweise wurden von Forschern der Hochschule Osnabrück und Universität Osnabrück 36 Use Cases für Smart-Glasses in nur 2 Logistikunternehmen identifiziert [12].

3.3.2 Hand-Held-Devices

Hand-Held-Devices sind Geräte, die ein Nutzer in der Hand hält. Diese Kategorie besteht heutzutage hauptsächlich aus 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. Dafür nutzen sie die Kamera und diverse Sensoren, um

die Umgebung des Nutzers aufzunehmen und digitale Informationen in das Kamerabild einzublenden. Das Smartphone oder Tablet ist dabei wie ein Fenster in die digitale Welt, durch das der Nutzer die erweiterte Realität sehen kann.

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

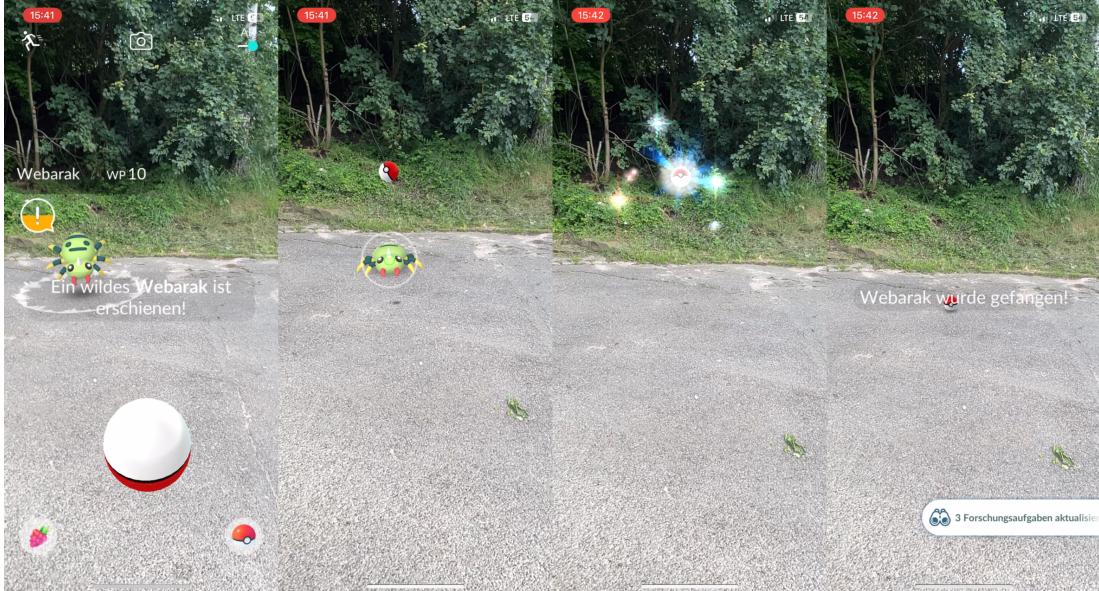


Abbildung 3: Screenshots aus dem AR-Spiel Pokémon Go

Jedoch ist die Nutzung von Smartphones für AR sehr weit verbreitet, da fast jeder ein Smartphone besitzt und so 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 zeigt das Potenzial der riesigen Nutzergruppe von Smartphones für AR-Anwendungen.

Diese Technologien können auch für praktische Anwendungen genutzt werden, wie bspw. für die Navigation in der Stadt oder für das Ausmessen von Längen mithilfe der Kameras und Sensoren.

Auch können Smartphones als Displays für HMDs genutzt werden. Dabei ist jedoch meist die Kamera des Smartphones nicht mehr nutzbar, wodurch normalerweise nur die Wiedergabe von VR-Inhalten und nicht 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 ist und daher starke visuelle Fehler auftreten.

3.3.3 Spatial Displays

Die bisher vorgestellten Konzepte für AR-Anzeigegeräte basieren auf Displays die direkt am Nutzer selbst befestigt sind. Im Kontrast dazu ist die meiste Technik bei Spatial Displays in der Umgebung des Nutzers verbaut. Der Nutzer selbst muss dabei 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.

Da in vielen Fällen keine spezielle Hardware am Nutzer selbst befestigt werden muss, kann die Immersion durch Spatial Displays sehr hoch sein. 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 Nachteil von Spatial Displays ist der Abstand der Displays zum Nutzer. Da sie weiter vom Nutzer entfernt sind als HMDs oder Hand-Held-Devices können sie mit der gleichen Displaygröße nur ein kleineres Sichtfeld abdecken. Daher werden bei Spatial Displays oft sehr große Displays bzw. Beamer genutzt, um trotzdem ein großes Sichtfeld abzudecken. Dadurch sind Spatial Displays deutlich teurer und aufwendiger in der Installation als HMDs oder Hand-Held-Devices. Sie werden daher meist in aufwendigen Anwendungen genutzt und sind für den privaten Gebrauch eher ungeeignet.



Abbildung 4: Innenraum des Mercedes-Benz Fahrsimulators

Quelle: [15]

Eine solche Anwendung ist in Abbildung 4 zu sehen, 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 Autos abgestimmt, sodass

der Fahrer das Gefühl hat, sich tatsächlich in einem Auto 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 aber auch mit extrem hohen Kosten und Aufwand im Vergleich zu anderen Darstellungsmöglichkeiten verbunden sind.

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 Autos zu den Spatial Displays, welche Informationen in das Sichtfeld des Fahrers projiziert, ohne dass sich dieser auf ein Display 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 er 3 Hauptbestandteile seines Ansatzes von Gamification von UX an, die in einer Anwendung vorhanden sein müssen, um das Nutzerengagement zu erhöhen:

- 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.

[16, S.66]

Der Ansatz von Hillmann versucht dabei stark, die Belohnungsmechanismen von Spielen zu nutzen, um den Nutzer zu motivieren. Eine Möglichkeit, die er beschreibt, ist das Nutzen von Punkten, um den Fortschritt des Nutzers zu visualisieren und ihn so zu motivieren, weiterzumachen.

Hillmann beschreibt auch, wie Gamification von UX bei der Orientierung, Beruhigung und Onboarding-Problemen der neuen Technologie von XR helfen kann. Neue Nutzer

schauen beispielsweise oft nicht in die richtige Richtung, um die Inhalte zu sehen, oder sind überfordert von der neuen Technologie. Daher kann durch intelligente Anrichtung der Objekten und eine spielähnliche Zielführung die Orientierung und User Experience des Nutzers 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 wird. Dabei wird auf der Hardwareseite auf das verwendete Anzeigegerät für AR, die Meta Quest 3, eingegangen, während auf Softwareseite die Engine und Frameworks vorgestellt werden, in welchen die Prototypen implementiert werden.

4.1 Meta Quest 3

Im Laufe der Bachelorarbeit wird als Anzeigegerät für die AR-Anwendung die Meta Quest 3 verwendet. Dabei 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, der speziell für VR- und AR-Anwendungen entwickelt wurde, und ist dadurch selbst in der Lage auch anspruchsvolle Inhalte zu rendern. Es muss nicht wie viele andere HMDs an einen Computer oder ein Smartphone angeschlossen werden, sondern kann direkt verwendet werden. Das Headset verfügt auch beispielsweise über einen eigenen Browser, wodurch AR Anwendungen auch 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.

Über zwei Displays mit jeweils einer Auflösung von 2064 x 2208 Pixeln und einer Bildwiederholrate von bis zu 120Hz zeigt das Headset Inhalte mit einer hohen Bildqualität und einer flüssigen Darstellung an. Durch die Kombination der geringen Latenz aufgrund des On-Board-Prozessors und der hohen Bildwiederholrate sollte das Gerät eine hohe Immersion und ein angenehmes Nutzungserlebnis bieten.

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



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

Zusätzlich zu den 2 RGB-Kameras sind insgesamt noch 4 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 auch die Position des Nutzers im Raum kontinuierlich getrackt und angepasst, sodass für das Headset keine externen Trackingstationen notwendig sind.

Die Meta Quest kann zudem alternativ zu Controllern als Eingabegeräte kann das Headset mithilfe der 4 IR-Kameras und 2 RGB-Kameras die Hände des Nutzers erkennen und tracken, sodass die eigenen Hände in VR- und AR-Anwendungen als Eingabegerät verwendet werden können [8]. Dafür sind 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 Oculus 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. Darunter sind auch die Unternehmen hinter den größten Browsern der Industrie: 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 Webbrowsern verwendet werden.

WebGL basiert dabei 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

WebXR ist eine standardisierte API für Webanwendungen, welche es ermöglicht, immersive VR und AR Anwendungen für das Internet zu erstellen. 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.“ [aus dem Englischen mit DeepL 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.

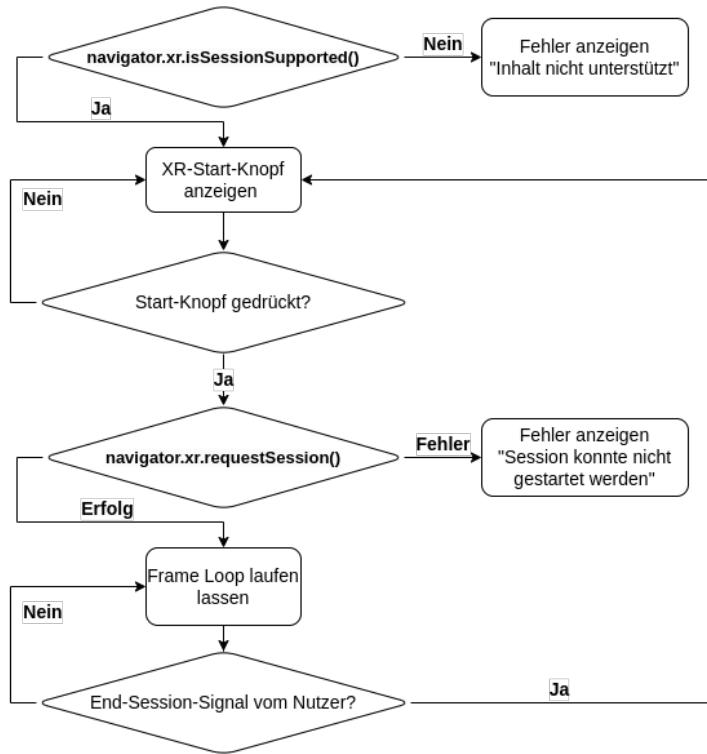


Abbildung 7: Application Flow von WebXR-Anwendungen

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

Der Flow von WebXR-Anwendungen ist in Abbildung 7 dargestellt. Dabei wird zuerst beim Aufrufen der Anwendung geprüft, ob die angegebene Art von XR-Inhalt von der Hardware und dem User Agent (UA) unterstützt werden. Mit dem User Agent (UA) der Software, ist dabei die Software gemeint, mit welcher der Nutzer auf die Anwendung zugreift, also der Webbrower 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. Startet die Session erfolgreich, 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.

4.3.1 Three.js

Three.js ist eine Open-Source JavaScript-Bibliothek unter der MIT-Lizenz, die auf WebGL basiert und die Erstellung und Darstellung von 3D-Grafiken in Webanwendungen vereinfacht. 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.

Es existieren auch einige Frameworks die auf der Three.js API basieren und die Entwicklung von AR- und VR-Anwendungen und 3D-Anwendungen generell weiter vereinfachen. Ein Beispiel dafür ist das Framework A-Frame, welches auf die Entwicklung von AR und VR Anwendungen spezialisiert ist und Features einer fast vollwertigen Game-Engine bietet [23].

Zudem wird Three.js von vielen Entwicklern und Unternehmen verwendet und hat dadurch eine große Community und so viele Ressourcen und Tutorials, die Entwicklern helfen können, ihre Anwendungen zu erstellen und zu debuggen.

Diese Vielfalt an Features und die große Community machen Three.js, vor allem in Kombination mit A-Frame, zu einer interessanten 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 Oculus Quest 3 Probleme mit der Erkennung der VR-Brille und der Darstellung von AR-Inhalten.

4.3.2 Babylon.js

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

Die Bibliothek bietet dabei auch viele Quality-of-Life-Features, wie beispielsweise einen Szenen-Explorer und -Inspektor, der sich für die Entwicklung und das Debugging von 3D-Szenen 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 Objekten betrachtet und verändert werden.

4 Technologien



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

5 Konzept

Die Bachelorarbeit beschäftigt sich mit der Darstellung von Requirements in AR. Dabei soll untersucht werden, wie Anforderungen in einer 3D-Umgebung dargestellt werden könnten. Dazu werden verschiedene Interaktionskonzepte entwickelt, welche als Basis für die Implementierung von Prototypen mit WebXR dienen. 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 Anforderungen von Produkten gedacht. Die Idee ist, ein Produkt in einer Animation in seine einzelnen Bauteile zu zerlegen und die Anforderungen auf 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. Beispielsweise könnte ein Auto so zerlegt werden, dass bei der Animation die Räder, die Karosserie, der Motor und die Innenausstattung einzeln als eigene Objekte sichtbar werden. So kann der Nutzer das gesamte Produkt betrachten und sich dann auf Wunsch einzelne Bauteile und deren Anforderungen genauer ansehen. Im Beispiel des Rubiks Cube aus Abbildung 9 wäre dann jeder Würfel ein Bauteil, der seine eigenen Requirements angehängt bekommen würde.

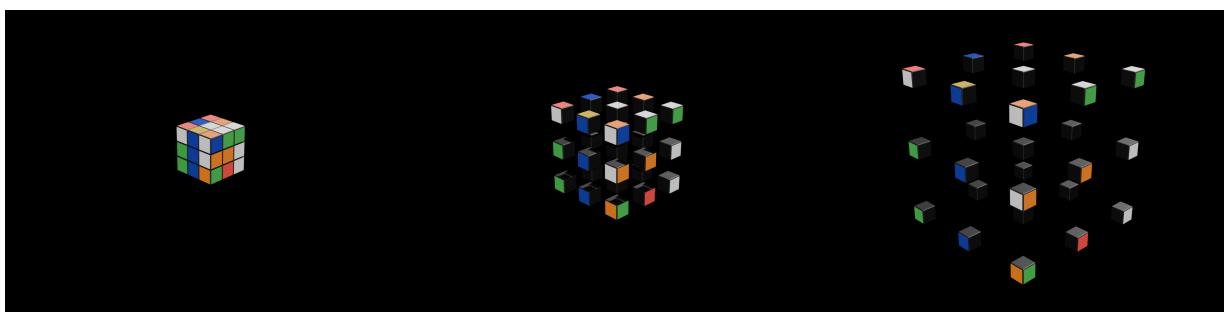


Abbildung 9: Beispiel einer Explosionsanimation anhand eines Rubiks Cube

Die Anforderungen sollen dabei als Text auf UI-Panelen dargestellt werden, die an den zugehörigen Bauteilen angebracht sind.

Am bereits genannten Beispiel des Autos wird klar, dass die Komplexität der Darstellung bei vielen Bauteilen schnell ansteigt und die Übersichtlichkeit verloren gehen kann. Daher ist es bei umfangreichen Produkten eventuell sinnvoll, die Darstellung der Bauteile in mehreren Schritten zu realisieren. Zum Beispiel könnte in einer Übersicht das gesamte Auto in wenigen Bauteilen dargestellt werden, indem beispielsweise ein Rad, das eigentlich aus Reifen, Felge, Radmuttern, Bremsscheibe etc. besteht, als ein einzelnes Objekt dargestellt wird. Will der Nutzer dann die Räder genauer betrachten, kann er in eine Detailansicht wechseln, in der nur ein Rad mit all seinen Bauteilen animiert wird. So lässt sich durch eine Verschachtelung von immer detaillierteren Ansichten eine hohe Komplexität der Darstellung erreichen, ohne dass die Übersichtlichkeit verloren geht.

Für diese Darstellung soll der Nutzer zunächst mithilfe eines Controllers einen Ort für die Darstellung im Raum auswählen. Dieser Ort wird als Ursprungspunkt für die Darstellung der Bauteile verwendet. Dann soll der Nutzer die Explosionsanimation des gerade angezeigten Modells per Knopfdruck aktivieren können. Dabei wird dann die Animation abgespielt und den Bauteilen sollen ihre Requirement-Paneele angehängt werden.

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

Schon bei der Konzeption des Konzepts wird klar, dass einige der Implementierungs schritte, wie beispielsweise das Erstellen der Animation, zeitaufwendig sein können. Daher soll bei diesem Interaktionskonzept die Realisierbarkeit solcher Darstellungen für physische Produkte untersucht werden. Dabei muss aufgrund des erwarteten hohen Zeitaufwands für die Implementierung die Kosten-Nutzen-Relation kritisch betrachtet werden. Um trotzdem noch sinnvoll für den tatsächlichen Einsatz zu sein, muss das Interaktionskonzept sonst einen großen Mehrwert bei der Vermittlung der Requirements bieten.

5.2 Wolken von Anforderungen

Der Ansatz der explodierenden Bauteile ist aufgrund der Individualität der Implementierung des Interaktionskonzepts sehr zeitaufwendig und komplex einzusetzen. Zudem ist dieser Ansatz nur für die Darstellung von Produkten, also physischen Systemen, geeignet. Daher wird ein weiteres Interaktionskonzept entwickelt, welches sich theoretisch auch automatisiert generieren lassen soll und für alle Arten von Anforderungen geeignet ist.

Außerdem soll versucht werden, eine Übersicht über mehr Requirements gleichzeitig zu erlangen, als im ersten Interaktionskonzept, um eine visuelle Darstellung vieler Anforderungen gleichzeitig zu ermöglichen.

Die grundlegende Idee ist, Anforderungen in Wolken von Texten darzustellen, also als wolkenförmige Gruppierungen von UI-Elementen im Raum. Die UI-Elemente sollen dabei Paneele sein, auf denen der Text der Anforderungen sowie weiter Information zu den Anforderungen stehen sollen. Diese Paneele sollen dann in einer Wolkenartigen Anrichtung, ähnlich wie die Punktewolke in Abbildung , dargestellt werden.

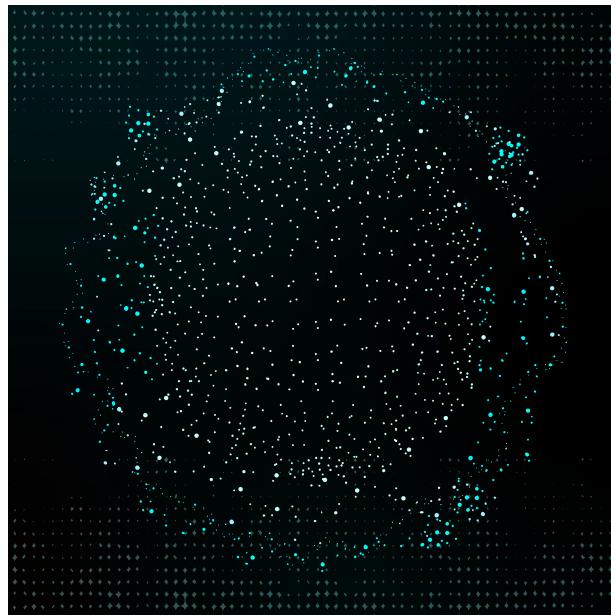


Abbildung 10: Beispiel einer Punktewolke

Quelle: [25]

Hierbei soll dann eine räumliche Gruppierung der Anforderungen nach verschiedenen Kriterien möglich sein. Beispielsweise könnten Anforderungen, die zu einem bestimmten Feature gehören, in einer Wolke gruppiert werden, während Anforderungen, die zu einem anderen Feature gehören, in einer anderen Wolke gruppiert werden. Durch die räumliche Anordnung der Wolken kann der Nutzer schnell erkennen, welche Anforderungen nach der aktuellen Filterung nah zueinander liegen und welche nicht.

Dabei soll es auch möglich sein in Wolken hinein- und herauszuzoomen, um die Granularität der angezeigten Anforderungen zu erhöhen. Beispielsweise könnten in großen Wolken initial nur Punkte als Repräsentationen von Requirements angezeigt werden, in welche man hereinzoomen kann um die Requirements zu lesen. Daraufhin kann wieder herauszoomt werden, um wieder eine Übersicht über mehr Requirements der Wolke zu erlangen. Eine weitere Möglichkeit wäre es, auf den Anforderungspanels Filtermöglichkeiten für verschiedene Beziehungen der Anforderung darzustellen, um dann bei einer Auswahl in die neue Anforderungswolke zu wechseln. Diese Interaktion und die räumliche Anordnung der Wolken sollen dem Nutzer helfen, auch bei einer großen Anzahl von Anforderungen einen Überblick zu behalten und schnell die gewünschten Anforderungen zu finden.

Bei diesem Konzept soll vor allem der Vorteil gegenüber einer 2D-Darstellung kritisch untersucht werden. Denn die Darstellung der Anforderungen in Wolken ist prinzipiell auch in 2D möglich, auch mit der Interaktionsmöglichkeit des Hinein- und Herauszoomens. Daher soll untersucht werden, ob die räumliche Anordnung der Anforderungen 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

Dieses Kapitel befasst sich mit dem Prozess der Implementierung der in Kapitel 5 erarbeiteten Interaktionskonzepte in Prototypen. Dabei wird zunächst ein Überblick über die Entwicklungsumgebung, welche für die Entwicklung mit WebXR auf einer Meta Quest 3 eingerichtet wurde. Daraufhin wird für jedes Interaktionskonzept die Implementierung der Prototypen mit einigen technischen Details beschrieben.

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. Dabei müssen verschiedene Technologien und Tools miteinander kombiniert werden, um eine reibungslose Entwicklung und ein möglichst schnelles Testen der Anwendung zu ermöglichen.

Der erste Schritt ist das Anzeigen der WebXR-Anwendung auf der Meta Quest 3. Das Problem dabei im Vergleich zu „normalen“ Webanwendungen ist, dass WebXR-Anwendungen nur 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. Hierfür gibt es verschiedene Möglichkeiten, wie beispielsweise das Erstellen eines eigenen Zertifikats für den lokalen Entwicklungsrechner oder das Hosting der Anwendung auf einem Server mit HTTPS-Unterstützung. Für den 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 erstellt einen temporären HTTPS-Link, über den die Anwendung aufgerufen werden kann, ohne dass ein eigenes Zertifikat oder ein eigener Server notwendig ist. Dafür muss die Anwendung nur lokal auf dem Entwicklungsrechner laufen und der LocalTunnel-Client gestartet sein, um den temporären Link zu generieren. Als zusätzliche Sicherheit muss dann beim Aufrufen der Seite noch die IP-Adresse des Entwicklungsrechners angegeben werden, um sicherzustellen, dass nur der Entwickler die Anwendung testen kann. Dies muss in der Regel jedoch nur einmal nach jedem Neustart oder Crash gemacht werden, da der Link für die Dauer der Sitzung gespeichert wird.

Der nächste Schritt, wenn Zugriff mit der Meta Quest 3 auf die WebXR-Anwendung besteht, ist die Anzeige von Entwickler-Tools und Debugging-Informationen der Meta

Quest 3. Da die Meta Quest 3 auf Android basiert, kann auf dem Entwicklungsrechner Android Debug Bridge (ADB) installiert werden, um über USB eine Verbindung zur Meta Quest 3 herzustellen. Die Verbindung muss noch in der VR-Brille bestätigt werden, um den Zugriff auf die Entwickleroptionen zu ermöglichen. Ist das geschehen, erscheint die Quest 3 mit ihren geöffneten Websites in der Geräteliste der Chrome DevTools, die über die URL `chrome://inspect/#devices` aufgerufen werden können. Dort können dann die Entwickler-Tools der Meta Quest 3 geöffnet werden, um beispielsweise die Performance der Anwendung zu überwachen und Fehlermeldungen zu sehen.

Für 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, die es ermöglicht, WebXR-Anwendungen direkt im Browser zu testen. Mit dieser Erweiterung wird für WebXR ein einfacher Raum mit einem Headset und den beiden Meta Quest Controllern simuliert. Das Headset sowie die beiden Controller können unabhängig voneinander über die Erweiterung positioniert und gesteuert werden. In dem in Abbildung 11 links dargestellten Panel können verschiedene Interaktionen, wie beispielsweise Klicks oder das Bewegen der Controller, simuliert werden, um die Anwendung zu testen. Der Emulator generiert dann eine live Vorschau der Anwendung, die direkt im Browser angezeigt wird.

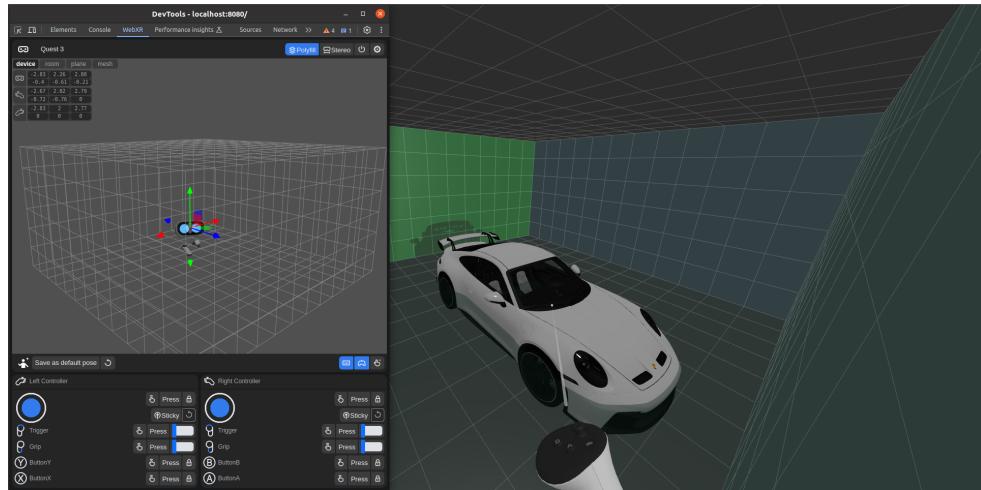


Abbildung 11: Screenshots der Immersive Web Emulator Erweiterung in Chrome

Das Testen der Anwendung mit dieser Erweiterung ist deutlich schneller und einfacher als das Testen auf der Meta Quest 3, da die Anwendung direkt im Browser getestet werden kann und keine Verbindung zur Meta Quest 3 notwendig ist. Zudem 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. Dennoch ist es wichtig, 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.

6.2 Implementierung der Interaktionskonzepte

In den folgenden Abschnitten wird auf die Implementierung der zuvor beschriebenen Interaktionskonzepte eingegangen. Dabei werden grundlegende Konzepte und Technologien vorgestellt und erklärt, die für die Implementierung notwendig sind. Zudem werden Screenshots der implementierten Konzepte gezeigt und die Funktionsweise der Interaktionen beschrieben. Dabei 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. Hierfür wird das Skelett einer WebXR-Anwendung aus einer Artikelserie des Taikonauten-Magazins verwendet, welches als Basis für die Implementierung dient [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 schon einige Interaktionen des Controllers, wie das Auswählen von Objekten durch Raycasting, implementiert, die als Basis für die Implementierung der Interaktionskonzepte dienen.

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. Dabei muss jedes Bauteil als eigenes Objekt im 3D-Modell vorhanden sein, um sie in der Animation separat darstellen und referenzieren zu können. Für die erste Implementierung wird ein einfaches 3D-Modell eines Tetris Blocks verwendet, welcher aus 4 verschiedenfarbigen Bauteilen besteht. Basierend auf der Implementierung des Tetris-Blocks soll dann das Konzept auf ein komplexeres Modell, wie beispielsweise ein Auto, übertragen werden.

Das Modell wurde 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 ist, 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 eingebettet werden. Das ist essenziell für die effektive technische Umsetzbarkeit des Konzepts, da die Animation der Bauteile so in speziellen Animationsprogrammen, wie beispielsweise Blender, erstellt werden kann und nicht in der WebXR-Anwendung selbst. Dadurch sind auch komplexere Animationen möglich, die in der WebXR-Anwendung selbst nicht oder nur sehr aufwändig realisierbar wären.

Der nächste Schritt ist das Platzieren des erstellten 3D-Modells in der WebXR-Umgebung. Dazu wird das Prinzip des Raycastings 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 zu Beginn einmalig in den Meta Quest Einstellungen 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. Dann wird vom Controller ein Strahl in die Szene geschossen, und der Punkt, an dem der Strahl ein Objekt trifft, wird als Event zurückgegeben.

An diesem Punkt wird dann ein Ankerpunkt erstellt, den das 3D-Modell als Referenzpunkt im AR-Raum nutzt. Dieser Ankerpunkt ist ein Babylon.js-Objekt, welcher einen Punkt im AR-Raum auch bei Bewegung des Nutzers am gleichen Ort halten kann. Bewegt sich dann der Ankerpunkt, bewegt sich auch das 3D-Modell mit, sodass es immer an der gleichen Stelle im Raum bleibt.

Ist das 3D-Modell platziert, kann die Animation des Produkts gestartet werden. Dafür wird der A-Knopf des Controllers als Start-Button für die Animation verwendet, mit dem sich die Animation vorwärts und wieder rückwärts abspielen lässt. Für die Animation wird die aus Blender in das glTF-Modell eingebackene Animation verwendet. Dafür wird in Babylon.js eine Animationsfunktion erstellt, die die Animation des Modells über einige Parameter, wie beispielsweise den Start- und Endframe der Animation, steuert. So kann für die Vorwärts- und Rückwärtsanimation die gleiche Funktion verwendet werden, indem die Start- und Endframe-Parameter basierend auf einem globalen Boolean, der den Animationsstatus speichert, gesetzt werden.

Beim Abspielen der Animation wird jedes Bauteil des Produkts in einer Schleife durchgegangen und dessen Animation gestartet. Diese Schleife wird dann auch genutzt, um den einzelnen Bauteilen ihre Anforderungen als Text-UI-Elemente zuzuweisen. Zunächst wird in dieser Schleife überprüft, ob es zu der ID des Bauteils Anforderungen gibt, die dargestellt werden sollen. Ist das der Fall, wird in Babylon.js eine Fläche erstellt, auf der der Text dargestellt wird, und diese Fläche an das Bauteil als Kindobjekt angehängt. So wie das gesamte Modell den Ankerpunkt als Referenzpunkt nutzt, nutzt dann jedes Requirement-Panel die Position seines Bauteils als Referenzpunkt für seine Position im AR-Raum. Die Fläche kann dabei auch als Knopf funktionieren, um bei einem Klick beispielsweise das Bauteil zu vergrößern oder auf eine Detailansicht des Bauteils zu wechseln. Diese UI-Elemente werden aber, wie in Abbildung 12 zu sehen ist, nur angezeigt, wenn das Produkt gerade „explodiert“ ist und nicht, wenn das Produkt gerade im Normalzustand ist.



Abbildung 12: Screenshots des explodierenden Tetris-Blocks mit Anforderungen in AR

Implementation 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 [27]. Das Modell, welches in Abbildung 13 zu sehen ist, wurde ausgewählt, da es viele verschiedene Bauteile enthält, die in der Animation separat dargestellt werden können.



Abbildung 13: Screenshot des in der Anwendung genutzten Porsche Modells

Bei dem in dieser Bachelorarbeit verwendetem Modell ist es jedoch wichtig zu beachten, dass es sich um kein vollständig akkurate und funktionales Modell eines Autos handelt. Beispielsweise existieren keine Achsen, an denen die Räder hängen. Für den anschaulichen

Zweck dieser Bachelorarbeit ist das Modell jedoch ausreichend, da es genug „reale“ Bauteile enthält, um das Interaktionskonzept zu demonstrieren.

Würde das Interaktionskonzept in einer echten Anwendung für Kunden verwendet werden, sollte jedoch mit möglichst akkurate CAD-Modellen gearbeitet werden, um die Anforderungen an die Bauteile möglichst genau darzustellen. Auf die Möglichkeit einer solchen professionellen Anwendung wird in der Diskussion eingegangen.

Für die Verwendung des Modells in der Anwendung muss als Nächstes eine Animation mit dem Modell erstellt werden, welche die Bauteile des Autos in ihre Einzelteile zerlegt. Dafür wird in Blender eine Animation erstellt, welche einige Bauteile, wie beispielsweise die Verkleidung, die Räder und der Spoiler, nach außen weg bewegt. Beim Erstellen der Animation musste zudem beachtet werden, dass sichmöglichst wenig Bauteile bei der Animation überschneiden. Dadurch sind haben einige Bauteile sehr einfache Animationen mit nur 2 Keyframes, wie bspw. die Räder die einfach nach außen verschoben werden, und andere Bauteile benötigen kompliziertere Animationen mit bis zu 5 Keyframes, um Überschneidungen zu verhindern.

Zur Veranschaulichung der Animation ist in Abbildung 14 eine Bildsequenz der Animation des Porsche Modells zu sehen.



Abbildung 14: Bildsequenz der Animation des Porsche Modells

Um den Bauteilen später ihre Anforderungen zuzuweisen, müssen die relevanten Bauteile identifizierbar sein. Dafür wird in Blender jedem animierten Objekt eine ID (bspw. #1_Rad, #2_Lichter) am Anfang des Objektnamens zugewiesen, die später in der Anwendung als Referenz für die Anforderungen dient.

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 Anforderungen des Hauptobjekts anzuzeigen. Beispielsweise besteht ein Rad aus Reifen, Felge, Bremsscheibe etc., wobei nur das Rad als Hauptobjekt die Anforderungen an das Rad anzeigt und die anderen Objekte die Anforderungen an sich selbst in der Detailansicht. Um die Objekte jedoch einfach ihren Elternobjekten zuordnen zu können, wie beispielsweise für die Klick-Abfrage des Rads, werden die IDs der Elternobjekte in den IDs der Kindobjekte gespeichert. Dabei jedoch ohne Unterstrich „_“ hinter der ID, wodurch sie beim Erstellen der Anforderungspanels ignoriert werden.

Die detaillierte Radansicht ist ein separates, ebenfalls in Blender erstelltes glTF-Modell, welches nur das Rad, seine Bauteile und seine eingebetteten Animationen enthält. Für die Änderung auf die detaillierte Radansicht wird eine Abfrage hinzugefügt, die prüft, ob der Nutzer mit seinem Controller auf ein Rad klickt. Dafür wird wieder das Raycasting-Prinzip verwendet, um den Punkt zu bestimmen, an dem der Controller auf ein Objekt trifft. Ist das getroffene Objekt Teil vom Rad, wird an der Stelle ein neuer Ankerpunkt erstellt, an dem die Detailansicht des Rads angezeigt wird. Der alte Ankerpunkt und die Bauteile des Autos werden dann ausgeblendet, und die Bauteile des Rads werden an den neuen Ankerpunkt angehängt. Wechselt der Nutzer dann wieder zurück zur Gesamtansicht des Autos, wird wieder das Auto-Modell an den Ankerpunkt angehängt und die Detailansicht des Rads ausgeblendet. So kann der Nutzer die Anforderungen an das Rad in einer Detailansicht betrachten und bei Bedarf wieder zur Gesamtansicht des Autos zurückkehren.

6.2.2 Wolken von Anforderungen

Das Interaktionskonzept der Wolken aus Anforderungspanels ist deutlich einfacher zu implementieren als das Konzept der explodierenden Bauteile. Der erste Schritt der Implementierung war die Erstellung eines Algorithmus, welcher einen Array von Anforderungen erhält und zu diesen eine 3D-Wolke von Punkten in einem gegebenen Bereich erstellt. Für jedes Requirementcluster wird dann eine Wolke von UI-Panels erstellt, die die Anforderungen als Texte enthalten.

Wie beim Konzept der explodierenden Bauteile wird auch hier ein Ursprungspunkt der Wolken durch Raycasting und auslösen des Triggers des Controllers bestimmt.

Bei der ersten Betrachtung der Anforderungen wurde jedoch klar, dass die Anforderungen sich bereits in kleinen Wolken stark überlappen und viele Anforderungen so nur schwer bis gar nicht lesbar sind. Wie in Abbildung 15 zu sehen ist, treten bereits bei Wolken von 5 Anforderungen starke Überlappungen auf, welche die Lesbarkeit der Anforderungen stark beeinträchtigen.

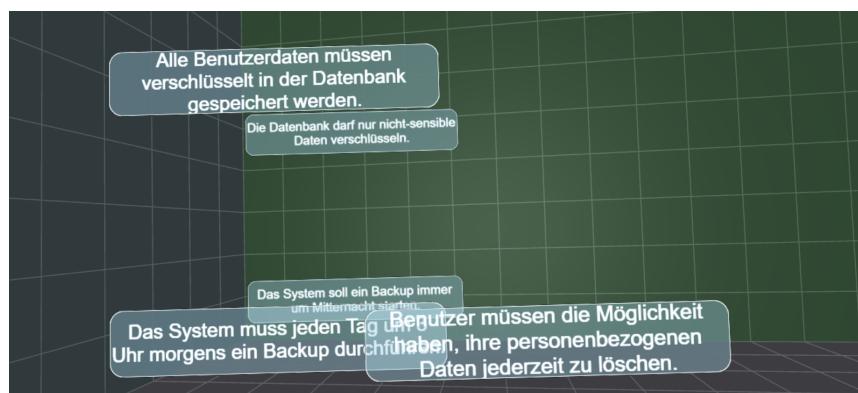


Abbildung 15: Wolke aus 5 Anforderungspanels

In Abbildung 15 fällt auf, dass durch die Transparenz der UI-Elemente die 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. Zudem wurde der Abstand der Anforderungspanels zueinander erhöht und die Anforderungen wurden durch eine Zufallsfunktion jeweils leicht von ihrem Ursprungspunkt verschoben, um die Überlappungen zu verringern. Bei kleinen Anforderungswolken führen diese Anpassungen zu einer verbesserten Lesbarkeit der Anforderungen.

Jedoch wird schon bei nur wenig größeren Anforderungswolken wie der in Abbildung 16 zu sehenden Wolke aus 9 Requirements klar, dass sich Überlappungen bei realistischen Wolkengrößen nicht vollständig vermeiden lassen.

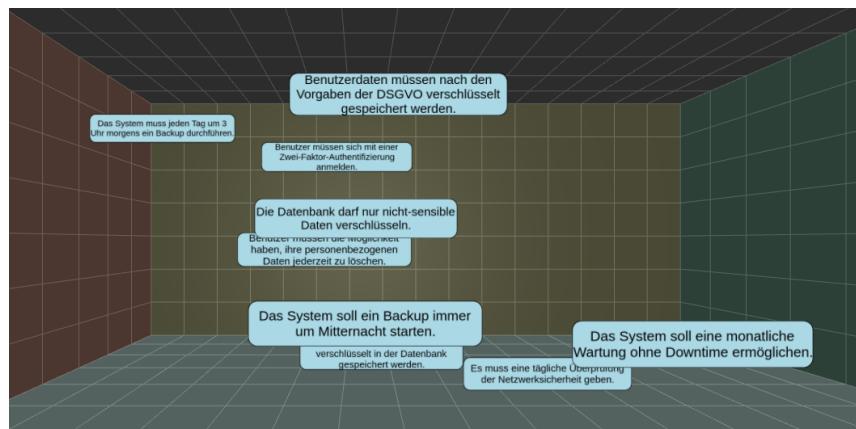


Abbildung 16: Wolke aus 9 Requirements mit undurchsichtigen Panels und größerem Abstand

Daher hat das Konzept trotz der initialen Anpassungen noch immer eine große Beeinträchtigung in der Lesbarkeit und damit der Usability. Da das Interaktionskonzept der Wolken vor allem für große Mengen an Requirements gedacht war, muss die Usability der Darstellung auch in großen Anforderungswolken gegeben sein. Um die Lesbarkeit der Anforderungen auch in großen Wolken zu verbessern wurden daher drei verschiedene weitere Ansätze in Betracht gezogen, welche im Folgenden erläutert werden.

Der erste Ansatz war das weitere Entfernen der Anforderungen, indem die Anforderungen so breitflächig wie möglich angerichtet werden, um so Überlappungen zu vermeiden. Dieser Ansatz führt jedoch dazu, dass die Anforderungen fast auf einer Ebene liegen müssen. Jedoch ist das navigieren durch große Flächen in AR unpraktisch im Vergleich mit einer ähnlichen Darstellung auf einem normalen Bildschirm. Daher wird durch diesen Lösungsansatz die Darstellung der Anforderungen im dreidimensionalen Raum weniger effizient als in einer 2D-Darstellung.

Der zweite Ansatz war das mögliche Fokussieren von Anforderungen mithilfe des Raycastings des Controllers. So könnten Anforderungen, die gelesen werden sollen, in den Fokus

gebracht und lesbar gemacht werden. Jedoch wird bei diesem Ansatz nicht der Überblick über alle Anforderungen gewährleistet, da nur eine Anforderung gleichzeitig fokussiert werden kann. Zudem kann es bei großen Anforderungswolken sein, dass gesuchte Anforderungen gar nicht gefunden und daher auch nicht fokussiert werden können, da sie durch andere Anforderungen verdeckt sind. Zieht man dabei in Betracht, dass vor allem dieses Interaktionskonzept für die Darstellung von vielen Anforderungen gleichzeitig gedacht ist, ist dieser Lösungsansatz auch keine Option.

Der dritte Ansatz war das Verkleinern der Anforderungspanels, um so mehr Anforderungen in einer Wolke darstellen zu können. Dabei wird jedoch schnell die Lesbarkeit der Anforderungen zum Problem. Durch den in Kapitel 3.3.1 beschriebenen Screen-Door-Effekt und andere Probleme bei der Schärfe von AR-Displays ist es schwer, kleine Texte in AR gut lesbar darzustellen. Dieses Problem der Unlesbarkeit wird bei großen Anforderungswolken, in denen viele Anforderungen 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 normalen Bildschirmen nicht auf, wodurch auf diesen eine Darstellung mit einer deutlich höheren Informationsdichte möglich ist. Daher ist auch dieser Lösungsansatz nicht besser als eine zweidimensionale Anwendung geeignet, um viele Anforderungen gleichzeitig darzustellen.

Da keiner der Ansätze eine Lösung für das Problem der Überlappung und Lesbarkeit der Anforderungen bietet, welches das Konzept nicht weniger sinnvoll macht als eine 2D-Darstellung, wird das Konzept der Wolken von Anforderungen in AR nicht weiter verfolgt.

Im Ausblick in Kapitel 7.3 wird noch auf die Möglichkeit der Implementierung des Konzepts als 2D-Anwendung eingegangen.

6.2.3 2D-Mockup für Wolken aus Anforderungen

Um ein Gefühl für die Darstellung des Interaktionskonzepts der Anforderungswolken zu erlangen, wurde im UI-Prototyping-Tool Figma Mockups erstellt, wie die Wolken in einer normalen Desktop-Anwendung dargestellt werden könnten. Im Folgenden werden anhand einiger Screenshots der Mockups Unterschiede zur Darstellung in AR diskutiert. Der Fokus liegt dabei darauf, die Unterschiede der Usability der beiden Darstellungen zu vergleichen.

Im ersten Screenshot aus Abbildung 17 ist eine Übersicht über eine große Anzahl an Requirements dargestellt. Da auch auf einem normalen Display nur eine begrenzte Anzahl von Requirements gleichzeitig lesbar sind, werden die Requirements in der Übersicht abstrakt als einfache Punkte dargestellt. Jeder Punkt ist dabei ein Requirement, welches gelesen werden kann wenn der Nutzer auf das Requirement hereinzoomt. Die Farben der Requirements sollen dabei Auskunft über den Bearbeitungsstatus der Requirements geben.

- **Grüne Requirements:** Requirements dieser Farbe sind bereits vollständig bearbeitet und überprüft, sie müssen momentan nicht mehr bearbeitet werden.
- **Rote Requirements:** Rote Requirements wurden noch gar nicht bearbeitet, sie sind meist neu oder wurden lang ignoriert.
- **Gelbe Requirements:** Requirements in gelb wurden bereits einmal bearbeitet, benötigen aber noch eine Überprüfung. Alternativ könnten in dieser Farbe alte Requirements mit neuen Änderungen dargestellt werden.

Mithilfe dieser Farbenkorrelation ist es mit dieser Darstellung sehr einfach Bereiche zu erkennen, in denen noch Verwaltungsaufgaben erledigt werden müssen. Zwar können offensichtlich nicht alle Requirements gleichzeitig mit ihrem Text betrachtet werden, doch durch die Vereinfachung der einzelnen Requirements als textlose Kugeln lässt sich ein gutes Gesamtbild erlangen.

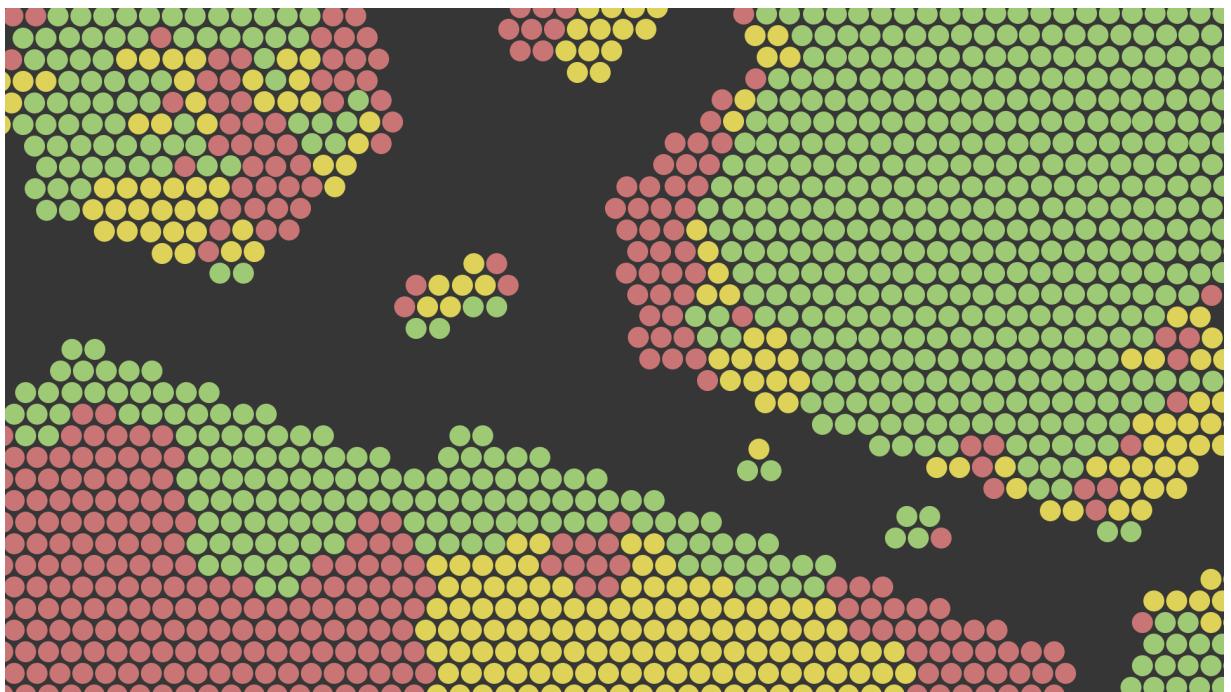


Abbildung 17: Mockup zur Übersicht über eine große Anzahl an Requirements

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. Die Größe, ab der die Texte dann als lesbar empfunden werden, hängt dabei dann von den nutzerspezifischen Bildschirm und einigen weiteren nutzerspezifischen Eigenschaften ab.

In Abbildung 18 ist ein Mockup einer solchen näheren Perspektive dargestellt. Hier können dann alle Texte der Anforderungen selbst gelesen werden und es können einzelne Requirements angeklickt werden, um weiter mit ihnen zu interagieren. Dabei wären dann Sei-

tenmenüs denkbar, welche aufklappen, wenn ein Requirement angeklickt wurde, um detailliertere Informationen und eventuelle Interaktionsmöglichkeiten zu den Requirements anzuzeigen.



Abbildung 18: Mockup der hereingezoomten Ansicht

Bei der Erstellung eines Mockups des Wolken-Interaktionskonzepts in 2D wurde klar, dass das Konzept zumindest bei erster Betrachtung fast keine Nachteile durch den Wechsel auf eine zweidimensionale Darstellung erfährt.

Tatsächlich lassen sich, wie in Abbildung 17 gezeigt, durch die höhere mögliche Informationsdichte in der zweidimensionalen Darstellung sogar größere Wolken aus Requirements darstellen als in AR. Zwar könnten durch die Abstraktion der Requirements als Punkte auch in AR deutlich mehr Requirements dargestellt werden, aber nicht so effizient wie in 2D. Schon durch die höhere Schärfe der meisten 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 auch zu beachten, dass sehr kleine Elemente mit einem Controller schwerer auszuwählen sind, als mit einer Maus, da eine Maus auf einer statischen Fläche abgelegt wird während der Controller frei in der Luft gehalten werden muss. Da durch diesen Effekt das Zielen mit einer Maus auf kleine Elemente einfacher und stabiler ist, kann von dem Interaktionskonzept in 2D sogar eine Verbesserung der Usability erwartet werden.

7 Zusammenfassung

In diesem Kapitel wird ein Rückblick auf die Arbeit gegeben und die Ergebnisse zusammengefasst. Zudem werden die Ergebnisse bewertet indem ein Fazit über die Effektivität und Usability der entwickelten Konzepte gezogen wird. Dabei wird auch auf die Herausforderungen und Schwierigkeiten eingegangen, die bei der Umsetzung der Konzepte aufgetreten sind. Zuletzt wird ein Ausblick auf mögliche Weiterentwicklungen der Konzepte gegeben.

7.1 Ergebnisse

Im Rahmen der Bachelorarbeit wurden zwei Interaktionskonzepte für die Visualisierung von Requirements in Augmented Reality ausgearbeitet und in einem Prototypen umgesetzt. Der Prototyp ist dabei 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.

Das erste Konzept ist die Visualisierung von Requirements in Kombination mit einem möglichst akkuraten 3D-Modell des zu entwickelnden Produkts. Dabei wird das 3D-Modell vom Nutzer auf einer freien Fläche im Raum platziert und kann in einer „Explosions“-Animation, welche in Abbildung 19 anhand einer Bildsequenz dargestellt ist, in seine Bauteile zerlegt werden. Ist das Modell in seine Einzelteile zerlegt, werden die Anforderungen als Panels an den einzelnen Bauteilen visualisiert. Der Nutzer kann dann die Animation wieder rückwärts abspielen, um das Modell wieder zusammenzusetzen. Dabei verschwinden die Anforderungen wieder, und das Produkt kann in seinem originalen Gesamtzustand ohne Requirements betrachtet werden. Im Prototyp wurde dafür ein Auto-Modell verwendet, welches in einer Animation des ganzen Autos in seine Einzelteile zerlegt wird. Zur Visualisierung der eingeplanten Detailansicht können im Prototyp außerdem die Räder des Modells angeklickt werden, um so eine Detailansicht der Anforderungen an die Räder zu erhalten. Das Interaktionskonzept zeigt dabei viel Potenzial, um viele Interaktionsmöglichkeiten mit dem 3D-Modell und den Anforderungen zu ermöglichen. Ein großer Mehrwert der Visualisierung in AR ist dabei, dass das 3D-Modell in Originalgröße dargestellt werden kann und so eine realistische Darstellung des Produkts ermöglicht wird.



Abbildung 19: Bildsequenz aus dem Prototyp der Explodierenden Bauteile in AR

Das zweite Konzept ist die Visualisierung von Requirements in Wolken aus Textpanels, die in der AR-Umgebung schweben. Die Wolken sind dabei Cluster von Anforderungen, die thematisch zusammengehören. Zusätzlich können die Anforderungen in den Wolken angeklickt werden, um zugehörige Anforderungen anzuzeigen. Dieses Konzept ist deutlich einfacher automatisiert umzusetzen, da die Anforderungen in den Wolken nur als Textpanels dargestellt werden und keine extra Animationen oder 3D-Modelle benötigt werden.

Jedoch 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 treten bei der dreidimensionalen Darstellung der Anforderungswolken Herausforderungen auf, die in einer 2D-Darstellung nicht auftreten würden. Beispielsweise werden oft Anforderungen hintereinander angezeigt, sodass der Nutzer die Anforderungen nur schwer oder gar nicht mehr lesen kann. Auch können nicht sehr viele Anforderungen in einer Wolke dargestellt werden, da die Anforderungen sonst zu klein werden und nicht mehr lesbar sind. Aufgrund der wenigen AR spezifischen Interaktionsmöglichkeiten und der Probleme der Darstellung des Konzepts im dreidimensionalen Raum ist fraglich, ob das Konzept in AR einen Mehrwert gegenüber einer 2D-Darstellung eines ähnlichen Konzepts bietet.

7.2 Fazit

Insgesamt konnte in dieser Arbeit gezeigt werden, dass die Visualisierung von Requirements in Augmented Reality zumindest in einem Prototypen umsetzbar ist. Das erste Konzept zeigt dabei auch, dass mit der Visualisierung in AR durch viele neue Interaktionsmöglichkeiten ein Mehrwert in der Visualisierung von Requirements erreicht 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 Anforderungen verknüpft werden. So entsteht ein neuer Ansatz,

um Anforderungen besser zu verstehen, zu kommunizieren und zu visualisieren, welcher in anderen Darstellungsformen nur schwer nachzuahmen ist.

Die Arbeit zeigt jedoch auch, dass die Umsetzung der untersuchten Konzepte noch viele Herausforderungen mit sich bringt. Visuell beeindruckende Darstellungen erfordern meist auch eine aufwendige und somit teure 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 Anforderungswolken 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.

Jedoch rate ich dazu ab, das Konzept der Anforderungswolken in AR bzw. in 3D weiter zu verfolgen, da das Konzept mehr für eine zweidimensionale Darstellung geeignet scheint und in AR keinen nennenswerten Mehrwert gegenüber einer 2D-Darstellung bietet.

7.3 Ausblick

In weiteren Arbeiten könnte die Umsetzung der beiden Konzepte in einer realen Anwendung weiter untersucht werden. Dabei würde sich vor allem eine Untersuchung der professionellen Umsetzbarkeit des Konzepts der explodierenden Bauteile anbieten, da dieses schon im Prototypen den größten Mehrwert der Darstellung bietet und noch stark ausgebaut werden kann. Das Konzept bietet auch viele Möglichkeiten zur Erweiterung der Interaktionen. Einige solcher möglicher Interaktionserweiterungen wären:

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

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

Das Konzept der Anforderungswolken sollte hingegen aufgrund der Probleme in 3D als 2D-Darstellung weiter untersucht werden, da sich die Wolken von Anforderungen auch in einer 2D-Darstellung realisieren lassen. In einer 2D-Darstellung könnte das Interaktionskonzept von einer hohen Informationsdichte und einer einfachen Navigation der Fläche an

7 Zusammenfassung

Anforderungen profitieren. Zudem kann es, wenn es als 2D-Darstellung umgesetzt wird, auch in anderen Anwendungen als der Visualisierung von Requirements eingesetzt werden. Beispielsweise könnten dafür Erweiterungen für Anforderungsmanagement-Tools wie Jira oder Confluence entwickelt werden, um so extra Visualisierungen als Alternative zu normalen Listen oder Tabellen anzubieten. So könnte der Kosten-Nutzen-Faktor der Anforderungswolken in einer 2D-Darstellung potenziell deutlich verbessert werden.

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. META PLATFORMS, INC. *Meta Quest 3 Homepage* [online]. 2024 [besucht am 2024-04-03]. Abgerufen unter: <https://www.meta.com/de/quest/quest-3/>.
9. APPLE INC. *iPhone 15 Pro - Technische Daten* [online]. 2023 [besucht am 2024-04-03]. Abgerufen unter: <https://support.apple.com/de-de/111829>.
10. 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>.
11. 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.

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. SHARIF, Abdul Azim. *Porsche 911 GT3* [<https://sketchfab.com/3d-models/porsche-911-gt3-very-high-quality-free-4eee314e142f4737872e08b0df0ff7f4>]. 2024. Auch verfügbar unter: <https://sketchfab.com/3d-models/porsche-911-gt3-very-high-quality-free-4eee314e142f4737872e08b0df0ff7f4> Published under the CC BY 4.0 license.