

CARL VON OSSIETZKY UNIVERSITÄT OLDENBURG

INFORMATIK
BACHELORARBEIT

Entwicklung einer Augmented Reality Anwendung zum Tracken und Erstellen von Markern für den Bildungsbereich

Autor:
Johannes Scheibe

Erstgutachter:
Prof. Dr.-Ing. Jürgen Sauer

Zweitgutachter:
M. Sc. B. Eng. Nils Hartmann

Abteilung Systemanalyse und -optimierung
Department für Informatik

Oldenburg, 23. Oktober 2020

Inhaltsverzeichnis

Abbildungsverzeichnis	v
Tabellenverzeichnis	vi
Abkürzungsverzeichnis	vii
1. Einleitung	1
1.1. Motivation	1
1.2. Zielsetzung	2
1.3. Methodik	2
1.4. Aufbau der Arbeit	3
2. Grundlagen	5
2.1. Augmented Reality	5
2.1.1. Einsatzbereiche	6
2.1.2. Technische Grundlagen	7
2.2. Android Entwicklung	10
2.2.1. Architektur	10
2.2.2. Grundlegende Konzepte	11
2.3. OpenGL	13
2.3.1. Grundlagen von OpenGL	13
2.3.2. Rendering-Pipeline	13
2.3.3. OpenGL ES	15
3. Verwandte Arbeiten	16
3.1. Augmented Reality in der Bildung	16
3.1.1. Geroimenko: Augmented Reality in Education	16
3.1.2. Hedberg: A Systematic Review of Learning through Mobile Augmented Reality	19
3.1.3. Billinghurst: Augmented Reality in Education	20
3.1.4. Diegmann: Benefits of Augmented Reality in Educational Environments	21
3.1.5. Dey: A Systematic Review of 10 Years of Augmented Reality Usability Studies	24

3.1.6. Damberger: Augmented Reality als Bildungenenhancement?	25
3.1.7. Buchner: Offener Geschichtsunterricht mit Augmented Reality	26
3.2. Frameworks und Tools	27
3.2.1. ARToolKitX	27
3.3. Beispielhafte Anwendungen	28
3.3.1. Atlas der Humananatomie	28
4. Anforderungsanalyse	29
4.1. Vision	29
4.2. Der Prototyp	30
4.3. Anforderungsanalyse	30
4.3.1. Beschreibung der Systemumgebung	31
4.3.2. Anwendungsfälle des Prototyps	32
4.3.3. Anforderungsliste	32
5. Entwurf	35
5.1. Allgemeine Technologieentscheidungen	35
5.1.1. Anwendungsart	35
5.1.2. Trackingverfahren	36
5.1.3. Tracking Bibliothek	36
5.2. Designentscheidungen	37
5.2.1. Grundlage der Entwicklung	37
5.2.2. Marker	38
5.2.3. Modelle	38
5.2.4. Texturen	39
5.2.5. Datenspeicherung	39
5.2.6. User Interface	40
6. Implementierung	41
6.1. MainActivity	41
6.2. Augmented Reality	41
6.2.1. ARCameraFragment	42
6.2.2. EducationARRender	42
6.3. Modellverwaltung	42
6.3.1. AdministrationFragment	43
6.3.2. UploadFragment	43
6.3.3. UploadViewModel	44
6.3.4. ConfirmationFragment	45
6.4. Markergenerierung	45
6.4.1. MarkerFragment	45

6.5.	Shader Implementierung	46
6.5.1.	MyShaderProgram	46
6.5.2.	MyVertexShader	46
6.5.3.	Vertex Shader	46
6.5.4.	MyFragmentShader	47
6.6.	Hilfsklassen	47
6.6.1.	FileManager	47
6.6.2.	MarkerGenerator	48
6.6.3.	Model	49
6.6.4.	draw()	49
6.6.5.	ObjLoader	50
6.6.6.	TextureLoader	51
6.7.	ARToolKit-Klassen	51
6.7.1.	ARController	51
6.7.2.	ARX_jni	51
6.7.3.	ARFragment	52
6.7.4.	ARRenderer	52
6.7.5.	ShaderProgram	52
6.8.	Tests	53
6.8.1.	Augmented Reality	53
6.8.2.	Markergenerierung	56
6.8.3.	Laden eigener Modelle	56
7.	Evaluation	57
7.1.	Evaluation des Prototyps	57
7.1.1.	Bewertung des finalen Prototyps	57
7.1.2.	Zusammenfassung	57
8.	Fazit	58
8.1.	Fazit	58
8.2.	Ausblick	58
Literaturverzeichnis		59
A. Beispielanhang		63

Abbildungsverzeichnis

2.1.	RV Kontinuum	5
2.2.	Snapchat AR	6
2.3.	Barcode Marker	9
2.4.	Android Architektur	11
2.5.	Lebenszyklus einer Aktivität	12
2.6.	OpenGL Rendering Pipeline	14
2.7.	OpenGL Triangle	15
3.1.	Komplexitätskontinuum von AR-Objekten	17
3.2.	ARToolKitX Architektur	27
3.3.	Atlas der Humananatomie	28
4.1.	Use Cases des Prototyps	32
5.1.	UI Mockup	40
6.1.	Markererstellung	48
6.2.	Obj-Dateiformat	50
6.3.	Testaufbau	54
6.4.	Perspektiven eines Markers	55
6.5.	Skalierungen eines Markers	55
6.6.	Rotationen eines Markers	55
6.7.	Belichtungen eines Markers	56

Tabellenverzeichnis

Abkürzungsverzeichnis

AR	Augmented Reality
VR	Virtual Reality
API	Application Programming Interface
ART	Android Runtime
IDE	Integrated Development Environment
IDE	Identifikator
SDK	Software Development Kit
HMD	Head-Mounted-Display
HHD	Hand-Hold-Display
GLSL	OpenGL Shading Language

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit der Entwicklung einer Augmented Reality Anwendung für den Bildungsbereich. Im folgenden wird die Thematik und die daraus resultierende Forschungsfrage genauer beleuchtet. Des weiteren werden die Ziele, sowie das Vorgehen der Arbeit definiert. Abschließend folgt ein Überblick über den Aufbau der Arbeit.

1.1. Motivation

In der heutigen Zeit werden immer neuere Methoden zur Informationsbeschaffung und -darstellung entwickelt, bei denen sich oft die Frage stellt, welche praktische Relevanz diese Verfahren besitzen.

Augmented Reality stellt eine dieser Technologien dar, die momentan immer mehr an Relevanz gewinnt. Sie bildet den Oberbegriff für die Erweiterung der Realität mit virtuellen Objekten (vgl. Kapitel 2.1).

Begünstigt wird diese Entwicklung vor allem durch den technischen Fortschritt im Bereich der Mobilgeräte. Diese ermöglichen Privatpersonen im alltäglichen Leben den Zugriff auf eine enorme Rechenleistungen und öffnen somit die Tür zu neuen Technologie, wie der Augmented Reality.

Eines der wohl verbreitetsten Anwendungsgebiete stellt dabei der Entertainmentbereich dar. Hier wird die Technologie vor allem im Bereich der Spielentwicklung für Mobilgeräte genutzt. Die grundlegenden Technologien und Methoden lassen sich jedoch auch auf andere Bereiche übertragen.

Im (Hoch-)Schulumfeld kann Augmented Reality eine neue Möglichkeit zur Visualisierung von Lerninhalten bieten, deren Vorteil unter anderem darin liegt, zweidimensionale Abbildungen durch dreidimensionale Modelle zu ersetzen.

So ist es mit Augmented Reality möglich den Lernenden einen weiteren Zugang zu den Lerninhalten zu bieten.

Solche alternativen, digitalen Lernmethoden werden in der heutigen Zeit immer wichtiger. Auch das 2019 ausgebrochene Covid-19 führt dazu, dass die Lehre zuneh-

mend ins Digitale verschoben und neue Lernmethoden, speziell für die Online-Lehre, gefunden werden müssen.

Vor diesem Hintergrund wurde die folgende Forschungsfrage aufgestellt:

„Wie lassen sich die Konzepte der Augmented Reality für den Bildungsbereich nutzen?“

Dabei beschäftigt sich diese Arbeit sowohl mit den konkreten Anwendungsfällen von Augmented Reality, als auch mit der Thematik, wie sich die Technologien und Verfahren der Augmented Reality für den Bildungsbereich nutzen lassen.

1.2. Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung einer Augmented Reality Anwendung für den Bildungsbereich.

Im Rahmen dieser Zielsetzung sollen bestehende Technologien, Methoden und Ideen zur Umsetzung von Augmented Reality im Bildungsbereich erfasst und evaluiert werden.

Auf dieser Grundlage soll dann eine eigenes Konzept für eine AR-Anwendung entworfen und die Realisierbarkeit durch eine prototypische Umsetzung validiert werden.

Der Prototyp soll dabei auf die grundlegenden Funktionen der Augmented Reality aufbauen und im Anschluss bezüglich der Einsetzbarkeit in der Praxis evaluiert werden.

1.3. Methodik

Diese Arbeit verfolgt die typischen Phasen der Softwareentwicklung, bestehend aus Anforderungsentwicklung, Entwurf, Implementierung und Evaluation.

Dazu wurde in einem ersten Schritt eine qualitative Untersuchung der Einsatzmöglichkeiten durchgeführt. Diese beruhte auf einer Literaturrecherche, die sich auf die bestehenden Einsatzmöglichkeiten, sowie der Vorteile für den Bildungsbereich, konzentrierte.

Bei der betrachteten Literatur beruhte zum einen auf dem Sortiment des „Oldenburgerischen Regionalen Bibliotheks- und Informationssystems (ORBISplus)“ und zum Anderen auf den Ergebnisse einer Stichwortsuche im wissenschaftlichen Suchportal Google Scholar. Insgesamt wurden sowohl wissenschaftliche Bücher, als auch Studien, und konkrete Anwendungen herangezogen.

Auf dieser Basis wurde im Rahmen der Anforderungsanalyse eine Anforderungsfall für den Einsatz von Augmented Reality im Bildungsbereich entwickelt. Mit Hilfe der zugehörigen Use Cases und den Ergebnissen der Literaturrecherche konnten

dann die Anforderungen an den Prototypen abgeleitet werden. Diese wurden in funktionale und nicht-funktionale Anforderungen unterteilt. Zu dem wurde auf eine präzise Formulierung und eine Überprüfbarkeit der Anforderungen Wert gelegt.

Während der Implementierung wurde auf einen zyklischen Testprozess zu Validierung und Dokumentierung der Ergebnisse zurückgegriffen. Dieser sah einen Test nach der erfolgreichen Implementierung einer neuen Funktion vor, um diese zu testen und eine Vergleichbarkeit zu älteren Versionen herzustellen. Dadurch sollten mögliche Mängel und Probleme durch neue Features frühzeitig erkannt werden.

Vorgehen bei Entwurf?

Die Evaluierung des Prototyps wurde in zwei Teile unterteilt. Im ersten Schritt wurde die Anwendung anhand der gestellten Anforderungen evaluiert. In diesem Schritt ging es darum zu testen, ob die Anforderungen erfüllt wurden und ob sich das Konzept eventuell im Laufe der Entwicklung geändert hat.

Der zweite Schritt bestand aus einer Usability-Evaluation mit ausgewählten Nutzern. Dabei sollte zum einen die Umsetzung, sowie der Einsatz der durch den Prototyp realisierten Funktionen in der Praxis getestet werden.

Methode und Vorgehen - Interview, Fragebogen

1.4. Aufbau der Arbeit

Insgesamt besteht diese Arbeit aus 8 Kapiteln.

Zu Beginn dieser Arbeit werden in dem Kapitel 2 die Grundlagen dieser Arbeit beleuchtet. Dabei geht das Kapitel auf die grundlegenden Prinzipien von OpenGL, der Augmented Reality und der Android App Entwicklung ein.

Anschließend folgt das Kapitel 3 in dem verwandte Arbeiten zum Thema dieser Arbeit beschrieben werden.

Diese beiden Kapitel bilden die Grundlage für die folgende Entwicklung des Prototyps. Im Kapitel 4 werden die Anforderungen für diesen erhoben und der Anwendungsfall spezifiziert.

Im folgenden Kapitel 5 werden aus den Anforderungen erste Designentscheidungen abgeleitet und ein Entwurf für die Implementierung entwickelt.

Die Konkrete Implementierung wird in Kapitel 6 behandelt. Dabei werden die einzelnen, konkreten Klassen, deren Methoden, sowie die Beziehungen zwischen den Klassen beschrieben.

Im Kapitel 7 folgt eine Evaluation des Prototyps. Dazu werden zunächst die in Kapitel 4 erhobenen Anforderungen betrachtet und die Realisierung durch den Prototypen geprüft. Im Anschluss wird die durchgeföhrte Usability-Evaluation und deren Ergebnisse beschrieben.

Das abschließende Kapitel 8 gibt eine inhaltliche Zusammenfassung über die Ergebnisse dieser Arbeit und einen Ausblick auf weiterführende Aufgaben. Des weiteren wird in diesem Kapitel auch eine kritische Reflexion des Verlaufs und der Ergebnisse

dieser Arbeit abgeben.

Kapitel 2

Grundlagen

Dieses Kapitel behandelt die Grundlagen, die für die Realisierung des Prototyps notwendig sind. Dabei werden Methoden und Eigenschaften der Augmented Reality erläutert, sowie ein Überblick über OpenGL gegeben.

2.1. Augmented Reality

In der Literatur lassen sich für den Begriff der Augmented Reality (AR, deutsch: „angereicherte Realität“) viele unterschiedliche Definitionen finden, die meisten stützen sich dabei auf das von Milgram u. a. (1994) definierte Reality-Virtuality (RV) Kontinuum, welches in Abbildung 2.1 dargestellt ist.

Um dieses zu verstehen muss zunächst der Begriff der Virtual Reality (VR, deutsch: „virtuelle Realität“) definiert werden. Nach Klein (2006, S.1) beschreibt die VR eine völlig künstliche, computergenerierte Welt, in die der Nutzer eintauchen kann.

Milgrams Definition fasst nun die Virtual Reality und die reale Welt als zwei, sich gegenüberliegende Enden eines Kontinuums auf. Dabei ist die reale Welt an die physikalischen Gesetze gebunden, während die virtuelle Welt diese überschreiten und sich von ihnen lösen kann (Milgram u. a., 1994, S. 283). Nach dem RV Kontinuum bewegt sich die Augmented Reality zwischen beiden Welten und stellt ein Kombination beider dar. Eine weitere Definition, die sich auch mit der von Milgram

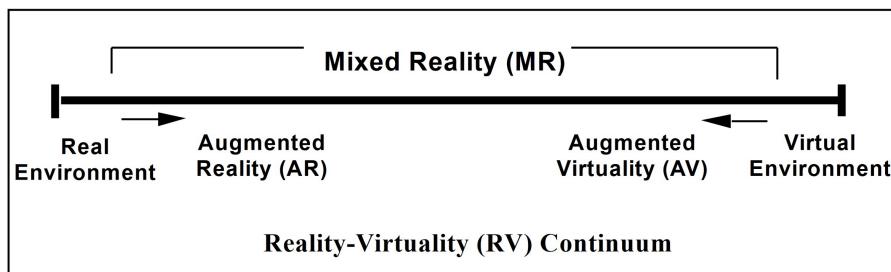


Abbildung 2.1.: Das Reality-Virtuality (RV) Kontinuum. (Quelle: Milgram u. a. (1994, S. 283))

vereinbaren lässt, beschreibt die Augmented Reality als eine computergestützte Erweiterung der wahrnehmbaren Realität um virtuelle Objekte (Kind u. a., 2019, S. 9). Auf dieser Grundlage kann man Augmented Reality als das Einbinden und Visualisieren digitaler, computergenerierte Objekte in der realen Welt auffassen. Oftmals wird dabei das Ziel verfolgt eine möglichst realistische Illusion für den Nutzer zu schaffen.

2.1.1. Einsatzbereiche

Die erste Assoziation, die die meisten mit dem Begriff Augmented Reality verbinden ist vermutlich der Unterhaltungsbereich. Große Firmen wie zum Beispiel Snapchat nutzen die Technologie um kleine Gimmicks für ihre Nutzer bereitzustellen (siehe Abbildung 2.2) Die meisten Personen, die den Begriff Augmented Reality hören, werden vermutlich an ein lustiges Gimmick zur Unterhaltung denken. Doch auch neben dem Bereich der Unterhaltung wird AR an vielen weiteren Stellen eingesetzt. Beispiellohaft zu nennen wären hier der Bereich der Produktion, in welchem AR unter Anderem als Hilfsmittel zum Prototyping (Kind u. a., 2019, S. 44) genutzt werden kann, oder der Bereich der Medizin. Im letzteren können mittels AR Therapiemaßnahmen für psychische Erkrankungen oder Assistenzsysteme zur Diagnose und Operation entwickelt werden (Kind u. a., 2019, S. 52, 54).

Der Einsatzbereich in dessen Rahmen sich diese Arbeit bewegt ist jedoch der Bereich der Bildung.

Hier bietet Augmented Reality die Möglichkeit eines neuen Informationsmediums, welches vor allem zur Betrachtung dreidimensionaler Objekte genutzt werden kann. Dieses ermöglicht ein verbessertes, räumliches Verständnis des Lerninhaltes. Laut einer systematischen Analyse der Universität Stockholm, in welchem basierend auf der Anzahl der wissenschaftlichen Veröffentlichungen Schlüsse für das Lernen mit AR gezogen wurden, sind vor allem die Naturwissenschaften relevant für den Einsatz von AR (Hedberg u. a., 2018, S. 81).



Abbildung 2.2.: AR Gimmick aus der Anwendung Snapchat. (Quelle: Screenshot aus der Anwendung Snapchat, 01.08.2020)

2.1.2. Technische Grundlagen

Zur Umsetzung der Augmented Reality ist eine Umgebungserfassung und -analyse des Systems mit Hilfe einer Tracking Software (auch Tracker genannt) notwendig. Auf Grundlage dieser können dann im Anschluss computergenerierte virtuelle Objekte in die Umgebung eingefügt werden.

Zur Analyse der Systemumgebung können verschiedene Eingabesysteme genutzt werden, mit deren Hilfe die Eigenschaften der Umgebung und der in ihr vorhandenen Objekte wahrgenommen werden können (Kind u. a., 2019, S. 22). Zu diesen Eigenschaften zählen neben statischen, wie der Größen und Position, auch dynamische Eigenschaften, welche die Veränderung der statischen Attribute einzelner Objekte umfassen. Beispielhafte Technologien, die zur Erfassung genutzt werden können, wären Kamerasysteme, Laser, Infrarot oder sonstige Sensoren (Kind u. a., 2019, S. 22). Neben der Umgebungserfassung ist auch die Verfolgung einzelner, in der Umgebung enthaltender Objekte notwendiger Bestandteil der Tracking Software, um die dynamischen Eigenschaften der realen Umgebung auf die virtuellen Objekte zu übertragen.

Umformulieren,
wegen Zitat

Eine wichtige Rolle beim Tracking spielt die Genauigkeit, mit der die Eigenschaften der Umgebung wahrgenommen werden, sie bestimmt wie akkurat virtuelle Objekte in die reale Umgebung eingefügt werden können und wie realistisch die Illusion erscheint (Klein, 2006, S. 2).

Ausgabegeräte

Grundsätzlich kann bei der Umsetzung von Augmented Reality eine Kategorisierung der Technologie durch die Art der Endgeräte vorgenommen werden. Dabei stehen vor allem die folgenden zwei Varianten zur Verfügung:

HMDs: Ein Head-Mounted-Display (HMD, deutsch: „am Kopf befestigter Bildschirm“) beschreibt eine Technologie, bei der die Sensorik, sowie das Ausgabegerät am Kopf des Nutzers befestigt wird (Mehler-Bicher und Steiger, 2014, S. 44). Populäre Umsetzungen dieser Technologie stellen spezielle Brillen dar, wie zum Beispiel die Google Glasses.

HHDs: Zu der Kategorie Hand-Hold-Display (HHD, deutsch: „Handhalteanzeige“) gehören Technologien, die durch den Nutzer in der Hand gehalten werden. Dazu gehören beispielsweise auch Smartphones auf denen AR-Anwendungen laufen.

Alternativ können die Sensoren und das Ausgabegerät auch fest in einem stationären Gerät montiert sein (Mehler-Bicher und Steiger, 2014, S. 27) oder Mischtechniken eingesetzt werden.

Trackingverfahren

Grundsätzlich kann bei der Tracking Software zwischen zwei Verfahren unterscheiden werden, dem nicht visuellen und dem visuellen Tracking (Mehler-Bicher und Steiger, 2014, S. 26). Während bei letzterem auf Daten von zum Beispiel einer Kamera zurückgegriffen wird, beruht das nichtvisuelle Tracking auf Sensoren, die direkten Zugriff auf die Eigenschaften der Umgebung, wie der Position, liefern.

Bei dem für diese Arbeit relevantem visuellen Tracking, müssen die Eigenschaften der Umgebung aus dem Kamerabild abgeleitet und verarbeitet werden. Dazu werden in der Regel nach Mehler-Bicher und Steiger (2014, S. 26) zwei Schritte benötigt:

1. Die Initialisierung, bei welcher nach einem bestimmten Muster im Kamerabild gesucht wird. Dieses kann dabei im Vorfeld definiert sein oder aus dem Kamerabild abgeleitet werden.
2. Die Verfolgung bzw. Antizipation der möglichen Bewegung, bei welcher das gefundene Muster in den einzelnen, aufeinanderfolgenden Videoframes verfolgt wird und eine Prognose der zukünftigen Position des Muster berechnet wird, um den Rechenaufwand zu verkleinern.

Des Weiteren kann beim Visuellen Tracking zwischen Marker Based Feature Tracking und Natural Feature Tracking unterschieden werden.

Marker Based Feature Tracking verwendet feste, im Vorfeld definierte Muster, die in der Umgebung platziert werden. Diese Muster werden als Marker bezeichnet. Meistens handelt es sich dabei um schwarze Quadrate, in deren Mitte eine ID als ein Muster aus schwarzen und weißen Vierecken codiert ist. Ein beispielhafter Marker ist in Abbildung 2.3 zusehen. Die Marker sind dabei optisch so angepasst, dass sie sehr einfach von einem Tracker erfasst werden können, um die Erkennungsgeschwindigkeit und somit die Performanz des gesamten Systems zu optimieren (Mehler-Bicher und Steiger, 2014, S. 28).

Owen u. a. stellen dazu folgende Anforderungen an einen optimalen Marker:

- Ein idealer Marker sollte die eindeutige Bestimmung seiner Position und Orientierung im Verhältnis zur Kamera unterstützen.
- Der Marker sollte alle Ausrichtungen unterstützen.
- Der Marker sollte Teil einer Reihe an Markern sein, die sich eindeutig von einander unterscheiden lassen.
- Ein Marker muss einfach lokalisierbar und identifizierbar sein.
- Die Marker müssen über einen weiten Aufnahmebereich funktionieren

(Nach Owen u. a. (2002, S. 2))

Durch diese Eigenschaften ist die Erkennung von einem Marker relativ simpel und funktioniert im Allgemeinen in drei Schritten:

1. Zunächst werden müssen die Kanten aus dem Bild extrahiert werden, um mit deren Hilfe alle Vierecke aus dem Bild zu filtern.
2. Dann kann mittels Template Matching oder ähnlichen Verfahren der Marker erkannt werden und die ID, falls vorhanden, dekodiert werden.
3. Im letzten Schritt muss dann noch die Position, Größe und Orientierung des Markers berechnet werden.

(In Anlehnung an Ćuković u. a. (2015))



Abbildung 2.3.: Ein Barcode Marker für ARToolKit mit der ID 10.
(Quelle: Generiert mit dem Marker Generator <https://augmented.com/app/marker/marker.php>)

link ins Literaturverzeichnis?

Natural Feature Tracking greift hingegen auf sogenannte Keypoints zurück, die natürlich im Bild enthalten sind. Keypoints beschreiben dabei markante Bildpunkte, die sich durch zum Beispiel starke Helligkeitsveränderungen in der direkten Nachbarschaft auszeichnen, wie es unter anderem bei Kanten oder Ecken der Fall ist. Zur Extraktion dieser Keypoints gibt es verschiedene Verfahren wie den SIFT-Algorithmus (Scale-Invariant-Feature-Tracking), welcher die Grundlage für viele weitere aktuelle Verfahren bildet und Merkmale extrahiert, die invariant gegenüber der Rotation, Translation, Skalierung und partiell invariant gegenüber Helligkeitsveränderungen sind (Nischwitz u. a., 2011, S. 345). Diese Invarianzen sind dabei auch für den Bereich der Augmented Reality sehr wichtig, da die virtuellen Objekte auch bei einer sich verändernden Umgebung korrekt eingebunden werden sollen.

Bei diesem Verfahren müssen in einem ersten Schritt zunächst die Keypoints für den Bildbereich, in dem das Objekt platziert werden soll, extrahiert und gespeichert werden. Diese Bildpunkte können dann mit Hilfe von Matchingverfahren in den folgenden Frames gefunden werden. Bei einer Übereinstimmung muss dann folglich noch die Position, Größe und Orientierung des Bildbereiches berechnet werden.

Eine Verallgemeinerung des gesamten Tracking Prozesses kann dabei wie folgt beschrieben werden:

Nachdem das System einen Frame von der Kamera bekommt wird dieser meist vorverarbeitet, um die wichtigsten Informationen zu extrahieren und den Tracking Prozess zu beschleunigen (Ćuković u. a., 2015, S. 5). Eine Möglichkeit ist dabei das Bild in ein Graustufenbild umzuwandeln. Dadurch lassen sich Helligkeitsveränderungen leichter detektieren und die Anzahl der Farbkanäle wird von drei oder mehr auf einen reduziert, wodurch die Geschwindigkeit der Tracking Algorithmen deutlich erhöht werden kann.

Nach der Vorverarbeitung kann dann mittels Natural Feature Tracking oder Marker Based Feature Tracking die Position, Größe und Orientierung der gesuchten Fläche, beziehungsweise des Markers ermittelt werden und das virtuelle Objekt entsprechend transformiert werden.

2.2. Android Entwicklung

Android ist ein Open Source Software Plattform, die vor allem bei Smartphones und anderen mobilen Geräten zum Einsatz kommt und ein Produkt der von Google gegründeten Open Handset Alliance ist (Gargenta, 2011, S. 4). Bekannt ist vor allem das Betriebssystem Android, das auf vielen Geräten ihren Einstaz findet.

Als Programmiersprache zur Entwicklung von Android Anwendungen stehen Kotlin, Java und C++ zur Verfügung Android (a).

2.2.1. Architektur

Abbildung 2.4 zeigt eine Übersicht der Software Architektur von Android.

Die Grundlage eines Android Systems bildet dabei der Linux-Kernel, der es Android unter anderem erlaubt auf die Sicherheitsfunktionen des Kernels zuzugreifen (Android, e).

Die Hardware Abstraktionschicht (HAL) stellt dem übergeordneten Java-API-Framework Standardschnittstellen zur Verfügung, über die eine Anwendung auf die Hardwarefunktionen des Gerätes zugreifen kann (Android, e). Übergeordnet finden sich die Laufzeitumgebung Android Runtime (ART) und nativen Bibliotheken auf. Bei letzteren handelt es sich um Bibliotheken, welche in C und C++ geschrieben sind (Android, e) und dem Entwickler unterschiedliche Funktionen zur Verfügung stellen. Die in C oder C++ geschriebenen Bibliotheken sind vor allem aus Geschwindigkeitsaspekten relevant.

Die Entwicklung von Android Anwendungen erfolgt dabei über das Java-API-Framework, welches dem Entwickler ein Grundgerüst sowie alle wichtigen Funktionen und An-

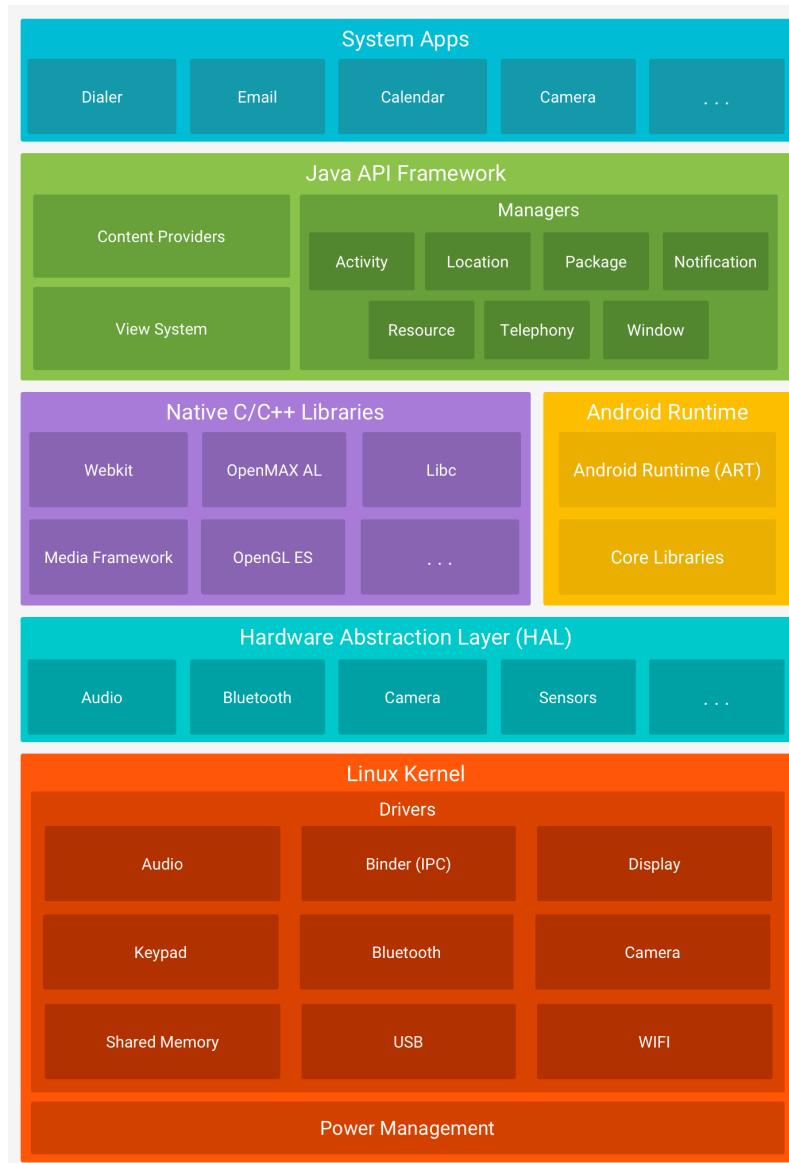


Abbildung 2.4.: Die Android Architektur. (Quelle: Android (e))

wendungsbausteine die Für die Entwicklung relevant sind zur Verfügung stellt (Android, e).

Die oberste Schicht stellen die System Anwendungen dar, bei welchen es sich um Standard-Anwendungen handelt, die Android dem Nutzer zur Verfügung stellt.

2.2.2. Grundlegende Konzepte

Alle Android Anwendungen basieren auf den selben Komponenten und Konzepten, die im folgenden einmal grundlegend betrachtet werden. Diese Komponenten sind als Klassen in Android verankert und können mittels Vererbung in Projekte eingebunden und angepasst werden

Activities

Activities (deutsch: „Aktivitäten“) stellen die Grundlage jeder Android Anwendung dar und werden durch die Klasse Activity implementiert. Dabei erzeugt jede Activity ein Fenster der Anwendung und ersetzt gleichzeitig die aus der Programmierung mit unter anderem Java bekannte main() Methode (Android, c). Anstelle einer main() Methode werden verschiedene callback Methoden aufgerufen, die einen bestimmten Zustand im Lebenszyklus einer Activity repräsentieren (Android, c). Der gesamte Lebenszyklus einer solchen Activity wird in Abbildung 2.5 dargestellt.

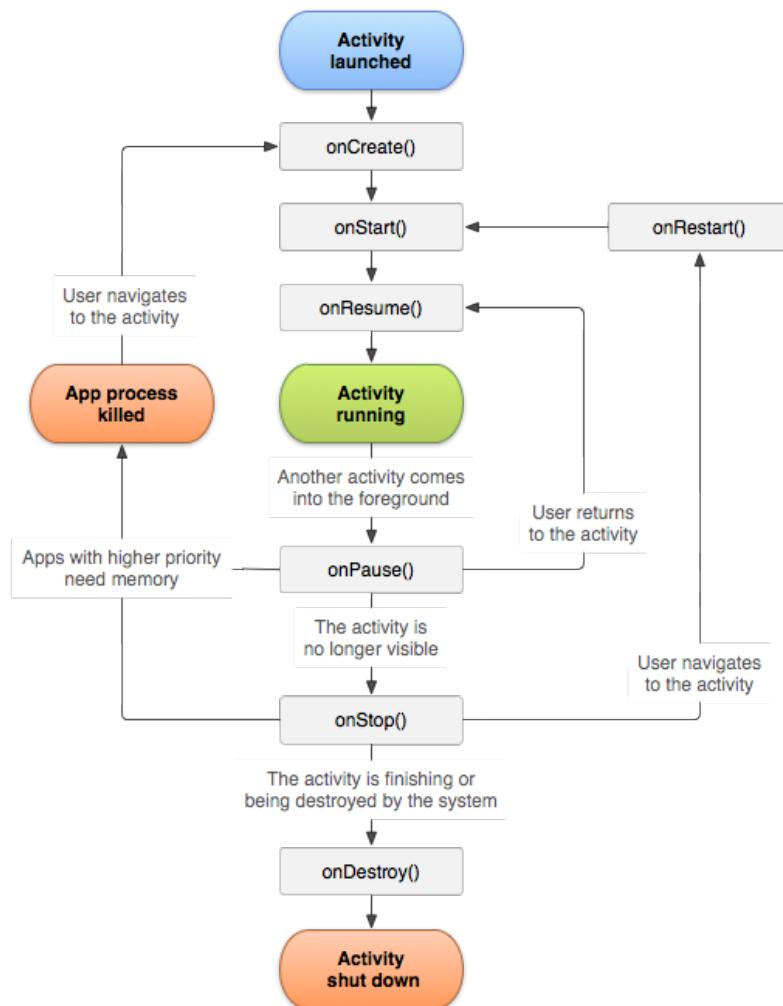


Abbildung 2.5.: Der Lebenszyklus einer Aktivität. (Quelle: Android (f))

Services

Ein Service (deutsch: „Dienstleistung“) ist eine Komponente die im Hintergrund läuft und dazu konzipiert wurde langfristige Operationen auszuführen (Android, a). Services können dabei unabhängig jeglicher Aktivitäten im Hintergrund laufen, auch

wenn die Anwendung geschlossen wurde (Murphy, 2009).

Content Providers

Content Providers (deutsch: „Inhaltsanbieter“) stellen ein Abstraktionslevel für Daten dar, auf die aus mehreren Anwendungen zugegriffen wird (Murphy, 2009). Mit ihrer Hilfe ist es möglich eigene Daten anderen Anwendungen zur Verfügung zu stellen (Murphy, 2009).

Dabei wird das Ziel verfolgt das Zusammenspiel zwischen verschiedenen Anwendungen zu verbessern, und den Datenaustausch zwischen ihnen zu verbessern.

Intents

Bei einem Intent (deutsch: „Absicht“) handelt es sich um eine asynchrone Systemnachricht, die dazu genutzt werden kann bestimmte Komponenten oder bestimmte Komponentenarten zu aktivieren (Android, a).

Fragments

Fragments (deutsch: „Fragmente“) stellen einzelne Teile des User Interfaces (deutsch: „Benutzeroberfläche“) einer Activity dar (Android, b). Sie können in die bestehende Oberfläche der Activity eingefügt werden und durch andere Fragments ausgetauscht werden.

2.3. OpenGL

OpenGL stellt die industriell meistgenutzte Programmierschnittselle (API) zur Entwicklung von 2D und 3D Anwendungen dar (Khronos Group, b). Mit Hilfe des Interfaces lassen sich komplexe dreidimensionale Objekte darstellen.

2.3.1. Grundlagen von OpenGL

Alle Objekte werden in OpenGL aus einem oder mehreren Polygons zusammengesetzt (Shreiner u. a., 2006, S. 5).

Dabei wird jedes Poligon durch eine Menge an Eckpunkten (Vertices) definiert und jeder Eckpunkt(Vertex) stellt dabei eine Menge an Daten für einen bestimmten Punkt in einem dreidimensionalen Koordinatensystem dar (de Vries, J.).

2.3.2. Rendering-Pipeline

Die folgende Abbildung 2.6 zeigt die Rendering-Pipeline von OpenGL. Alle Objekte und Modelle, die mit Hilfe von OpenGL dargestellt werden, durchlaufen diese Pipe-

line.

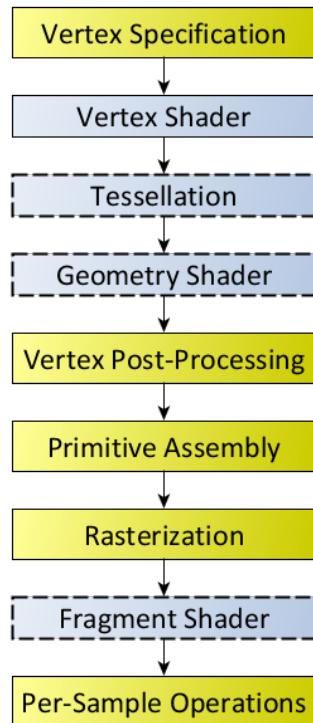


Abbildung 2.6.: Die Rendering-Pipeline von OpenGL. (Quelle: Khronos Group (c))

Die Objekte werden dabei durch eine Menge an Vertices definiert. Die zwei wichtigsten Eingriffspunkte in der Rendering-Pipeline sind der sec:vertex-shader und der sec:fragment-shader. Beide können von Programmierer implementiert werden und dienen dem Zweck verschiedene Operationen auf den einzelnen Vertices beziehungsweise den Pixeln durchzuführen. Um beide nutzen zu können müssen sie mit einem Shader Programm verknüpft werden, welches den Output von jedem Shader mit dem Input des nächsten Shaders verbindet (de Vries, J.).

Im folgenden werden einmal die einzelnen Schritte der Pipeline genauer betrachtet. Bevor das Objekt dargestellt werden kann muss es zunächst definiert werden. Dieser Schritt wird als **Vertex Specification** bezeichnet. Dazu muss das Modell in viele einzelne Dreiecke unterteilt werden, welche zusammen das gesamte Objekt bilden. Diese einzelnen Flächen werden als Grundelemente (Primitives) bezeichnet (Khronos Group, c) und können durch jeweils drei Eckpunkte (Vertices) beschrieben werden (siehe Abbildung 2.7).

Dabei können jedem Vertex neben der Position, auch weitere Eigenschaften, welche für die Berechnungen in den nächsten Schritten der Rendering-Pipeline notwendig sind, wie Texturkoordinaten zugeordnet werden.

Alle Informationen werden in einer Listenstruktur gespeichert und werden im Anschluss an den **Vertex Shader** übergeben. Dieser verarbeitet jeden einzelnen Eckpunkt, indem er bestimmte Operationen auf diesem durchführt bevor er an den

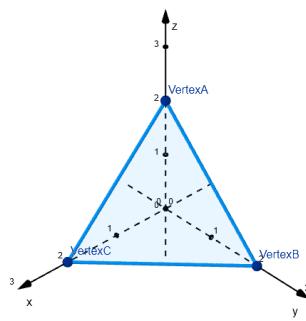


Abbildung 2.7.: Zusammensetzung eines einzelnen Dreiecks. (Quelle: Eigene Darstellung)

nächsten Schritt der Rendering Pipeline weitergegeben wird. (de Vries, J.). Dabei können einzelne Attribute des Punktes transformiert werden, während andere einfach nur weitergeleitet werden.

Die **Tessellation** stellt einen weiteren, optionalen Schritt in der Pipeline dar. Er arbeitet mit einer Menge an Vertices, die als Patch bezeichnet wird und erzeugt aus diesen kleinere Grundelemente (Khronos Group, d). Mit Hilfe dieser Zerkleinerung können Geometrien, wie zum Beispiel Rundungen verstärkt werden.

Der nächste Schritt in der Rendering Pipeline ist der **Geometry Shader**, welcher ebenfalls eine optionale Station darstellt und vor allem für das Layered Rendering, bei welchem ein Grundelement in mehreren Bildern gerendert wird, genutzt werden kann (Khronos Group, a). Im Anschluss folgt das **Vertex Post-Progressing**, welches aus einer Reihe fester Funktionen besteht (Khronos Group, c), die im Kontext dieser Arbeit nicht zu brachten sind. Der nächste Schritt ist das **Primitive Assembly** bei welchem aus der Reihe an Vertices eine geordnete Sequenz an Grundelementen geformt wird (Khronos Group, c). Während der **Rasterization** werden dann die Grundelemente in eine Pixeldarstellung für den entsprechenden Bildschirm umgewandelt (de Vries, J.).

Der **Fragment Shader** ordnet dann jedem Pixel eine Farbe zu, in deren Berechnung meist Werte wie Schatten, Licht und dessen Farbe einfließen (de Vries, J.). Zu letzt werden dann noch eine Reihe an **Per-Sample Operations** durchgeführt, welche Tests beschreiben die testen, ob ein Pixelwert aktualisiert werden muss oder nicht (Khronos Group, c).

2.3.3. OpenGL ES

OpenGL ES ist eine OpenGL Version speziell für eingebettete Systeme (Android, d). Sie erlaubt es bei der Android Programmierung auf die OpenGL Funktionalitäten zuzugreifen.

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten betrachtet, die sich mit der Forschungsfrage „Wie lassen sich die Konzepte der Augmented Reality für den Bildungsbereich nutzen?“ oder einer angrenzenden Thematik beschäftigen. Dabei wird sowohl auf allgemeine Literatur, Studien, wissenschaftliche Arbeiten, als auch konkrete Anwendungen eingegangen und ihr Inhalt zusammengefasst.

3.1. Augmented Reality in der Bildung

Im folgenden werden Veröffentlichungen betrachtet, die sich mit dem Einsatz von Augmented Reality beschäftigen. Zum Teil beleuchteten die Arbeiten konkrete Einsatzmöglichkeiten von Augmented Reality, zum Anderen werden die Vorteile von Augmented Reality oder Richtlinien für AR-Anwendungen betrachtet.

3.1.1. Geroimenko: Augmented Reality in Education

Das Buch (Geroimenko, 2020) von Geroimenko beschäftigt sich mit dem Einsatz von Augmented Reality im Bildungsbereich. Dabei wird zum einen auf den aktuellen Stand und aktuelle Einsatzbereiche eingegangen, zum Anderen wird aber auch die Konkrete Entwicklung von AR-Anwendungen für den Bildungsbereich thematisiert. Im folgenden werden einmal die Grundaussagen des Buches zusammengefasst:

Aktueller Stand

Smartphones stellen aufgrund des Vorhandenseins verschiedener Software Development Kits (SDK) und ihrer Allgegenwart die aktuelle Hauptplattform für Augmented Reality dar. Während Head Mounted Displays vor allem an kleinere Zielgruppen und den professionellen Markt richten. (Geroimenko, 2020, Kapitel 1.2)

Generell kann bei AR-Objekten zwischen fundamentalen und generativen AR-Lerneinheiten unterschieden werden, die jeweils eine Ende eines Kontinuums darstellen (vergleiche Abbildung 3.1). Der ausschlaggebende Faktor ist dabei die Kom-

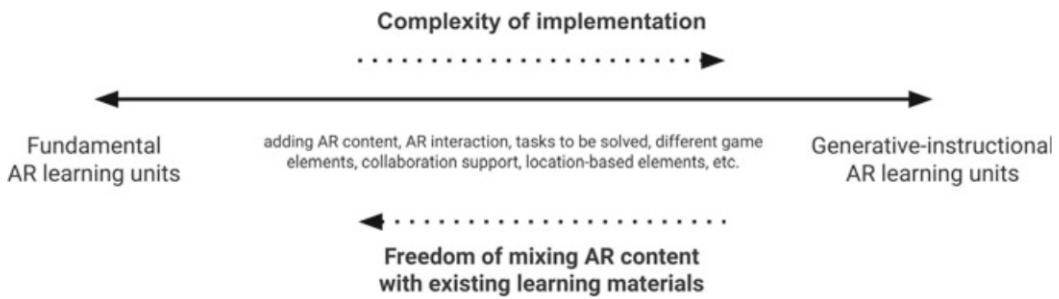


Abbildung 3.1.: Das Komplexitätskontinuum von AR-Objekten. (Quelle: (Geroimenko, 2020, S. 9))

plexität des Objektes, je höher diese ist desto generativer ist die Lerneinheit, aber umso geringer ist die Möglichkeit die AR-Inhalte mit den existierenden Lerninhalten zu verbinden. Generative AR-Lerneinheiten stehen dementsprechend mehr für sich alleine und decken ein größeres inhaltliches Feld ab. Der Zugriff auf diese fundamentalen AR-Inhalte ist heutzutage bereits sehr einfach und die Entwicklung in der Thematik wird sich vermutlich ähnlich analog zu der Entwicklung der VR-Inhalte, bei welcher die generative Inhalte nach und nach hinzukamen, verhalten.(Geroimenko, 2020, Kapitel 1.3)

Die aktuelle Hauptzielgruppe im schulischen Bereich ist dabei die weiterführende Schule. Außerhalb des schulischen Umfeldes kann AR zur Ausbildung von Fachkräften eingesetzt werden. (Geroimenko, 2020, Kapitel 1.5)

Entwicklung von Augmented Reality Anwendungen

Vor der Entwicklung einer Augmented Reality Anwendung für den Bildungsbereich, sollten folgende Entscheidungen getroffen werden:

1. Die erste Entscheidung, die bei der Entwicklung getroffen werden muss bezieht sich auf die eingesetzte Technologie. Dabei muss überlegt werden, welche Geräte die Schüler bereits besitzen und welche alternativ angeschafft werden können. Des weiteren muss bedacht werden, dass auch die Lehrenden im Umgang mit den neuen Technologie geschult werden müssen.
2. Als zweites muss überlegt werden, was durch den Einsatz der Anwendung erreicht werden soll. Dabei muss grundsätzlich zwischen fundamentalen Lerneinheiten, die angeschaudt und vom Nutzer manipuliert werden können, und generativen, instruktiven Lerneinheiten, welche verschiedenen Interaktionsmöglichkeiten, Spielemente, Beteiligungsformen und vieles mehr enthalten, abgewogen werden.
3. Des weiteren muss bedacht werden, dass für die Erstellung von AR-Lernobjekten unter Umständen komplexe Fähigkeiten notwendig sind. Während einfache,

dreidimensionale Objekte einfach anhand von Bildern realer Objekte generiert werden können, ist die Erstellung komplexer Lerninhalte aufgrund mangelnder Werkzeuge sehr aufwendig und erfordert weitere Spezialkräfte.

4. Der letzte Punkt ist die Zielgruppe. Hier muss zwischen einem breitem Feld an Einsatzgebieten, wie Schulen, Museen und vielem mehr, abgewogen werden, da sich die Anforderungen der Teilgebiete stark unterscheiden können.

(Geroimenko, 2020, Kapitel 1.7)

Augmented Reality in der Lehre

In der Lehre kann Augmented Reality ein breites Feld bedienen. Ein Einsatzbereich stellen dabei die Naturwissenschaften und die Medizin dar. Besonders Letzteres kann von Augmented Reality in den folgenden Bereichen profitieren:

- Anatomie: Hierbei kann AR von Medizinstudenten dazu genutzt werden die Anatomie des Körpers besser zu verstehen, indem sie 3D Modelle im AR-Bereich betrachten können.
- Mentoring: Mit Hilfe von AR ist es möglich das Studenten Prozeduren kennenzulernen und erfahren, in denen sie noch nicht geschult sind.
- Klinische Fertigkeiten: Auch die klinischen Fähigkeiten der Studenten können verbessert werden, indem mit der Hilfe von Augmented Reality klinische Simulationen durchgeführt werden.

(Geroimenko, 2020, Kapitel 7-9)

Augmented Reality kann zudem für die Umweltbildung im Freien genutzt werden. Dabei bietet AR die Möglichkeit die Komplexität der Umgebung aus verschiedenen Perspektiven zu visualisieren, detaillierte und wissenschaftliche Informationen anzuzeigen oder das „unsichtbare“ sichtbar zumachen. Dabei ermöglicht es dem Lernenden eine andere Perspektive auf das Sichtbare, in dem es Parameter, wie die Zeit (Vergangenheit, Zukunft), die Skalierung (mikroskopische Sicht, Vogelperspektive) oder die Wahrnehmbarkeit (Unsichtbares enthüllen, Sichtbares verdecken), verändert. (Geroimenko, 2020, Kapitel 17)

Auch in Bereichen der Archäologie kann AR als Forschungsinstrument für die Rekonstruktion und Interpretation eingesetzt werden. (Geroimenko, 2020, Kapitel 17) Ein weiterer Bereich, den Augmented Reality abdecken kann, sind die Geisteswissenschaften. Hier stellt die Erweiterung der Umgebung basierend auf der aktuellen Position des Nutzers durch AR eine wichtige Einsatzmöglichkeit dar. Dadurch lassen sich beispielsweise Zusatzinformationen zu historischen Gebäuden anzeigen und es

ist möglich dem Nutzer einen Eindruck zu vermitteln, wie ein Ort in der Vergangenheit einmal aussah. (Geroimenko, 2020, Kapitel 11-12)

Im Bereich des Lernens einer Fremdsprache, bietet AR die Möglichkeit das konventionelle Lernen im Klassenzimmer um eine virtuelle Welt erweitern, um die Motivation der Lernenden zu steigern. (Geroimenko, 2020, Kapitel 11-12)

Konkrete Anwendungsbeispiele

Das FeDiNAR-Projekt stellt ein konkretes Anwendungsbeispiel der Augmented Reality zu Ausbildung von Personen dar. Es hat das Ziel, die Fehler mit Hilfe von Augmented Reality zu einer Lernmöglichkeit umzuwandeln. Das Konzept beruht darauf Fehler nicht durch den Ausbilder zu verhindern, sondern sie zuzulassen. Dieses soll den Ausbilder möglich sein, indem die Auszubildenden zwar an echten Maschinen arbeiten, die Auswirkungen ihrer Handlungen dabei jedoch lediglich in der Augmented Reality dargestellt werden. (Geroimenko, 2020, Kapitel 5)

Ein weiteres Anwendungsbeispiel ist die im Rahmen des Buches beschriebene Entwicklung einer Online-Plattform zur verbesserten Lernerfahrung. Die Anwendung beruht auf „durchsichtige“ Markern, die in ein Bild eingefügt werden konnten. Wurde der Marker innerhalb eines Bildes erkannt wird ein dreidimensionales Modell des Bildes mit Hilfe von Augmented Reality angezeigt. (Geroimenko, 2020, Kapitel 3)

3.1.2. Hedberg: A Systematic Review of Learning through Mobile Augmented Reality

In der wissenschaftlichen Arbeit von Hedberg u. a. (Hedberg u. a., 2018) wurde mit Hilfe einer systematischen Literaturauswertung eine Einschätzung des Einsatzes von Augmented Reality zu Bildungszwecken vorgenommen.

Dabei wurde die Relevanz einzelner Einsatzgebiete anhand der Anzahl an Veröffentlichungen zum jeweiligen Thema gemessen.

Die Veröffentlichungen stammen dabei aus den Datenbanken „IEEE Xplore“, „Elsevier“ und der „ACM digital library“ und wurden über Schlüsselwörter herausgesucht. Die Ergebnisse der Suche wurden im Anschluss noch gefiltert und kategorisiert, um irrelevante Veröffentlichungen auszuschließen. Anhand der gesammelten wissenschaftlichen Arbeiten kam die Studie zu den folgenden Ergebnissen in den für diese Arbeit relevanten Kategorien:

Bildungsniveau

Diese Kategorie zielte darauf ab die Zielgruppen der AR-Systeme herauszufinden. Dabei kam die Studie zu dem Ergebnis, dass die meisten Systeme (fast 50 %) bezogen

auf das amerikanische Bildungssystem für die Universität geschaffen wurden. Mit einem deutlichen Abstand folgt die Grundschule, die Junior High School, die High School und die Vorschule. (Hedberg u. a., 2018, S. 78)

Mobiles Endgerät

In dieser Kategorie ging es darum herauszufinden, welches die meist genutzten Endgeräte für Augmented Reality im Bildungsbereich darstellen. Hierbei kam heraus, dass das Handy bzw. das Smartphone (43,84 %) gefolgt von dem Tablet (27,40 %) diese Kategorie dominieren, wobei ebenfalls 27 % der wissenschaftlichen Arbeiten keine spezifische Plattform angaben. (Hedberg u. a., 2018, S. 80)

Unterrichtsfach

Diese Kategorie untersuchte die fachliche Ausrichtung der Augmented Reality und kam zu dem Ergebnis, dass die Systeme vor allem in den Naturwissenschaften genutzt werden. Darauf folgen mit signifikantem Abstand die Sprachen, Geschichte und Technik. (Hedberg u. a., 2018, S. 81)

Lernerfolge

Hierbei wurden Studien untersucht, die sich mit dem pädagogischen Nutzen von Augmented Reality beschäftigt haben. Dabei kamen von 73 Studien lediglich zwei nicht zu dem Ergebnis, dass AR einen positiven Einfluss auf das Lernen besitzt. Allgemein stellten 45 Veröffentlichungen (54,88 %) eine erhöhte Motivation und ein verbessertes Engagement fest. 28 Studien (34,15 %) konnten verbesserte Lernergebnisse bei den Studenten aufzeigen. (Hedberg u. a., 2018, S. 81-82)

Pädagogische Methoden

In der letzten Kategorie, wurden die Einsatzarten von Augmented Reality im Bildungsbereich untersucht und das Ergebnis erarbeitet, dass die drei Hauptanwendungsmethoden das interaktive, das forschungsbasierte und das kollaborative Lernen sind. (Hedberg u. a., 2018, S. 82)

3.1.3. Billinghurst: Augmented Reality in Education

Billinghurst untersuchte in seiner wissenschaftlichen Arbeit (Billinghurst, 2002) den Einsatz von Augmented Reality im Bildungsbereich.

Dabei geht er in den folgenden drei Unterkapiteln auf unterschiedliche Eigenschaften der AR im Bildungsbereich ein.

Nahtlose Interaktion

Billinghurst führt hier auf, dass Schüler besser zusammenarbeiten, wenn sie sich gemeinsam auf einen Arbeitsplatz fokussieren. Dieses ist bei computerbasierten Lernen schwierig um zusetzen, da sich jeder auf den Bildschirm vor sich fokussiert. Dadurch fehlt eine wichtige Eigenschaft, die die Kommunikation in der Gruppe verbessert: die gegenseitige Sichtbarkeit. Wenn Schüler gleichzeitig das Objekt der Diskussion und ihre Diskussionspartner sehen, werden auch die nichtverbalen Gesprächsmerkmale, wie Gesten oder die Mimik wahrgenommen. Diese Merkmale bilden einen essentiellen Teil der menschlichen Kommunikation. Durch den Einsatz von Augmented Reality können diese Eigenschaften bei behalten werden und trotzdem gleichzeitig computerbasierte Inhalte angezeigt werden. (Billinghurst, 2002, S. 2-3)

Greifbare Schnittstelle

Hier führt Billinghurst auf, dass im Bildungsbereich physische Objekte dazu genutzt werden Bedeutung von theoretischem Wissen zu übermitteln, in dem sie die Eindrücke des Schülers, durch ihre Erscheinung, ihre physikalischen Eigenschaften, ihren räumlichen Beziehungen und der Fähigkeit, die Aufmerksamkeit zu fokussieren, verstärken.

Diese Eigenschaften lassen sich zu großen Teilen auch auf die virtuellen Objekte in der Augmented Reality beziehen. Auf Grund der direkten Beziehungen zwischen der virtuellen Objekten und der Augmented Reality ist zum Beispiel eine physikalisch basierte Interaktion mit den computergenerierten Objekten möglich. (Billinghurst, 2002, S. 3)

Übergangsschnittstelle

Im dritten Abschnitt bezieht sich Billinghurst auf das bereits in Kapitel 2.1 eingeführte RV-Kontinuum und führt an, dass man mittels AR den Nutzer entlang des Kontinuums in die virtuelle Welt führen kann. Besonders Kinder können so in die Seiten eines Buches eintauchen und die Fantasie Realität werden lassen. Dadurch werden aus statischen Unterrichtsbüchern dynamische interaktive Umgebungen. (Billinghurst, 2002, S. 3-4)

3.1.4. Diegmann: Benefits of Augmented Reality in Educational Environments

In dem Artikel (Diegmann u. a., 2015) von Diegmann u. a. werden die Vorteile von Augmented Reality betrachtet.

Dabei wurde eine systematische Evaluation von Augmented Reality anhand der

Veröffentlichungen zu diesem Thema durchgeführt.

Basierend auf der Literatur stellte der Artikel zunächst in den folgenden Bereichen positive Effekte der Augmented Reality fest:

Geisteszustand

In verschiedenen Veröffentlichungen wurde eine Verbesserung der Motivation festgestellt. Dieses umfasst ein erhöhtes Engagement und Interesse gegenüber den Lerninhalten und der Technologie.

Des weiteren wurde auch eine Steigerung der Aufmerksamkeit und der Konzentration, bezogen auf die Inhalte mit denen die Lernenden konfrontiert wurden, festgestellt.

Zu dem verbesserte sich auch die Zufriedenheit der Schüler bezüglich der Bildungsfortschritte und des Lernprozesses. (Diegmann u. a., 2015, Kapitel 4.1)

Lehrkonzept

Das Lehrkonzept verbessert sich durch AR dahingehend, dass ein schülerzentriertes Lernen ermöglicht, wodurch das unabhängige, eigenverantwortliche Lernen der Schüler, sowie die Fähigkeit Wissensinhalte aufzunehmen verbessert wird.

Auch eine Verbesserung des kollaborativen Lernens kann durch AR erreicht werden. Dabei kann Schülern durch AR die Möglichkeit zur gemeinsamen Problemlösung und Kommunikation gegeben werden. (Diegmann u. a., 2015, Kapitel 4.2)

Darstellung

Auch die Darstellung von Lerninhalten kann mit Hilfe von AR-Anwendungen verbessert werden. So können AR-Inhalte einen höheren Detailgrad ermöglichen und die Zugänglichkeit von Lerninhalten und die Interaktivität mit diesen Inhalten. verbessern. Besonders die neuen Interaktionsmöglichkeiten unterstützen ein Lernen durch eigene Erfahrungen. (Diegmann u. a., 2015, Kapitel 4.3)

Lerntyp

Ein weiterer Vorteil ist die Verbesserung der Lernkurve dar. Schüler die mit Hilfe von Augmented Reality Unterrichtsinhalte erfassen konnten lernten schneller und einfacher als Schüler ohne diese Möglichkeit.

Auch eine Verbesserung der Kreativität, der Wissensaufnahme und der Problemlösung konnte in verschiedenen Veröffentlichungen festgestellt werden. (Diegmann u. a., 2015, Kapitel 4.4)

Verständnis

Dieser Bereich bezieht sich auf das Wissen, das durch Augmented Reality erlangt werden kann.

Dabei kann, neben einem deutlich gesteigerten, räumlichen Verständnisses, auch eine Steigerung des Gedächtnisses beobachtet werden. So konnten Schüler mit Hilfe von AR sich besser an Inhalte erinnern.

(Diegmann u. a., 2015, Kapitel 4.5)

Zuordnung der Vorteile zu verschiedenen Anwendungsarten

Diese Vorteile lassen sich auf Basis der Literatur zu den folgenden Anwendungsarten zuordnen:

- Entdeckungsbasiertes Lernen: Bei entdeckungsbasierten Anwendungen erhält der Benutzer „Informationen über einen realen Ort, während er gleichzeitig das interessierende Objekt betrachtet“ (Diegmann u. a., 2015, Kapitel 2.2)
Bei Art von Anwendungen konnte vor allem eine Verbesserung der Lernkurve und des Geisteszustandes bei den Lernenden beobachtet werden. Insgesamt deckt diese Kategorie die meisten Vorteile im Vergleich zu den anderen Anwendungsarten ab. (Diegmann u. a., 2015, Kapitel 5)
- Objektmodellierung: Diese Gruppe beschreibt Modellierungsanwendungen, die mit Hilfe von Augmented Reality dem Nutzer direktes Feedback zur Optik der modellierten Objekte geben. (Diegmann u. a., 2015, Kapitel 2.2)
Bei dieser Anwendungsart konnte in der Literatur eine erhöhte Motivation, eine verbesserte Zufriedenheit und eine angestiegene Lernkurve bei den Studierenden festgestellt werden. Jedoch konnte trotz der starken Interaktion mit der Augmented Reality, keine Artikel gefunden werden die von einer verbesserten Interaktivität oder Kreativität berichteten. (Diegmann u. a., 2015, Kapitel 5)
- AR-Bücher: AR-Bücher bezeichnen Bücher, die durch Augmented Reality angereichert werden. Dabei stellen sie dem Leser 3D-Darstellungen der Lerninhalte zur Verfügung, wenn dieser das Buch mit Hilfe von speziellen, AR-fähigen Geräten betrachtet. (Diegmann u. a., 2015, Kapitel 2.2)
Die wenigsten Veröffentlichungen konnten diese Kategorie abdecken und dem entsprechend wenig Vorteile wurden in den Arbeiten herausgestellt. Lediglich sechs der zuvor definierten Vorteil waren in der Literatur wieder zu finden. (Diegmann u. a., 2015, Kapitel 5)
- Fähigkeitentraining: Diese Kategorie umfasst Anwendungen die das Trainieren spezieller Fähigkeiten trainieren, in dem sie die Abläufe, Trainingsobjekte

oder Ähnliches bereitstellen. (Diegmann u. a., 2015, Kapitel 2.2)

Anwendungen die in den Bereich des Fähigkeitstrainings fallen haben in der Literatur die meisten Erwähnungen eines verbesserten Verständnisses. Des weiteren wird häufig eine Verbesserung der Lernkurve erwähnt. (Diegmann u. a., 2015, Kapitel 5)

- AR-Spiele: In dieser Gruppe befinden sich Video-Spiele, die mit Hilfe von Augmented Reality dem Schüler Lerninhalte vermitteln sollen (Diegmann u. a., 2015, Kapitel 2.2). Die Vorteile von AR-Spielen liegen laut der Literatur vor allem im Bereich Bereich des Geisteszustands, der Lernkurve und der Zugänglichkeit zu den Lerninhalten (Diegmann u. a., 2015, Kapitel 5).

3.1.5. Dey: A Systematic Review of 10 Years of Augmented Reality Usability Studies

In dem Artikel (Dey u. a., 2018) von Dey u. a. wird eine Auswertung von verschiedenen Studien, die sich mit dem Einsatz von Augmented Reality beschäftigen, durchgeführt. Dabei wird ein Überblick über die Veröffentlichungen aus den Jahren von 2005 bis 2014 gegeben.

Insgesamt wurden in dem Artikel die folgenden Ergebnisse herausgestellt.

Display

Allgemein betrachtet stellen Head Mounted Displays (HMD, deutsch: „Am Kopf befestigter Bildschirm“) und Hand-Held Displays (HHD, deutsch: „Handdisplay“) die meistgenutzten Wiedergabegeräte dar.

Im Bildungsbereich kommen HMDs hingegen kaum zum Einsatz. Hier dominieren vor allem HHDs, gefolgt von Desktopbildschirmen und verschiedene Arten von großformatigen Anzeigen. (Dey u. a., 2018, Kapitel 3.7)

Bildungsbereich

Alle Studien die sich mit dem Einsatz im Bildungsbereich beschäftigten, bezogen sich auf eine Methode des Unterrichtens oder des Lernens. Basierend auf den Schlüsselwörtern der Studien sind vor allem das Lernen, die Interaktivität, die Nutzer und die Umgebung relevant für den Bildungsbereich.

Eine Studie von Fonseca u. a, die im Rahmen des Artikels genauer betrachtet wurde, untersuchte den Einsatz einer Smartphone basierten AR-Anwendung, die zur Visualisierung von 3D-Modellen genutzt werden kann, in einem schulischen Umfeld. Dabei beobachteten sie eine erhöhte Motivation und eine Verbesserung der akademischen Leistungen.

Diese Anwendung stellt nur ein Beispiel der untersuchten Einsatzmethoden von AR dar. Insgesamt wurden in den Studien die verschiedensten Einsatzmöglichkeiten von Anwendungen untersucht. Darunter waren beispielsweise Applikationen, die Personen mit Amputationen im Umgang mit Prothesen schulen, Anwendungen, die kognitiv beeinträchtigte Menschen bei der Erwerbung von beruflichen Fähigkeiten unterstützen oder AR-Systemen, welche zum Unterrichten der historischen Seiten einer Stadt genutzt werden können.

Auf dieser Grundlage betonen die Autoren, die Variabilität in den Einsatzmethoden, die im Bildungsbereich genutzt werden können. (Dey u. a., 2018, Kapitel 4.2)

3.1.6. Damberger: Augmented Reality als Bildungenhancement?

In dem Artikel (Damberger, 2016) von Damberger wird die Frage thematisiert, ob Augmented Reality die Bildungsprozesse verbessern kann. Eine solche Verbesserung, die auf dem Einsatz von Technologien beruht, bezeichnet der Autor als Bildungenhancement.

Dabei wird jedoch betont, dass mit dem Bergriff nicht die Verbesserung der Bildung selbst gemeint ist. Dieses ist mit Hilfe von AR ebenso wenig möglich, wie durch einen gesteigerten Konsum an (Fach-)Literatur, so der Autor.

„Bildungenhancement soll vielmehr einen durch Technik ermöglichten besonderen Zugang zur Welt beschreiben, der reflexive Prozesse eröffnet, in deren Folge ein Mehr an Bildung möglich werden kann.“ (Damberger, 2016, S. 5)

Im Laufe des Artikels betrachtet der Autor, die oben genannte Frage aus einer philosophischen Sicht.

Augmented Reality als Erweiterung der Realität

Augmented Reality bietet dem Menschen die Möglichkeit seine Umgebung mit virtuellen Objekten zu ergänzen. Dieses Fähigkeit bietet großes didaktisches Potential, indem sie es dem Menschen erlaubt den realen Objekten, mit dem sie arbeiten um Informationen zu erweitern.

Da diese Informationen nicht nur subjektiver Art sein müssen, sondern auch objektive Eindrücke und Erfahrungen anderer Menschen repräsentieren können, erlaubt es Augmented Reality mehr Welt zu erfassen und diese auf eine andere Weise wahrzunehmen. Dadurch „enhanced“ AR die Bildung zwar nicht unbedingt, bietet aber einen anderen Zugang zu den Repräsentationen der Welt und stellt somit eine Erweiterung der Bildung dar. (Damberger, 2016, S. 22-23)

3.1.7. Buchner: Offener Geschichtsunterricht mit Augmented Reality

Der Artikel (Buchner, 2017) von Buchner beschäftigt sich mit einem Unterrichtsszenario zum Einsatz von Augmented Reality im Fach Geschichte, Sozialkunde und Politik. Grundlage des Artikels ist die Problematik, dass für einen kompetenzorientierten im Vergleich zum inhaltsorientierten Unterricht, eine intensive und aktive Auseinandersetzung mit dem Unterrichtsthemen notwendig ist und eine reine Wissensrezeption nicht mehr ausreicht.

Des weiteren wird angezweifelt, dass der klassische Lehrervortrag zur Kompetenzvermittlung nicht geeignet ist, auch wenn er im folgenden noch einen Großteil der Unterrichtszeit einnehmen wird. Der Artikel stellt hier heraus, dass die Methodenvielfalt ein wichtiges Qualitätsmerkmal der Wissensvermittlung darstellt und das beim Einsatz von digitalen Medien nicht das Tool im Mittelpunkt stehen sollte, sondern das didaktische Problem.

Lehren und Lernen mit Augmented Reality

Eine Einsatzmöglichkeit von Augmented Reality stellt das Unterstützen von authentischen Geschichten dar, so wird in dem Artikel das Beispiel angebracht, dass in einem Museum in Salzburg mit Hilfe von Markern, die in den Schaukästen angebracht sind, animierte Figuren im AR-Bereich angezeigt werden können, die den Besuchern Geschichten über beispielsweise das Leben der Kelten erzählen.

Laut verschiedener Studien hat Augmented Reality positive Auswirkungen auf den kognitiven Wissenserwerb, die Motivation, das Interesse, aber auch auf überfachliche Kompetenzen, wie das Suchen, Organisieren und Evaluieren von Informationen.

Des weiteren wurden Studien heran gezogen, die das größte Potential von AR in der Gestaltung authentischer, flexibler und mobiler Lernumgebungen sehen. (Buchner, 2017, Kapitel 3)

Anwendung im Geschichtsunterricht

Im Rahmen des Artikels wurde ein beispielhafter Einsatz in der dritten Klasse eines Wiener Gymnasiums im Fach Geschichte getestet.

Dafür wurden Videos zu den entsprechenden inhaltlichen Thematiken gedreht, dessen Startbilder anschließend als Marker fungierten. Haben die Schüler nun diese Bilder eingescannt wurden sie so mit dem Video überlagert als ob das Bild zum Leben erwachen würde.

Am Ende des entsprechenden Videos wurde jeweils einen Aufgabe für die Schüler

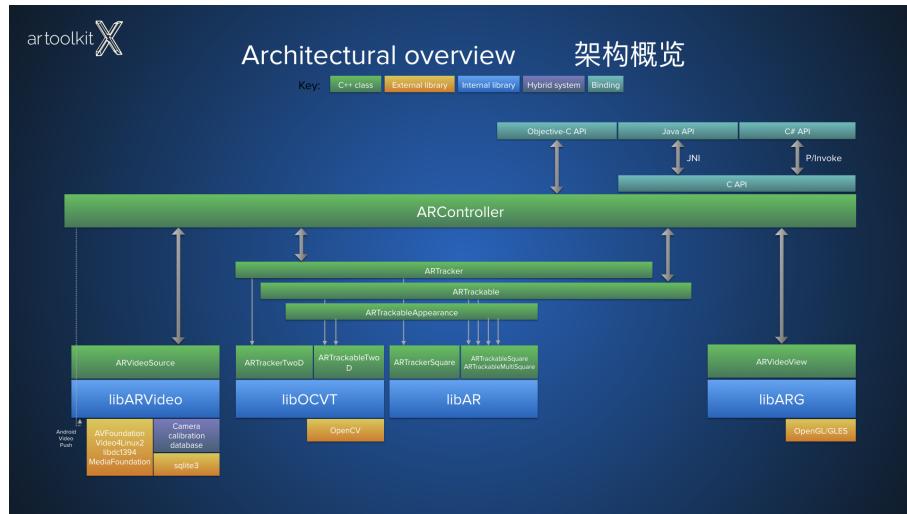


Abbildung 3.2.: Ein Überblick über die Architektur von ARToolKitX. (Quelle: Lamb u. a. (a))

eingebaut, so öffnete sich beispielsweise automatisch ein Quiz. (Buchner, 2017, Kapitel 4)

Fazit

Buchner konnte in beiden Klassen positive Einflüsse der AR-Lernumgebung auf das Interesse, die wahrgenommene Kompetenz und Wahlfreiheit feststellen.

Das Ganze Experiment traf auch über das Projekt hinaus auf Anspruch, sodass sich an dem entsprechenden Gymnasium anschließend Lehrer zusammengeschlossen haben um weitere Schülerzentrierte Lernumgebungen zu entwerfen. (Buchner, 2017, Kapitel 5-6)

3.2. Frameworks und Tools

3.2.1. ARToolKitX

ARToolKitX ist die aktuellste Version der Augmented Reality Bibliothek ARToolKit. Es wurde von Ben Vaughan und Phil Lamb zusammen mit der Unterstützung von Realmax Inc erstellt, um ARToolKit weiter als Open Source Bibliothek zur Verfügung zu stellen (Lamb u. a., b).

Dabei stellt es Entwicklern verschiedene Methoden und Klassengerüste zur Umsetzung von Augmented Reality auf verschiedenen Plattformen zur Verfügung.

Grundlegend besteht ARToolKitX aus einer Reihe an C++ Klassen, mit Schnittstellen zu den verschiedenen Programmiersprachen (vergleiche Abbildung 3.2).

3.3. Beispielhafte Anwendungen

3.3.1. Atlas der Humananatomie

Visible Body stellt mit der Anwendung „Atlas der Humananatomie 2020“ eine Anwendung zur Visualisierung von anatomischen Modellen bereit. Diese steht auch Studenten der Universität Oldenburg über den Bibliothekszugang zur Verfügung und kommt im medizinischen Bereich zum Einsatz.

[zitieren?](#)

Anwendung

Innerhalb der Applikation stehen dem Nutzer verschiedene anatomische Modelle zur Auswahl. Wird eines dieser Modelle ausgewählt wird es zunächst vor einem eintönigen Hintergrund angezeigt. Allerdings besteht zusätzlich die Möglichkeit das Modell im Augmented Reality Bereich darzustellen (siehe ??).

Mit Hilfe von Gesten kann der Nutzer mit dem Modell interagieren.

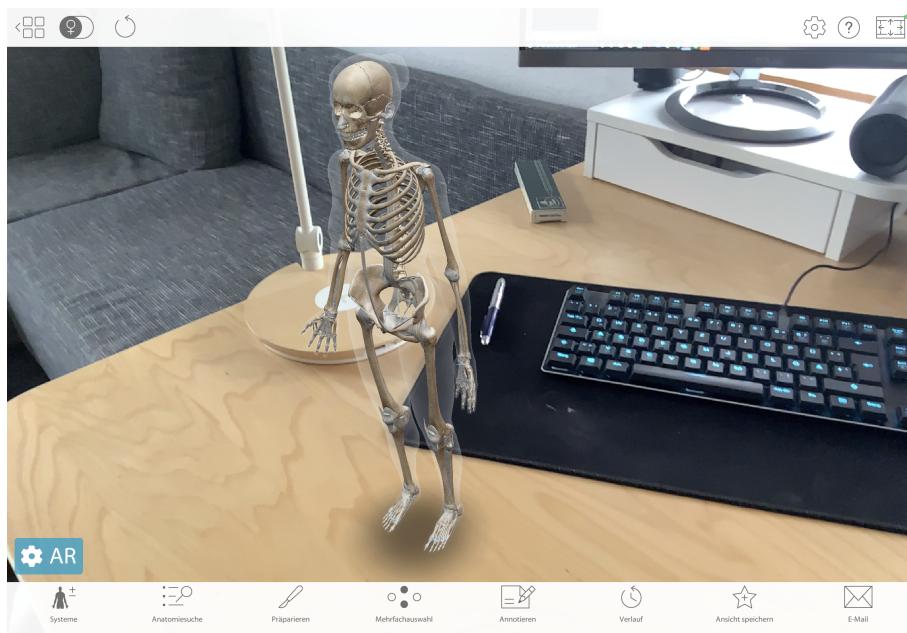


Abbildung 3.3.: Die Darstellung des menschlichen Skeletts in der Anwendung „Atlas der Humananatomie 2020“. (Quelle: Screenshot aus der Anwendung „Atlas der Humananatomie 2020“)

Kapitel 4

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die Anwendung und den zu entwickelnden Prototypen erhoben. Dazu wird zunächst auf Grundlage der Literatur ein Anwendungsfall erstellt und für diesen dann eine Anforderungsanalyse durchgeführt.

4.1. Vision

Die Anwendung hat den Bildungsbereich und das Lernen mit Hilfe von Augmented Reality als Zielgruppe. In der Arbeit von Diegmann u. a. werden dazu verschiedene Anwendungsarten definiert (vergleiche 3.1.4). Diese stellt zudem auch das entdeckungsbasierte Lernen als die Einsatzmöglichkeit mit den meisten Vorteilen für das AR-gestützte Lernen heraus. In diese Kategorie fällt auch die in Abschnitt 3.3.1 vorgestellte Anwendung „Atlas der Humananatomie 2020“.

Die Anwendungen soll in der Art der Wissensvermittlung auf den selben, bereits im praktischen Einsatz getesteten Grundlagen beruhen. Sie soll Lernende unterstützen, indem sie ihnen dreidimensionale Modelle anzeigt.

Jedoch soll das Konzept von Anwendungen, wie „Atlas der Humananatomie 2020“ dabei wie folgt erweitert werden:

Zum einen soll die Anwendung so aufgebaut werden, dass sie vom Nutzer speziell an die eigenen Bedürfnisse angepasst werden kann. Dazu soll die Anwendung kein festes Set an Modellen zur Verfügung stellen, sondern die Modelle können vom Nutzer hochgeladen werden.

Der zweite Punkt ist, dass anstelle des meistens verwendeten Matural Feature Tracking ein Markerbasiertes Tracking verwendet werden soll. So dass die Modelle speziellen Dokumenten, Textpassagen oder Lerninhalten mit Hilfe der Marker zugeordnet werden können. Diese Marker könnten dann schnell und einfach mit der Kamera getrackt werden.

Außerdem ist es die Vision, dass dieses Markerbasierte Tracking in einem großen Kontext genutzt werden kann.

Dazu wäre eine Umsetzung angelehnt an die Webseite Socrative.com denkbar. Diese

Webseite bietet Professoren die Möglichkeit Umfragen über einen Raumcode mit den Studierenden zu teilen.

Ein ähnliches System könnte auch zum Teilen der Modelle genutzt werden, so dass der Professor die Marker verknüpft mit den entsprechenden Modellen in eine Präsentation zusammen mit einem Raumcode einfügen kann und der Studierende dann, sobald er den Raumcode eingegeben hat, die Modelle in seinem Kamerabild sieht, wenn der entsprechende Marker getrackt wurde.

Ein solcher Einsatz von Augmented Reality bietet die Möglichkeit eindimensionale Vorlesungen oder Unterrichtseinheiten um eine interaktive Komponente zu erweitern und sollte viele der in der Literatur erwähnten positiven Auswirkungen, wie eine Verbesserung der Motivation, der Aufmerksamkeit des Verständnisses der Lerninhalte übertragen können (vergleiche Kapitel 3.1.4). Auch Faktoren wie die von Billinghurst beschriebene Verbesserung der Kommunikation während der Interaktion mit digitalen Lernmethoden, wird unterstützt (vergleiche 3.1.3).

4.2. Der Prototyp

Im Rahmen der Arbeit wurde dabei das folgende Konzept für einen vertikalen Prototypen entwickelt.

Dieser soll die Funktionalität des Verarbeitens eigener Modelle sowie das Tracken von selbst erstellten Markern testen. Dazu soll der Fokus die Umsetzung der Augmented Reality und das Erstellen der zu trackenden Marker gelegt werden und die hoch geladenen Modelle sollen lediglich lokal gespeichert werden.

Dadurch handelt es sich bei dem Prototyp um die Umsetzung eines „privaten Raumes“. Er bietet dem Anwender nicht die Möglichkeit die Modelle zuteilen.

Eine ansonsten notwendige Datenspeicherung auf einem Server, das Implementieren eines Raumsystems mit Zugangscode und weitere Funktionen zum Teilen von Daten fallen dadurch weg.

4.3. Anforderungsanalyse

Im folgenden wird eine Anforderungsanalyse für den zu entwickelnden Prototyp durchgeführt. Dazu wurde das in der Softwaretechnik I Vorlesung vorgestellte Verfahren zur Anforderungsdefinition genutzt (Winter, 2018, Folie 209-214). Dieses Verfahren sieht den folgenden Aufbau vor:

1. Vision
2. Machbarkeitsstudie

3. Beschreibung der Systemumgebung
4. Anwendungsfälle
5. Anforderungsliste
6. Prototypen
7. Glossar

Im folgenden wird jedoch nur eine vereinfachte, an diese Arbeit angepasste Version dieses Verfahrens genutzt.

4.3.1. Beschreibung der Systemumgebung

Bei dem Prototyp handelt es sich um eine mobile Anwendung. Das Endgerät für den Nutzer ist also ein Smartphone.

Der relevante Sensor, den das Smartphone für das Trackingverfahren bereitstellt, ist die Kamera. Die Anwendung muss die Videoframes analysieren und eine Ausgabe in Form von einem gerenderten 3D-Modell auf dem Display darstellen.

Insgesamt besitzt der Prototyp drei Unterfunktionen:

1. Das Generieren von Markern. Hierbei erstellt der Prototyp Marker für den Anwender, die alle eine unterschiedliche ID besitzen und aus jeder Rotation eindeutig zu erkennen sind. Dafür müssen sich zwei Marker auch unterscheiden, wenn einer von ihnen beliebig rotiert wurde.
2. Das Tracken von Markern. Bei dem die Anwendung einen Marker in einem Videoframe erkennt und seine Position und Transformation im Videoframe berechnet. Dabei liegt der Marker in gedruckter Form auf einem Blatt Papier vor oder wird auf einem Bildschirm angezeigt.
3. Das Rendern von Modellen. Das anzuzeigende Modell wurde dabei vom Nutzer als Datei hochgeladen und muss zunächst vom System verarbeitet werden. Um im Anschluss das Modell realistisch in Relation zum Marker zu platzieren, muss die beim Tracking berechnete Position und Transformation des Markers verwendet werden.

Da der Prototyp auf einem mobilen Endgerät genutzt wird, variiert die Umgebung in welcher dieser zum Einsatz kommt stark. Deshalb ist es notwendig, dass das System robust gegenüber verschiedenen Umwelteinflüssen ist.

4.3.2. Anwendungsfälle des Prototyps

Ein Anwendungsszenario des Prototypen ist die mit dem Prototyp generierten Marker auf einem auf einem begleitenden Unterrichtsdokument einzufügen und ein oder mehrere relevante Modelle zu verlinken. Dazu werden die Modelle in dem Prototyp hochgeladen und lokal auf dem Gerät gespeichert. Die von der Anwendung generierten Marker können dann in einem Dokument eingefügt werden. Über die Kamerafunktion ist es im Anschluss möglich die Marker zu scannen, um sich die entsprechenden, dreidimensionale Modelle anzeigen zu lassen.

Allgemein lassen sich die Funktionen und Anwendungsfälle in dem in Abbildung 4.1 gezeigtem Anwendungsfalldiagramm visualisieren.

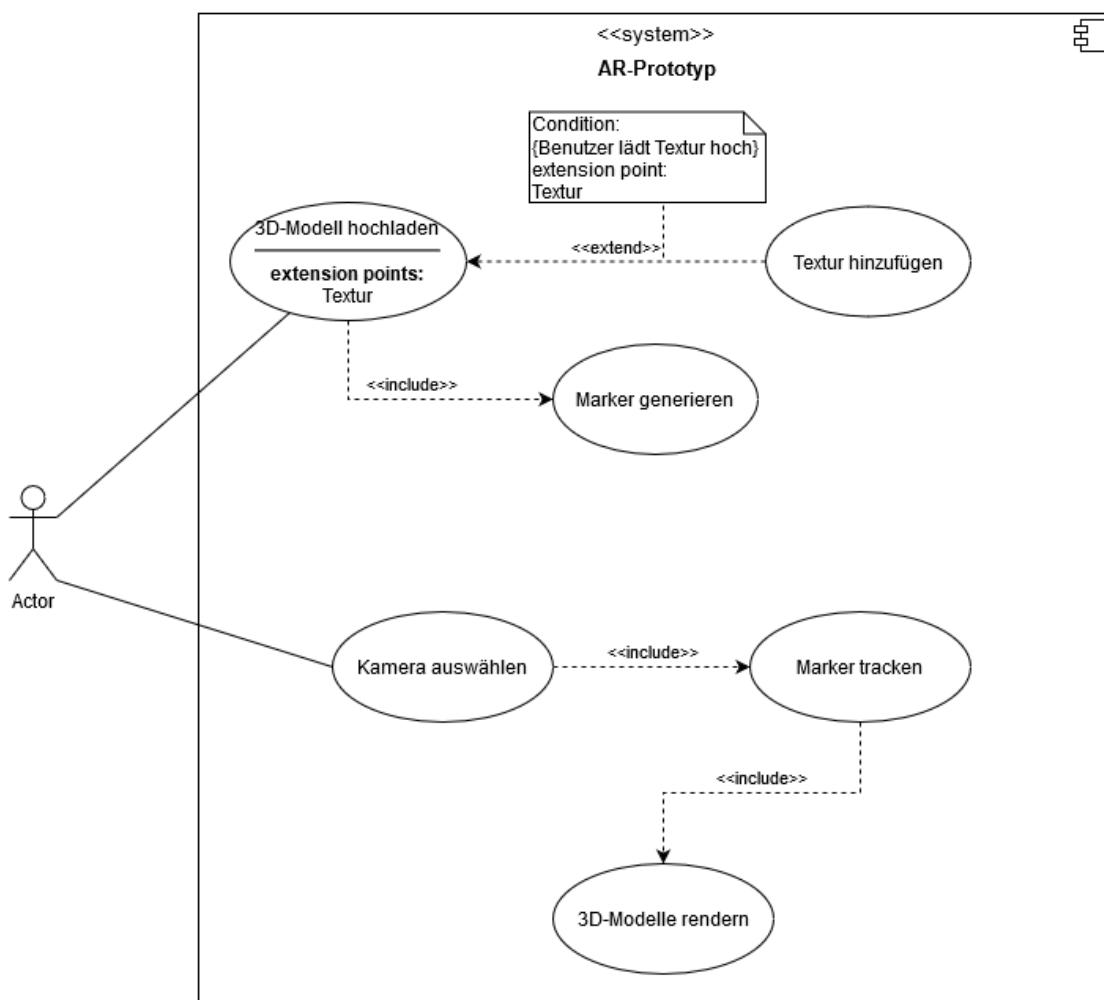


Abbildung 4.1.: Use Case Diagramm des Prototyps. (Quelle: Eigene Darstellung)

4.3.3. Anforderungsliste

Im folgenden Abschnitt werden die Anforderungen an den Prototyp definiert. Dabei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

„Eine funktionale Anforderung ist eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems [...] bereitgestellt werden soll.“ (Rupp und die SOPHISTen, 2014, S. 17)

Nicht-funktionale Anforderungen umfassen im Vergleich dazu die Anforderungen an die Technologie, die Qualität, die Benutzeroberfläche, die durchzuführende Tätigkeiten und rechtlich vertragliche Anforderungen (Rupp und die SOPHISTen, 2014).

Sie können als Eigenschaften, die das System besitzen muss, um den Nutzer zufrieden zu stellen, beschrieben werden. (Robertson und Robertson, 2012, S. 10). Diese Eigenschaften beziehen sich meistens auf die Bereiche Service Level, Zugriffsbeschränkungen, Sicherheit, Monitoring, Kontrolle, Schnittstellen, Archivierung, Benutzerfreundlichkeit und Konversion (Böhm und Fuchs, 2002, S. 139).

Auf Grundlage dieser Definition wurden die folgenden funktionalen und nicht-funktionalen Anforderungen mit Hilfe der von Rupp und die SOPHISTen, S. 219 definierten Anforderungsschablonen aufgestellt:

Funktionale Anforderungen

- FA01 Die Anwendung muss fähig sein, ein Marker zu generieren.
- FA02 Die Anwendung muss fähig sein, Marker in dem Kamerabild erkennen zu können.
- FA03 Die Anwendung muss fähig sein, generierte Marker zu unterscheiden.
- FA04 Die Anwendung muss dem Benutzer die Möglichkeit bieten, 3D-Modelle in Form von OBJ-Dateien hochzuladen können.
- FA05 Die Anwendung muss dem Benutzer die Möglichkeit bieten, Textur-Dateien im Bildformat(jpeg, png) hochzuladen.
- FA06 Die Anwendung muss fähig sein, hochgeladenen Dateien lokal zu speichern.
- FA07 Die Anwendung muss fähig sein, jedem Modell einen individuellen Marker zuzuordnen.
- FA08 Die Anwendung muss fähig sein, die dreidimensionalen Modelle im Kamerabild anzuzeigen.
- FA09 Die Anwendung muss fähig sein, die Transformation der Marker zu berechnen.
- FA10 Die Anwendung muss fähig sein, die Modelle basierend auf der berechneten Transformation zu rendern.

Nichtfunktionale Anforderungen

- NF01 Das Endgerät der Anwendung sollte ein Android Smartphone(Huawei P30 Pro) sein.
- NF01.1 Die genutzte Entwicklungsumgebung der Anwendung sollte Android Studio sein.
- NF01.2 Die genutzte Programmiersprache der Anwendung sollte Java sein.
- NF02 Die Markergenerierung der Anwendung sollte so gestaltet sein, dass es mindestens 10 verschiedene Marker generieren kann.
- NF03 Das Tracking der Anwendung sollte kamerabasiert sein.
- NF04 Das Tracking der Anwendung sollte auf einem markerbasiertem Verfahren beruhen.
- NF05 Das Tracking der Anwendung sollte mit mindestens 30FPS laufen.
- NF06 Das Tracking der Anwendung sollte so gestaltet sein, dass es alle generierten Marker der Anwendung erkennen kann.
- NF07 Bei vollständig erkanntem Marker sollte das Tracking robust gegenüber Rotation, Skalierung, Perspektive und Belichtung sein.
- NF08 Die Benutzeroberfläche der Anwendung sollte eine einfache Navigation zu allen Komponenten der Anwendung bereitstellen.
- NF09 Die Benutzeroberfläche der Anwendung sollte eine Ansicht zum Hochladen und Verwalten der Modelle bieten.
- NF10 Die Benutzeroberfläche der Anwendung sollte einen Zugriff auf die Kamera ermöglichen.

Kapitel 5

Entwurf

Dieses Kapitel beschäftigt sich mit dem Entwurf des Prototyps. Die Aufgabe des Entwurfs eines Softwaresystems ist es aus den gegebenen Anforderungen softwaretechnische Lösungen zu entwickeln (Balzert, 2011). Die Grundlage bilden dabei die in Kapitel ?? definierten Anforderungen.

5.1. Allgemeine Technologieentscheidungen

Im ersten Teil des Entwurfs werden grundlegende Technologieentscheidungen betrachten, die die Grundlage für das folgende Design bieten.

5.1.1. Anwendungsart

Die Anwendung soll Lernenden eine weitere Zugriffsmöglichkeit auf Wissensinhalte in Form von Augmented Reality bieten. Wie bereits in den Anforderungen festgelegt, soll dafür die Anwendung auf einem mobilen Endgerät laufen.

Dieses bietet im Bezug auf AR mehrere Vorteile. Zum einen ist es wichtig, dass das Endgerät frei im Raum bewegbar ist, damit der Anwender gut mit den AR-Objekten interagieren kann. Dieses ist durch ein mobiles Endgerät gewährleistet. Eine Alternative dazu wären sogenannte Head-Mounted-Displays (siehe Kapitel ??). Diese besitzen aber im Vergleich eine deutlich geringere Verfügbarkeit. Ein Handy oder Tablet besitzt dagegen heutzutage fast jeder Student und hat dieses auch so gut wie immer parat. Dadurch ermöglicht ein Smartphone zusätzlich einen schnellen, ortsunabhängigen Zugriff.

Ein weiteren Vorteil den ein mobiles Endgerät mit sich bringt ist die Kamera verfügt, welche für die meisten Trackingverfahren essentiell ist.

Betriebssystem

Für mobile Anwendungen gibt es vor allem zwei Plattformen, die für diese Arbeit in Frage kommen. Auf der einen Seite steht Apples IOS und auf der Anderen Android von Google. Insgesamt gibt es im Bezug auf die AR-Anwendung nicht viele Argumente, die für eine bestimmte Seite sprechen. Letztendlich wurde Android als Zielsystem gewählt, da bereits ein Endgerät, welches dieses Betriebssystem nutzt, vorliegt. Dadurch kann die entwickelte Software direkt auf diesem Gerät getestet werden.

Entwicklungsumgebung

Basierend auf der Entscheidung das Betriebssystem Android zu nutzen, wurde die Entwicklungsumgebung (IDE) wurde Android Studio gewählt, weil dieses die offizielle Entwicklungsumgebung für Android Anwendungen darstellt und viele praktische Features bereitstellt.

5.1.2. Trackingverfahren

Für Augmented Reality Anwendungen auf einem Smartphone kommen vor allem visuelle Tracking Verfahren, bei denen das Kamerabild als Eingabe für den Tracker genutzt wird, in Frage. Dabei gibt es jedoch nochmal die Unterscheidung zwischen Natural Feature Tracking und Markerbased Feature Tracking (vgl. Kapitel 2.1.2). Für den konkreten Anwendungsfall dieser Arbeit ist das markerbasierte Verfahren am besten geeignet. Es ermöglicht, dass Speichern von Informationen über den Identifikator (ID) der einzelnen Marker. Dadurch kann man jedem individuellem Marker ein AR-Objekt zu ordnen. Diese Zuordnung ist mit Hilfe von natürlichen Bildpunkten nicht direkt möglich, da bei diesem Verfahren lediglich ein Objekt im Bild positioniert wird und anhand markanter Bildpunkte der Umgebung ausgerichtet und transformiert wird.

Ein weiterer Vorteil den die Marker bieten ist dass sie sich einfach in verschiedene Arten von Dokumenten einbinden lassen und dann im Anschluss lediglich mit Hilfe einer Kamera eingescannt werden müssen.

5.1.3. Tracking Bibliothek

Mittlerweile gibt es viele Bibliotheken, die Methoden und Klassen zur Umsetzung von Augmented Reality bereitstellen und die Tracking Verfahren müssen nicht selbst erstellt werden. Jedoch unterstützen nicht alle von Ihnen auch das Markerbasierte Verfahren. Eines der vermutlichen bekanntesten Bibliotheken, die die Anforderungen erfüllen, ist ARToolKit. Es stellt umfangreiche Klassen zur Implementierung von

Augmented Reality zur Verfügung und unterstützt dabei die verschiedensten Markervarianten von simplen Hiro Markern bis hin zu Barcode Markern mit verschiedenen IDs. Im Rahmen dieser Arbeit wurde die Version ARToolKitX genutzt. Weitere mögliche Bibliotheken wären Googles ARCore, welches jedoch keine explizite Marker Unterstützung besitzt, oder OpenCV, welches viele Methoden zum Tracken bereitstellt, jedoch im Vergleich zu ARToolKit einen deutlich aufwendigeren Implementierungsprozess benötigt. ARToolKit bietet zusätzlich den Vorteil, dass es bereits an die Konzepte von Android angepasst wurde und sich somit gut in die bestehende oder neue Android Anwendungen einfügen lässt.

5.2. Designentscheidungen

Im folgenden Abschnitt werden konkrete Designentscheidungen für die Implementierung der Anwendung getroffen.

5.2.1. Grundlage der Entwicklung

ARToolKitX stellt neben dem Methoden und Klassen zur Umsetzung von Augmented Reality auch eine simple AR-Anwendung bereit, die die zuvor definierten allgemeinen Technologieentscheidungen erfüllen kann.

Diese Anwendung kann als Grundlage für die folgende Implementierung genutzt werden und im Rahmen dieser Arbeit so erweitert werden kann, dass sie die in Kapitel 4 entwickelten Anforderungen erfüllt.

Die Anwendung von ARToolKit „ARSquareTrackingExample“ trackt einen einzelnen Hiro-Marker und projiziert einen bunten Würfel auf den Marker.

Damit die gestellten Anforderungen erfüllt werden können, müssen in der Implementierung die folgenden, grundlegenden Punkte umgesetzt werden:

1. Marker: Die genutzten Hiro-Marker der Anwendung können keine ID speichern. Also müssen andere Marker verwendet werden. Des weiteren muss die Anwendung die Marker selbst generieren können, damit der Nutzer eigene Modelle Tracken kann.
2. Modelle: Die Anwendung verfügt lediglich über ein einzelnes hardgecoded Modell, welches angezeigt werden kann. Dieses muss durch das Laden eigener, variabler Modelle ersetzt werden.
3. Texturen: Auch die Textur die in Form fest definierter Farben im Text verankert ist muss, durch das Laden und anwenden der Texturdatei ersetzt werden.

4. Datenspeicherung: Basierend auf den beiden vorherigen Punkten muss die Anwendung ein Konzept zur Speicherung der hochgeladenen Modelle implementieren, damit diese dem Nutzer auch nach einem Neustart der Anwendung zu Verfügung stehen.
5. User Interface: Die Anwendung von ARToolKitX besitzt noch kein User Interface, sie zeigt lediglich die Kamera und rendert das Modell in dieses Bild. Damit die finale Anwendung jedoch alle Funktionen erfüllen kann muss ein User Interface zur Navigation zwischen den einzelnen Komponenten eingeführt werden.

Im folgenden werden die aufgeführten Punkte einmal genauer betrachtet und konkrete Designentscheidungen getroffen.

5.2.2. Marker

Die Hauptanforderung an die Marker ist, dass sie über eine ID differenziert werden können. ARToolKitX stellt hier die sogenannten Barcode-Marker bereit, die aus einem schwarzen Rand und einem Pattern, das eine codierte ID beinhaltet, bestehen. Diese Marker sind speziell auf das Tracking von ARToolKitX zugeschnitten und sollen auch in der Implementierung genutzt werden.

Bei der Verwendung dieser Marker müssen zwei Entscheidungen getroffen werden. Zum einen muss über die Größe des Patterns entschieden werden. Hierbei unterstützt ARToolKitX Matrizen der Größe 3x3 bis 6x6. Die Größe der Matrix beeinflusst dabei die Anzahl der Marker, die generiert werden sollen. Dabei sind kleinere Matrizen vermutlich jedoch weniger fehleranfällig.

Der zweite Eigenschaft, die variieren kann ist die Codierung der Marker. Dazu nutzt ARToolKitX je nach Matrixgröße verschiedene Codierungsmethoden, die den Sinn verfolgen die Fehlertoleranz, bei Auslesen des Patterns, zu erhöhen. Dieses geschieht indem sich die einzelnen Marker durch die Codierung optisch deutlicher unterscheiden. Zwar ist die Fehlertoleranz des Trackings umso höher je größer die Unterschiede sind, allerdings sinkt dadurch auch die Anzahl der verschiedenen Marker.

Da die Fehleranfälligkeit zu Beginn noch nicht eingeschätzt werden kann, werden zunächst einmal die simplen 5x5 BarcodeMarker ohne Codierung genutzt. Allerdings sollte bei der Implementierung auf eine leichte Austauschbarkeit der Marker geachtet werden, um auf mögliche Probleme reagieren zu können.

5.2.3. Modelle

Da es dem Nutzer mit der Anwendung möglich sein soll eigene Modelle hochzuladen, muss die Software die entsprechenden Dateien einlesen und verarbeiten können. Für

diese Modelldateien stehen dem Nutzer verschiedene Datei-Formate zur Verfügung. Der zu entwickelnde Prototyp soll zunächst lediglich das OBJ-Format unterstützen, dieses könnte dann in der Zukunft recht einfach erweitert werden.

Diese Entscheidung beruht darauf, dass dieses Format von den meisten Grafikprogrammen unterstützt wird. So ist es auch mit dem kostenlosen Tool Blender, welches zum Erstellen der Testmodelle genutzt wurde, möglich die Objekte im OBJ-Format zu exportieren und somit auch andere Formate in das OBJ-Format zu konvertieren. Eine weitere Anforderung an das Modell ist, dass die sogenannten Faces, welche die einzelnen Flächen (Polygons), trianguliert werden. Dieses ist in den meisten Grafikprogrammen ohne Probleme vor dem Export möglich. Auch hier könnte in späteren Versionen des Prototyps über eine erweiterte Unterstützung nachgedacht werden.

5.2.4. Texturen

Neben der OBJ-Datei bestehen die Modelle die in diesem Format exportiert werden aus einer Materialdatei und einer Textur in Form einer Bilddatei.

Beide stellen eine Variante dar um einem Modell eine Textur zuzuweisen.

Für den Prototyp wurde die Entscheidung getroffen, die MTL-Datei, welche Materialinformationen des Objektes speichert nicht zu betrachten, sondern dem Nutzer die Möglichkeit zu bieten, die Textur des Modells über eine Bilddatei zu definieren. Diese Entscheidung beruht zum einen auf der Tatsache, dass die viele der Modelle die Online zur Verfügung stehen primär auf diese Variante zurückgreifen und zum Anderen das Erstellen eines eigenen Modells mit einer Texturdatei für einen Nutzer, ohne Vorkenntnisse im Bereich der Modellierung, einfacher ist.

5.2.5. Datenspeicherung

Die Anwendung muss die Modelle, die vom Anwender hochgeladen werden auf geeignete Weise abspeichern.

Da dieses zunächst nur auf lokaler Ebene passieren soll, kann für diesen Zweck auf den internen Speicher zurückgegriffen werden. Dieser bezeichnet den Speicher der einer Anwendung und dessen Inhalt bei Deinstallation der Anwendung gelöscht wird. Zusätzlich muss noch eine Referenzierung über die MarkerID auf den zugehörigen Speicherort der kopierten Dateien im internen Speicher erfolgen, damit zum einen alle Modelle beim Start der Anwendung geladen werden können und zusätzlich jedem Modell ein Marker zugeordnet ist.

5.2.6. User Interface

Damit der Nutzer auf alle Funktionen des Prototyps zugreifen kann, soll dieser simples User Interface besitzen, welches dem Nutzer grundlegende Interaktionsmöglichkeiten bietet. Ein erstes Mockup dieses Interfaces wird in Abbildung 5.1 dargestellt. Es

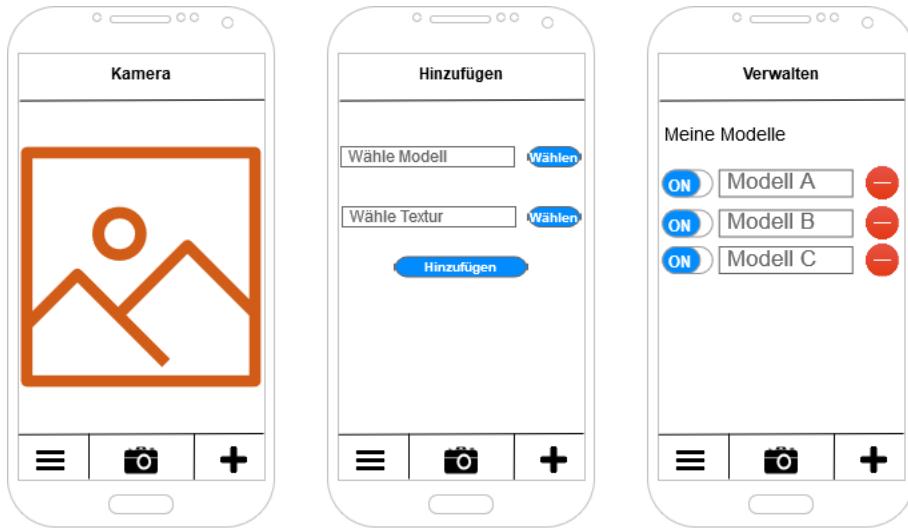


Abbildung 5.1.: Das Mockup des User Interfaces. (Quelle: Eigene Darstellung)

zeigt die drei Ansichten (Fragments) der Anwendung, welche von der Navigation überlagert werden.

Die Navigation besteht aus zwei Teilen am oberen Bildschirmrand ist der Titel des aktuellen Fragments zusehen und am unteren Rand befindet sich die eigentliche Navigation. Diese besteht aus drei repräsentativen Icons, welche jeweils ein Fragment repräsentieren. Über einen Druck auf den entsprechenden Knopf erscheint die zugehörige Ansicht.

Das Hauptfragment stellt die Kamera dar. Hier wird das Kamerabild analysiert und mittels Augmented Reality, um die entsprechenden Objekte erweitert. Zusätzlich ist noch ein Fragment zum Hinzufügen von Modellen und eins zum Verwalten der Fragments vorgesehen.

Letzteres bietet dem Nutzer die Möglichkeit bestehende Modell wieder zu entfernen.

Kapitel 6

Implementierung

Die Anwendung besteht grundlegend aus einer *MainActivity* in welche die verschiedenen Fragments geladen werden.

Jedes Fragment wurde dabei nach dem in Android üblichen Pattern für Fragments entwickelt, welches aus einem *Fragment*, das für die optische Darstellung der Informationen zuständig ist, und einen *ViewModel*, welches die Daten verarbeitet und an das Fragment übergibt, implementiert.

Im folgenden werden die implementierten Klassen und Konzepte der entwickelten Anwendung einmal genauer betrachtet.

6.1. MainActivity

Die *MainActivity* wird ausgeführt sobald die Anwendung vom Nutzer gestartet wird. In dieser Klasse wird der Hauptbildschirm erzeugt, der aus einem *Hostfragment*, in welches dann später die einzelnen Fragments geladen werden, und der *BottomNavigationView* besteht.

Um der Navigationsleiste ein Funktion zuzuweisen, muss in der *onCreate*-Methode der *NavController* erstellt werden. Dieser steuert die einzelnen Buttons der *BottomNavigationView* und lädt die zugehörigen Fragments in das *Hostfragment*. Über diese Leiste stehen dem Nutzer auf die Ansichten Augmented Reality, Modellverwaltung und Markergenerierung zur Verfügung.

6.2. Augmented Reality

Die wichtigste Ansicht der Anwendung stellt die AR-Funktion dar.

Sie zeigt dem Nutzer das Kamerabild und fügt die 3D-Modelle in dieses ein.

Diese Ansicht wird durch das *ARCameraFragment* bereitgestellt. Die Struktur, die zur Umsetzung der Augmented Reality genutzt wurde, beruht zum Großteil auf den Konzepten und Klassen von ARToolKit und wurde dann an die Anforderungen dieser Arbeit angepasst.

6.2.1. ARCameraFragment

Das *ARCameraFragment* bildet den Grundbaustein des Augmented Reality Fragments und erweitert das *ARFragment*. Es erzeugt die visuelle Oberfläche in Form eines *FrameLayouts*, welches ein *GLSurfaceView* enthält, auf dem später das Kamerabild, sowie die Modelle projiziert werden.

Des weiteren überschreibt die Klasse die abstrakten Methoden des *ARFragments* und weist diesem einen *ARRenderer* und das *FrameLayout* zu.

6.2.2. EducationARRenderer

Die *EducationARRenderer*-Klasse erweitert ARToolKits *ARRender*. Die Aufgabe des Renderes ist es die AR-Szene zu konfigurieren und die Visualisierung der AR-Modelle einzuleiten.

Dazu werden zunächst alle Modelle geladen und die Instanzen der *Modell*-Klasse in einer ersten Hashmap gespeichert. Des weiteren werden die folgenden Klassen der *ARRenderer*-Klasse überschrieben.

configureARScene()

In dieser Klasse wird die AR-Szene konfiguriert. Dazu wird zunächst für jedes Modell der Marker mit der zugehörigen ID erzeugt und die sogenannte UID, die zurückgeliefert wird und auf den Marker verweist, zusammen mit dem Modell in einer zweiten Hashmap gespeichert.

draw()

Diese Methode dient dazu die *draw*-Funktion der *Modell*-Klassen aufzurufen, wenn der Marker des Modells im Kamerabild gefunden wurde.

Dafür durchläuft die Methode mit Hilf der UIDs alle Marker und generiert die Model-, View- und Projektionsmatrizen der gefundenen Marker. Diese Matrizen werden dann an die entsprechenden *draw*-Methoden der Modelle weitergegeben, um die Modelle in der richtigen Transformation zu visualisieren.

6.3. Modellverwaltung

Eine der drei Hauptfunktionen der Anwendung stellt Modellverwaltung dar. Diese besteht eigentlich zwei einzelnen Fragments, dem *AdministrationFragment* und einem weiterführenden Fragment, dem *UploadFragment*. Das *AdministrationFragment* wird geladen, sobald der Nutzer in der Navigation die Modellverwaltung aufruft. Hier kann der Nutzer eine Liste mit seinen hoch geladenen Modellen betrachten,

und diese gegebenenfalls löschen.

Über einen Plus-Button wird ein Nutzer auf das *UploadFragment* weitergeleitet. Dieses erlaubt es dem Nutzer Modelle aus den Dateien hochzuladen.

6.3.1. AdministrationFragment

Wie bereits beschrieben ist es die Aufgabe dieser beiden Klassen die Modelle des Nutzers zu Verwalten und wird aufgerufen sobald der Nutzer über die Navigation die Verwaltung auswählt.

Dafür wird dem Nutzer über das *AdministrationFragment* eine graphische Oberfläche bereit gestellt. Dafür werden über die *ModelManager*-Klasse alle Modelle des Nutzers eingeholt und für jedes Modell über die *createNewElem()*-Methode eine Zeile im Layout in Form eines horizontalen linearen Layouts erstellt.

Des weiteren wird ein Button erzeugt, welcher über einen *OnClickListener* das *UploadFragment* aufruft.

createNewElem()

Diese Methode erzeugt die graphische Darstellung eines Modelles im *AdministrationFragment*.

Dazu wird auf Grundlage der übergebenen Parameter eine horizontales lineares Layout erstellt, welches beginnt mit einem Switch-Button, der zunächst nicht implementiert wurde, aber später dazu genutzt werden soll ein Modell zu aktivieren oder zu deaktivieren. Anschließend folgt die eindeutige MarkerID des Modelles und der Name. Zuletzt folgt ein Button zum Löschen des Markers, der über einen *OnClickListener* die *deleteModel()*-Methode der *ModelManager*-Klasse aufruft.

6.3.2. UploadFragment

Dieses Fragment wird über den Hinzufügen-Button der *AdministrationFragment*-Klasse aufgerufen und erzeugt dem Nutzer eine graphische Oberfläche zum hochladen von Modellen.

Dazu werden dem Nutzer verschiedene Eingabefelder zur Verfügung gestellt.

Das erste ist ein Texteingabefeld, über welches dem Modell ein Name zugeordnet wird. Initial wird dieses Feld mit dem Namen „Modell-MarkerID“erzeugt.

Anschließend folgen zwei Buttons zum Auswählen der Texturdatei und der Modelldatei. Über einen *OnClickListener* wird ein *Action Open Document*-Intent erzeugt, sobald der Button gedrückt wird. Dieser Intent erzeugt ein Dialog zur Dateiauswahl, über den der Nutzer auf seinen lokalen Speicher, sowie GoogleDrive, zugreifen kann. Das Ergebnis des Intents kann dann über die Callback-Methode *onActivityResult()*

abgerufen werden.

Des weiteren wird ein Button zum Hinzufügen des Modells bereitgestellt, welcher die Methode `addModel()` der *UploadViewModel*-Klasse aufruft.

onActivityResult()

Bei dieser Methode handelt es sich um eine Callback-Methode, die die Ergebnisse eines Intents abruft.

Dazu wird anhand eines Codes geprüft, um was für eine Rückgabe es sich bei den empfangen Daten handelt und ob das Ergebnis erfolgreich war. Anschließend wird dann, falls es sich bei der Rückgabe um den gewählten Dateipfad zu einer der Modelldateien handelt, der entsprechende Pfad an das *UploadViewModel* übergeben.

6.3.3. UploadViewModel

Das *UploadViewModel* stellt die Daten für das *UploadFragment* zur Verfügung. Dazu wird das Konzept der *MutableLiveData* genutzt, welche vom *UploadFragment* über einen *Observer* beobachtet werden können. Diese werden im Viewmodel initial erzeugt und werden dann über Nutzereingaben vom *UploadFragment* aktualisiert. Lediglich die MarkerId wird anfänglich generiert und im Anschluss nicht mehr verändert. Dazu wird die `getNextID()`-Methode der *ModelManager*-Klasse aufgerufen. Diese liefert dem *UploadViewModel* die nächste freie ID zurück.

addModel()

Diese Methode wird vom *UploadFragment* aufgerufen und fügt das Modell zu den Modellen des Nutzers hinzu, falls alle benötigten Daten vom Nutzer angegeben wurden.

Ist dies der Fall werden die Modell über die `transferToInternalStorage()`-Methode der *FileManager*-Klasse in den internen Speicher der Anwendung kopiert. Des weiteren wird über die Methode `addModel()` der **ModelManager**-Klasse eine Referenz auf die Dateien abgespeichert.

Anschließend wird der Nutzer auf das *ConfirmationFragment* weitergeleitet.

requierdDataAvailable()

Diese Methode prüft, ob der Nutzer alle Daten angegeben hat, die benötigt werden, um das Modell hochzuladen.

correctFileType()

Diese Methode wird dazu genutzt, um zu überprüfen, ob es sich bei der Modelldatei, die der Nutzer wählt, um eine Obj-Datei handelt. Dazu wird der Dateiname der Modelldatei abgefragt und die Endung der Datei geprüft. Eine solche Überprüfung ist bei der Texturdatei nicht nötig, da an dieser Stelle über den MIME-Type, die Auswahl eingeschränkt werden kann.

6.3.4. ConfirmationFragment

Das *ConfirmationFragment* wird aufgerufen wenn der Nutzer ein Modell hochgeladen hat. Dabei wird die MarkerID des Modells und dessen Name an das Fragment übergeben. Mit Hilfe der ID generiert das Fragment über die *generateMarker()*-Methode der *MarkerGenerator*-Klasse den Marker der zum Modell gehört und zeigt diesen in einem *ImageView* an.

Über einen Button hat der Nutzer die Möglichkeit den Marker abzuspeichern. Per *OnClickListener* wird ein *Intent* erzeugt der dem Nutzer einen Dialog öffnet in dem er einen Ordner zum Abspeichern des Markers auswählen kann.

onActivityResult()

Diese Methode stellt die Callback-Methode zur Behandlung von *Intents* dar. Hier wird die *saveBitmap()*-Methode der *Filemanager*-Klasse aufgerufen, nachdem der Nutzer einen Dateipfad zum Abspeichern des Markers gewählt hat.

6.4. Markergenerierung

Diese Funktion stellt die letzte Ansicht dar, die über die Navigationsleiste aufgerufen werden kann.

Sie besteht lediglich aus dem *MarkerFragment*, mit dessen Hilfe der Nutzer einen Marker, den er zum Beispiel gelöscht hat, erneut generieren kann.

6.4.1. MarkerFragment

Das *MarkerFragment* stellt dem Nutzer eine Oberfläche mit einem Eingabefeld und einem Button zur Verfügung.

In das Eingabefeld kann der Nutzer, die ID eines Modelles, welche er aus der Modelldatenbank entnehmen kann, eintragen. Drückt dieser anschließend den Button, wird über einen *OnClickListener* der zugehörige Marker zu der ID mit der *generateMarker()*-Methode der *MarkerGenerator*-Klasse aufgerufen und die zurückgelieferte Bitmap des Markers in einem *ImageView* angezeigt.

Des weiteren wird ein Button zum Speichern des Markers mit den analogen Funktion des Buttons im *ConfirmationFragments* zur angezeigt.

6.5. Shader Implementierung

Zum Rendern der Modelle mussten die folgenden eigene Shader implementiert werden. Diese beruhen im Grundgerüst auf den von ARToolkit bereitgestellten Beispieleimplementierungen der Shader und wurden um die Anforderungen dieser Arbeit erweitert. So wurde beispielsweise eine Unterstützung von Texturen implementiert.

6.5.1. MyShaderProgram

Die Klasse *MyShaderProgram* erweitert ARToolKits Klasse *ShaderProgram*.

Die wichtigste Eigenschaft dieser Klasse ist es, dass sie die *render()*-Methode der *ShaderProgramm*-Klasse überschreibt.

render()

Diese Methode wird aufgerufen um ein Modell zu rendern.

Dazu werden zunächst die Verticies, Farbinformationen, Normalenvektoren und Texturkoordinaten an OpenGL übergeben.

Im letzten Schritt wird dann das Modell aus vielen einzelnen Dreiecken mit OpenGLs *drawArrays()*-Funktion gezeichnet.

6.5.2. MyVertexShader

Damit OpenGL den Shader ausführen kann muss dieser in der Programmiersprache OpenGL Shading Language (GLSL) geschrieben sein.

Der String *vertexShader* enthält diesen in GLSL geschriebenen Shader.

6.5.3. Vertex Shader

In dem Shader selbst wird der finale Punkt von jedem Vertex aus der *Projection*-Matrix, der *ModelView*-Matrix und dem ursprünglichen Vertexvektor berechnet. Dieser wird dann zusammen mit der Farbe, der Texturkoordinate und des Normalenvektors an den Fragment Shader, der in der Klasse *MyFragmentShader* implementiert wird, weitergeleitet.

configureShader()

Diese Methode konfiguriert einen Shader und übergibt den String mit dem Vertex Shader an OpenGL.

6.5.4. MyFragmentShader

Auch dieser Shader muss über den String *fragmentShader* in GLSL implementiert werden.

Fragment Shader

In dem Shader, der über den String definiert wird, wird die Farbe für jedes Fragment aus der vom Vertex Shader übergeben Farbe, und übergebenen Textur berechnet.

configureShader()

Diese Methode konfiguriert einen Shader und übergibt den String mit dem Fragment Shader an OpenGL.

6.6. Hilfsklassen

Im Folgenden werden ein paar Hilfsklassen vorgestellt, die genutzt wurden, um den Restlichen Klassen wichtige Funktionen zur Verfügung zustellen.

6.6.1. FileManager

Der *FileManager* dient dem Zweck alle Funktionen bereitzustellen, die benötigt werden um die Dateien zu verwalten, die die Anwendung benötigt.

saveBitmap()

Die *saveBitmap()* Methode speichert ein Bild, welches in Form einer Bitmap übergeben wird, an einem bestimmten Dateipfad, der über die Methodenparameter übergeben wird.

Dazu wird ein *OutputStream* erzeugt auf welchen die Datei dann in Form einer jpeg-Datei geschrieben wird.

transferToInternalStorage()

Diese Methode kopiert eine Datei von einem Ort in den internen Speicher. Dazu wird ein *InputStream* zum lesen der Datei und ein *OutputStream* zum schreiben der Datei geöffnet. Anschließend wird die Datei Stück für Stück eingelesen und auf den *OutputStream* geschrieben.

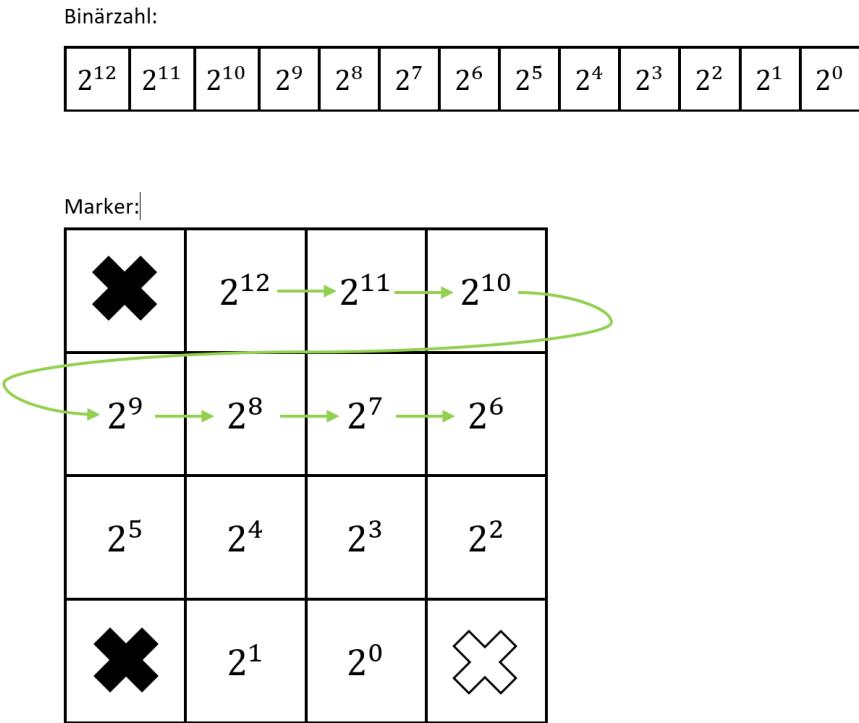


Abbildung 6.1.: Visualisierung des Vorgehens zur Erstellung eines 4x4 Barcodepatterns. (Quelle: Eigene Darstellung)

printInternalStorage()

Dieses ist eine Hilfsmethode zum ausgeben des internen Speichers. Sie wurden zum Debuggen verwendet.

6.6.2. MarkerGenerator

Der *MarkerGenerator* ist eine Hilfsklasse, die dazu zuständig ist einen Marker mit einer bestimmten ID zu generieren, der dann von den AR-Klassen getrackt werden kann.

Die Hauptfunktion ist dabei die Methode *generateMarker()*.

generateMarker()

Die *generateMarker*-Methode erzeugt einen Marker mit einer bestimmten ID und Patterngröße n , die über die Methodenparameter übergeben werden. Dazu wird zunächst eine quadratische Bitmap der Größe $n \times n$ erzeugt. Anschließend muss aus der übergebenen ID ein Barcodemuster erzeugt werden. Dazu muss die ID mit der *getEncodedID()*-Methode in eine kodierte Binärzahl in Form eines Strings umgewandelt werden.

Im nächsten Schritt kann dann die Bitmap Feld für Feld durchlaufen werden und der Wert des Strings an der Stelle $i * n + j$ eingefügt werden, wobei i die aktuelle

Zeile und j die die aktuelle Spalte repräsentiert.

In einem letzten Schritt wird die Grafik dann hochskaliert, damit sie besser in Dokumente eingefügt werden kann, und ein schwarzer Rahmen mit Hilfe der *addBorder*-Methode um das Barcodepattern gezogen werden, um die Anforderungen von ARToolKit zu befriedigen.

getCodedID()

Diese Methode erzeugt aus einer ID einen kodierten String der das Pattern des Barcodemarkers repräsentiert. Dieses geschieht indem die ID zunächst in eine Binärzahl mit der Länge $n^2 - 3$ umgewandelt wird. Jede Null in der Binärzahl repräsentiert ein weißes Feld und jede Eins ein schwarzes Feld im Barcodepattern. Damit das Pattern am Ende jedoch Rotationsinvariant ist sind insgesamt drei Felder des Musters bereits eine feste Farbe zugeordnet(in Abbildung 6.1 durch Kreuze in der entsprechenden Farbe dargestellt). Für diese vordefinierten Felder werden deshalb entweder eine Null oder eine Eins an den entsprechenden Stellen des Strings, der die Binärzahl repräsentiert , eingefügt.

6.6.3. Model

Diese Klasse repräsentiert ein Modell, welches vom Nutzer hochgeladen werden kann. Dabei speichert je eine Instanz der Klasse die für das Rendering relevanten Informationen von je einem Modell.

Die *Model*-Klasse beruht auf dem Grundgerüst von der *Cube*-Klasse, welche ein Teil von ARToolKit ist und einen simplen Würfel darstellt.

Diese Klasse wurde im Rahmen der Arbeit so modifiziert, dass sie beliebige Modelle speichern kann.

Dazu werden mittels der *ObjLoader*-Klasse die Daten aus der Obj-Datei geladen und in Buffern gespeichert. Des weiteren wird mit Hilfe der *TextureLoader*-Klasse die Textur geladen und eine Referenz auf diese erzeugt.

6.6.4. draw()

Die *draw*-Methode ist dafür zuständig das Modell zu rendern, in dem die Daten des Modells, sowie die Model-, View- und Projektionsmatrix des Markers, an eine Instanz der *ShaderProgram*-Klasse des Modells übergeben werden.

```

30 vn 0.0000 -1.0000 0.0000
31 vn 1.0000 0.0000 0.0000
32 vn 0.0000 0.0000 -1.0000
33 usemtl Material
34 s off
35 f 5/1/1 3/2/1 1/3/1
36 f 3/2/2 8/4/2 4/5/2

```

Abbildung 6.2.: Ausschnitt aus einer Obj-Datei. (Quelle: Eigene Darstellung)

6.6.5. ObjLoader

Die *ObjLoader*-Klasse ist dafür zuständig die Daten aus der Obj-Datei des Modells, welches der Nutzer hochgeladen hat, einzulesen und in das passende Format für OpenGL umzuwandeln.

Das Ganze geschieht über den Konstruktor 6.6.5, der die Daten ausliest und als Klassenvariablen des ObjLoaders speichert, sodass diese aus anderen Klassen abgefragt werden können.

ObjLoader()

Der Konstruktor der Klasse bekommt über einen Parameter den Pfad zur Obj-Datei übergeben.

Nachdem diese geladen wurde, muss die Datei Zeile für Zeile durchlaufen werden und anhand der Abkürzungen, mit denen jede Zeile beginnt, der Datentyp bestimmt werden.

Der Aufbau einer Obj-Datei unterliegt dabei dem folgenden Muster (vergleiche dazu auch Abbildung 6.2):

Jede Zeile enthält ein Datenobjekt, welches über die die Kennung zu einem bestimmten Datentyp zugeordnet werden kann. Zu Beginn der Datei werden alle Einzelteile des Modells definiert:

Vertex (v): Ein Vertex repräsentiert einen Eckpunkt des Objektes. Dazu werden in der Zeile die drei Koordinaten des Punktes gespeichert.

Textur (vt): Diese Zeile besteht aus zwei Koordinaten, welche einen Punkt in einem zweidimensionalen Bild, der Texturdatei, repräsentieren.

Normalenvektor (vn): Dieser Kennung folgenden die drei Koordinaten eines Normalenvektors

Damit OpenGL mit diesen Einzelteilen arbeiten kann, müssen diese noch zusammengesetzt werden.

Die dafür nötigen Informationen speichern die sogenannten Faces (deutsch: „Gesichter“), welche die Kennung f besitzen. Sie beschreiben die einzelnen Flächen des Modells. Da zuvor im ??) die Anforderung an die Obj-Dateien gestellt wurde, dass diese trianguliert werden, handelt es sich bei den Flächen immer um Dreiecke.

Die Zeile eines dieser Dreiecke speichert dafür für jeden Eckpunkt der Fläche drei Indices die auf jeweils einen der zuvor definierten Vertices, Texturkordinaten und Normalenvektoren verweisen. Dadurch wird neben der Koordinaten der Fläche, auch der zugehörige Ausschnitt der Textur und der Normalenvektor der Datei definiert. Das Programm erstellt aus diesen Daten drei Listen (jeweils eine für Vertices, Textrukoordinaten und Normalenvektoren), indem es für jede Fläche die Daten jedes Punktes an die entsprechende Liste anhängt.

Auf diese Listen kann dann über die Klassenvariablen des *ObjLoader*-Objektes zugegriffen werden.

6.6.6. TextureLoader

Die *TextureLoader*-Klasse hat die Aufgabe die Textur-Datei einzulesen und OpenGL zur Verfügung zu Stellen. Dazu wird zunächst mit *GLES20 glGenTextures* eine Textur erzeugt und eine Referenz in *textureHandle* gespeichert. Anschließend wird die Texturdatei des Modells als Bitmap an die Textur gebunden und die Filter zur Texturverarbeitung gesetzt.

6.7. ARToolKit-Klassen

Dieses Kapitel deckt die wichtigsten Klassen, die zur Umsetzung der Augmented Reality genutzt wurden ab. Diese Klassen wurden im Laufe der Arbeit an vielen Stellen angepasst oder erweitert, um die Anforderungen der Arbeit zu erfüllen.

6.7.1. ARController

Der *ArController* ist ein *Singelton* der ARToolKit-Bibliothek, welcher den Zugriff auf die nativen Funktionen, die in der *ARX_jni*-Klasse definiert werden, regelt.

6.7.2. ARX_jni

Die *ARX_jni*-Klasse enthält die *Java Native Interface*-Funktionen und stellt die Schnittstelle zur Nativen ARToolKit Bibliothek dar. Diese Bibliothek enthält alle grundlegenden Funktionen von ARToolKit in C++ geschrieben.

6.7.3. ARFragment

Die *ARFragment*-Klasse wurde im Laufe der Arbeit implementiert und ist von ihrer Funktionalität identisch zu der *ARActivity*-Klasse von ARToolKit. Sie wurde lediglich an die Anforderungen eines Fragments angepasst, um in das Userinterface der Anwendung eingefügt werden zu können.

Bei ihr handelt es sich um eine Abstrakte Klasse, die das Grundgerüst des Fragments bildet, dass die AR-Inhalte anzeigen soll.

Die Hauptfunktionalität dieser Klasse ist es die graphische Oberfläche für die AR-Anwendung zu erzeugen. Dazu erzeugt die Klasse ein *GLSurfaceView* und weist diesem einen Renderer zu, der später dazu zuständig ist die Modelle auf dem *GLSurfaceView* anzuzeigen.

Des weiteren wird der Kamerastream geöffnet und dem *GLSurfaceView* zugewiesen.

supplyRenderer()

Mit Hilfe dieser Methode wird der Klasse ein *ARRenderer* übergeben. Diese Methode muss von der Klasse, die von dem *ARFragment* erbt überschrieben werden, um den *ARRenderer* zuzsetzen.

supplyFrameLayout()

Diese abstrakte Methode übergibt das *FrameLayout* auf dem die Augmented Reality erzeugt werden soll an die *ARFragment*-Klasse. Dazu muss diese Methode von der Erbenden Klasse überschrieben werden.

6.7.4. ARRender

Der *ARRenderer* ist eine abstrakte Klasse aus der ARToolKit-Bibliothek und bildet das Grundgerüst für den Renderer, welcher in einer Anwendung die ARToolKit nutzt implementiert werden muss.

Dazu wird unter anderem das Rendern des Videohintergrunds initialisiert.

6.7.5. ShaderProgram

Diese abstrakte Klasse bildet das Grundgerüst eines Shader Programms.

Diese Klasse definiert einige Konstanten die OpenGL die Anzahl an Koordinaten, aus denen ein einzelnes Datenelement besteht, mitteilt. Dazu gehören die Positions-, die Farb-, der Textur- und der Normalenvektorkoordinaten.

Diese Werte werden dann noch in die Anzahl an Bytes, die ein einzelnes Datenelement umfasst, umgerechnet.

createProgram()

Diese Methode erzeugt ein Shader Programm, in dem es einen Vertex und einen Fragment Shader zu einem Programm verknüpft.

Dazu wird zu nächst ein OpenGL Shader Programm erstellt und anschließend werden die beiden Shader an das Programm gebunden. Im letzten Schritt können dann beide Shader miteinander verbunden werden.

setProjectionMatrix()

Diese Methode erlaubt es die *ProjectionMatrix* des Shader Programms zusetzen.

setModelViewMatrix()

Diese Methode erlaubt es die *ModelViewMatrix* des Shader Programms zusetzen.

6.8. Tests

Während der Implementierung wurde mit Hilfe von ausführlichen Tests die Funktionalität neuer Features sichergestellt und eine ausführliche Testdokumentation angefertigt. Dadurch sollten Fehler, Probleme und mögliche Verbesserungen entdeckt und festgehalten werden.

Das Ziel war es für jedes neue Feature einen Test durchzuführen.

Dafür wurde die Anwendung mit Hilfe von Android Studio auf ein Androidgerät geladen und dort getestet. Dieses bot die Möglichkeit die Tests in einer realistischen, praxisnahen Umgebung durchzuführen, um noch bessere Erkenntnisse über die Alltagstauglichkeit der neuen Features zu erhalten.

Als Testgerät diente dabei ein Huawei P30 Pro. Dabei wurden die einzelnen Grundfunktionalitäten auf verschiedenen getestet.

6.8.1. Augmented Reality

Bei dem Testen neuer AR-Funktionalitäten wurde darauf Wert gelegt, dass die einzelnen Durchläufe in der selben Testumgebung und unter den gleichen Umständen stattfinden, um eine Vergleichbarkeit zwischen den verschiedenen Versionen herzustellen. So konnte nach jedem Test zusätzlich festgestellt werden, ob sich eventuell die bestehende Eigenschaften im Vergleich zur Vorgängerversion verschlechtert oder verbessert hatten.

Zu diesem Zweck wurde für jeden Durchlauf der in Abbildung 6.3 gezeigte Versuchsaufbau gewählt.

Zum Testen des Trackings wurde ein Testdokument angefertigt. Dieses Dokument

Verweis auf
Testdoku-
ment(Anhang)

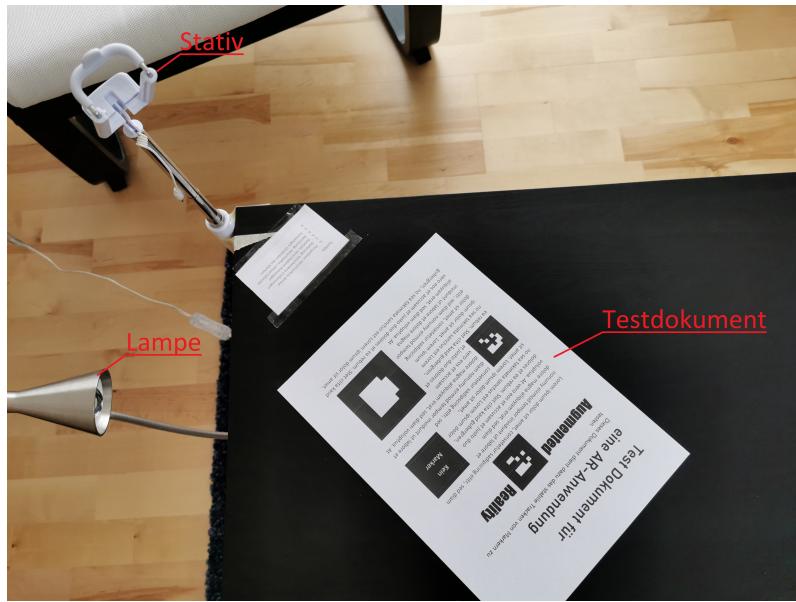


Abbildung 6.3.: Der Testaufbau für jeden Durchlauf. (Quelle: Eigene Darstellung)

wurde im Laufe der Entwicklung an die neuen Features angepasst und optimiert. Grundlegend bildet das Dokument einen oder mehrere Marker ab. Gegebenenfalls sind die Marker in Textpassagen eingebunden, um den Schwierigkeitsgrad für die Markererkennung zu erhöhen.

Während des Testlaufs wurden jedes mal vier Testfälle durchlaufen, die in Abschnitt ?? beschrieben werden. Dabei wurden sowohl die Neuerungen getestet, als auch Veränderungen in den bereits bestehenden Features festgehalten.

Jeder Testfall wurde dabei mit Hilfe der in Android enthaltenen Funktion „Bildschirmrekorder“ aufgezeichnet und in dem entsprechenden Testbericht dokumentiert. Die Testfälle an sich beruhen auf den Eigenschaften des SIFT-Algorithmus, welcher Bildmerkmale extrahiert, die invariant gegenüber Rotation, Translation, Skalierung und partiell invariant gegenüber Helligkeitsveränderungen sind (Nischwitz u. a., 2011, S. 345). Mithilfe des Algorithmus können dieselben Objekte in zwei verschiedenen Aufnahmen wiedererkannt werden.

Da auch beim Marker Tracking ein ähnliches Verfahren angewandt werden muss, sind auch hierbei die genannten Eigenschaften relevant. Deshalb wurden die folgenden Testfälle gewählt, die primär das Marker Tracking testen sollten, aber zusätzlich auch eine Evaluation des Renderns der Modelle ermöglichten:

Perspektivische Invarianz

Dieser Testfall diente dazu das Tracking aus verschiedenen Perspektiven zu testen. Dazu wurde die Kamera auf das Testdokument gerichtet und anschließend der Winkel zum Dokument so verändert das verschiedenen, perspektivische Verzerrungen der Marker erzeugt wurden.

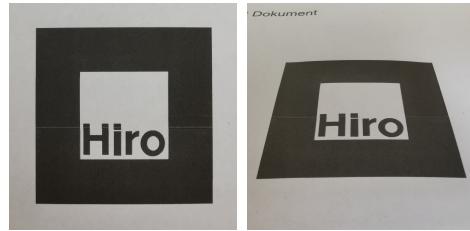


Abbildung 6.4.: Perspektivische Verzerrung eines Markers. (Quelle: Eigene Darstellung)

Skalierungsinvarianz

Dieser Testfall diente dazu das Tracking aus verschiedenen Entfernungen zu testen. Dazu wurde die Kamera langsam auf das Testdokument zu- und wegbewegt, um verschiedenen Markergrößen bzw. -auflösungen zu erhalten.

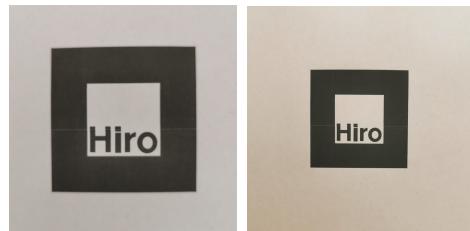


Abbildung 6.5.: Verschiedene Skalierungen eines Markers. (Quelle: Eigene Darstellung)

Rotationsinvarianz

Dieser Testfall diente dazu das Tracking von Markern mit unterschiedlichen Rotationen zu testen, dazu wurde die Kamera auf das Dokument gerichtet und anschließend wurde letzteres langsam rotiert, um zu evaluieren, ob die Marker auch mit unterschiedlichen Rotationen korrekt erkannt werden.

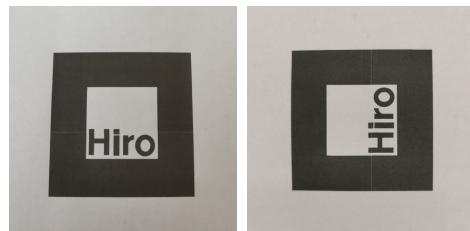


Abbildung 6.6.: Verschiedene Rotationen eines Markers. (Quelle: Eigene Darstellung)

Belichtungsinvarianz

Dieser Testfall diente dazu das Tracking von Markern gegenüber unterschiedlichen Belichtungen zu testen. Dazu wurde mit Hilfe einer Lampe die Belichtung des Testdokumentes verändert.



Abbildung 6.7.: Verschiedene Belichtungen eines Markers. (Quelle: Eigene Darstellung)

Trackinggeschwindigkeit

Besserer Name

Dieser Testfall sollte die Geschwindigkeit des Trackings, sowie die Robustheit testen. Dazu wurden ein oder mehrere Marker kurzzeitig mit der Hand verdeckt, um zu testen, ob anschließend alle Marker wieder erfolgreich getrackt wurden und wie schnell dieses erfolgte.

6.8.2. Markergenerierung

Um die Markergenerierungsfunktion des Prototyps zu testen wurden über der Programmcode der Anwendung so angepasst, dass die Nummer des getrackten Markers in der Konsole ausgegeben wurde. Anschließend wurden mit Hilfe der Anwendung eine Stichprobe von fünf Markern generiert. Für die Stichprobe wurden dabei sowohl die äußeren Grenzen 0 und 4194303, als auch drei zufällig generierte Werte zwischen den beiden Grenzen gewählt.

Nun wurden zuletzt die generierten Marker manuell in das Tracking eingefügt und die Ausgabe mit der ID verglichen.

6.8.3. Laden eigener Modelle

Zum Testen des Ladens wurde eine Sammlung an verschiedenen Modellen zusammengestellt.

Diese bestanden zum einen Teil aus simplen Modellen, die mit der Anwendung Blender selbst erstellt wurden, und zum anderen Teil aus Modellen aus dem Internet.

Diese Modelle wurden dann in der Anwendung hochgeladen und der zugehörige Marker wurde auf ein Dokument gedruckt.

Anschließend wurde getestet, ob das Modell korrekt gerendert wird, wenn der zugehörige Marker getrackt wurde.

Kapitel 7

Evaluation

In diesem Kapitel werden die Ergebnisse der Arbeit evaluiert. Dabei wird zum einen auf den entwickelten Prototypen eingegangen und zum anderen das Konzept und die Anwendung von Augmented Reality im Bildungsbereich evaluiert.

7.1. Evaluation des Prototyps

Dieses Kapitel dient dazu den Prototypen anhand der in Kapitel 4 gestellten Anforderungen zu evaluieren. Dazu wird zunächst das Testverfahren, das parallel zu der Implementierung genutzt wurde beschrieben und im Anschluss der finale Prototyp evaluiert.

7.1.1. Bewertung des finalen Prototyps

7.1.2. Zusammenfassung

-
- Belichtung nicht einfach zu testen - starke Belichtungswechsel teilweise kurze Aussetzer

Am Ende
schreiben

Kapitel 8

Fazit

8.1. Fazit

8.2. Ausblick

Literaturverzeichnis

- [Android a] ANDROID: *Application Fundamentals*. Webquelle. – URL <https://developer.android.com/guide/components/fundamentals?authuser=1>. – letzter Abruf: 16.08.2020
- [Android b] ANDROID: *Fragments*. Webquelle. – URL <https://developer.android.com/guide/components/fragments>. – letzter Abruf: 21.10.2020
- [Android c] ANDROID: *Introduction to Activities*. Webquelle. – URL <https://developer.android.com/guide/components/activities/intro-activities>. – letzter Abruf: 16.08.2020
- [Android d] ANDROID: *OpenGL ED*. Webquelle. – URL <https://developer.android.com/guide/topics/graphics/opengl>. – letzter Abruf: 21.10.2020
- [Android e] ANDROID: *Platform Architecture*. Webquelle. – URL <https://developer.android.com/guide/platform>. – letzter Abruf: 16.08.2020
- [Android f] ANDROID: *Understand the Activity Lifecycle*. Webquelle. – URL <https://developer.android.com/guide/components/activities/activity-lifecycle>. – letzter Abruf: 16.08.2020
- [Balzert 2011] BALZERT, H.: *Lehrbücher der Informatik*. Bd. 1: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. 3. Spektrum Akademischer Verlag, 2011. – ISBN 978-3-8274-1706-0
- [Böhm und Fuchs 2002] BÖHM, R. ; FUCHS, E.: *Wirtschaftsinformatik*. Bd. 1: *System-Entwicklung in der Wirtschaftsinformatik*. 5. vdf Hochschulverlag, 2002. – ISBN 978-3-7281-2762-4
- [Billinghurst 2002] BILLINGHURST, M.: Augmented Reality in Education / New Horizons for Learning. 2002. – Forschungsbericht
- [Buchner 2017] BUCHNER, J.: Offener Geschichtsunterricht mit Augmented Reality. In: *Medienimpulse* 55 (2017), Nr. 1. – URL <https://journals.univie.ac.at/index.php/mp/article/view/mi1061>

- [Damberger 2016] DAMBERGER, T.: Augmented Reality als Bildungsenhancement? In: *Medienimpulse* 54 (2016), Nr. 1. – URL <https://journals.univie.ac.at/index.php/mp/article/view/1532>
- [de Vries, J.] DE VRIES, J.: *Learn OpenGL - Hello Trinagle*. Webquelle. – URL <https://learnopengl.com/Getting-started>Hello-Triangle>. – letzter Abruf: 20.08.2020
- [Dey u. a. 2018] DEY, A.m ; BILLINGHURST, M. ; LINDEMAN, R. W. ; SWAN, J. E.: A Systematic Review of 10 Years of Augmented Reality Usability Studies: 2005 to 2014. In: *Frontiers in Robotics and AI* 5 (2018), S. 30. – URL <https://www.frontiersin.org/article/10.3389/frobt.2018.00037>
- [Diegmann u. a. 2015] DIEGMANN, P. ; SCHMIDT-KRAEPELIN, M. ; EYNDEN, S. an den ; BASTEN, D.: Benefits of Augmented Reality in Educational Environments - A Systematic Literature Review. In: *Wirtschaftsinformatik Proceedings 2015*, URL <https://aisel.aisnet.org/wi2015/103>, 2015
- [Gargenta 2011] GARGENTA, M.: *Learning Android - Building Applications for the Android Market*. 1. O'Reilly, 2011. – ISBN 978-1-449-39050-1
- [Geroimenko 2020] GEROIMENKO, V.: *Augmented Reality in Education : A New Technology for Teaching and Learning*. 1. Springer, 2020. – ISBN 9783030421564
- [Hedberg u. a. 2018] HEDBERG, H. ; NOURI, J. ; HANSEN, R.: A Systematic Review of Learning Through Mobile Augmented Reality. In: *International Journal of Interactive Mobile Technologies* 12 (2018), Nr. 3, S. 75–85. – URL <https://www.online-journals.org/index.php/i-jim/article/view/8404>
- [Khronos Group a] KHRONOS GROUP: *Geometry Shader*. Webquelle. – URL https://www.khronos.org/opengl/wiki/Geometry_Shader. – letzter Abruf: 02.09.2020
- [Khronos Group b] KHRONOS GROUP: *OpenGL Overview*. Webquelle. – URL <https://www.opengl.org/about/>. – letzter Abruf: 09.08.2020
- [Khronos Group c] KHRONOS GROUP: *Rendering Pipeline Overview*. Webquelle. – URL https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview. – letzter Abruf: 20.08.2020
- [Khronos Group d] KHRONOS GROUP: *Tessellation*. Webquelle. – URL <https://www.khronos.org/opengl/wiki/Tessellation>. – letzter Abruf: 02.09.2020

- [Kind u. a. 2019] KIND, H. ; FERDINAND, J. ; JETZKE, T. ; RICHTER, S. ; WEIDE, S.: Virtual und Augmented Reality: Status quo, Herausforderungen und zukünftige Entwicklungen / TAB - Büro für Technikfolgen-Abschätzung beim Deutschen Bundestag. 2019. – TAB-Arbeitsbericht
- [Klein 2006] KLEIN, G.: *Visual Tracking for Augmented Reality*, University of Cambridge, Dissertation, 2006
- [Lamb u. a. a] LAMB, L. ; VAUGHAN, B. ; BELL, D. ; BUX, T.: *Architecture overview*. Webquelle. – URL <https://github.com/ar toolkitx/ar toolkitx/wiki/Architecture-overview>. – letzter Abruf: 22.10.2020
- [Lamb u. a. b] LAMB, L. ; VAUGHAN, B. ; BELL, D. ; BUX, T.: *What is ar toolkitX?* Webquelle. – URL <http://www.ar toolkitx.org>. – letzter Abruf: 22.10.2020
- [Mehler-Bicher und Steiger 2014] MEHLER-BICHER, A. ; STEIGER, L.: *Augmented Reality - Theorie und Praxis*. Bd. 1. 2. De Gruyter Oldenbourg, 2014. – ISBN 978-3-11-035385-3
- [Milgram u. a. 1994] MILGRAM, P. ; TAKEMURA, H. ; UTSUMI, A. ; KISHINO, F.: Augmented Reality: A class of displays on the reality-virtuality continuum. In: *Telemanipulator and Telepresence Technologies* Bd. 2351, SPIE, 1994, S. 282–292
- [Murphy 2009] MURPHY, M. L.: *Beginning Android - Master Android from first principles and begin the journey toward your own successful Android applications!* 1. Apress, 2009. – ISBN 978-1-4302-2420-4
- [Nischwitz u. a. 2011] NISCHWITZ, A. ; FISCHER, M. ; HABERÄCKER, P. ; SOCHER, G.: *Computergrafik und Bildverarbeitung - Band II: Bildverarbeitung*. Bd. 2. 3. Vieweg+Teubner Verlag, 2011. – ISBN 978-3-8348-1712-9
- [Owen u. a. 2002] OWEN, C. ; XIAO, F. ; MIDLIN, P.: What is the best fiducial?, 2002, S. 8 pp.. – ISBN 0-7803-7680-3
- [Robertson und Robertson 2012] ROBERTSON, S. ; ROBERTSON, R.: *Mastering the Requirements Process: Getting Requirements Right*. Bd. 1. 3. Pearson Education, 2012. – ISBN 978-0-321-81574-3
- [Rupp und die SOPHISTen 2014] RUPP, C. ; SOPHISTEN die: *Requirements-Engineering und -Management : aus der Praxis von klassisch bis agil*. 6. Hanser, 2014. – ISBN 9783446438934
- [Shreiner u. a. 2006] SHREINER, D. ; WOO, M. ; NEIDER, J. ; DAVIS, T.: *OpenGL programming guide - the official guide to learning OpenGL, version 2.5*. Addison-Wesley, 2006. – ISBN 0-321-33573-2

[Ćuković u. a. 2015] ĆUKOVIĆ, S. ; GATTULLO, M. ; PANKRATZ, F. ; DEVEDZIC, G. ; CARRABBA, E. ; BAIZID, K.: Marker Based vs. Natural Feature Tracking Augmented Reality Visualization of the 3D Foot Phantom, 2015

[Winter 2018] WINTER, Andreas: *SRS Anforderungen*. Vorlesung Softwaretechnik I. 2018

Anhang A

Beispielanhang

Beispiel für einen Anhang!

Erklärung

Hiermit versichere ich, Johannes Scheibe, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Johannes Scheibe