

B A C H E L O R A R B E I T

Entwicklung von Darstellungs- und
Interaktionsmöglichkeiten in Virtual Reality
für das Cranach Digital Archive

Vorgelegt an der TH Köln
Campus Gummersbach
im Studiengang
Medieninformatik

ausgearbeitet von:
NIKOLAS BECKEL
(Matrikelnummer: 11103435)

Erster Prüfer: Prof. Christian Noss
Zweiter Prüfer: Matthias Groß

Gummersbach, im November 2020

Inhaltsverzeichnis

Abbildungsverzeichnis	3
Tabellenverzeichnis	4
1 Einleitung	5
2 Grundlage und Datenbasis	5
2.1 Das Cranach Digital Archive	5
2.1.1 Lucas Cranach der Ältere	6
2.1.2 Martin Luther als Junker Jörg	7
2.2 Konzept und Nutzung von Virtual Reality	7
3 Theoretische Einordnung	8
3.1 Aktueller Forschungsstand von Virtual Reality	9
3.1.1 Entstehung	9
3.1.2 Wahrnehmungsaspekte	10
3.1.3 VR-Brillen	12
3.1.4 Native und webbasierte Anwendungen	14
3.2 Virtual Reality im musealen Kontext	16
3.2.1 Chancen und Risiken	17
3.2.2 Optimierung des Lernprozesses	19
3.2.3 Aktuelle Lösungen und Umsetzung	19
4 Prototyping	21
4.1 Anforderungen an das VR-System	21
4.2 Auswahl der Technologien	23
4.2.1 Unity	24
4.2.2 Unreal Engine	25
4.2.3 React 360	27
4.2.4 A-Frame	27
4.2.5 Three.js	29
4.2.6 Ergebnis	29
4.2.7 Weitere Hilfsmittel	30
4.3 Entwicklung der Prototypen	30
4.3.1 Technischer Prototyp	31
4.3.2 Lucas Cranachs Werkstatt	44
4.3.3 Neuronales Netz	48
5 Ergebnisse	50

6	Diskussion	51
7	Fazit	52
8	Nachwort	53
9	Quellenverzeichnis	53
	Erklärung über die selbständige Auffassung der Arbeit	55

Abbildungsverzeichnis

1	Google Cardboard für Smartphones.	13
2	Unity Entwicklungsumgebung.	24
3	Unreal Engine Entwicklungsumgebung.	26
4	Unreal Engines Blueprint-System.	26
5	HTML-Element aus A-Frame.	28
6	Grafische Benutzeroberfläche von A-Frame.	28
7	Erste Version einer <code>index.html</code> -Datei.	31
8	Erste virtuelle Welt.	33
9	Image-Tag von A-Frame für das Einbinden von Bildern.	33
10	Gemälde innerhalb der virtuellen Welt.	34
11	Event Registrierung innerhalb von HTML.	34
12	Aufbau einer Komponente in A-Frame.	35
13	A-Frame Entität mit hinzugefügter Komponente.	35
14	a-box mit der Komponente <code>standing-area</code> .	36
15	TypeScript-Code der Komponente <code>standing-area</code> .	37
16	Funktionsfähige Standing Area zum Teleportieren.	38
17	Beschreibung zum Gemälde der ID <code>DE_MdbKL_946</code> .	39
18	Beschreibung zum Gemälde der ID <code>DE_MdbKL_946</code> innerhalb der virtuellen Welt.	39
19	JSON-Struktur eines Gemäldes.	41
20	Gemälde-Interface in TypeScript.	41
21	Abrufen der Daten innerhalb der <code>paintings.json</code> .	42
22	Codeabschnitt für die Generierung des Gemäldes und dessen Informationen.	42
23	<code>createDetailPoints</code> innerhalb der <code>painting</code> -Komponente.	43
24	Ansicht des ersten fertigen Prototypen.	44
25	Detailpunkte auf dem Gemälde nach einem Klick auf den „Detailansicht“-Button.	45
26	Eine Nahaufnahme des Gemäldes mit der ID <code>DE_MdbKL_946</code> .	45
27	Einbinden eines glTF-Modells in A-Frame über <code><a-assets></code> .	46
28	Aktuelles Gemälde von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.	47
29	Dokumente von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.	47
30	Sound-Button der Archivalien von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.	48
31	<code>document</code> -Komponente als HTML-Implementation.	49
32	Übersicht des neuronalen Netzes aus der Inspektor-Perspektive von A-Frame.	50

Tabellenverzeichnis

1	Funktionsumfang aller Technologien.	29
2	Informationen eines Gemäldes auf http://lucascranach.org	40

1 Einleitung

Virtual Reality ist nicht mehr nur Zukunftsmusik oder ein Anwendungsbereich für Forscher großer Konzerne; immer mehr kommt Virtual Reality auf den Massenmarkt. Wo zu Beginn leistungsfähige Computer benötigt wurden, entstehen nun All-in-One VR-Brillen¹. Die Anwendungsbereiche von Virtual Reality sind vielfältig: Gaming, Unterhaltung durch VR-Filme oder das Treffen von Freunden in virtuellen Welten². Die Pandemie durch das Virus SARS-CoV-2 hat uns auch gezeigt, dass immer mehr digitale Lösungen benötigt werden. Durch solche Krisen bekommen plötzlich Anwendungsbereiche, die nicht als notwendig betrachtet wurden, völlig neue Relevanz. Durch Virtual Reality ist man nicht mehr an einen Ort gebunden und kann gewisse Aktionen statt in der realen Welt, in einer virtuellen Welt ausführen. Von der realen Welt losgelöst, entstehen so auch völlig neue Möglichkeiten, diese aufzuführen. In dieser Bachelorarbeit wird das digitale Archiv der Cranachs³ als Ressource genutzt, um neue Möglichkeiten der Darstellung für Kunst und Kultur zu erforschen. Dabei sollen aktuelle Entwicklungen, unter anderem aus dem musealen Bereich, miteinbezogen und deren Erkenntnisse berücksichtigt werden. Auf Basis der recherchierten Ergebnisse sollen mehrere Prototypen entwickelt werden, die zeigen, wie Virtual Reality im Bereich Kunst und Kultur angewendet werden kann. Dabei sollen die aktuellsten Technologien miteinbezogen und abgewägt werden, mit welcher die besten Ergebnisse erzielt werden können.

Frage: Wie lassen sich Gemälde und Archivalien des Cranach Digital Archives in einer virtuellen Welt darstellen? Und wie kann mit diesen interagiert werden?

2 Grundlage und Datenbasis

In diesem Kapitel wird auf die Grundlagen von Virtual Reality, auf den Künstler Lucas Cranach der Ältere und auf das Cranach Digital Archive eingegangen. Letzteres stellt die Datenbasis für dieses Projekt dar, welches in der Entwicklung der Prototypen eingesetzt wird. Auf diesem Grundwissen baut das weitere Kapitel auf, welches genauer die allgemeine Technologie von Virtual Reality thematisiert und weitergehend den musealen Anwendungsbereich miteinbezieht.

2.1 Das Cranach Digital Archive

Das Cranach Digital Archive ist eine Initiative und ein visionäres Forschungsprojekt, welches sich zum Ziel gesetzt hat, alle relevanten Informationen, Dokumente und Werke der Cranachs in einer digitalen Datenbank der Forschung und Öffentlichkeit zur Verfügung

¹Oculus Quest 2 - All-in-One VR-Brille | <https://www.oculus.com/quest-2/> (21.09.2020)

²Mozilla Hubs | <https://labs.mozilla.org/projects/hubs/> (21.09.2020)

³Cranach Digital Archive | <http://lucascranach.org/> (21.09.2020)

zu stellen. So konnten bisher über 1600 Gemälde und 14000 Abbildungen digitalisiert und auf der Webseite lucascranach.org veröffentlicht werden, welche frei über das Internet zugänglich ist. Bei den digitalisierten Aufnahmen der Werke handelt es sich nicht nur um hochauflösende Gemälde, sondern auch Röntgenaufnahmen, Infrarotreflektogramme und Archivalien. Dadurch lassen sich verschiedene Gemälde der Cranachs aus einer völlig neuen Perspektive betrachten, denn durch die Infrarotreflektogramme lassen sich zum Beispiel Unterzeichnungen eines Gemäldes erkennen, welches das bloße Auge niemals sehen könnte. [Heydenreich u. a., 2017]

2.1.1 Lucas Cranach der Ältere

Lucas Cranach d. Ä. war nicht nur ein erfolgreicher Künstler während der Renaissance, sondern auch ein guter Freund Martin Luthers. Gemeinsam ermöglichten sie eine moderne Auffassung der Kunst. Lucas Cranach d. Ä. wurde 1472 als Sohn des Malers Hans Moller geboren, der ihn in der Zeichenkunst unterrichtete. Obwohl er früh mit dem Malen begonnen hatte, wurde Lucas Cranach d. Ä. erst in Wien um 1502 als Künstler bekannt. Um 1504/05 verließ er Wien und nahm den Beruf des Hofmalers in Wittenberg für Friedrichs III. entgegen, von welchem er auch sein zukünftig verwendetes Wappen verliehen bekommen hatte. Dieses Wappen, eine „geflügelte, bekrönte und einen Ring im Maul tragende Schlange“ [Heydenreich u. a., 2017, S. 15], sollte fortan als Signet von Lucas Cranach und seiner Werkstatt werden. Nach dem Tod von Friedrich III., diente er dessen Sohn und Nachfolger Johann I., welcher das Amt jedoch nur sieben Jahre lang führen konnte und schließlich an seinen Sohn, Johann Friedrich I. abgegeben hat. Bis zu seinem Tod stand Lucas Cranach d. Ä. im Dienst dieser drei Kurfürsten. Lucas Cranach d. Ä. zeichnete sich nicht nur durch die Qualität seiner Gemälde aus, sondern war auch dafür bekannt, produktiv zu arbeiten. Im Gegensatz zu seiner Konkurrenz malte er seine Gemälde nicht in freier Natur, sondern baute sich bereits in Wittenberg eine Werkstatt auf, die es ihm erlaubte, seine Produktivität auf eine neue Ebene zu bringen. Auch die Kunstgattung „Druckgrafik“, die Martin Luther als Graben bezeichnete, erlaubte es Lucas Cranach d. Ä. einzelne Gemälde mehrfach zu drucken. Er war jedoch nicht nur als Künstler erfolgreich, sondern belegte in Wittenburg zwischen 1519 und 1544/45 das Amt des Ratsherren und einige Jahre davon auch als Kämmerer oder Bürgmeister. Die letzten Jahre von Lucas Cranach d. Ä. waren durch den Schmalkaldischen Krieg geprägt und beeinflusst, da Johann Friedrich I. mit seinen Truppen gegen den Kaiser kämpfte. Nach einer Niederlage und Verlusten seines Territoriums, folgte Lucas Cranach d. Ä. Johann Friedrich I. in die Gefangenschaft und verblieb dort von 1547 bis 1552. Nach der Gefangenschaft kehrte Lucas Cranach d. Ä. nach Weimar zurück und starb am 16. Oktober 1553 im Haus seiner Tochter. Zusammenfassend war Lucas Cranach d. Ä. nicht nur ein einfacher Künstler, sondern aufgrund seiner Qualität, Produktivität und humanistischem Denken seiner Zeit voraus. So legte er einen wichtigen Grundstein für die moderne Kunst. [Heydenreich u. a.,

2017]

2.1.2 Martin Luther als Junker Jörg

Als Martin Luther 1511 nach Wittenburg zurückkehrte und 1512 die Professur für Bibelauslegung übernommen hatte, traf er zum ersten Mal auf Lucas Cranach d. Ä. Die beiden pflegten nicht nur engen Kontakt zueinander, sondern ergänzten sich auch in ihrer Arbeit. Martin Luthers Gedanken der Reformation konnten nicht nur in Schrift verbreitet werden, sondern auch durch das künstlerische Talent und die Produktivität von Lucas Cranach d. Ä. grafisch unterlegt werden. Martin Luther nahm nachweislich die Dienste von Lucas Cranach d. Ä. für Tiefenholzschnitte entgegen, welche auch für theologische Argumentationen benutzt wurden. Lucas Cranach d. Ä. trug auch dazu bei, dass es ein öffentliches Bild von Martin Luther gab und versuchte dieses zu manifestieren. Als 1521 die Reichsacht über Martin Luther verhängt wurde, musste er aus Schutz seinen Tod durch einen inszenierten Überfall vortäuschen. Sein enger und guter Freund Lucas Cranach d. Ä. war jedoch über die Inszenierung informiert. Als Martin Luther zurück nach Wittenburg kehrte, um die dortigen Unruhen zu besänftigen, trat er unter dem Pseudonym „Junker Jörg“ auf. Auch hier nahm Lucas Cranach d. Ä. eine entscheidende Rolle ein, da er in dieser Zeit Bildnisse von Martin Luther anfertigte. Beispielsweise wurde zu dieser Zeit ein Bildnisholzschnitt von Martin Luther als Junker Jörg gefertigt, welche absichtlich zur Medienstrategie verwendet wurde. So überbrachte Lucas Cranach d. Ä. die Nachricht, dass Martin Luther den inszenierten Überfall überlebte und vermittelte damit auch das Bild eines entschlossenen und visionären Mannes. [Heydenreich u. a., 2017]

In diesem Forschungsprojekt wird sich auf die Datenbasis von Martin Luther als Junker Jörg des Cranach Digital Archive bezogen. Zur Entwicklung einer Antwort auf die Forschungsfrage wird nicht die gesamte Datenbasis des Cranach Digital Archive benötigt, da eine geringe Datennenge zur Entwicklung von Virtual Reality-Szenen bereits ausreichen. Das Team hinter lucascranach.org hat die Daten bereits digitalisiert und aufgearbeitet, sodass Werke und Gemälde, die miteinander verwandt sind oder in Beziehung stehen, bereits zusammenhängend verknüpft sind.

2.2 Konzept und Nutzung von Virtual Reality

Virtual Reality stammt aus dem Wissenschaftsgebiet der Computergrafik, denn eine virtuelle Realität besteht immer aus einer dreidimensionalen Simulation [Dörner u. a., 2013, S. 13]. Es gibt mehrere Wege, eine virtuelle Realität zu simulieren. Nicht immer ist eine VR-Brille notwendig, doch ist heutzutage das Nutzen solch einer Brille, um in eine virtuelle Realität einzutauchen, state of the art. Das zeigt auch die schnelle Entwicklung dieser Branche, die vermehrt auf VR-Brillen setzt, welche ohne einen zusätzlichen Computer auskommen. Die gesamte Echtzeitsimulation der virtuellen Realität wird innerhalb der

VR-Brille berechnet⁴. Sollte die Leistung der Brille nicht ausreichen, lassen sich diese optional mit einem Computer verbinden. Bei VR-Displays werden stereoskopische Verfahren angewandt [Dörner u. a., 2013, S. 13], wodurch Objekte, die durch die VR-Brille gesehen werden, einen Tiefeneffekt erhalten. Dieser Tiefeneffekt lässt virtuelle Objekte real erscheinen, da sie erstmals für unser Gehirn eine erkennbare räumliche Position aufweisen. Ein weiterer wichtiger Bestandteil von Virtual Reality ist die blickabhängige Bildgenerierung [Dörner u. a., 2013, S. 13]. Bei der Bewegung des Kopfes, und damit des Blickpunktes des menschlichen Auges, wird die virtuelle Welt aus dem Blickwinkel, in welchen das menschliche Auge in der virtuellen Welt schaut, neu generiert. Da Virtual Reality es erlaubt, Objekte aus verschiedenen Perspektiven zu sehen und auch mit diesen zu interagieren, gibt es vielfältige Einsatzmöglichkeiten. Das Virus SARS-CoV-2 hat gezeigt, dass es immer wichtiger wird von zu Hause aus arbeiten zu können oder Dienstleistungen von zu Hause aus entgegenzunehmen. Auch Museen oder Kunstmuseen können von Virtual Reality profitieren, da so die Kunst auf verschiedenen Wegen übermittelt werden kann. Die Darstellung ist nicht mehr an das klassische Bild eines großen Raumes mit Gemälden an der Wand gebunden, sondern kann räumliche Limitierungen der realen Welt überwinden. Das Cranach Digital Archive ist eine große Datenbank mit vielen Gemälden und Archivalien. Einige Gemälde haben eine Beziehung zu anderen Gemälden, da sie zum Beispiel zur selben Zeit entstanden sind, oder auf demselben Bildträger angebracht wurden. Virtual Reality kann Forschern, aber auch Kulturinteressierten, dabei helfen, die großen Datensets besser aufzunehmen und zu verstehen [Dörner u. a., 2013, S. 9]. Im nächsten Kapitel wird der aktuelle Forschungsstand von Virtual Reality erläutert und aufgezeigt wie dieser aktuell für Museen eingesetzt werden kann.

3 Theoretische Einordnung

In diesem Kapitel wird das Thema Virtual Reality genauer behandelt, auf welchem Forschungsstand sich diese Branche befindet und welche Lösungen bereits auf dem Markt sind. Mit Hilfe von Fachliteratur zu diesem Themenbereich sollen die aufgeworfenen Forschungsfragen beantwortet werden. Dieses Kapitel stellt das wissenschaftliche Fundament für die Entwicklung der Prototypen, die anhand der in diesem Kapitel gewonnenen Fakten entwickelt werden sollen. Zunächst wird der aktuelle Forschungsstand der VR-Branche betrachtet, welche technologischen Möglichkeiten und welche Produkte im musealen Bereich bereits existieren. Durch andere VR-Produkte, die ebenfalls im musealen Bereich entwickelt wurden, lassen sich Fehler vermeiden und etablierte Ideen für das eigene Produkt umsetzen.

⁴Oculus Quest 2 - All-in-One VR-Brille | <https://www.oculus.com/quest-2/> (21.09.2020)

3.1 Aktueller Forschungsstand von Virtual Reality

Virtual Reality ist mittlerweile eine Branche, welche Einzug in viele Bereiche der Industrie gefunden hat. Die Automobilindustrie nutzt Virtual Reality um Prototypen zu entwickeln. Durch dieses Vorgehen lassen sich Materialkosten sparen und auftretende Fehler frühzeitig erkennen. Die Medizin kann anhand von Virtual Reality die Ausbildung von angehenden Ärzten vereinfachen, da diese in einer virtuellen Welt den menschlichen Körper besser betrachten und dementsprechend verstehen können. Und die Spiele-Industrie, die einen großen Teil zur Entwicklung von VR-Brillen auf dem Massenmarkt beigetragen haben, erfährt ganz neue Möglichkeiten den Spielern ein einzigartiges Spielerlebnis zu geben. Virtual Reality wird in weiteren Branchen Anwendung finden und in naher Zukunft zu einer Technologie gehören, auf die sich nicht mehr verzichten lässt.

3.1.1 Entstehung

Virtual Reality ist keine neuartige Technologie, sondern existiert bereits seit den 60er Jahren. Ivan Sutherland forschte als Erster an immersiven Technologien und führte so mit seinem Buch „The Ultimate Display“ den Rechner, das Design, die Konstruktion und Navigation virtueller Welten zusammen. In den 80er Jahren entwickelte die NASA erstmals mit dem Projekt VIEW (Virtual Environment Interface Workstations) einen multisensorischen Arbeitsplatz, welcher für die Simulation virtueller Weltraumstationen genutzt wurde. Der Begriff Virtual Reality wurde erstmals 1987 von einem Wissenschaftler namens Jaron Lanier verwendet. Er und Thomas Zimmermann gründeten gemeinsam die Firma VPL, welche am „DataGlove“ arbeitete und diesen verkaufte. Dieser Handschuh konnte mit Hilfe von Glasfasern Fingerdaten erfassen. Neben dem Handschuh verkauften sie auch das „EyePhone“, eine Weiterentwicklung des Head-Mounted-Displays von Ivan Sutherland aus den 60ern. 1989 wurde ein weiterer Meilenstein erreicht, denn die Firma Polhemus entwickelte einen elektromagnetischen Tracker, welcher ein Ziel in bestimmter Entfernung vom Rechner bestimmen konnte. Zeitgleich entstand BOOM (Binocular Omni-Orientation Monitor) von Fake Space Labs. BOOM war ein 3D-Sichtgerät mit einem 1280x1024 Pixel Display, welches 1991 zum ersten Mal Anwendung im „Virtual Windtunnel“ von Steve Bryson fand, einem Bereich der Luft- und Raumfahrt. 1988 kamen mehr und verschieden hochwertige Arbeitsplätze für den Bereich Grafik auf den Markt. Silicon Graphics konnte sich mit ihrer SGI Reality Engine 1995 weltweit durchsetzen und wurde so zum Standard dieser Branche. Damit kamen dann die ersten kommerziellen VR-Softwaresysteme auf den Markt. Die nächste große technologische Innovation wurde daraufhin vom Unternehmen SensAble Technologies Inc. vorgestellt, welches vom Massachusetts Institut of Technologies gegründet wurde. SensAble entwickelte ein haptisches Gerät namens „PHANTom“, welches berührt werden musste, um eine Kraftrückkopplung spüren zu können. Anfang der 90er Jahre, nachdem schon einige technische Innovationen

auf dem Markt erschienen sind, wurden viele wichtige Forschungen im Bereich der Virtual Reality unternommen, die sich vor allem auf Stereoleinwände konzentriert haben. Nach den Tracking-Systemen auf Basis von Elektromagnetismus folgten Systeme auf Basis von Ultraschall. Gegen 2000 wurde der Ultraschall jedoch von Infrarot abgelöst. Tracking-Systeme auf Basis von Infrarot finden noch heute Anwendung im Bereich der modernen VR-Brillen. Die von Silicon Graphics entwickelte SGI Reality Engine, welche in vielen VR-Softwaresystemen verwendet wurde, wurde langfristig auch von Computern abgelöst. Die damit einhergehende Preissenkung (circa um $\frac{1}{5}$) erlaubte umfangreichere Forschungen. Historisch betrachtet hat auch Deutschland einige Unternehmen vorzuweisen, die sich bereits seit 1998 mit VR-Systemen beschäftigen. So sind beispielsweise VRCOM, RTT und IC:IDO zu nennen. Ein ständiger Austausch zum Wissenschaftsgebiet Virtual Reality fand demnach international und auf Länderebene statt. Schließlich konnte sich die IEEE VR Konferenz international durchsetzen und ist jährlich mit etwa 500 Teilnehmern einer der größten Konferenzen zu diesem Thema. Seit 2003 hat zudem die Gesellschaft für Informatik eine Fachgruppe für das Thema Virtual und Augmented Reality. [Dörner u. a., 2013, S. 19–21]

Heutzutage haben sich die Head-Mounted-Displays, oder einfach VR-Brillen genannt, mit Infrarot als Sensorik auf dem Massenmarkt durchgesetzt. Diese sind sehr kompakt und benötigen, je nach Anbieter, keinen Computer mehr. Sie funktionieren autark und übernehmen jegliche Berechnung in der VR-Brille. Vorreiter dieser Branche sind unter anderem Oculus von Facebook und Vive von HTC in Kooperation mit Valve.

3.1.2 Wahrnehmungsaspekte

Wie Menschen Informationen wahrnehmen, ist ein wichtiger Aspekt für die Gestaltung von virtuellen Welten. In einer perfekten virtuellen Realität würde der Konsument alle Sinnesindrücke, die er aus der realen Welt kennt, in der simulierten Welt in gleicher Qualität und Quantität empfinden können [Dörner u. a., 2013, S. 17]. Doch so eine perfekte virtuelle Realität zu entwickeln ist nach aktuellem Stand der Technik nicht möglich. Die heutigen VR-Systeme und -Technologien konzentrieren und beziehen sich auf den visuellen, akustischen und haptischen Sinn des Menschen [Dörner u. a., 2013, S. 34]. Dementsprechend nutzt ein ideales VR-System jene Effekte in Kombination mit einem Tracking-System [Slater, 2009]. Die heutige Technologie ist bereits auf dem Stand, dass alle drei genannten Sinne angesprochen und beeinflusst werden können. Die VR-Brillen selbst besitzen ein Tracking-System für das Bewegen des Kopfes sowie die Veränderung der räumlichen Position und mehrere 3D-Eingabegeräte, die das Bewegen der Hände analysieren können. Die Immersion einer VR-Erfahrung wird auch nicht zwingend durch die Bildschirme und das Tracking-System bestimmt, sondern durch die Anzahl der ausführbaren Aktionen innerhalb der virtuellen Welt [Slater, 2009]. So hilft der technologische Fortschritt der heutigen VR-Brillen sehr bei der Entwicklung von virtuellen Welten, jedoch müssen auch

die durchführbaren Aktionen gut umgesetzt sein. Denn einer der wichtigsten Aspekte für das Wahrnehmen von virtuellen Welten ist das Gefühl der Präsenz [Slater, 2009]. Die Präsenz beschreibt das Gefühl, in der virtuellen Welt zu sein [Slater, 2009]. Dabei weiß der Benutzer jedoch zu jedem Zeitpunkt, dass er nicht wirklich dort ist [Slater, 2009]. Grundlegend lässt sich der Aspekt der Präsenz in drei Unteraspekte aufteilen: Das Gefühl der Ortsillusion, die Plausibilitätsillusion und die Involviertheit [Dörner u. a., 2013, S. 18–19]. Eine virtuelle Welt sollte immer unter Berücksichtigung dieser drei Gefühle entwickelt werden, da eine gute Umsetzung dieser die beste Nutzungserfahrung für einen Benutzer sicherstellen kann.

Bei der Ortsillusion handelt es sich um die starke Illusion an einem Ort zu sein, mit dem Wissen, dass man nicht wirklich dort ist [Slater, 2009]. Einer der wichtigsten Kriterien, um diese Illusion beim Benutzer zu erreichen, ist die in Kapitel 2.2 erwähnte blickpunkt-abhängige Bildgenerierung der VR-Brille [Dörner u. a., 2013, S. 18].

Bei der Plausibilitätsillusion handelt es sich um das Gefühl, dass die Geschehnisse innerhalb der virtuellen Welt wirklich passieren, mit dem Wissen, dass dies nicht wirklich passiert [Slater, 2009]. Eine Schlüsselkomponente, um diese Illusion hervorzurufen, sind Geschehnisse, die nicht vom Benutzer ausgelöst wurden, sich jedoch auf ihn beziehen [Slater, 2009]. Das können beispielsweise auf den Benutzer zufliegende Projektilen sein oder ein virtueller Mensch, der den Benutzer anspricht [Dörner u. a., 2013, S. 18–19]. Die Geschehnisse die innerhalb der virtuellen Realität stattfinden müssen jedoch nicht physikalisch realistisch sein [Slater, 2009] und können dementsprechend auch fantasievoll und unrealistisch sein. Als Beispiel kann ein Spiel genommen werden, in welchem der Benutzer von einem Magier mit einem Feuerball angeschossen wird; der Benutzer hat das Ereignis nicht selber ausgelöst, wird jedoch in das Geschehen eingebunden. Eine Plausibilitätsillusion entsteht, obwohl das Szenario unrealistisch ist. So kann diese durch das Einbauen von realistischen Effekten, wie Schatten oder die Darstellung des eigenen Körpers, verstärkt werden [Slater, 2009].

Das Anzeigen des Körpers des Benutzers kann sogar den Effekt der Orts- und Plausibilitätsillusion verstärken, da das Gefühl in der virtuellen Realität zu sein realer erscheint und dementsprechend auch Geschehnisse die den Benutzer betreffen [Slater, 2009]. Genauso können beide Illusionen durch das Ausnutzen der Ängste eines Menschen hervorgerufen werden [Slater, 2009]. Leidet der Benutzer unter Höhenangst, dann wird er ebenfalls in der virtuellen Welt unter Höhenangst leiden, wenn er sich beispielsweise auf einem Hochhaus befindet und die Kante herunterschaut. Demnach verstärkt sich das Gefühl, dass die virtuelle Realität Wirklichkeit sei, obwohl der Benutzer zu jedem Zeitpunkt weiß, dass es sich um eine Simulation handelt. So kann einer immersiven virtuellen Realität gesprochen werden, wenn der Benutzer so handelt und agiert, wie er es in einer realen Welt tun würde („Response-as-if-real“ (RAIR)) [Slater, 2009]. Diese genannten Aspekte (Immersion, Ortsillusion, Plausibilitätsillusion und das Darstellen des eigenen Körpers) stellen

ein Framework dar, wonach virtuelle Realitäten entwickelt werden können [Slater, 2009]. Diese Aspekte stellen auch Gefühle dar, welche der Benutzer während seiner Erfahrung empfinden kann und die es gilt gut umzusetzen. Fehlerhafte Darstellungen können die Ortsillusion negativ beeinflussen. Darüber hinaus ist es durch das Beheben der Fehler jedoch möglich, jene Illusion wiederherzustellen [Slater, 2009]. Anders verhält sich die Plausibilitätsillusion, welche schwer zu erreichen ist. Sofern diese einmal gebrochen ist, kann sie in der aktuellen Erfahrung des Benutzers nicht mehr oder nur schwer wieder aufgenommen werden [Slater, 2009]. Ein Beispiel für den Bruch der Plausibilitätsillusion ist ein virtueller Mensch, mit dem kommuniziert werden kann, er aber nur in sehr einfachen Phrasen antwortet [Dörner u. a., 2013, S. 19]. So geht die Glaubwürdigkeit der Illusion verloren. Deshalb sollten virtuelle Realitäten so wenig Raum für Fehler bieten wie nur möglich [Slater, 2009].

Die Involviertheit ist neben der Orts- und Plausibilitätsillusion ein weiteres Gefühl, welches die Qualität der Nutzungserfahrung steigert. Sie beschreibt das Verhältnis der Aufmerksamkeit und des Interesses eines Benutzers zur virtuellen Realität [Witmer und Singer, 1998, S. 227]. Je höher die Aufmerksamkeit und das Interesse des Benutzers, desto höher ist die Qualität der Nutzungserfahrung. Geräusche aus der realen Umgebung, oder das unbequeme Sitzen der VR-Brille, können bereits die Aufmerksamkeit und das Interesse stark senken und somit die Qualität der Erfahrung beeinträchtigen [Witmer und Singer, 1998, S. 227]. Abgesehen der drei oben genannten Aspekte, lässt sich die Nutzungserfahrung durch Berücksichtigung technischer Aspekte verbessern. Darunter fällt die Bildwiederholungsrate der Anwendung, die Bildschirmauflösung, das Gesamtausmaß und die Latenz des Trackings, der Grad des Sichtfelds und die visuelle Qualität der Umgebung und ihrer Objekte [Slater, 2009]. Unter Berücksichtigung dieser Gesichtspunkte können qualitativ hochwertige virtuelle Realitäten entwickelt werden, welche durch das Ansprechen der richtigen Gefühle des Benutzers eine erfolgreiche Nutzungserfahrung hervorbringen können.

3.1.3 VR-Brillen

Neben den gesamten Aspekten, die es für die Erstellung von Nutzungserfahrungen innerhalb virtueller Welten zu berücksichtigen gibt, muss das technische System ebenfalls jene Aspekte abbilden können. In der heutigen Zeit haben sich VR-Brillen (auch Head-Mounted-Displays genannt) in Kombination mit 3D-Eingabegeräten durchgesetzt. Diese sind mittlerweile sehr kompakt, leistungsstark und preislich erschwinglich. Dadurch, dass Hardware kleiner und günstiger wird, kann der Trend beobachtet werden, dass VR-Brillen auch ohne einen stationären Computer funktionieren. Den Vorteil von VR-Brillen, im Gegensatz zu Projektionssystemen, bildet die Möglichkeit, diese überall anziehen und nutzen zu können [Dörner u. a., 2013, S. 129]. Dementsprechend haben sich Head-Mounted-Displays als VR-Systeme auf dem Massenmarkt durchgesetzt, da diese kompakt sind und



Abbildung 1: Google Cardboard für Smartphones.

nicht zwingend einen Computer benötigen. Zudem ist je nach Anwendungsfall eine VR-Brille nicht notwendig, da heutige mobile Endgeräte, wie Smartphones, bereits genügend Leistung für eine Echtzeit-Simulation mitbringen. So gibt es Aufsätze für das Smartphone, mit welchen es sich in eine VR-Brille umfunktionieren lässt (siehe Abb. 1⁵). Da diese Lösung kostengünstig und einfach umzusetzen ist, gilt dies als gute Einstiegsmöglichkeit für Personen, die gerne VR-Erfahrungen erleben würden, sich jedoch nicht eine autarke VR-Brille leisten möchten. Doch ist durch das Fehlen der 3D-Eingaberäte bei dieser Lösung, je nach Anwendungsfall, eine Einschränkung festzustellen. Abhängig vom dem jeweiligen Smartphone kann auch eine geringe Bildschirmauflösung die in Kapitel 3.1.2 genannten Aspekte beinträchtigen und einen Bruch der Illusionen hervorrufen. Projektionssysteme, die die simulierte Umgebung auf ganzen Leinwänden darstellen, haben den Vorteil, dass sich der Benutzer nicht unmittelbar vor dem Bildschirm befindet [Dörner u. a., 2013, S. 134]. Dadurch können keine einzelnen Pixel mit dem Auge erfasst werden.

⁵Google Cardboard | <https://arvr.google.com/cardboard/> (18.10.2020)

Daraus ergibt sich, dass die Bildschirmauflösung ein wichtiges Kriterium zur guten Anwendung von VR-Brillen ist [Dörner u. a., 2013, S. 134]. Aktuelle Unternehmen, die sich mit VR-Brillen beschäftigen und diese Technologie maßgeblich vorgeben, sind unter anderem Facebook mit Oculus⁶, HTC mit Vive⁷, Playstation mit PlaystationVR⁸ und Valve Corporation mit der Valve Index⁹. Samsung beschäftigt sich ebenfalls mit Virtual Reality, jedoch bieten diese nur eine hochwertige Brille ohne Rechenleistung an. Für den vollständigen Funktionsumfang wird zusätzlich ein Samsung Smartphone vorausgesetzt¹⁰. Auch Nintendo hat mit der Konsole Nintendo Switch den Weg zu Virtual Reality gefunden¹¹. Das VR-Set für die Nintendo Switch bietet sich hilfreich für Kinder an, um erste Erfahrungen mit Virtual Reality machen zu können. Auch Google ist mit dem Google Cardboard in diesen Markt involviert, bietet jedoch keine VR-Brille mit integrierter Hardware an.

3.1.4 Native und webbasierte Anwendungen

Viele VR-Brillen bedeuten auch viele Software-Development-Kits (SDK) die damit einhergehen. VR-Anwendungen können heutzutage mit Hilfe von vielen Programmen oder Frameworks entwickelt werden. So kann ein Produkt nativ für eine Plattform entwickelt werden oder durch webbasierte Anwendungen plattformunabhängig funktionieren. Beide Ansätze bieten ihre Vor- sowie Nachteile und sind daher vom jeweiligen Anwendungsfall abhängig. Native Lösungen bieten in der Softwareentwicklung eine geeignete Performance, da diese Anwendungen mit Programmiersprachen entwickelt werden die effizient und maschinennah sind. Je nach Hersteller gibt es auch für das Betriebssystem eigene Programmiersprachen, die dann für das jeweilige Ökosystem optimiert werden können. Webanwendungen, die in der Regel auf Basis von JavaScript laufen, werden nicht kompiliert sondern interpretiert und der Quellcode wird zur Laufzeit vom Browser gelesen und ausgeführt. Durch eine große und aktive Community hat sich JavaScript in den letzten Jahren zu einer universellen Programmiersprache entwickelt und findet Anwendung in fast allen Gebieten der Softwareentwicklung. Diese Entwicklung der letzten Jahre hat gezeigt, dass der Mensch weniger Zeit vor dem PC im Internet verbringt, sondern hauptsächlich mobil darauf zugreift [Ater, 2017, S. 1]. Durch das Betrachten von Internetseiten auf einem mobilen Endgerät wurde der Ansatz mobile-first immer wichtiger [Ater, 2017, S. 1]. Die Vorteile, die native Anwendungen mehrere Jahre mit sich brachten, waren, neben der besseren Leistung, erweiterte Grafikmöglichkeiten, Standortermittlung, Push-

⁶Oculus | https://www.oculus.com/?locale=de_DE (18.10.2020)

⁷HTC VIVE | <https://www.vive.com/de/> (18.10.2020)

⁸PlaystationVR | <https://www.playstation.com/de-de/explore/playstation-vr/> (18.10.2020)

⁹Valve Index | <https://www.valvesoftware.com/de/index/headset> (19.10.2020)

¹⁰Samsung Gear VR | <https://www.samsung.com/de/wearables/gear-vr-r323/> (18.10.2020)

¹¹Nintendo Labo VR-SET | <https://www.nintendo.de/Nintendo-Labo/Nintendo-Labo-1328637.html> (18.10.2020)

Benachrichtigungen, Offlineverfügbarkeit und Startbildschirm-Verknüpfungen [Ater, 2017, S. 3]. Diese Funktionen waren damals notwendig für erfolgreiche Apps, da diese dem Benutzer eine bessere Nutzungserfahrung ermöglichen. Heutzutage hat sich das Web als Plattform weiterentwickelt, sodass jene Funktionen nicht mehr nur nativen Anwendungen vorenthalten sind. Sogenannte Progressive Web Apps erlauben sich an nativen Funktionen der Plattform zu bedienen, bieten jedoch weiterhin die Flexibilität, die das Web mit sich bringt [Ater, 2017, S. 2]. So können webbasierte Anwendungen offlinefähig gestaltet werden. Darüber hinaus können Push-Benachrichtigungen von Webseiten verschickt werden und zudem lassen sich Desktop- und Startbildschirmverknüpfungen über moderne Browser anlegen [Ater, 2017, S. 5–6]. Durch das Ausblenden der URL-Leiste des Browsers und das Ausführen der webbasierten Anwendung im Vollbildmodus, ähneln sich den native Anwendungen [Ater, 2017, S. 6]. Für das Ziel einer Anwendung, diese möglichst zugänglich zu machen, ist die Entwicklung einer webbasierten Lösung vorteilhaft, da plattformunabhängig entwickelt werden kann und der Benutzer direkt am Ziel ist, sobald dieser die Webseite öffnet. Es muss kein zusätzlicher Download oder eine zusätzliche Installation erfolgen und auch das Pflegen mehrerer Versionen in den verschiedenen App Stores entfällt. So kann der Benutzer unabhängig seiner Plattform und seinem Gerät die Anwendung benutzen. Da der Benutzer demnach an keine Plattform gebunden ist, ist die Möglichkeit höher, ein vielfaches mehr an potenziellen Benutzern zählen zu können [Ater, 2017, S. 3]. Es ist zunehmend schwieriger geworden Benutzer für den Download einer App zu motivieren [Ater, 2017, S. 3]. Die meisten Benutzer gehen bereits auf dem Weg zum jeweiligen Store und der darauffolgenden Installation verloren [Ater, 2017, S. 4]. Application Programming Interfaces (API) wie die Web Share und Web Share Target API sind nur ein Teil der zukünftigen Entwicklung, die immer mehr Funktionen ins Web bringen, welche zuvor nur für native Anwendungen zur Verfügung standen [Ater, 2017, S. 245]. So findet auch Virtual Reality, durch die WebXR-Schnittstelle (ehemals WebVR), Anwendung im Browser [Ater, 2017, S. 245]. In Kombination mit WebGL lassen sich komplexe, rechenintensive und überzeugende VR-Erfahrungen entwickeln [Ater, 2017, S. 245]. Die Schnittstelle WebXR ist kompatibel mit den gängigsten VR-Brillen und kann auch ihre jeweiligen 3D-Eingabegeräte lesen, wodurch der Entwickler Zugang zu allen Informationen bekommt, die auch eine native Desktop-Anwendung mitbringen würde [Ater, 2017, S. 245]. Im Punkt Leistung sind nach wie vor native Anwendungen einige Schritte voraus, doch gibt es jetzt schon bereits Lösungen, die an diesem Problem arbeiten. WebAssembly¹² ist eine niedere Programmiersprache, welche wenig Abstraktion zur Befehlssatzarchitektur des Computers bietet [Haas u. a., 2017, S. 185]. Dadurch, dass WebAssembly zu ByteCode kompiliert wird, kann nahezu die native Leistung mit einer webbasierten Anwendung erreicht werden [Haas u. a., 2017, S. 186]. So bleibt weiterhin die Plattformunabhängigkeit geboten, da WebAssembly über den Browser ausgeführt werden kann. Jedoch ist WebAs-

¹²Offizielle WebAssembly Webseite | <https://webassembly.org/> (02.11.2020)

sembly noch nicht so verbreitet wie JavaScript und bietet deshalb weniger Technologien und Möglichkeiten, zur Entwicklung von VR-Anwendungen. Zum aktuellen Zeitpunkt bietet die Unity Engine die Möglichkeit, VR-Anwendungen mit Hilfe ihrer Engine zu entwickeln und diese für moderne Browser zu exportieren¹³. Die Unreal Engine kompiliert den geschriebenen Code ebenfalls zu WebAssembly, hat die HTML5-Entwicklung jedoch als Erweiterung ausgelagert, die durch die Community weiterentwickelt werden soll¹⁴. Da es in der klassischen Softwareentwicklung nicht sinnvoll ist in Assemblersprache zu entwickeln, macht es ebenfalls wenig Sinn selbstständig eine VR-Anwendung in WebAssembly zu entwickeln. Deshalb wird eine höhere Programmiersprache benötigt, die eine höhere Abstraktion bietet. So ist es am einfachsten Unity oder Unreal Engine als Software zu nutzen, um VR-Anwendungen auf Basis von WebAssembly zu generieren. Nach wie vor ist die Entscheidung, ob nativ oder für das Web entwickelt werden soll, abhängig vom Anwendungsfall. Für das Cranach Digital Archive bietet sich eine webbasierte Lösung an, da die Darstellung von Gemälden und Archivalien keine hochkomplexen Berechnungen benötigt. Das aktuelle Archiv ist über das Internet zu erreichen und lässt sich auf Desktop-PCs und mobilen Endgeräten betrachten. Das Archiv ist bereits über die größte Plattform aufzurufen, dementsprechend sollte die VR-Lösung ebenfalls über das Web zugänglich sein. So können auch Kunstinteressierte, die sich keine hochwertige VR-Brille leisten möchten, die Gemälde und Archivalien mit ihrem Smartphone erkunden. Durch eine plattformunabhängige Lösung ist weniger Wartung notwendig und das Bereitstellen in die jeweiligen App Stores entfällt, sodass viel Zeit und Ressourcen gespart werden können [Ater, 2017, S. 248]. Webbasierte Anwendungen benötigen die wenigsten Ressourcen für eine hohe Zugänglichkeit, wodurch sich dieser Entwicklungsansatz auch für das Forschungsprojekt anbietet, da die vorhandenen Ressourcen gering sind.

3.2 Virtual Reality im musealen Kontext

Die Idee Kunst in einer virtuellen Realität darzustellen, ist keine neue Idee. Immer wieder wagten sich Museen daran, stellten Konzepte auf, sammelten Erfahrungsberichte und entwickelten eigene Produkte [Heidsiek, 2019]. Es ist ein wichtiger und verständlicher Schritt, dass Virtual Reality im musealen Bereich vermehrt und umfangreich Anwendung findet. Museen versuchen Gefühle, Erfahrungen sowie Emotionen zu übermitteln und durch den Einsatz von Virtual Reality besteht die Möglichkeit, den Besucher in Szenarien zu versetzen, die in der Vergangenheit geschehen sind. Der Besucher kann auch Fantasiewelten der Künstler wahrnehmen, um ein besseres Verständnis für ihre Kunst zu erlangen. Umfragen ergaben, dass ein generelles Interesse an neuartigen Technologien im Museum

¹³WebAssembly in Unity | <https://blogs.unity3d.com/2018/08/15/webassembly-is-here/> (02.11.2020)

¹⁴Unreal Engine Dokumentation für HTML5 Development | <https://docs.unrealengine.com/en-US/Platforms/HTML5/index.html> (02.11.2020)

vorhanden ist [Heidsiek, 2019, S. 34]. Die Besucher erhoffen sich dadurch, dass Museen unterhaltsamer, lehrreicher und zugänglicher werden [Heidsiek, 2019, S. 34]. Gerade Virtual Reality lässt Museen und Informationen für die breite Masse zugänglicher werden, denn die Besucher sind nicht mehr gezwungen das Museum physisch zu besuchen. Der Anwendungsbereich ist vielfältig und die Technologie noch nicht ausgereizt. Doch bergen neue Technologien neben ihren Chancen auch Risiken, die berücksichtigt und diskutiert werden müssen. Deshalb wird in diesem Kapitel auf die Risiken und auch auf die Chancen eingegangen, die ein Museum bei einem Einsatz von Virtual Reality erwarten kann. Da das Cranach Digital Archive nicht das erste Museum ist, welches Kunst digitalisiert, sollen bereits entwickelte Produkte analysiert werden, deren Erkenntnisse und Ideen dabei helfen, bereits begangene Fehler zu vermeiden und geeignete Prototypen zu entwickeln.

3.2.1 Chancen und Risiken

Museen sind Stätten des Wissens. Ihre Aufgabe besteht darin, Menschen Wissen zu vermitteln. Meist über vergangene Erfahrungen oder Geschehnisse. Durch die Digitalisierung und das Internet war es erstmals möglich, historische Gegenstände oder Gemälde mit der ganzen Welt zu teilen, unabhängig des Standortes eines Betrachters. Dementsprechend liegt der Mehrwert von digitalen Museen darin, eine erhöhte Vervielfältigung und Verbreitung von Wissen zu betreiben [Hünnekens, 2002, S. 17]. Auch das Anwendungsgebiet Virtual Reality bietet nach dem Internet und der Digitalisierung eine neue Möglichkeit Informationen innovativ zu übermitteln [Heidsiek, 2019, S. 52]. So wird das Museum als Stätte des Wissens effizienter gestaltet und ermöglicht mehr Menschen zu erreichen, als es vorher möglich war. Das Cranach Digital Archive geht hier mit gutem Beispiel voran und bietet eine umfangreiche Datenbank zu den Werken und Archivalien der Cranachs. Im Vergleich zu statischen Objekten, wie hängende Gemälde an Wänden, lassen sich durch digitale Lösungen verschiedene und neue Darstellungen ermöglichen [Hünnekens, 2002, S. 17]. Gerade Virtual Reality bietet hier eine sehr interessante Möglichkeit, völlig neue Perspektiven zu erschaffen, die niemals in der realen Welt erreicht werden können. Ein Vorteil in Virtual Reality liegt darin, dass die Gesetze von Raum und Zeit nicht in einer virtuellen Realität gelten [Hünnekens, 2002, S. 140]. So kann der Mensch seiner Begierde, alltägliche Erfahrungen zu überschreiten, nachkommen [Hünnekens, 2002, S. 140]. Diese unendlichen Möglichkeiten bringen jedoch auch eine ethische Verantwortung zur Präsentation musealer Gegenstände mit sich. Das Loslösen von Raum und Zeit kann sich negativ auf die historischen Inhalte auswirken [Hünnekens, 2002, S. 141]. So könnte sogar von Entweihung der historischen Gegenstände gesprochen werden [Hünnekens, 2002, S. 141]. Deshalb liegt es in der Verantwortung der Museen die historische Integrität eines Gegenstandes oder einer Erfahrung nicht zu verletzen [Heidsiek, 2019, S. 38]. Dies kann vermieden werden, indem kenntlich gemacht wird, was original ist und was selbstständig hinzugefügt wurde [Heidsiek, 2019, S. 38]. Sobald virtuelle Realitäten erfundene Aspekte

beinhalten, und diese nicht als solche gekennzeichnet wurden, handelt es sich nicht mehr um eine historische Überlieferung, sondern um persönliche und subjektive Wahrnehmungen [Heidsiek, 2019, S. 38]. Fotorealistische Darstellungen von unwahren Ereignissen oder Objekten können verstärkt dazu führen, dass der Benutzer von der fälschlichen Tatsache überzeugt wird [Heidsiek, 2019, S. 39]. So hat jedes Museum, dass Virtual Reality als Möglichkeit zur Betrachtung kunstrelevanter Objekte anbieten möchte, die ethische Verantwortung, dem Benutzer zu jedem Zeitpunkt kenntlich zu machen, was eine historische Übermittlung und was virtuelle Realität ist. Zudem besteht durch Virtual Reality die Gefahr, dass Museen, auf lange Sicht betrachtet, obsolet werden und alles über eine VR-Brille erforscht werden kann [Hünnekens, 2002, S. 142–143]. Besucher die Virtual Reality in Museen genutzt haben, betrachten diese Technologie jedoch als Ergänzung zum Museum und nicht als einen vollständigen Ersatz, da der Hauptgrund eines Museumsbesuchs die Betrachtung der Originalobjekte sei [Heidsiek, 2019, S. 79]. Originale Ausstellungsstücke haben demnach weiterhin einen nicht reproduzierbaren Charakter, welcher sich nicht in einer virtuellen Realität darstellen lässt [Heidsiek, 2019, S. 93]. So schwächt die Darstellungen von Originalobjekten als Kopie in einer virtuellen Welt die Authentizität dieser [Heidsiek, 2019, S. 93]. Besucher, die vorher eine Online-Ausstellung eines Museums gesehen haben, entschieden sich sogar aufgrund dessen für das Besuchen des Museums in der realen Welt [Halpern und Katz, 2015, S. 777–778]. So kann eine VR-Erfahrung, die über das Web zu erreichen ist, nicht als Ersatz für das Museum gesehen werden, sondern auch als innovatives und modernes Marketingwerkzeug. Dementsprechend ist nach aktuellem Stand nicht davon auszugehen, dass die Technologie Virtual Reality das Museum in naher oder mittlerer Zukunft obsolet macht. Die Museumslandschaft sieht in Virtual Reality die Chancen, neue und andere Geschichten zu vermitteln, langfristig für Besucher attraktiv zu bleiben und auch eine neue Zielgruppe zu erreichen [Heidsiek, 2019, S. 34–35]. Da Emotionen Teil eines Museumsbesuchs sind, lassen sich durch den Einsatz von Virtual Reality neue und intensivere Emotionen hervorrufen. Empfundene Emotionen steigern bei einem Lernprozess nachweislich den erhöhten Lernerfolg [Heidsiek, 2019, S. 29]. So konnte nachgewiesen werden, dass VR-Erfahrungen bei musealen Ausstellungen zu erhöhtem Interesse, erhöhter Zufriedenheit und Vergnügen geführt haben [Heidsiek, 2019, S. 69–72]. Dadurch können schwer verständliche Themen leichter übermittelt werden und der Informationsaustausch zwischen Museum und Mensch optimiert werden. Das deckt sich auch mit der Chance, langfristig für Besucher attraktiv zu bleiben und eine neue Zielgruppe anzusprechen, die vorher nichts mit Museen anfangen konnte. Aber auch die aktuelle Zielgruppe eines Museums kann, unabhängig des Alters, etwas mit Virtual Reality anfangen [Heidsiek, 2019, S. 75]. Dadurch wird keine bestimmte Zielgruppe ausgeschlossen.

3.2.2 Optimierung des Lernprozesses

Wie bereits im vorherigen Unterkapitel erwähnt, tragen Emotionen nachweislich zum Lernerfolg bei. Um diese Emotionen bei Besuchern zu aktivieren und entsprechende Gefühle hervorzurufen, bietet es sich an, Lebensräume der darzustellenden Personen oder Objekte in der virtuellen Welt nachzustellen [Heidsiek, 2019, S. 35–36]. Anhand dessen kann sich der Benutzer noch besser in die Situation hineinbegeben und komplexe Gefühle und Emotionen besser wahrnehmen. Storytelling stellt hierbei eine mögliche Methode dar [Heidsiek, 2019, S. 35–39]. Bei VR-Erfahrungen im Museum lassen sich zwei Emotionen beobachten: Die erfahrungsbezogene und inhaltsbezogene Emotion [Heidsiek, 2019, S. 69]. Erstere beschreibt die empfundene Emotion, die während der gesamten Nutzungs-erfahrung entstanden ist und letztere die Emotion, die empfunden wird, während sich mit einem genauen Ausstellungsstück beschäftigt wird. Virtual Reality ruft vermehrt die erfahrungsbezogenen Emotionen hervor [Heidsiek, 2019, S. 69], was im Hinblick auf die Technologie logisch erscheint. Denn durch Virtual Reality liegt der Fokus nicht nur auf genau einem einzigen Objekt, sondern meist auf einer ganzen Szene und die Erfahrung innerhalb dieser. Die erfahrungsbezogenen Emotionen sind diejenigen, die einen besseren Lerneffekt mit sich bringen, sodass Museen dadurch als unterhaltsamer und aufregender bezeichnet [Heidsiek, 2019, S. 69]. Traditionelle Museen rufen hauptsächlich inhaltsbezogene Emotionen hervor [Heidsiek, 2019, S. 69], da der Fokus bei Ausstellungen oft auf einzelnen Ausstellungsstücken liegt. Immersive Umgebungen, die Virtual Reality in hoher Form bieten kann, tragen ebenfalls zu einem optimierten Lernprozess bei, da der Lernende den Inhalt selbstständig steuern und manipulieren kann [Halpern und Katz, 2015, S. 777]. Ebenso kann der Lernende die Lerngeschwindigkeit eigenmächtig bestimmen [Halpern und Katz, 2015, S. 777], was die Konzentration erhöht. Nichtsdestotrotz muss berücksichtigt werden, dass nicht jeder Anwender sofort perfekt mit der virtuellen Umgebung agieren kann. So kann durch zu komplizierte Interaktionen und Darstellungen eher eine negative Erfahrung hervorgerufen werden, die dem Lernprozess eher schadet [Halpern und Katz, 2015, S. 777]. An dieser Stelle ist die Studie des Deutschen Auswandererhauses zu erwähnen, welche eine einfache Bedienung durch blickgesteuerte Interaktion gewährleistet [Heidsiek, 2019, S. 38]. Dadurch müssen die Benutzer lediglich eine Brille aufsetzen und können sofort innerhalb der Anwendung navigieren, da keine Tasten der Eingabegeräte auswendig gelernt werden müssen. Auch das Gefühl der Präsenz, welches bereits in Kapitel 3.1.2 behandelt wurde, trägt dazu bei, wie gut der Lernende Informationen aufnimmt [Halpern und Katz, 2015, S. 777].

3.2.3 Aktuelle Lösungen und Umsetzung

Da es viele unterschiedliche Technologien und Hardware für die Umsetzung von VR-Ausstellungen gibt, lassen sich auch viele verschiedene Wege fassen, solche VR-Erfahrungen

zu entwickeln. Das Deutsche Auswandererhaus beispielsweise entschied sich für das Samsung Odyssey Virtual Reality Headset, welches ohne Controller funktioniert und intuitiv bedienbar ist [Heidsiek, 2019, S. 35]. Sie entwickelten eine VR-Erfahrung, die durch Storytelling vermittelt wurde [Heidsiek, 2019, S. 35–39]. Dabei konnten zwei VR-Erfahrungen erlebt werden. Eine der beiden bestand aus einer auditiven und blickgesteuerten Interaktion, während sich die Andere aus einem 360-Grad-Video zusammensetzte [Heidsiek, 2019, S. 38]. Aufgrund der positiven Studienergebnisse und Bewertungen der Probanden, scheint die Entwicklung einer VR-Erfahrung mit blickgesteuerter Interaktion sinnvoll, da keine Einarbeit in die Eingabegeräte der VR-Brille notwendig ist. Dadurch erhält die Anwendung eine intuitive Steuerung. Auch das Vermitteln einer Geschichte innerhalb der VR-Erfahrung fördert das Interesse der Benutzer [Heidsiek, 2019, S. 35–39]. Google bietet ebenfalls eine simple Anwendung, die über ein Google Daydream kompatibles Smartphone ausgeführt werden kann¹⁵. Hierbei lassen sich Gemälde aus verschiedenen Perspektiven betrachten, so zum Beispiel durch Heranzoomen. Besondere Funktionen oder ein Storytelling weist diese App nicht auf. Sie immittiert letztlich einen Museumsbesuch in einer virtuellen Welt. Das Archäologische Museum in Hamburg bietet ebenfalls einen Rundgang durch das Museum in 3D und Virtual Reality an¹⁶. Dabei handelt es sich auch über eine Webanwendung, welche mit einem Smartphone und Google Cardboard erkundet werden kann. Wer eine vollwertige VR-Brille besitzt, unabhängig vom Hersteller, kann dieses VR-Erlebnis ebenfalls über einen VR-Browser erfahren. Somit spricht das Archäologische Museum aus Hamburg die größtmögliche Nutzerzahl an. Die Umsetzung relativ simpel, da es sich um eine einfache Museumsrundführung ohne bestimmte Funktionen handelt. Das Deutsche Museum in München bietet ein VRlab an, in dem Besucher in die Vergangenheit reisen und dort mit Objekten interagieren können¹⁷. Auch eine Fahrt über den Mond ist möglich. Das Deutsche Museum in München zeigt an dieser Stelle, dass die Möglichkeiten von Virtual Reality vor allem zur Erkundung physikalisch unmöglich Szenarien genutzt werden sollten. An diesem Beispiel lässt sich gut erkennen, dass Virtual Reality nicht als Ersatz, sondern als Ergänzung zu einem klassischen Museumsbesuch genutzt werden kann. Das Städel Museum bietet auch eine Zeitreise an, mit der das Museum im 19. Jahrhundert betrachtet werden kann¹⁸. Die entwickelte Lösung wurde in Kooperation mit Samsung entwickelt, weshalb die App nur mit einem Samsung Smartphone oder einer VR-Brille von Oculus betrachtet werden kann. So schränkt das Städel Museum ihre Zielgruppe enorm ein. Darüber hinaus bieten sie jedoch den Benutzern, durch das Veröffentlichen der App in einem Store, die Möglichkeit, die Erfahrung auch von zu Hause

¹⁵Google Arts & Culture VR | <https://play.google.com/store/apps/details?id=com.google.vr.museums&hl=de> (04.11.2020)

¹⁶Archäologisches Museum Hamburg | <https://amh.de/digitales-angebot/google-art-project/> (05.11.2020)

¹⁷Deutsches Museum | <https://www.deutsches-museum.de/ausstellungen/sonderausstellungen/vrlab/> (05.11.2020)

¹⁸Zeitreise Städel Museum | <https://zeitreise.staedelmuseum.de/vr-app/> (05.11.2020)

aus zu erleben. Eine weitere Möglichkeit, um den Inhalt einer VR-Erfahrung verständlicher zu übermitteln ist der Einsatz von Museumsführern. Entweder durch eine einfache Vertonung oder das zusätzliche Einbauen eines Avatars. Virtuelle Museumsführer können nachweislich zu einem höheren Engagement und höherer Aufmerksamkeit des Betrachters führen, da der Inhalt verständlicher wird [Carrozzino u. a., 2018, S. 299–300]. Den Ansatz, mit Vertonung zu arbeiten, hat das Deutsche Auswandererhaus ebenfalls in ihrer Studie durchgeführt. Soll ein ideales VR-System entwickelt werden, muss, wie in Kapitel 3.1.2 bereits erwähnt, auch der akustische Sinn miteinbezogen werden. Je mehr Sinne miteinbezogen werden, desto besser sei die VR-Erfahrung für den Benutzer.

4 Prototyping

Das Prototyping ist eine Methode in der Softwareentwicklung, um schnell und mit wenig Ressourcen an Ergebnisse zu gelangen. Diese Ergebnisse können als frühzeitiges Feedback zur Eignung eines Lösungsansatzes und der genutzten Technologien dienen. Diese Methode wird im vorliegenden Forschungsprojekt angewandt, um mehrere Ergebnisse im Hinblick auf technische Möglichkeiten der Darstellung erzielen und erörtern zu können. Diese Ergebnisse können weiterführend verwendet werden, um eine vollwertige Software in diesem Anwendungsbereich entwickeln zu können. Zu Beginn müssen die Anforderungen des VR-Systems analysiert und gestellt werden. Da die Forschungsfrage keine eindeutige Lösung fordert, sondern herausfinden möchte, wie solch ein System für das Cranach Digital Archive entwickelt werden kann, werden die Anforderungen auf Basis von Erfahrungen der bereits entwickelten Produkte und der Literaturergebnisse festgelegt. Zunächst muss dazu eine geeignete Technologie gewählt werden, um eine VR-Anwendung entwickeln zu können. So werden die aktuellsten und populärsten Frameworks und Librarys in Betracht gezogen und kritisch hinterfragt, welche sich für den vorliegenden Anwendungsfall am besten eignen. Sobald die genutzten Technologien feststehen, werden die ersten Prototypen entwickelt und mit einer VR-Brille getestet. Sofern diese die festgelegten Ziele erfüllen, können anhand dieser Erkenntnisse die Prototypen entwickelt beziehungsweise weiterentwickelt werden. Die Ergebnisse werden, basierend auf den fertigen Prototypen, im nächsten Kapitel vorgestellt und analysiert.

4.1 Anforderungen an das VR-System

Das aktuelle Cranach Digital Archive ist bereits eine Webanwendung und kann über jedes browserfähige Gerät aufgerufen werden. Somit soll dieser Ansatz der maximalen Zugänglichkeit im vorliegenden Forschungsprojekt weiterhin aufgegriffen werden. Zudem sollte das Ziel eines Museums, dem Verbreiten von Wissen, keiner Zielgruppe durch die Technologieauswahl verwehrt werden. Deshalb ist die erste Anforderung an das VR-System

die Zugänglichkeit über das Web als webbasierte Anwendung. Diese Anwendung, welche unter Umständen vom Benutzer auch auf einem Smartphone aufgerufen werden möchte, muss dementsprechend ohne 3D-Eingabegerät funktionieren. Darüber hinaus ist durch eine blickorientierte Steuerung die Anwendung für den Benutzer zugänglicher und kann daher, wie bereits in Kapitel 3.2.2 und 3.2.3 erwähnt, ohne erschwerte Einarbeitung genutzt werden. So ist eine blickorientierte Steuerung intuitiv und benötigt weniger Hardware. Dies stellt auch die zweite Anforderung an das VR-System dar. Um dem Benutzer eine immersive und interessante VR-Erfahrung zu ermöglichen, sollen möglichst viele Sinne miteinbezogen werden. In Kapitel 3.1.2 wurde festgehalten, dass ein perfektes VR-System alle Sinne des Menschen miteinbezieht. Da dies aktuell technisch jedoch nicht möglich ist, sollen so viele Sinne wie möglich angesprochen werden. Deshalb soll, neben dem visuellen Sinn der bei einer VR-Erfahrung immer verwendet werden muss, zusätzlich der akustische Sinn angesprochen werden. Das Cranach Digital Archive ist neben kunstinteressierten Benutzern auch für Forscher und Mitarbeiter des Teams zugänglich. In diesem Kontext soll der Fokus der Prototypen nicht nur auf den visuell schönen VR-Erfahrungen liegen, sondern auch auf der Verbesserung des generellen Datenverständnisses. Wie bereits in Kapitel 2.2 erwähnt wurde, kann Virtual Reality dabei helfen, komplexe Datenstrukturen besser zu verstehen. Demnach gibt es zwei Arten von VR-Systemen: Jene die für die Zielgruppe eines Museums entwickelt werden und sich daher auf immersive und visuell ansprechende VR-Erfahrungen fokussieren und Andere, die versuchen das Verständnis für die Daten zu verbessern. Jedoch ist zu erwähnen, dass beide Ansätze der VR-Systeme auch von beiden Zielgruppen angenommen werden können. So kann die Zielgruppe eines Museums ebenfalls das Bedürfnis für ein besseres Verständnis der Daten empfinden.

Die folgende Auflistung fasst alle Anforderungen der zu entwickelnden VR-Systeme zusammen:

Die Anforderungen

- Webbasierte Anwendung
- Blickorientierte Steuerung
- Bezug auf den akustischen Sinn
- VR-System mit Fokus auf die Zielgruppe der Museumsbesucher
- VR-System mit Fokus auf besseres Verständnis von Daten

4.2 Auswahl der Technologien

In diesem Kapitel werden die Vor- und Nachteile der einzelnen Frameworks, Librarys oder Softwares gegenübergestellt, die zur Umsetzung des Forschungsprojektes genutzt werden sollen. Anhand der zuvor aufgelisteten Anforderungen sowie der vorhandenen Ressourcen dieses Projektes wird eine entsprechende Technologielösung ausgewählt. Die Technologien werden durch eine einfache Internetrecherche über WebVR gesucht. Damit eine Technologie in Betracht gezogen werden kann, muss diese eine gewisse Verbreitung und Community mit sich bringen, denn Software, die regelmäßig erweitert und von einer Community unterstützt wird, bringt viele Vorteile mit sich. Durch die umfangreiche Nutzung durch eine Community ist es einfacher Lösungen für auftretende Entwicklungsprobleme zu finden. Zudem erleichtert eine große Anzahl an bereits geschriebener Code-Pakete die Entwicklung des vorliegenden Forschungsprojektes.

Unity¹⁹ und die Unreal Engine²⁰ sind Softwarelösungen, hinter welchen zwei große Firmen stehen, die diese seit Jahren entwickeln. Beide Spiele-Engines bieten eine große Community und zudem eine ausführliche Dokumentation²¹. So werden jene Softwarelösungen in die Auswahl der Technologie des vorliegenden Forschungsprojektes miteinbezogen.

Ein weiteres populäres Framework ist React 360²² (8.400 Stars auf GitHub²³) und wird von Facebook entwickelt. Zusätzlich zu dem React-Ökosystem bietet dieses in Kombination mit WebGL und WebXR die Möglichkeit, VR-Systeme für das Web zu entwickeln²⁴. Da das React-Ökosystem sehr beliebt und verbreitet ist, wird dieses ebenfalls in die Auswahl der vorliegenden Entwicklung miteinbezogen.

Das nächste Framework, welches laut GitHub (12.000 Stars²⁵) populärer als React 360 ist, ist A-Frame²⁶ von Mozilla und Supermedium. Mit A-Frame lassen sich ebenfalls dank WebGL und WebXR VR-Systeme für das Web entwickeln²⁷. A-Frame ist für seine große und hilfsbereite Community bekannt und hat eine sehr ausführliche und umfangreiche Dokumentation²⁸. Da dies wichtige Kriterien für eine Entwicklung darstellen, wird auch dieses Framework in die Auswahl miteinbezogen.

Als letzte Library wird Three.js²⁹ in Betracht gezogen. Three.js bietet eine umfangreiche Library für das Entwickeln von 3D-Anwendungen im Web. Virtual Reality wird ebenfalls

¹⁹Unity | <https://unity.com/de> (09.11.2020)

²⁰Unreal Engine | <https://www.unrealengine.com/en-US/> (09.11.2020)

²¹Unreal Engine Dokumentation | <https://docs.unrealengine.com/en-US/index.html>; Unity Dokumentation | <https://docs.unity3d.com/Manual/index.html> (11.11.2020)

²²React 360 | <https://facebook.github.io/react-360/> (09.11.2020)

²³React 360 auf GitHub | <https://github.com/facebook/react-360> (09.11.2020)

²⁴React 360 auf GitHub | <https://github.com/facebook/react-360> (11.11.2020)

²⁵A-Frame auf GitHub | <https://github.com/aframevr/aframe> (09.11.2020)

²⁶A-Frame | <https://aframe.io/> (09.11.2020)

²⁷A-Frame Dokumentation | <https://aframe.io/docs/1.0.0/introduction/> (11.11.2020)

²⁸A-Frame Community | <https://aframe.io/community/>; A-Frame Dokumentation | <https://aframe.io/docs/1.0.0/introduction/> (11.11.2020)

²⁹Three.js | <https://threejs.org/> (09.11.2020)

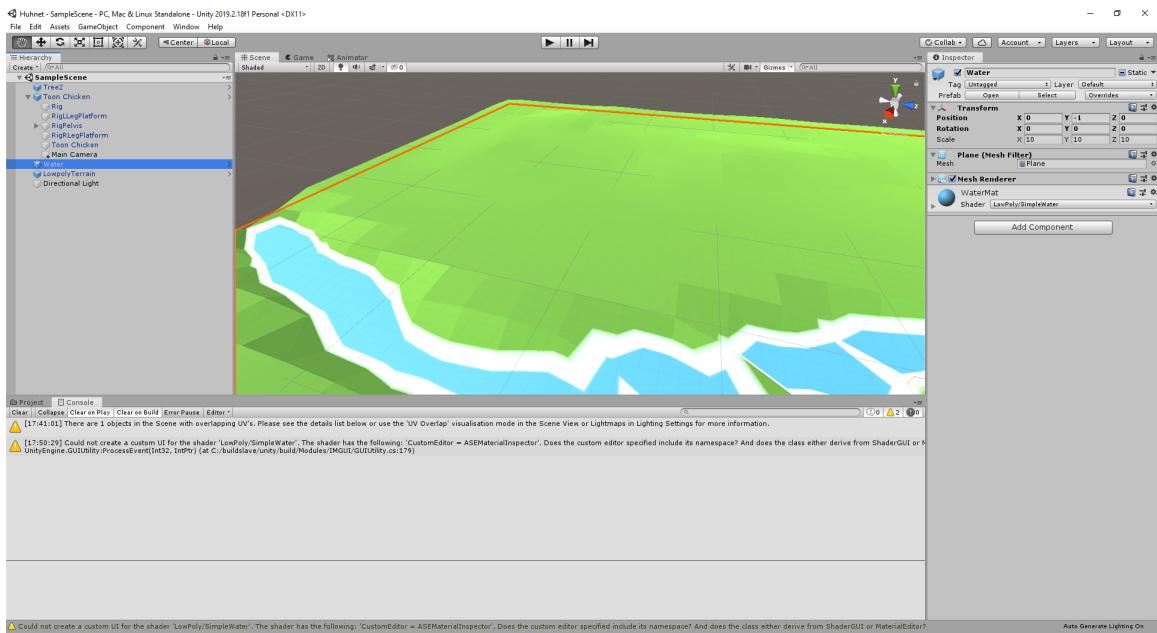


Abbildung 2: Unity Entwicklungsumgebung.

unterstützt³⁰. Da Three.js sehr mächtig ist, wird diese Library sogar von React 360 und A-Frame genutzt, um komplexe Berechnungen und Darstellungen im Browser möglich zu machen³¹. Three.js ist mit fast 65.000 Stars³² auf GitHub die populärste der auf GitHub vorhandenen Librarys / Frameworks, um WebXR-Anwendungen zu entwickeln. Dementsprechend groß ist auch die Community von Three.js. Deshalb fällt auch diese Library in die engere Auswahl zur Entwicklung des vorliegenden Forschungsprojektes.

In den folgenden Kapiteln wird genauer auf die einzelnen Technologien eingegangen sowie deren Vor- und Nachteile erörtert.

4.2.1 Unity

Unity ist eine Laufzeit- und Entwicklungsumgebung für Spiele der Firma Unity Technologies. Dem Namen zufolge bedeutet dies jedoch nicht, dass damit hauptsächlich Spiele entwickelt werden können. Durch den großen Funktionsumfang von Unity lassen sich auch Anwendungen entwickeln, welche sinnvoll für Industrie aber auch Museen sind. Die Entwicklungsumgebung bietet viele grafische Oberflächen, womit sich ganze 3D-Szenen per Maus gestalten lassen (siehe Abb. 2). Zudem bietet Unity nicht nur eine grafische Benutzeroberfläche, sondern auch viele weitere Funktionen, die im Folgenden aufgelistet werden:

³⁰Three.js WebVR Dokumentation | <https://threejs.org/docs/index.html#manual/en/introduction/How-to-create-VR-content> (11.11.2020)

³¹Three.js in A-Frame | <https://aframe.io/docs/1.0.0/introduction/developing-with-threejs.html>; Three.js in React 360 | <https://facebook.github.io/react-360/docs/what-is.html> (11.11.2020)

³²Three.js auf GitHub | <https://github.com/mrdoob/three.js/> (09.11.2020)

- Grafik gemäß dem aktuellen Stand der Technik
- Animationen, um Charaktere oder andere Modelle animieren zu können
- Physik für das Verhalten von Objekten in einer Welt
- Musik und Geräusche, welche Stereofonie ermöglichen
- Programmierung, um die einzelnen Entitäten durch eigene Skripte zu erweitern
- Toolkits für User Interfaces
- Unterstützung für Netzwerk- und Multiplayer-Anwendungen
- Cross-Plattform-Entwicklung, um die entwickelte Anwendung auf mehrere Plattformen zu veröffentlichen

Unity bietet eine umfangreiche Dokumentation für jede dieser Funktionen. Entwickelt werden die sogenannten Skripts, welche dann in der Programmiersprache C# zu verschiedenen Entitäten hinzugefügt werden können. Dementsprechend ist eine gewisse Erfahrung in dieser Programmiersprache notwendig, um effektiv Prototypen entwickeln zu können. Unity stellt zusätzlich zu ihrem 3D-Editor die Entwicklungsumgebung MonoDevelop zur Verfügung, mit welcher die Skripte geschrieben sowie Fehlersuche und -beseitigung betrieben werden können. Durch die umfangreiche Benutzeroberfläche und den Funktionen benötigt Unity jedoch eine lange Einarbeitungszeit. Im Rahmen dieses Forschungsprojektes werden jedoch nicht alle Funktionen, die Unity überliefert, benötigt.³³ Unity ist kostenlos nutzbar und bietet dabei den vollen Funktionsumfang. Betrug der Umsatz in den letzten zwölf Monaten jedoch über 100.000\$, muss eine kommerzielle Lizenz erworben werden³⁴.

4.2.2 Unreal Engine

Die Unreal Engine ist, verglichen mit den anderen Technologien, ähnlich zur Unity Engine. Dennoch hat diese Software einige andere Ansätze. Was sehr ähnlich zur Unity Engine ist, ist der weitläufige Funktionsumfang und die grafische Benutzeroberfläche (siehe Abb. 3). Auch mit Hilfe dieser Technologie lassen sich über einen 3D-Editor Entitäten bearbeiten und in Szenen positionieren. Zusätzlich verfügt die Unreal Engine, für Personen die nicht programmieren können, über ein sogenanntes Blueprint-System (siehe Abb. 4). Damit lassen sich per Drag and Drop Algorithmen grafisch zusammenstellen. Somit ist Unreal Engine einsteigerfreundlich für Nicht-Entwickler. Wenn jedoch komplexere Algorithmen gewünscht sind, wird empfohlen die Programmiersprache C++ anzuwenden. Dazu sollten

³³Unity Dokumentation | <https://docs.unity3d.com/Manual/> (11.11.2020)

³⁴Unity Pläne | <https://store.unity.com/#plans-individual> (11.11.2020)

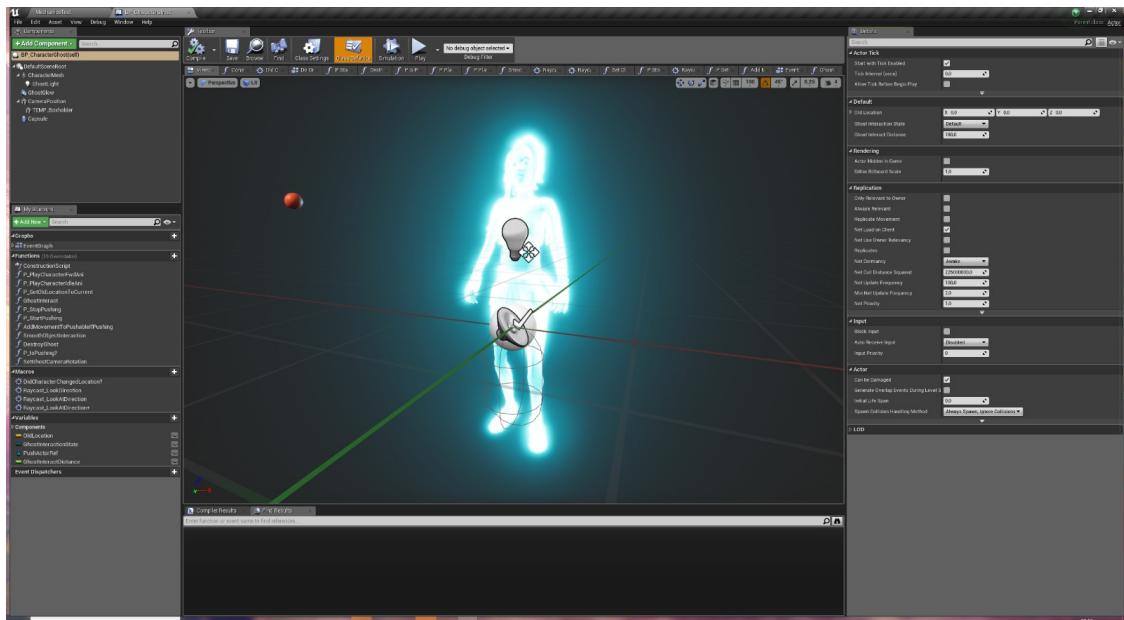


Abbildung 3: Unreal Engine Entwicklungsumgebung.

Kenntnisse im Bereich C++ mitgebracht werden. Auch dabei muss, wie bereits bei der Unity Engine aufgezeigt wurde, eine gewisse Einarbeitungszeit berücksichtigt werden.

Das Endprodukt einer Software kann daraufhin sowohl auf viele verschiedene Plattfor-

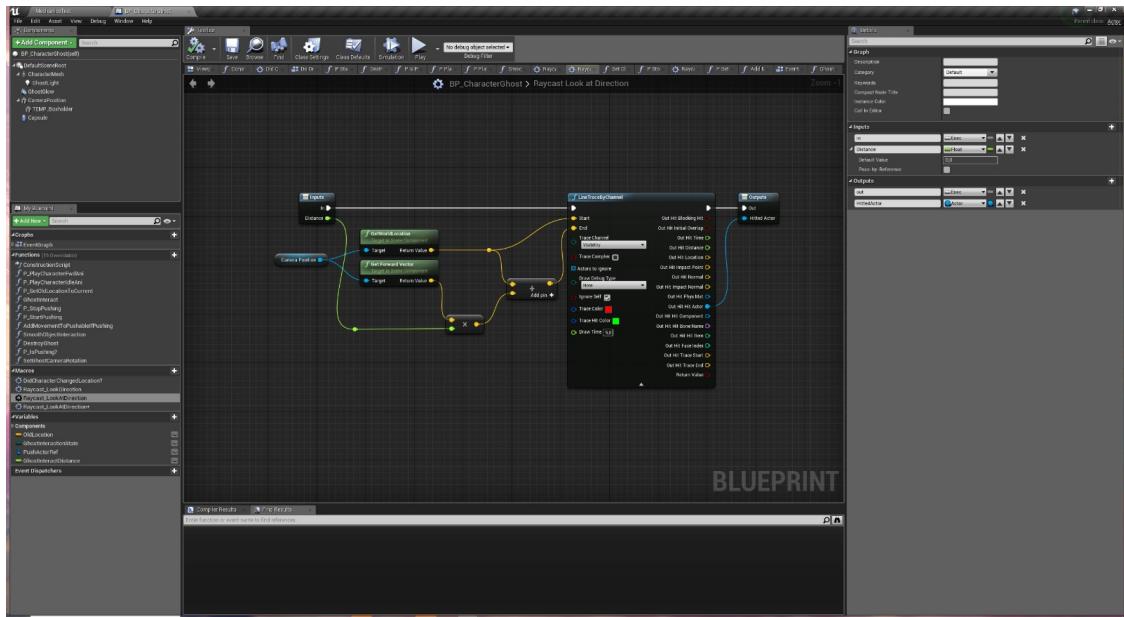


Abbildung 4: Unreal Engines Blueprint-System.

men als auch in das Web portiert werden. Die Entwicklung von Webanwendungen in der Unreal Engine ist jedoch nur über eine Erweiterung möglich, welche von der Community entwickelt wird. Darin zeichnet sich nicht unbedingt ein Nachteil ab, kann aber unter Umständen problematisch sein. Dies ist von der Aktivität der Community abhängig, die an dieser Erweiterung arbeitet. Zum aktuellen Zeitpunkt wird die Erweiterung von der

Community weiterentwickelt³⁵.

Die Unreal Engine ist ebenfalls kostenlos nutzbar, sofern kein kommerzieller Hintergrund bei der Entwicklung einer Anwendung ist³⁶.

4.2.3 React 360

React 360 ist ein Framework zur Entwicklung von Cross-Plattform-Anwendungen auf Basis von JavaScript. Wie der Name bereits sagt, stammt React 360 von React ab. Da das React-Ökosystem eine große Community mit sich bringt, gibt es viele Code-Pakete die zur Entwicklung von Webanwendungen mit React genutzt werden können. React 360 baut auf Three.js auf, bietet jedoch ein weiteres Abstraktionslevel und vereinfacht dadurch die Entwicklung. Durch die JavaScript Syntax Extension (JSX) ist es möglich, einfaches HTML mit JavaScript in einer Datei zu kombinieren³⁷. Dadurch ist der Programmcode für den Entwickler lesbarer und einfacher nachzuvollziehen. Ein Nachteil ist jedoch, dass React 360 keinen 3D-Editor anbietet und dementsprechend auch keine grafische Benutzeroberfläche. 3D-Modelle die an eine bestimmte Position sollen, müssen über den Programmcode positioniert und daraufhin kontrolliert werden. Das bringt einen Mehraufwand mit sich.³⁸

React 360-Anwendungen werden mit JavaScript entwickelt, was einen leichten Einstieg erlaubt. C# und C++ sind im Gegensatz zu JavaScript komplexe Programmiersprachen. Unter anderem aufgrund der strengen Typsicherung. Dieser Punkt ist aber auch für einige Entwickler ein Nachteil von JavaScript. Dieses Problem lässt sich aber durch Hilfsmittel für JavaScript und die passende Entwicklungsumgebung beheben.

React 360 ist Open Source und steht unter der BSD-Lizenz. Demnach darf man diese Software kostenlos verwenden, verändern und bei Bedarf kommerziell vertreiben. Der Vorteil an Open Source Software ist, dass der Quellcode frei einsehbar ist und sich bei Bedarf sogar verändern lässt. Das ist auch einer der Gründe, weshalb React und React 360 eine große und aktive Community besitzt.

4.2.4 A-Frame

A-Frame ist, wie React 360, ein Framework, mit welchem WebXR-Anwendungen entwickelt werden können. Dazu muss ebenfalls in der Programmiersprache JavaScript entwickelt werden. A-Frame erweitert HTML um eigene HTML-Elemente, was das Benutzen von geometrischen Entitäten oder 3D-Modelle vereinfacht (siehe Abb. 5). Damit schafft

³⁵Unreal Engine Dokumentation | <https://docs.unrealengine.com/en-US/index.html> (11.11.2020)

³⁶Unreal Engine Pläne für Nicht-Spiele | <https://www.unrealengine.com/en-US/get-now/non-games> (11.11.2020)

³⁷JSX bei React | <https://reactjs.org/docs/introducing-jsx.html> (11.11.2020)

³⁸React 360 | <https://facebook.github.io/react-360/> (11.11.2020)

A-Frame ein weiteres Abstraktionslevel, was das Programmieren einer VR-Anwendung deklarativ macht. Prototypen lassen sich bereits ohne das Anwenden von JavaScript entwickeln, was vor allem den Einstieg immens vereinfacht. Ein weiterer Vorteil von A-Frame

```
<a-box position="-1.692 0.018 -1.460" rotation="0 28.938 0"></a-box>
```

Abbildung 5: HTML-Element aus A-Frame.

ist die mitgelieferte grafische Benutzeroberfläche die im Browser läuft (siehe Abb. 6). Darüber lassen sich Entitäten positionieren und Komponenten zu diesen hinzufügen. Wie bereits erwähnt verfügt A-Frame über eine aktive und große Community und bietet deshalb einen Discord-Server an³⁹. Auf diesem können jederzeit Fragen zu A-Frame gestellt werden. A-Frame baut ebenfalls auf Three.js auf, weshalb es auch die Funktionen dieser

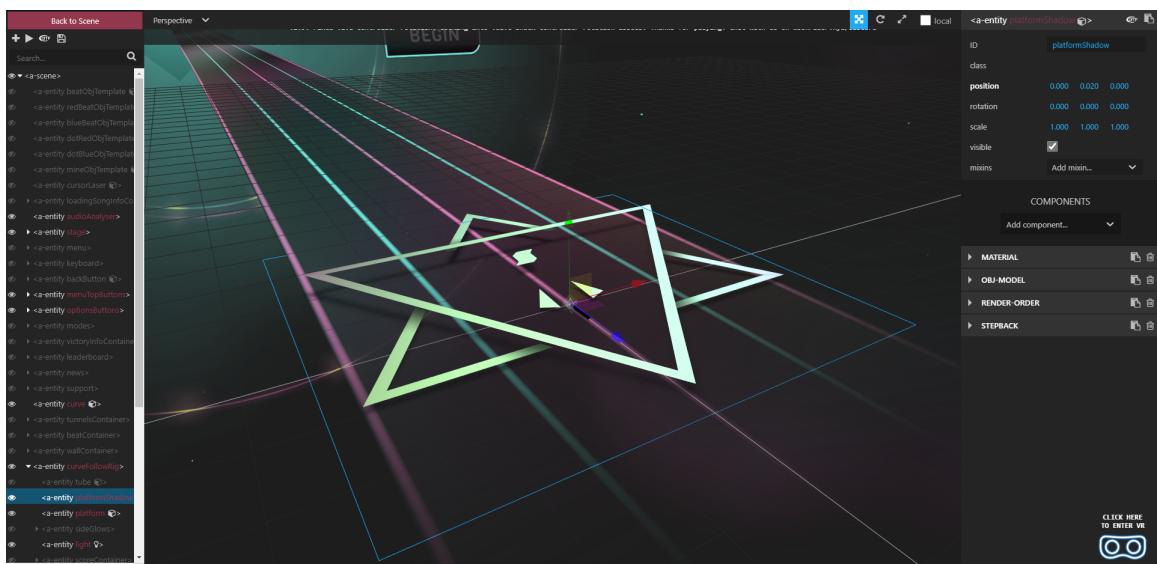


Abbildung 6: Grafische Benutzeroberfläche von A-Frame.

Library mitbringt. Bei Bedarf lässt sich über JavaScript auf den vollen Funktionsumfang von Three.js innerhalb von A-Frame Komponenten zugreifen. Das macht das Framework sehr flexibel und mächtig.

A-Frame ist Open Source und steht unter der MIT-Lizenz. Somit kann dieses Framework kostenlos und ohne Einschränkungen zur Entwicklung der Prototypen verwendet werden.

³⁹Discord ist eine Software zum Schreiben und Telefonieren. Es können öffentliche Server eingerichtet werden mit verschiedenen Chat-Kanälen, in denen sich ausgetauscht werden kann. | <https://discord.gg/tGYjkYr> (12.11.2020)

4.2.5 Three.js

Three.js ist eine Library die auf JavaScript basiert und demnach auch für diese Programmiersprache vorgesehen ist. Ähnlich wie A-Frame bietet Three.js eine Anlaufstelle für ihre Community. Neben einem Discord-Server lässt es sich auch über ein Forum austauschen. Auf der Webseite findet man auch viele Beispielprojekte, die Three.js nutzen. Darunter finden sich auch A-Frame und React 360.

Three.js bietet wie Unity und Unreal Engine vorprogrammierte Funktionen an wie Physik, Animationen oder einen 3D-Editor. Der Editor ist jedoch, anders als bei A-Frame, nicht im eigenen Projekt integriert, sondern über die Webseite von Three.js erreichbar. In diesem Editor können die Funktionen anhand einer grafischen Benutzeroberfläche ausprobiert und erprobt werden.

Auf der Webseite wird eine eigene Tutorialreihe angeboten in der auch auf die Entwicklung von WebVR-Anwendungen eingegangen wird.

Da keine grafische Benutzeroberfläche mitgeliefert wird und alles über den Programmcode erstellt werden muss, steigt die Komplexität bei dieser Technologie.

Three.js ist Open Source, steht unter der MIT-Lizenz und lässt sich daher kostenfrei ohne Einschränkungen nutzen für die Entwicklung der Prototypen.⁴⁰

4.2.6 Ergebnis

Nachfolgend lassen sich in der Tabelle 1 die Vor- und Nachteile der einzelnen Technologien festhalten.

	Unity	Unreal Engine	React 360	A-Frame	Three.js
Graf. Benutzeroberfläche	✓	✓	X	✓	X
3D-Editor	✓	✓	X	✓	✓
Einsteigerfreundlich	X	X	✓	✓	✓
Große Community	✓	✓	✓	✓	✓
Open Source	X	X	✓	✓	✓
Kostenlos nutzbar	✓	✓	✓	✓	✓

Tabelle 1: Funktionsumfang aller Technologien.

Diese Tabelle zeigt, dass die meisten Vorteile A-Frame und Three.js mitbringen. Da in diesem Forschungsprojekt auch Kenntnisse im Bereich JavaScript und Webentwicklung mitgebracht werden, spricht dies zusätzlich für die Entwicklung in einer der beiden Technologien.

A-Frame bietet zusätzlich zu Three.js eine grafische Oberfläche, welche in jedem Projekt automatisch integriert und über den Browser zu bedienen ist. Der volle Funktionsumfang

⁴⁰Three.js Webseite | <https://threejs.org/>

von Three.js ist auch weiterhin in A-Frame enthalten, da das Framework auf dieser Library aufbaut. Dadurch, dass A-Frame auf ein weiteres Abstraktionslevel setzt und viele vorgefertigte Funktionen mitbringt, die in Three.js selbst implementiert werden müssten, ist der Einsatz von A-Frame sinnvoll und benötigt keine große Einarbeitungszeit.

4.2.7 Weitere Hilfsmittel

Zusätzlich zur Wahl der Technologie lassen sich weitere Hilfsmittel in die Entwicklung der Prototypen hinzuziehen. Diese sollen das Entwickeln und Verwalten des Codes und der Dateien vereinfachen und auch Optimierungen der Performance der Anwendung mit sich bringen.

Zur Versionsverwaltung wird **Git** in Kombination mit GitHub verwendet. Dadurch können Funktionen implementiert werden ohne den aktuellen und lauffähigen Code zu beeinflussen. Bei erfolgreichen Tests der neuen Funktionen kann dieser Code in die aktuelle Version hinzugefügt werden. Bei Bedarf kann auf alten Code zugegriffen und der Code auch auf diesen Stand zurückgebracht werden.

Für eine verbesserte Typsicherheit in JavaScript und produktiveres Programmieren wird **TypeScript** verwendet. So finden Konstrukte wie Interfaces aus der objektorientierten Programmierung auch Anwendung in JavaScript. Das vereinfacht die Entwicklung mit JavaScript und ist skalierbarer.

Das letzte Hilfsmittel welches verwendet wird ist **Webpack**. Dies hilft bei der Optimierung des Codes, da es den gesamten JavaScript-Code in eine Datei zusammenfügt und diese keine Leerzeichen, Tabulatoren oder Umbrüche beinhaltet. Dadurch ist die Größe der Datei minimiert.

4.3 Entwicklung der Prototypen

Zu Beginn wird sich mit dem Framework auseinandergesetzt und ein Bereich zum Entwickeln geschaffen, in dem verschiedene Ansätze zum Darstellen von Gemälden und Archivalien in einer virtuellen Welt ausprobiert werden können.

Entwickelt wird in der Entwicklungsumgebung WebStorm von Jetbrains. Die VR-Brille, mit der die Anwendungen getestet werden, ist die Oculus Quest, welche von der Technischen Hochschule Köln gestellt wird. Diese verwendet den Browser Firefox Reality, da dieser viele Funktionen für Entwickler bietet.

Das erste Ziel ist einen technischen Prototypen zu entwickeln, welcher das Fortbewegen in der virtuellen Welt durch das Angucken von Objekten ermöglicht. Auch sollen Gemälde und deren Beschreibung angezeigt werden können. Sobald die ersten technischen Anforderungen erfüllt wurden, sollen weitere Prototypen entwickelt werden.

4.3.1 Technischer Prototyp

Der erste technische Prototyp soll zeigen, wie eine blickorientierte Steuerung und die Gemälde von Lucas Cranach mit Hilfe von A-Frame entwickelt werden können. Dieser soll auch das Fundament für die weiteren Prototypen darstellen.

Um den ersten Prototypen zu entwickeln, muss vorerst das Prinzip von A-Frame verstanden werden und wie Inhalte mittels HTML und JavaScript dargestellt werden können.

Bevor mit der Entwicklung eines Prototypen mit JavaScript begonnen werden kann, soll erstmal ein Gemälde von Lucas Cranach innerhalb einer virtuellen Welt dargestellt werden. Dafür wird eine statische `index.html`-Datei benötigt, welche die aktuelle stabile Version von A-Frame importiert (siehe Abb. 7). Neben den notwendigen HTML-Tags, die für

```
<html lang="de">
  <head>
    <meta charset="UTF-8">
    <script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
    <title>Virtual Reality</title>
  </head>
  <body>
    <a-scene>
      <a-assets>
        
      </a-assets>

      <a-entity
        id="camera"
        position="0 0 0"
      >
        <a-camera>
          <a-cursor
            color="white"
            fuse="true"
            fuseTimeout="1000"
          >
          </a-cursor>
        </a-camera>
      </a-entity>

      <a-sky src="#sky"></a-sky>
    </a-scene>
  </body>
</html>
```

Abbildung 7: Erste Version einer `index.html`-Datei.

eine valide HTML-Datei notwendig sind, kommen `<a-scene>`, `<a-assets>`, `<a-entity>`, `<a-camera>`, `<a-cursor>` und `<a-sky>` hinzu. A-Frame-HTML-Tags lassen sich durch das Präfix `a` erkennen.

Das HTML-Tag `<a-scene>` ist immer das erste und notwendige einer A-Frame-Anwendung.

Durch dieses Tag weiß das Framework, dass es sich hierbei um eine A-Frame-Szene handelt. Alles weitere, welches im Rahmen von A-Frame entwickelt wird, muss innerhalb dieses Tags platziert werden.

<a-assets> beinhaltet alle notwendigen Assets für die VR-Anwendung. Darin können 3D-Modelle oder Bilder (Texturen, Icons, etc.) eingefügt werden. A-Frame empfiehlt, sofern möglich, alle Assets die benötigt werden in **<a-assets>** hinzuzufügen, da diese im Cache gespeichert werden und ein Neuladen der Anwendung beschleunigt.

Bei **<a-entity>** handelt es sich um eine einfache Entität, welche vorerst keine Darstellung innerhalb der VR-Anwendung haben muss. Oft kann dieses Tag auch als Wrapper für mehrere Tags verwendet werden. Das hat zum Beispiel den Vorteil, dass nur diese Entität angesprochen werden muss, um alle Entitäten innerhalb dieser bewegen zu können. Das HTML-Attribut **id** innerhalb des **a-entity**-Tags ist keine A-Frame-spezifische Definition, sondern hat die selbe Funktion wie in normalem HTML. **position** hingegen gibt die Position als 3D Vektor, relativ zum Nullpunkt der Weltkoordinaten an. In diesem Fall befindet sich die Entität genau auf dem Nullpunkt der Weltkoordinaten. Das HTML-Tag **<a-camera>** stellt die Kamera und somit die Augen des Benutzers innerhalb der VR-Anwendung dar. Die Position der Kamera ist auch die des Benutzers. Eine Kamera muss platziert werden, da ansonsten keine virtuelle Welt gesehen werden kann. Somit ist dieser HTML-Tag zwingend notwendig.

Innerhalb der **<a-camera>**-Entität befindet sich **<a-cursor>**. Mit diesem Cursor kann der Benutzer auf bestimmte Entitäten innerhalb einer virtuellen Welt zeigen. Durch die Attribute **fuse** und **fuseTimeout** lassen sich mit diesem auch Entitäten anklicken. A-Frame bietet an der Stelle bereits eine Lösung für die blickorientierte Steuerung. Durch **fuse=true** wird die blickorientierte Steuerung aktiviert. **fuseTimeout="1000"** gibt an, wie viel Millisekunden auf eine Entität geschaut werden muss, damit bei dieser der Event-Listener **click** ausgelöst wird. Somit muss bei einer Entität ein EventListener für **click** festgelegt werden, damit diese auf einen blickorientierten Klick reagieren kann. Das Attribut **color** legt die Farbe des Cursors fest.

Das HTML-Tag **<a-sky>** bietet eine fertige Lösung, um einen Himmel innerhalb der virtuellen Welt anhand einer Bilddatei anzuzeigen. Durch das Attribut **src="#sky"** wird die Bilddatei mit der ID **#sky** aus den Assets geladen. Diese Schreibweise ersetzt letztlich die Angabe eines Pfades.

Beim Aufrufen der **index.html** erscheint folgendes Bild (siehe Abb. 8). Der Cursor der Anwendung befindet sich in der Mitte und ist als weißer Kreis dargestellt. Unten rechts befindet sich der VR-Button, welcher immer von A-Frame mitgerendert wird. Ein Klick auf diesen Button aktiviert den VR-Modus der Webanwendung.

Als nächster Schritt soll ein Gemälde angezeigt werden. In diesem Beispiel wird das Gemälde aus dem Cranach Digital Archive mit der ID **DE_MdbKL_946** verwendet. A-Frame bietet den HTML-Tag **<a-image>**, um Bilder anzuzeigen. Das Seitenverhältnis der Bilder

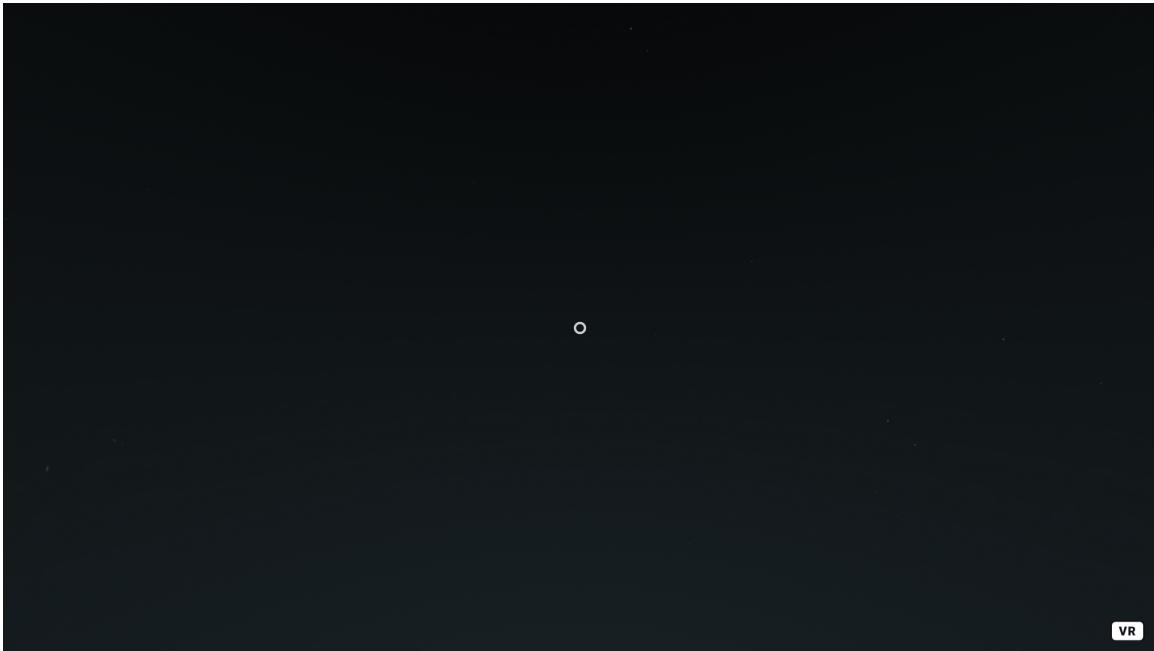


Abbildung 8: Erste virtuelle Welt.

wird nicht beibehalten, weshalb diese durch die Attribute `width` und `height` festgelegt werden muss. Das Tag wird einfach innerhalb der `<a-scene>` platziert (siehe Abb. 9). An dieser Stelle bietet A-Frame die Möglichkeit, häufig benutzte Eigenschaften von Three.js wie `height`, `width` oder `color` über HTML-Attribute festzulegen. Durch diese Abstraktion muss nicht über JavaScript die Entität manipuliert werden und vereinfacht den Prozess des Prototypings. Bei den Einheiten innerhalb des Attributs `position` handelt es

```
<a-image
  src="#gemaelde"
  width="2"
  height="2.4866"
  position="0 1.5 -4"
></a-image>
```

Abbildung 9: Image-Tag von A-Frame für das Einbinden von Bildern.

sich um Angaben in Meter. Die Verschiebung einer Entität ist immer vom Mittelpunkt dieser ausgehend. Betrachtet man das Ganze innerhalb der virtuellen Welt, erscheint das Gemälde vor dem Benutzer (siehe Abb. 10). Als nächstes soll eine Steuerung durch das Anschauen von Entitäten ermöglicht werden. Da A-Frame bereits eine Lösung für das blickorientierte Klicken zur Verfügung stellt, muss zunächst eine Entität erzeugt werden, welches auf Klicken reagiert. Das Framework bietet zwei Möglichkeiten EventListener zu registrieren. Die eine Möglichkeit besteht darin, die EventListener ebenfalls in HTML als Attribut festzulegen. In Abbildung 11 wird ein `click`-EventListener registriert, welches die Komponente sichtbar über `visible=true` macht. Die zweite Möglichkeit ist das Registrieren eines EventListeners über JavaScript. Diese bietet mehr Möglichkeiten und

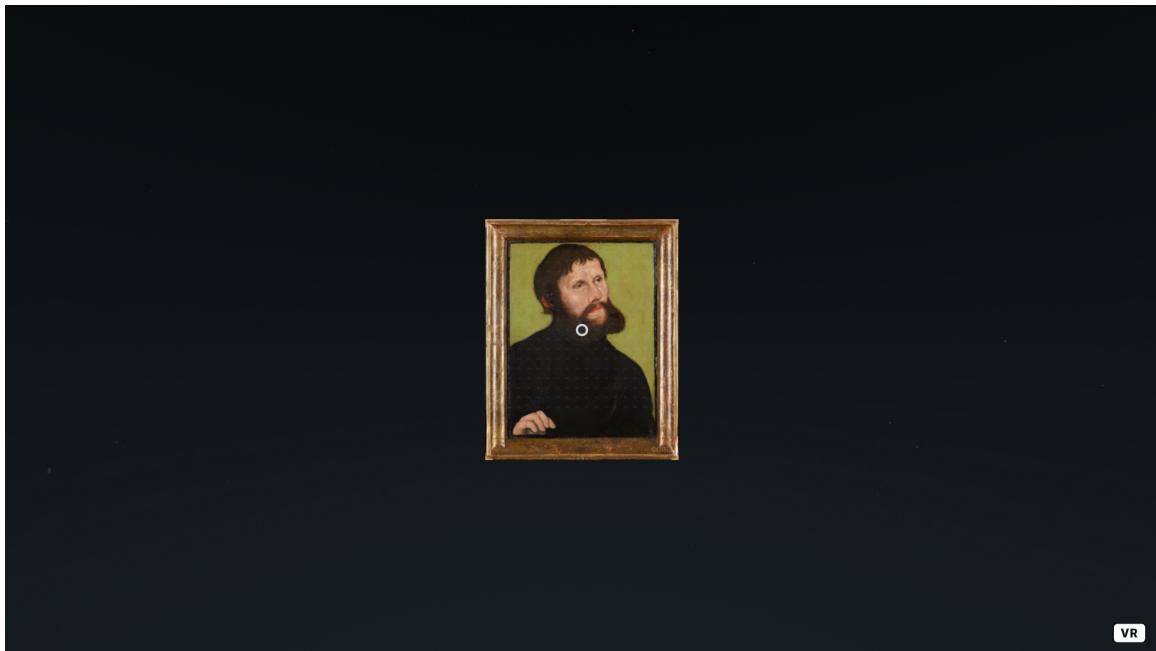


Abbildung 10: Gemälde innerhalb der virtuellen Welt.

erlaubt den Zugriff auf Three.js innerhalb einer Entität. Um einen EventListener mit

```
<a-image
  src="#gemaelde"
  width="2"
  height="2.4866"
  position="0 1.5 -4"
  event-set="_event: click; visible: true"
></a-image>
```

Abbildung 11: Event Registrierung innerhalb von HTML.

JavaScript umzusetzen, muss eine Komponente entwickelt werden. Komponenten in A-Frame sind wiederverwendbare Code-Module, welche zu beliebigen Entitäten hinzugefügt werden können. Diese Entitäten übernehmen dann die Logik der Komponente. Eine Komponente hat folgenden Aufbau (siehe Abb. 12). Um auf A-Frame Funktionen innerhalb der TypeScript-Datei zugreifen zu können, muss A-Frame zuerst mit einem Paketmanager wie yarn oder npm installiert werden. Aufgrund dessen kann die Importierung von A-Frame innerhalb der HTML-Datei entfernt werden, da nun nur noch die `bundle.js`, welche von Webpack generiert wird, importiert werden muss. Diese beinhaltet jeglichen Code, auch den der importierten Module innerhalb einer Komponente.

Die in Abbildung 12 gezeigte Komponente lässt sich dann über HTML einer Entität hinzufügen (siehe Abb. 13). Mit `AFRAME.registerComponent(...)` wird eine Komponente für A-Frame erstellt. Das erste Argument ist vom Typ `string` und beinhaltet den Namen der Komponente. Das zweite Argument ist ein Objekt, welches die Lifecycle-Methoden und ein Schema einer A-Frame Komponente enthält.

```

AFRAME.registerComponent('standing-area', {
  schema: {
    feld1: {type: 'string', default: 'Defaultwert'},
    feld2: {type: 'number', default: 1337},
  },
  init() { },
  update() { },
  remove() { },
  tick() { },
});

```

Abbildung 12: Aufbau einer Komponente in A-Frame.

```
<a-entity standing-area></a-entity>
```

Abbildung 13: A-Frame Entität mit hinzugefügter Komponente.

Das Feld **schema** gibt an, welche Werte innerhalb des HTML über das Attribut **standing-area** in die Komponenten hineingeben werden können. **feld1** und **feld2** sind die Namen der Eigenschaften. Mit **type** lässt sich der Typ für den Wert festlegen und **default** gibt einen Standardwert zurück, sofern keine Information zu dieser Eigenschaft mitgegeben wurde. In Abbildung 13 müsste das Attribut um **standing-area="feld1: Hallo!; feld2: 1338"** erweitert werden, damit der Standardwert überschrieben und ein neuer hineingegeben wird. Die Werte in **schema** können dann über **this.data.feld1** oder **this.data.feld2** innerhalb der Komponente aufgerufen werden.

Die **init**-Methode wird beim ersten Laden einer Entität mit dieser Komponente ausgeführt. Darin lassen sich Werte festlegen, die sich im Laufe der Anwendung nicht mehr verändern.

Die **update**-Methode wird einmalig nach der **init**-Methode ausgeführt und immer, wenn sich Attribute der Entität verändern.

Wird die Entität und ihre Komponente gelöscht, wird einmalig die **update**-Methode ausgeführt.

Die **tick**-Methode ist abhängig von der Bildrate der Anwendung. Beträgt die FPS (Bilder pro Sekunde) der Anwendung 60, dann wird diese Methode 60 mal ausgeführt.

Um das Teleportieren eines Benutzers auf die Position der Entität, welche angeguckt wird, umzusetzen, wird eine **<a-box>** verwendet. Diese erhält die Komponente **standing-area** (siehe Abb. 14). Der Code innerhalb der Komponente **standing-area** sieht wie folgt aus (siehe Abb. 15). In diesem Beispiel zeigen sich wieder einige Vereinfachungen die A-Frame bietet. Soll die Oberflächentextur einer Entität verändert werden, so lässt sich das über das HTML-Attribut **src** machen (siehe Abb. 15, Zeile 30-33). Diese Schreibweise erin-

```

<a-box
  |   standing-area
  |   position="0 0 -3"
  ></a-box>

```

Abbildung 14: a-box mit der Komponente `standing-area`.

nert an das klassische HTML-Attribut von einem `img`-Tag. HTML-Attribute, wie `width`, `height` oder `src` können auch über den Code festgelegt werden (siehe Abb. 15, Zeile 28-33).

A-Frame Tags lassen sich über JavaScript wie normale HTML-Tags generieren und auch beispielsweise mit der Methode `querySelector` selektieren (siehe Abb. 15, Zeile 27). Über `this.el` lässt sich auf die Entität der Komponente zugreifen.

`color` legt die Farbe einer Entität fest und `color` und `height` (siehe Abb. 15, Zeile 6-7) sind zwei selbst festgelegte Eigenschaften, welche nicht zwingend in einer A-Frame Komponente vorhanden sein müssen. Selbst festgelegte Eigenschaften können auch genutzt werden, um den `state` (Zustand) einer Komponente zu speichern. `color` gibt die Farbe der `<a-box>` an und `height` die Höhe.

Die Komponente `standing-area` führt in ihrer `init`-Methode zunächst die beiden internen Methoden `setAttributes` und `addFootsteps` aus. Erste fügt die notwendigen Attribute hinzu, im das Erscheinungsbild der Entität in der virtuellen Welt festzulegen. `class` hingegen ist nicht zwingend notwendig, erleichtert jedoch zukünftig das Selektieren dieser Entität. Die zweite Methode erzeugt das Element `<a-plane>` (siehe Abb. 15, Zeile 27). Dabei handelt es sich um eine zweidimensionale Fläche. Durch das Attribut `src` erhält es Fußabdrücke als Textur (siehe Abb. 15, Zeile 30-33). Anders als die Attribute `height` oder `width`, sollen laut A-Frame Eigenschaften wie `position` und `rotation`, sofern diese über den Code angegeben oder verändert werden, über Three.js manipuliert werden⁴¹. Dies ist schneller als über HTML und spart Leistung ein.

Das `<a-plane>`-Tag wird etwas über die `<a-box>` positioniert. Dadurch, dass das Element der `<a-box>` angehängt wird, befindet sich dieses Element im DOM innerhalb des `<a-box>`-Tags. Wenn sich Entitäten innerhalb einer anderen Entität befinden, werden deren Positionsangaben relativ zum Mutterelement angegeben (siehe Abb. 15, Zeile 35-36).

Zuletzt wird in der `init`-Methode der `click`-Event-Listener hinzugefügt (siehe Abb. 15, Zeile 13-17). Durch den Aufruf von `.object3D` eines HTML-Elements kann auf die Object3D Repräsentation von Three.js zugegriffen werden. Bei einem Klick auf die Entität wird die aktuelle Weltposition ausgelesen und der Kamera hinzugefügt (siehe Abb. 15, Zeile 15-16). Die `getCamera`-Methode ist eine globale Methode, welche das Kamera-Element zurückliefert.

⁴¹ A-Frame Best Practices | <https://aframe.io/docs/1.0.0/introduction/best-practices.html> (17.11.2020)

```

5  AFRAME.registerComponent<StandingAreaComponent>('standing-area', {
6    color: '#f4d6be',
7    height: 0.075,
8
9    init() {
10      this.setAttributes();
11      this.addFootsteps();
12
13      this.el.addEventListener('click', () => {
14        const worldPosition = new THREE.Vector3();
15        const { x, y, z } = this.el.object3D.getWorldPosition(worldPosition);
16        getCamera().object3D.position.set(x, y, z);
17      });
18    },
19
20    setAttributes() {
21      this.el.setAttribute('class', 'standing-area');
22      this.el.setAttribute('color', this.color);
23      this.el.setAttribute('height', this.height.toString());
24    },
25
26    addFootsteps() {
27      const footsteps = document.createElement('a-plane');
28      footsteps.setAttribute('width', '0.5');
29      footsteps.setAttribute('height', '0.5');
30      footsteps.setAttribute(
31        'src',
32        'https://localhost:8080/public/assets/icons/footsteps.png'
33      );
34      footsteps.object3D.position.set(0, 0.038, 0);
35      footsteps.object3D.rotation.set(degToRad(-90), 0, 0);
36      this.el.appendChild(footsteps);
37    }
38  });

```

Abbildung 15: TypeScript-Code der Komponente `standing-area`.

Somit wird der Benutzer nun zu den Standfläche teleportiert, wenn er diese für 1 Sekunde anschaut (siehe Abb. 16). Auf der linken Seite der Abbildung 16 schaut der Benutzer auf



Abbildung 16: Funktionsfähige Standing Area zum Teleportieren.

die Standing Area. Rechts daneben zeigt die neue Position des Benutzers auf der Standing Area an. Er befindet sich nun vor dem Gemälde und kann dies genauer betrachten. Damit der Benutzer nicht vor einem Gemälde ohne Informationen steht, soll neben dem Gemälde einige Details zu diesem angezeigt werden. A-Frame bietet das Tag `<a-text>`, womit man zweidimensionalen Text in der virtuellen Welt anzeigen lassen kann. Damit die Inhalte nicht unübersichtlich werden, wurden die Inhalte auf Autor, Datierung, Zuschreibung und Bildträger beschränkt. Über das Attribut `value` kann ein Text festgelegt werden, welches in der virtuellen Welt angezeigt wird (siehe Abb. 17).

Durch die richtige Positionierung erscheint der Text neben dem Gemälde (siehe Abb. 18). Neben den Informationen zu einem Gemälde besitzt dieses im Cranach Digital Archive auch mehrere Detailaufnahmen. Diese zeigen besondere Stellen im Gemälde auf und bieten eine hochauflösende Nahaufnahme. Für den Benutzer soll es die Möglichkeit geben diese auch ansehen zu können. Dafür soll ein Button unter den Informationen angezeigt werden, welcher sich anklicken lässt. Wurde dieser angeklickt, erscheinen Punkte auf dem Gemälde, die dort positioniert sind, an denen sich die Nahaufnahmen befinden. Wird einer dieser Punkte angeklickt, erscheint statt dem Gemälde die Nahaufnahme vor dem Benutzer. Um diese Funktion umzusetzen, müssen zwei weitere Komponenten entwickelt werden. Zum einen der Button, mit welchem die Punkte innerhalb des Gemäldes erscheinen und zum anderen die Punkte selbst, welche das Gemälde durch eine Nahaufnahme ersetzen. Dafür muss das `src`-Attribut des Gemäldes verändert werden. A-Frame reagiert auf Veränderungen innerhalb des DOM und aktualisiert die Anwendung in Echtzeit.

```
<a-entity position="1.25 1.1 -4">
  <a-text
    value="Bildnis Luthers als Junker Joerg"
    position="0 0.9 0"
  ></a-text>
  <a-text
    value="Datierung: 1521"
    position="0 0.6 0"
  ></a-text>
  <a-text
    value="Zuschreibung: Lucas Cranach der Aeltere"
    position="0 0.3 0"
  ></a-text>
  <a-text
    value="Bildtraeger: Malerei auf Buchenholz"
  ></a-text>
</a-entity>
```

Abbildung 17: Beschreibung zum Gemälde der ID DE_MdbKL_946.

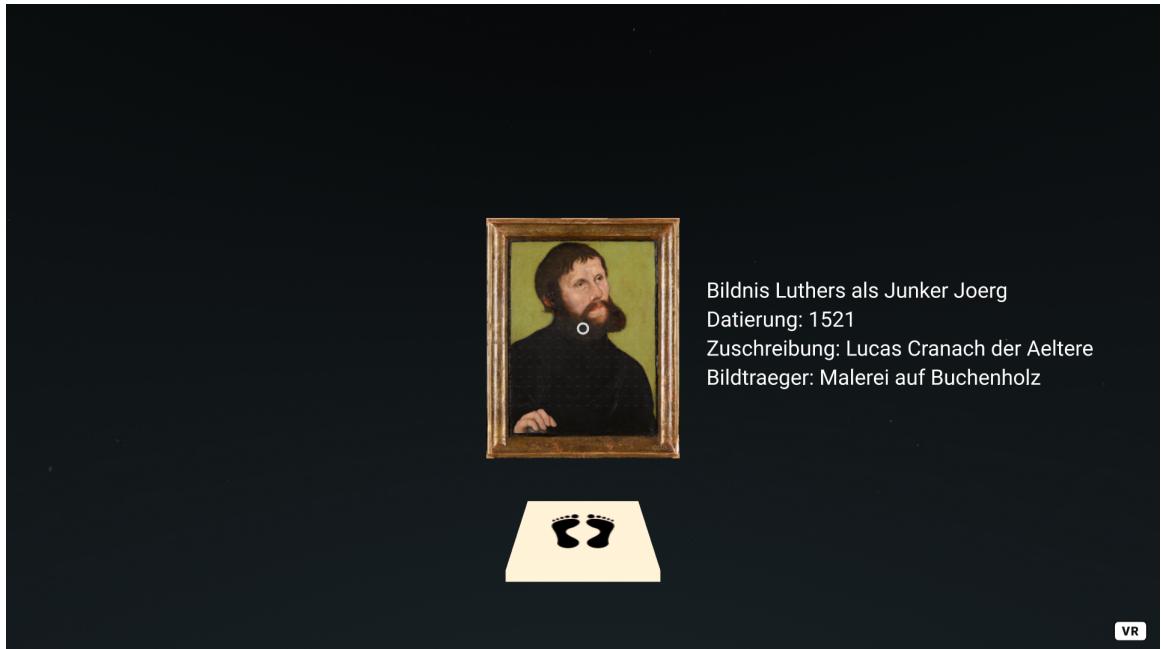


Abbildung 18: Beschreibung zum Gemälde der ID DE_MdbKL_946 innerhalb der virtuellen Welt.

Mittlerweile wird der technische Prototyp komplexer und eine Datenstruktur für die Gemälde wird notwendig, da mehrere Nahaufnahmen zu einem Gemälde existieren. Auch die Punkte, die auf dem Gemälde angezeigt werden, haben eine statische Position innerhalb des Gemäldes. Da die Datenpflege über HTML langfristig zu komplex und unübersichtlich wird, muss diese ausgelagert werden. Zuerst muss, anhand der Daten die zur Verfügung stehen, eine Struktur für ein einzelnes Gemälde entwickelt werden. Betrachtet man das aktuelle Cranach Digital Archive und deren Struktur, kommen folgende Informationen zusammen (siehe Tabelle 2). Nun müssen die folgenden Informationen auf die beschränkt

Gemäldeinformationen
Gemälde ID
Titel
Zuschreibung
Datierung
Eigentümer / Besitzer / Standort
Maße
Bildträger
Signatur / Datierung
Inscriptions / Stempel / Siegel / Beschriftungen
Kurzbeschreibung
Provenienz
Ausstellungen
Quellen / Publikationen
Forschungsgeschichte / Diskussion
Verwandte Arbeiten
Material / Technik
Erhaltungszustand
Restaurierungsgeschichte

Tabelle 2: Informationen eines Gemäldes auf <http://lucascranach.org>.

werden, welche in einer virtuellen Realität angezeigt werden sollen. Dabei sollen nur die Informationen genutzt werden, die sich von ihrer Inhaltsmenge abbilden lassen und für den Benutzer interessant sind. Möchte der Benutzer weitere Informationen erhalten, soll dieser auf einen Button klicken können, um zum Archiv weitergeleitet zu werden. Da es sich um eine Webanwendung und handelt und der Benutzer bereits im Browser ist, stellt dies kein Problem dar. Nur muss gekennzeichnet werden, dass ein neues Fenster geöffnet wird, denn dies führt zur Brechung der Ortsillusion aus Kapitel 3.1.2.

Um die Informationen in der Anwendung auslesen zu können, werden diese im JavaScript Object Notation (JSON) Format gespeichert. Dieses Format erlaubt das Auslesen von Informationen in JavaScript und anschließend das Speichern dieser in JavaScript-Objekte. Folgende Informationen werden für ein Gemälde innerhalb einer virtuellen Welt zur Darstellung benötigt (siehe Abb. 19). Da JavaScript beim Auslesen einer JSON-Datei nicht

```
{
  "id": "DE_MdbKL_946",
  "src": "https://localhost:8080/public/assets/paintings/DE_MdbKL_946/Overall.jpg",
  "title": "Bildnis Luthers als Junker Joerg",
  "attribution": "Lucas Cranach der Aeltere",
  "dating": "1521",
  "support": "Malerei auf Buchenholz",
  "description": "Brustbildnis Martin Luthers als Junker Jörg vor grünem Hintergrund, nach rechts...",
  "closeUps": [
    {
      "id": "020",
      "src": "https://localhost:8080/public/assets/paintings/DE_MdbKL_946/Detail-020.jpg",
      "position": {
        "x": -0.771,
        "y": -1.090,
        "z": 0
      }
    }
  ]
}
```

Abbildung 19: JSON-Struktur eines Gemäldes.

wissen kann, um was für ein Objekttyp es sich handelt, wird zusätzlich ein Objekt in TypeScript angelegt, um innerhalb des Codes Typsicherheit und eine verbesserte Unterstützung der Code Completion innerhalb einer Entwicklungsumgebung gewährleisten zu können (siehe Abb. 20).

Um ein Gemälde anhand einer JSON-Datei zu generieren, muss das Generieren voll-

```
interface Painting {
  id: string;
  src: string;
  title: string;
  attribution: string;
  dating: string;
  support: string;
  description: string;
  closeUps: CloseUp[];
}

interface CloseUp {
  id: string;
  src: string;
  position: THREE.Vector3;
```

Abbildung 20: Gemälde-Interface in TypeScript.

ständig auf JavaScript basieren, da das dynamische Auslesen aller Informationen nicht im HTML umgesetzt werden kann. Dafür wird eine Komponente benötigt, welche die JSON-Datei ausliest und anhand der Daten ein Gemälde, dessen Informationen und Nahaufnahmen-Buttons generiert. Die Komponente `paintings-builder` wird der `<a-scene>` hinzugefügt, da dies beim Erstellen der Szene ausgeführt werden soll. Die Komponente ist im Plural geschrieben, da die Option freigehalten werden soll, mehrere Gemälde generieren zu können. In ihrer `init`-Methode wird zuerst die JSON-Datei ausgelesen und bei Erfolg `createPainting(...)` ausgeführt (siehe Abb. 21). Diese Methode bekommt

```

| fetch('https://localhost:8080/public/data/paintings.json')
|   .then(result => result.json())
|   .then((paintingsData: Painting[]) => {
|     if (paintingsData) {
|       paintingsData.forEach((it) => this.createPainting(it));
|     }
|   });

```

Abbildung 21: Abrufen der Daten innerhalb der `paintings.json`.

```

46   createPainting(paintingData: Painting) {
47     const painting = document.createElement('a-image');
48     painting.setAttribute(
49       'painting',
50       'id:' + paintingData.id + ';' +
51       'src:' + paintingData.src + ';' +
52       'closeUps:' + JSON.stringify(paintingData.closeUps)
53     );
54
55     const description = document.createElement('a-entity');
56     description.setAttribute(
57       'painting-description',
58       'title:' + paintingData.title + ';' +
59       'dating:' + paintingData.dating + ';' +
60       'attribution:' + paintingData.attribution + ';' +
61       'support:' + paintingData.support + ';' +
62       'id:' + paintingData.id + ';' +
63       'src:' + paintingData.src
64     );

```

Abbildung 22: Codeabschnitt für die Generierung des Gemäldes und dessen Informationen.

ein Objekt des Typs `Painting` (Typdefinition siehe Abb. 20). Zum aktuellen Zeitpunkt befindet sich in der JSON-Datei die Informationen zu nur einem Gemälde. Der Code ist jedoch so entwickelt wurden, dass er ohne viel Mehraufwand mehrere Gemälde entgegennehmen kann. Innerhalb der `createPainting`-Methode wird das Gemälde, die Informationen und die Standing Area erzeugt. Bei der Standing Area handelt es sich um das selbe HTML, nur wurde dies in das JavaScript übersetzt. Da jedes Gemälde eine Standing Area benötigt, wird diese immer mit einem Gemälde mitgeneriert. Für das Verständnis der Umsetzung und um nicht den Rahmen zu sprengen, wird nicht vollständig auf die `paintings-builder`-Komponente eingegangen.

Die relevantesten Codestellen innerhalb der `paintings-builder` sind die, welche das Gemälde und die Informationen dazu generiert (siehe Abb. 22). Im Gegensatz zum vorherigen HTML, bekommen die Entitäten ihre Daten in ihre zugehörigen Komponenten hineingegeben. Das `<a-image>` -Tag bekommt nun eine Komponente namens `painting` und die In-

```

64      createDetailPoints(id: string, closeUps: CloseUp[]) {
65          const detailSphereElements: Entity[] = [];
66          closeUps.forEach(it => {
67              const sphere = document.createElement('a-sphere');
68              sphere.setAttribute('class', 'detail ' + it.id);
69              sphere.setAttribute('radius', '0.1');
70              sphere.setAttribute('color', 'blue');
71              sphere.setAttribute(
72                  'image-switcher',
73                  'id:' + id + ';' +
74                  'src:' + it.src
75              );
76              sphere.object3D.position.set(it.position.x, it.position.y, it.position.z);
77              sphere.object3D.visible = false;
78              detailSphereElements.push(sphere);
79          });
80          detailSphereElements.forEach(it => this.el.appendChild(it));
81      }

```

Abbildung 23: `createDetailPoints` innerhalb der `painting`-Komponente.

formationen zum Gemälde werden weiterhin in einem `<a-entity>` -Tag gebündelt, jedoch bekommt dieses ebenfalls eine Komponente mit dem Namen `painting-description`. Beide Komponenten bekommen über ihr Schema die notwendigen Informationen, die sie benötigen (siehe Abb. 22, Zeile 48-53 und Zeile 56-64). So wird kein HTML benötigt und die Daten können dynamisch ausgelesen und in die einzelnen Komponenten hineingegeben werden.

Die `painting`-Komponente bekommt neben ihrer eigenen Gemälde-ID und deren Pfad zur Bilddatei noch die Informationen der einzelnen Nahaufnahmen, da diese innerhalb des Gemäldes generiert werden müssen. In der `init`-Methode dieser Komponente wird die `createDetailPoints`-Methode aufgerufen (siehe Abb. 23). Diese nimmt folgende Argumente entgegen: `id: string` und `closeUps: CloseUp[]`. Die `id` des Gemäldes wird innerhalb der `image-switcher` Komponente benötigt. Die Informationen der Nahaufnahmen sind in dem Parameter `closeUps` gespeichert. Dieses Objekt beinhaltet neben der `id` und `src` auch die Positionen der Punkte relativ zum Gemälde. In der `image-switcher`-Komponente wird der `click`-EventListener registriert, welcher bei einem Klick das Gemälde mit einer Nahaufnahme ersetzt und die aktivierten Punkte ausblendet. Durch `visible = false` werden die Punkte initial unsichtbar gestellt (siehe Abb. 23, Zeile 77).

Die `painting-description`-Komponente aus Abbildung 22 hat sich nicht stark zu seiner HTML-Version verändert. Er generiert zusätzlich noch eine `<a-box>`, welche den Button darstellen soll. Dieser Button erhält eine Komponente namens `detail-button`, welche bei Klicken das Anzeigen der möglichen Nahaufnahme-Punkten ermöglicht. Dabei erhält der Button den Schriftzug „Zurück“, um von einer Nahaufnahme wieder zurück zum Gemälde

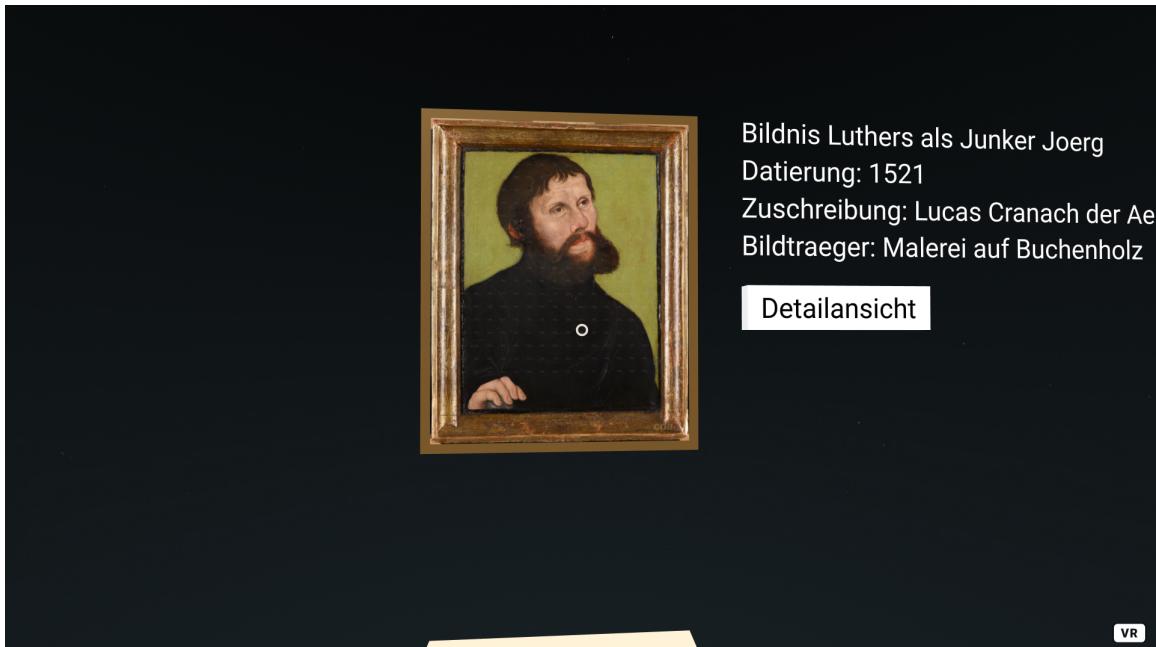


Abbildung 24: Ansicht des ersten fertigen Prototypen.

umzuschalten.

Der erste Prototyp ist somit fertig und zeigt ein Gemälde mit Informationen an und einen Button, um die Details einzublenden (siehe Abb. 24). Klickt der Benutzer auf den Button „Detailansicht“, erscheinen blaue Punkte auf dem Gemälde. (siehe Abb. 25). Zu diesen markierten Stellen existieren Nahaufnahmen von dem Gemälde. Durch einen weiteren Klick auf einer der Punkte, wird das aktuelle Gemälde durch eine Nahaufnahme ersetzt (siehe Abb. 26). Der erste technische Prototyp soll das Fundament der weiteren Prototypen darstellen. Durch das komponentenbasierte System von A-Frame, lässt sich diese Art der Darstellung in andere Szenen und in einem anderen Kontext wiederverwenden.

4.3.2 Lucas Cranachs Werkstatt

Lucas Cranach d. Ä. entwarf vieler seiner Gemälde in seiner eigenen Werkstatt. Ähnlich wie das Deutsche Auswandererhaus soll bei diesem Prototyp der Benutzer in ein Szenario aus der Vergangenheit versetzt werden, um das Erlebnis immersiver zu gestalten. Da Archivalien von Lucas Cranach d. Ä. zur Verfügung stehen, wird ein Gemälde und Dokumente aus dem selben Jahr genommen, um den Benutzer in einen Tag von Lucas Cranach d. Ä. zu versetzen. Die Schrift der Archivalien sind für den Benutzer unleserlich. Durch einen Klick auf einen Button soll die Beschreibung aus dem Cranach Digital Archive zu diesen Dokumenten vorgetragen werden. Das soll dem Benutzer das Gefühl geben, zu wissen, was bei Lucas Cranach d. Ä. an diesem Tag vorgegangen ist.

Dieser Prototyp wird qualitativ nicht an eine hochwertige Produktion gelangen, wird aber zeigen, wie durch A-Frame solche Konzepte umgesetzt werden können.

Um eine Werkstatt als Szene in A-Frame umzusetzen, werden 3D-Modelle benötigt. Nach



Abbildung 25: Detailpunkte auf dem Gemälde nach einem Klick auf den „Detailansicht“-Button.

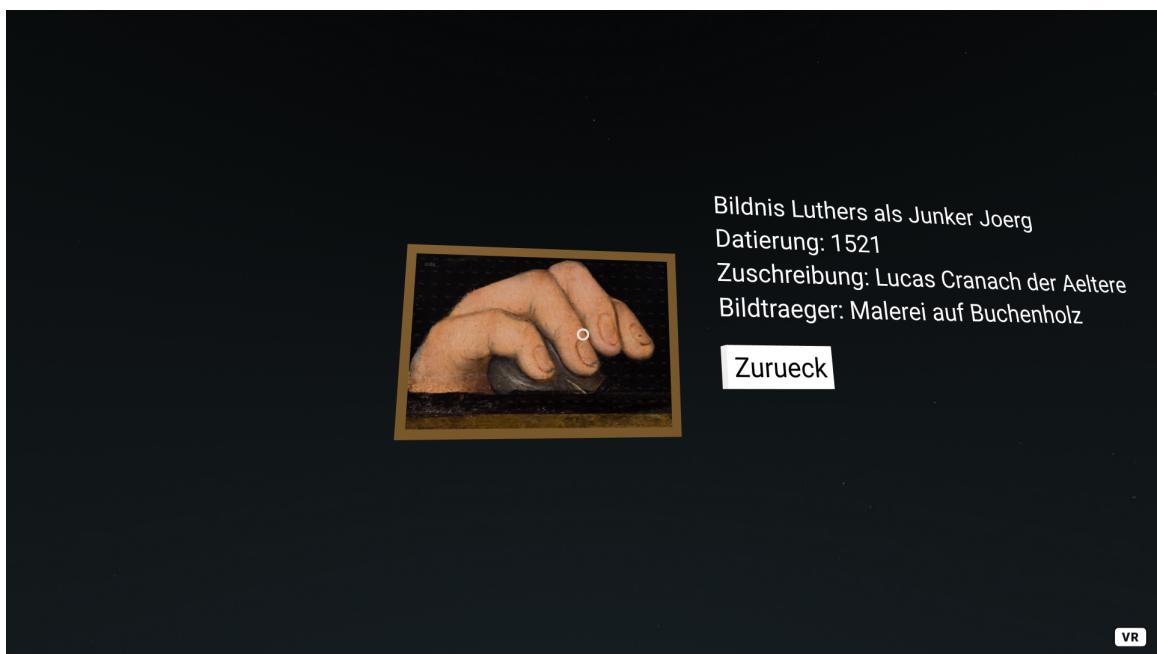


Abbildung 26: Eine Nahaufnahme des Gemäldes mit der ID DE_MdbKL_946.

```

13     <a-assets>
14         <a-asset-item id="dog" src=".../.../assets/models/dog/dog.gltf"></a-asset-item>
15     </a-assets>
16
17     <a-entity id="camera">
18         <a-camera>
19             <a-cursor
20                 color="white"
21                 fuse="true"
22                 fuseTimeout="1000"
23             >
24             </a-cursor>
25         </a-camera>
26     </a-entity>
27
28     <a-gltf-model src="#dog" position="0 1 3" rotation="90 0 0"></a-gltf-model>

```

Abbildung 27: Einbinden eines glTF-Modells in A-Frame über `<a-assets>`.

einer kurzen Recherche bei Google haben sich Poly von Google⁴² und Sketchfab⁴³ als Plattformen für kostenfreie 3D-Modelle hervorgehoben.

A-Frame bietet die Möglichkeit, ganze 3D-Modelle ebenfalls über ein HTML-Tag einzubinden. Dabei erlaubt das Framework das Einbinden von mehreren Datentypen, wie zum Beispiel glTF oder OBJ. In diesem Projekt wurde sich für glTF entschieden, da dies eine royalty-free Spezifikation für 3D-Modelle, leichtgewichtig und interoperabel ist⁴⁴. Sketchfab wie auch Poly bieten glTF als Dateiformat an.

Das Einbinden eines glTF-Modells erfolgt über das A-Frame-Tag `<a-gltf-model>`. Das Modell lässt sich über das Attribut `src` einbinden. Wie auch bei Bildern, kann der Pfad oder die ID aus dem `a-assets`-Tag genommen werden. Wird das 3D-Modell über `a-assets` eingebunden, muss dieses als `<a-asset-item>` hinzugefügt werden. Die Abbildung 27 zeigt wie ein glTF-Modell eingefügt werden kann. Komponenten wie `painting`, `painting-description` und `standing-area` können in diesem Prototypen wiederverwendet werden. Der einzige Mehraufwand entsteht durch die Modellierung der Werkstatt. Da fertige und kostenfrei Modelle aus dem Internet benutzt werden, entfällt dieser Aufwand ebenfalls. Die Werkstatt kann mittels A-Frame-Editor, welcher mit STRG + ALT + I innerhalb der Webanwendung aufgerufen werden kann, gestaltet werden.

Die Werkstatt besteht aus zwei Teilen. Zum einen der Bereich des Gemäldes (siehe Abb. 28) und zum anderen der Bereich der Dokumente (siehe Abb. 29).

Der Benutzer kann über die Standing Area zu den jeweiligen Bereichen teleportiert werden. In Abbildung 28 erkennt man die Komponente `painting` und `painting-description`,

⁴²Poly von Google | <https://poly.google.com/> (18.11.2020)

⁴³Sketchfab | <https://sketchfab.com/> (18.11.2020)

⁴⁴glTF Overview | <https://www.khronos.org/gltf/> (18.11.2020)



Abbildung 28: Aktuelles Gemälde von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.



Abbildung 29: Dokumente von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.

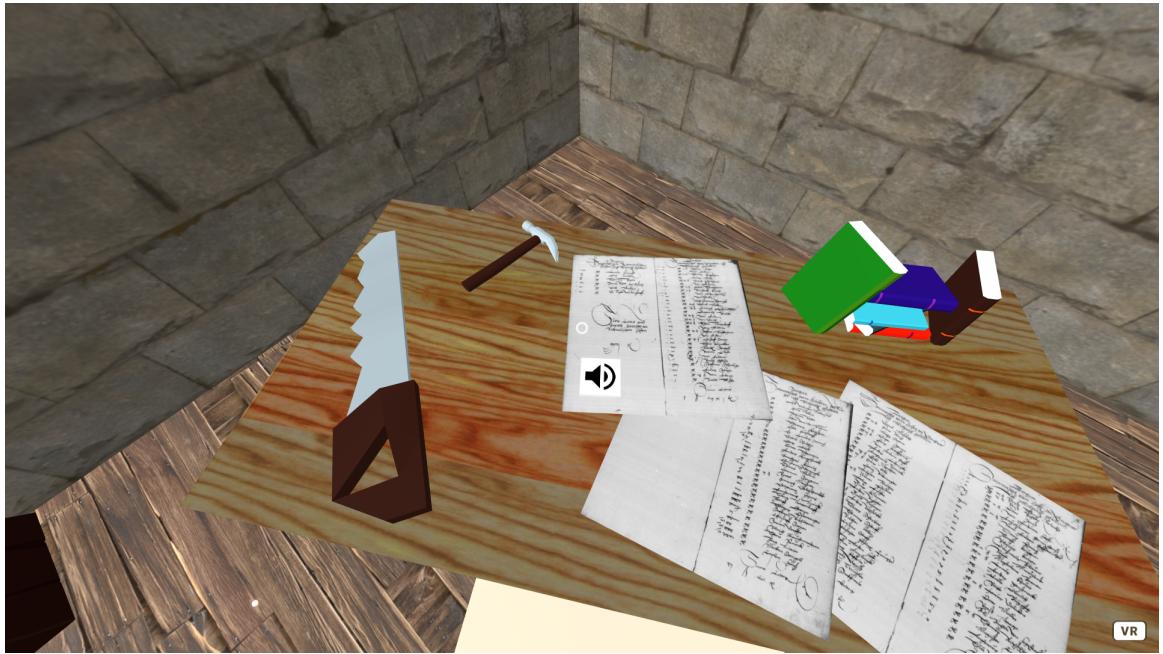


Abbildung 30: Sound-Button der Archivalien von Lucas Cranach d. Ä. in seiner fiktiven Werkstatt.

die im letzten Kapitel entwickelt wurden. Die Beschreibung des Gemäldes befindet sich an der Wand und kann von dort aus gut von dem Benutzer erkannt werden. Die Detailaufnahmen lassen sich ebenfalls betrachten.

Teleportiert sich der Benutzer zu dem Bereich der Archivalien, kann er diese von dort aus betrachten. Über einen Button, welcher sich auf einem Dokument befindet, kann er die Beschreibung zu diesem Dokument anhören (siehe Abb. 30).

Die Darstellung der Dokumente von Lucas Cranach wird über eine neue Komponente `document` ermöglicht. Diese nimmt über das `schema` die Eigenschaften `id` und `page` entgegen. Im Cranach Digital Archive haben Archivalien von Lucas Cranach d. Ä. mehrere Seiten. Durch die Eigenschaft `page` können verschiedene Darstellungen eines Dokuments der selben ID ermöglicht werden. Über `<audio>` lassen sich in `<a-assets>` Sounddateien der Anwendung hinzufügen. A-Frame bietet die Möglichkeit, mehrere Dateiformate anzunehmen. Jedes Dokument hat die Möglichkeit einen Soundbutton darzustellen. Abhängig ist dies davon, ob das Dokument eine Sound-Entität als `children`-Element besitzt (siehe Abb. 31, Zeile 92). Innerhalb der Komponente wird abgefragt, ob diese eine Sound-Entität besitzt. Ist dies der Fall, wird der Sound-Button angezeigt.

4.3.3 Neuronales Netz

Bei dem letzten Prototypen handelt es sich um eine Darstellung die einem neuronalem Netz ähnelt. In diesem soll die Beziehung zwischen den Gemälden und den jeweiligen Nahaufnahmen dargestellt werden. In diesem Prototypen befinden sich alle verfügbaren

```

74      <a-plane
75          document="id:DE_StSWR_KR-1520;page:1"
76          position="-1.956 0.700 -2.186"
77          rotation="-90 15.477 0"
78          scale="0.5 0.5 0.5"
79      ></a-plane>
80      <a-plane
81          document="id:DE_StSWR_KR-1520;page:2"
82          position="-1.562 0.693 -2.310"
83          rotation="-90 0 0"
84          scale="0.5 0.5 0.5"
85      ></a-plane>
86      <a-plane
87          document="id:DE_StSWR_KR-1520;page:3"
88          position="-2.265 0.707 -2.368"
89          rotation="-90 38.006 0"
90          scale="0.5 0.5 0.5"
91      >
92          <a-entity sound="src: #DE_StSWR_KR-1520-sound"></a-entity>
93      </a-plane>

```

Abbildung 31: document-Komponente als HTML-Implementation.

Gemälde von Martin Luther als Junker Jörg des Cranach Digital Archive und sind deshalb miteinander verwandt, da Lucas Cranach d. Ä. nicht nur eine Darstellung von Junker Jörg gezeichnet hatte, sondern verschiedene. Somit haben die Gemälde die Darstellung von Martin Luther als Junker Jörg gemeinsam. Die einzelnen Nahaufnahmen sind immer einem Gemälde zuzuordnen.

In einem neuronalen Netz werden Verbindungen meist durch eine einfache Linie dargestellt. A-Frame bietet hier die Möglichkeit mit `<a-entity line=start: x, y, z; end: x, y, z;"` eine Linie zu zeichnen. Dabei muss ein Start und Endpunkt in der virtuellen Welt angegeben werden. Zusätzlich lässt sich durch das Attribut `color` noch eine Farbe für die Linie angeben.

Um diesen Prototypen zu entwickeln, kann auf die Komponente `painting` zurückgegriffen werden. Die Nahaufnahmen sollen nun zu jedem Zeitpunkt sichtbar sein und nicht mehr durch einen Klick auf einen Button angezeigt werden. Somit können die Nahaufnahmen ebenfalls die `painting`-Komponente verwenden, um eine Darstellung in der virtuellen Welt zu erhalten.

Die roten Linien stellen die Verbindungen von einem Gemälde zu einer Nahaufnahme dar und die blauen Linien die thematische Verbindungen zwischen Gemälden (siehe Abb. 32).

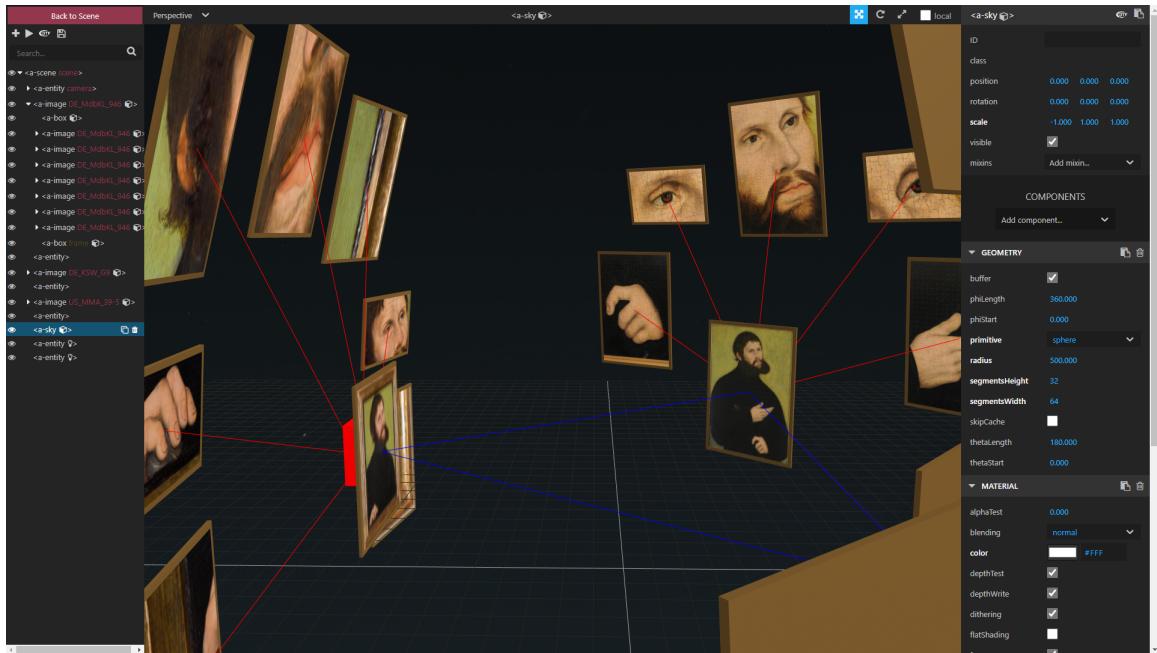


Abbildung 32: Übersicht des neuronalen Netzes aus der Inspektor-Perspektive von A-Frame.

5 Ergebnisse

A-Frame als Framework hat gezeigt, dass sich das Entwickeln von Prototypen für das Web einem fortgeschrittenen Stadium befindet. Das Framework bietet Möglichkeiten alle notwendigen Sinne anzusprechen, um eine immersive VR-Erfahrung zu bieten. Durch den Einsatz von HTML für das Prototyping und die zusätzliche Option JavaScript einzubinden gibt dem Framework alle notwendigen Funktionen, um eine simple aber auch umfangreiche Anwendung zu entwickeln.

Der **Technische Prototyp** war der Einstieg in die Entwicklung mit A-Frame und bot eine gute Übersicht darüber, was möglich ist und wie sich die Gemälde und Nahaufnahmen von Lucas Cranach d. Ä. darstellen lassen. Durch die einfache Schnittstelle über HTML ließen sich sehr schnell erste Prototypen entwickeln und durch JavaScript in ihrer Funktion erweitern. Dadurch, dass A-Frame bereits viele notwendige Funktionen wie das Darstellen von Bildern oder das Klicken über einen Cursor, welcher durch die Blickrichtung gesteuert wird, implementiert und durch einen HTML-Tag abstrahiert hat, konnte während der Entwicklungszeit viel Zeit gespart werden. Durch das Komponentensystem welches A-Frame bietet, kann redundanter Code vermieden werden. In diesem Prototypen konnten die wichtigsten Komponenten und ihre Funktionen implementiert werden, sodass die Entwicklung der weiteren Prototypen vereinfacht wurde. Die Entwicklung von **Lucas Cranachs Werkstatt** stellt einen Tag von Lucas Cranach d. Ä. aus dem Jahr 1521 dar. Die Dokumente auf dem Tisch sollen den Eindruck verstärken, dass der Benutzer weiß, was in diesem Jahr und an diesem Tag bei Lucas Cranach d. Ä. passiert ist. Durch die Ver-

tonung der Beschreibung zu diesem Dokument kann der Benutzer nachvollziehen, worum es bei diesem Dokument geht. Durch die Umgebung, die einer Werkstatt ähnelt, wird das Gefühl verstärkt, sich an diesem Ort zu befinden. Der letzte Prototyp **Neuronales Netz** verstärkt das Datenverständnis des jeweiligen Benutzers, da auf einem Blick alle vorhandenen Daten und deren Beziehungen zueinander erkennbar werden. Der Benutzer sieht alle Gemälde über Junker Jörg, die aktuell im Cranach Digital Archive verfügbar sind.

6 Diskussion

Die Prototypen haben erfolgreich gezeigt, dass virtuelle Welten innerhalb eines Browsers ohne viel Ressourcen entwickelt werden können. Auch das Aufrufen in der VR-Brille oder mit einem mobilen Endgerät haben funktioniert. Je nach Anzahl der Gemälde oder Archivalien wird eine schnelle Internetverbindung benötigt und ausreichend Datenvolumen, da die zu herunterladenden Inhalte umfangreich sein können. Das Navigieren ohne 3D-Eingabegeräte funktioniert und stellt keine Hürde bei der Verwendung dar. Sofern die Webanwendung auf einem Server hochgeladen und über eine IP oder Domain erreichbar ist, kann jeder auf diese virtuelle Welt zugreifen. Auch die Leistung die benötigt wird kann von einer eigenständigen VR-Brille oder einem modernen Smartphone erbracht werden. Obwohl es sich bei der Implementation um eine Webtechnologie handelt und diese in der Theorie von jedem modernen Browser aufgerufen werden kann, konnte nicht mit jedem Browser eine erfolgreiche VR-Erfahrung erreicht werden. Die Browser Microsoft Edge, Google Chrome, Samsung Internet und Firefox für das Smartphone ermöglichen es, die Anwendung auch ohne einen VR-Modus zu betrachten. Ein Klick auf den VR-Button schaltete erfolgreich auf die VR-Ansicht um. Die Browser Brave und Opera für das Smartphone jedoch konnten nur die normale 3D-Ansicht der Anwendung darstellen und ermöglichen nur eine Rotation innerhalb der y-Achse per Touch auf dem Smartphone.

Innerhalb der VR-Brille konnte die Webanwendung erfolgreich mit dem Oculus Browser und Firefox Reality getestet werden. Auch die Navigation innerhalb der VR-Anwendung erfolgte problemlos.

Beim ersten Laden der Anwendung konnte manchmal jedoch ein kurzer Ruckler festgestellt werden. Auch der Wechsel zwischen einem Gemälde und der Nahaufnahme passiert nicht direkt. Die Bildhöhe passt sich an die der Detailaufnahme an, aber gleichzeitig wird nicht die Detailaufnahme eingeblendet. Beide Probleme lassen sich wahrscheinlich auf die umfangreichen Dateigrößen zurückführen, da die Anwendung etwas Zeit benötigt, diese zu laden. Dieses Problem sollte sich aber beheben lassen, wenn die Anwendung als PWA installiert wird und sich dadurch die Bilddateien auf das Gerät laden.

Der Prototyp **Lucas Cranach Werkstatt** kann durch das Erstellen von professionel-

len und hochauflösenden 3D-Modellen und Texturen einen höheren Realismus erhalten, was zu einer immersiveren VR-Erfahrung beitragen würde. Durch das Gestalten mehrerer Räume, durch welche sich der Benutzer navigieren kann, könnten verschiedene Gemälde gezeigt werden, die in diesem Jahr entstanden sind oder an welchen Lucas Cranach d. Ä. aktuell arbeitet. Da er auch über Maschinen für die Druckgrafik verfügte, könnte solch ein Prozess in einer virtuellen Welt nachgestellt werden. Die Darstellung von Lucas Cranach in seiner Werkstatt, wie er an einem neuen Gemälde arbeitet, würde die Authentizität der VR-Erfahrung ebenfalls fördern.

Da das Cranach Digital Archive über eine große Datenbank verfügt, könnte im Stil des Prototyps **Neuronales Netz** ein umfangreiches und eindrucksvolles Netz aufgebaut werden, welches automatisiert generiert wird. Dabei könnten verschiedene Filtermöglichkeiten für die einzelnen Verbindungen zwischen den Gemälden implementiert werden. So könnten komplexe Beziehungen zwischen Gemälden und auch Archivalien visuell dargestellt werden. Durch eine dreidimensionale freie Bewegung in dem virtuellen Raum wäre dem Benutzer keine Grenzen in der Betrachtung der Daten gesetzt. Jedoch muss bei dieser Lösung auch eine schnelle Internetverbindung gewährleistet werden oder die Webanwendung muss als PWA installiert werden können.

7 Fazit

Um eine immersive und qualitativ wie auch quantitativ hochwertige VR-Erfahrung zu gestalten, müssen zunächst die Wahrnehmungsaspekte des Menschen berücksichtigt werden. Je mehr Sinne angesprochen werden, desto immersiver die Erfahrung für den Benutzer. Dabei gibt es drei Aspekte zu beachten: Orts- und Plausibilitätsillusion und die Involviertheit.

Die **Ortsillusion** ist das Gefühl, sich an dem Ort innerhalb der virtuellen Welt zu befinden, mit dem Wissen, dass man nicht wirklich dort ist. Dazu ist die blickpunktabhängige Bildgenerierung innerhalb einer VR-Brille zwingend notwendig.

Die **Plausibilitätsillusion** beschreibt das Gefühl, dass die Geschehnisse in der virtuellen Welt gerade im Moment passieren, obwohl der Benutzer zu jedem Zeitpunkt weiß, dass dies nicht tatsächlich passiert. Diese Illusion kann beispielsweise durch Geschehnisse hervorgerufen werden, die den Benutzer betreffen, welche er aber nicht selbst ausgelöst hat. Beispielsweise wenn ihn plötzlich eine Person innerhalb der virtuellen Welt anspricht. Dabei müssen die Ereignisse aber nicht physikalisch korrekt sein.

Bei der **Involviertheit** geht es um das Interesse und die Aufmerksamkeit des Benutzers. Je höher diese beiden Eigenschaften sind, desto höher ist die Qualität der VR-Erfahrung für den Benutzer.

Für Museen steckt sehr viel Potenzial in Virtual Reality, da diese ganze Ausstellungen in das Wohnzimmer der Zielgruppe bewegen können. Jedoch muss bei der Darstellung der

Ausstellungsstücke gekennzeichnet werden, ob es sich um eine Originaldarstellung handelt oder ob diese künstlich verändert wurde, da ansonsten ein objektives Bild vermittelt werden kann.

Die Aufgabe, Informationen und Emotionen zu verbreiten, kann durch Virtual Reality an eine große Zielgruppe getragen werden. Durch den Einsatz von Webtechnologien für die Umsetzung von VR-Anwendungen muss auch niemand durch seine Technologiewahl ausgeschlossen werden. Jeder mit einem modernen Smartphone oder einer beliebigen VR-Brille kann diese Anwendung aufrufen. Es ist auch keine Installation notwendig und die VR-Erfahrung kann sofort aufgerufen werden.

Ein Museum würde nicht weniger Besucher durch eine online abrufbare Anwendung bekommen, da originale Ausstellungsstücke einen nicht reproduzierbaren Charakter besitzen. Die virtuellen Darstellungen haben nicht die gleiche Authentizität wie ihre originalen Gegenstücke. Solch eine Webanwendung kann sogar als Marketingwerkzeug betrachtet werden, da viele Besucher, die vorher eine Onlineausstellung betrachtet haben, eher dazu neigen, dass Museum auch in der realen Welt zu besuchen. Demnach ist es für die Museen eine Chance, um eine neue Zielgruppe anzusprechen.

Der heutige Fortschritt von Browsern ist so fortgeschritten, dass sich umfangreiche VR-Erfahrungen entwickeln lassen. Dank Schnittstellen wie WebGL und WebXR und Frameworks wie A-Frame, können diese auch teilweise mit nativen Entwicklungen mithalten. Sofern die Ressourcen vorhanden sind, können professionelle und realistische virtuelle Welten gestaltet werden. Dabei können die Schnittstellen auf die meisten VR-Geräte zugreifen und ihre 3D-Eingabegeräte erkennen.

8 Nachwort

Ich danke allen.

9 Quellenverzeichnis

Literatur

Ater, Tal (2017). *Building Progressive Web Apps - Bringing the Power of Native to the Browser*. United States of America, 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc. ISBN: 978-1-491-96165-0. URL: <https://www.oreilly.com/library/view/building-progressive-web/9781491961643/>.

Carrozzino, Marcello u. a. (2018). „Comparing Different Storytelling Approaches for Virtual Guides in Digital Immersive Museums“. In: *Augmented Reality, Virtual Reality*,

- and Computer Graphics*. Hrsg. von Lucio Tommaso De Paolis und Patrick Bourdot. Cham: Springer International Publishing, S. 292–302. ISBN: 978-3-319-95282-6.
- Dörner, Ralf u. a. (2013). *Virtual und Augmented Reality (VR/AR)*. Berlin, Deutschland: Springer Vieweg. ISBN: 978-3-662-58860-4. DOI: [10.1007/978-3-662-58861-1](https://doi.org/10.1007/978-3-662-58861-1). URL: <https://www.springer.com/gp/book/9783662588604>.
- Haas Andreas und Rossberg, Andreas u. a. (Juni 2017). „Bringing the web up to speed with WebAssembly“. In: *Proceedings. of the 38th ACM SIGPLAN Conference. on Programming. Language. Design. and Implementation.* 52.6, S. 185–200. ISSN: 0362-1340. DOI: [10.1145/3062341.3062363](https://doi.org/10.1145/3062341.3062363).
- Halpern, Daniel und James Katz (2015). „Can Virtual Museums Motivate Students? Toward a Constructivist Learning Approach“. In: *Journal of Science Education and Technology* 24, S. 776–788. ISSN: 1573-1839. DOI: [10.1007/s10956-015-9563-7](https://doi.org/10.1007/s10956-015-9563-7). URL: <https://doi.org/10.1007/s10956-015-9563-7>.
- Heidsiek, Katie (2019). *Berührt es mich? Virtual Reality und ihre Wirkung auf das Besuchserlebnis in Museen – eine Untersuchung am Deutschen Auswandererhaus*. Deutsches Auswandererhaus Bremerhaven, edition DAH. ISBN: 978-3-9817-8619-4. URL: <https://dah-bremerhaven.de/beruehrt-es-mich-virtual-reality-und-ihre-wirkung-auf-das-besuchserlebnis-in-museen-eine-untersuchung-am-deutschen-auswandererhaus/>.
- Heydenreich, G. u. a. (2017). *Lucas Cranach der Ältere: Meister, Marke, Moderne*. Museum Kunsthalle. ISBN: 978-3-777-42744-7. URL: https://www.hirmerverlag.de/de/titel-87-2/lucas_cranach_der_aelttere-1463/.
- Hünnekens, Annette (2002). *Expanded Museum - Kulturelle Erinnerung und virtuelle Realität*. Bielefeld: transcript. ISBN: 978-3-8394-0089-0. URL: <https://www.transcript-verlag.de/978-3-933127-89-1/expanded-museum/>.
- Slater, Mel (2009). „Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments“. In: *Phil. Trans. of the Royal Society B* 364: 3549–3557. DOI: <http://doi.org/10.1098/rstb.2009.0138>.
- Witmer, Bob G. und Michael J. Singer (1998). „Measuring Presence in Virtual Environments: A Presence Questionnaire“. In: *Presence: Teleoperators and Virtual Environments* 7.3, S. 225–240. DOI: [10.1162/105474698565686](https://doi.org/10.1162/105474698565686). eprint: <https://doi.org/10.1162/105474698565686>. URL: <https://doi.org/10.1162/105474698565686>.

Erklärung über die selbständige Abfassung der Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.

Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

(Ort, Datum, Unterschrift)

