IU Internationale Hochschule GmbH

Bachelorarbeit Informatik

Entwicklung einer Webanwendung zur Animation von Gesichtsausdrücken mit Three.js und Action Units nach dem FACS

Florian Lessig

Betreuer

Prof. Ph.D. Stephan de la Rosa

Eingereicht am 23.06.2024

Anmerkungen

In Absprache mit Prof. Ph.D. Stephan de la Rosa wurde die Seitenzahl der Arbeit auf Grund der zahlreichen Abbildungen überschritten.

Der Programmcode zu dieser Anwendung wird in dem GitHub Repository

https://github.com/flow217/FaceEx

zur Verfügung gestellt.

Alle Dateien des Repository, mit Ausnahme der Ordner three.js und jema.js sowie die wasm und glb Dateien im Ordner Releases wurden durch den Autor erstellt.

Zusammenfassung

Zur einfachen Darstellung von Gesichtsausdrücken, dem Speichern statischer Bilder und dem komfortablen Erstellen von Animationen entwickelt diese Arbeit einen Prototypen, der ein Blendshape Set auf das Kodierungssystem des FACS abbildet. Hierzu wird das JavaScript Framework three.js verwendet und das darin enthaltene Face Mesh. Als Interface wird eine Webseite erstellt, die den Nutzer sowohl die Animation des Gesichts als auch die Steuerung ermöglicht. Es wird besonderer Wert auf eine einfache und freie Software gelegt. Hierzu gehört auch eine installationsfreie Nutzung, indem die Anwendung vollständig im Browser ausgeführt wird. Es war möglich zu zeigen, dass ein so umfangreiches System wie das FACS prinzipiell in einer kleinen Webanwendung abgebildet werden kann. Ebenso konnte gezeigt werden, dass auch die Darstellung eines Gesichts, das eine komplexe geometrische Struktur ist, mit teils erheblichem Berechnungsaufwand in der Animation möglich ist. Die Performanz von WebGL im Browser ist ausreichend, um ein Gesicht mit einfachen Texturen anhand eines Blendshape Sets in beliebiger Weise zu animieren. Zur vollständigen Darstellung sind noch einige Erweiterungen notwendig und bis eine funktionierende Software daraus entsteht noch weitere Entwicklungsschritte erforderlich, aber die grundlegenden Funktionalitäten sind erreicht worden.

Abstract

For the simple representation of facial expressions, the storage of static images, and the convenient creation of animations, this work develops a prototype that fulfills the mapping of Blendshapes to the FACS system. The JavaScript framework three.js and its included face mesh are used for this purpose. An interface is created as a webpage that allows the user to animate the face and control it. Emphasis is placed on the easy and free use of the software. This also includes an installation free usage by running the application entirely in the browser. It was possible to demonstrate that a system as extensive as FACS can, in principle, be mapped in a small web application. It was also shown that the representation of a face, which is a complex geometric structure, with sometimes significant computational effort in the animation, is possible. The performance of WebGL in the browser is sufficient to animate a face with simple textures in any way using a blendshape set. Some extensions are still necessary for a complete representation, and further development steps are required before a fully functioning software can be created, but the basic functionalities have been achieved.

Keywords:

FACS, Facial Expressions, Action Units, Face Animation, Emotions, Browser Application

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

FACS, Facial Action Coding System

JSON, JavaScript Object Notation

HTML, Hypertext Markup Language

CSS, Cascading Style Sheets

OpenGL, Open Graphics Library

WebGL, Web Graphics Library

AU, Action Unit/Action Units

AD, Action Descriptor

SPA, Single Page Application

GPU, Graphics Processing Unit

API, Application Programming Interface

ISI, Interstimulus-Interval

AR, Augmented Reality

DOM, Document Object Model

WHATWG, Web Hypertext Application Technology Working Group

glTF, Graphics Language Transmission Format

JS, JavaScript

1.Einleitung

1.1 Problemstellung

Ausdrücke des menschlichen Gesichts und Rückschlüsse, die daraus gezogen werden können, sind ein Forschungsgegenstand, der in der Psychologie eine bedeutende Rolle spielt. Diese Arbeiten benötigen messbare und reproduzierbare statische und dynamische Abbildungen von Gesichtern, Gesichtsausdrücken und Emotionen. Dabei werden teils Aufnahmen von Schauspielern oder Freiwilligen verwendet, aber auch synthetische Abbildungen, die durch einen Computer erzeugt wurden. Die künstlich generierten Bilder bieten den Vorteil, dass sie in großem Umfang und Variationen erzeugt werden können und dabei eine gute Vergleichbarkeit haben. So können beispielsweise zwar durch geübte Schauspieler Emotionen nach dem FACS (Facial Action Coding System) dargestellt werden, das ist aber mit einem gewissen Aufwand verbunden. In Aufzeichnungen ist es nur in begrenztem Umfang möglich Änderungen in der Darstellung vorzunehmen. Hier bieten Computeranimationen eine Alternative zu den klassischen Fotos und Videos. Diese Anwendungen sind aber teils mit einem erheblichen Aufwand verbunden und erfordern vom Anwender meist eine Einarbeitungszeit sowie Kosten und sind für kleinere Versuche oder im Rahmen von Studierendenexperimenten weniger geeignet. So ist eine Software wie Blender zwar in der Lage alles als Animation darzustellen was sich der Mensch vorstellen kann, aber mit einer steilen Lernkurve und viel Aufwand verbunden. Blendshapes, die in solchen Anwendungen verwendet werden, sind kein standardisiertes System, haben keine allgemeinen Bezeichnungen und unterscheiden sich teils erheblich in ihrer Anzahl. Das FACS wurde entwickelt, um ein subjektives Feld wie das des menschlichen Gesichts zu kategorisieren und damit messbar zu machen. Applikationen, welche die Detektion und Zuordnung von Gesichtsausdrücken zu den Kodierungen des FACS durchführen, existieren zahlreich. Anwendungen, die dem Nutzer das Modellieren von Gesichtsausdrücken und die Transition zwischen ihnen, auf Basis der FACS-Kodierung, ermöglichen, sind dagegen nicht vorhanden. Es konnte keine Anwendung gefunden werden, die das schnelle und einfache Darstellen von Gesichtsausdrücken und Animationen ermöglicht und dabei intuitiv und uneingeschränkt benutzbar sowie kostenlos ist. Die akademische Forschung und Lehre profitierten aber erheblich von dem Open Access Gedanken, der jedem die Nutzung von Tools und Wissen ermöglichen soll.

1.2 Zielsetzung

In dieser Arbeit soll ein Lösungsansatz entwickelt werden, der eine einfache und nachvollziehbare Generierung von Bildern und Animationen beschreibt. Dieser soll kostenlos, Open Source, leicht verständlich, ohne Installationsaufwand, schnell anpassbar und vielseitig genug sein, um die verschiedenen Anforderungen, welche die Untersuchung von Gesichtsausdrücken bereit hält, zu unterstützen. Dieses Ziel soll durch eine Anwendung erreicht werden, welche im Webbrowser mit guter Performanz, ohne Vorkenntnisse in 3D Modellierung, von Menschen mit Grundkenntnissen in der Verwendung des FACS direkt benutzt werden kann. Anstelle der Steuerung über Blendshape Werte sollen die AU (Action Units) und auf diesen aufbauende Kombinationen verwendet werden. Das

Erstellen von Keyframes für eine gewünschte Animation soll intuitiv erfolgen. Dem Nutzer soll es ermöglicht werden eine Konfiguration von Action Units zu erstellen und damit die Ausprägungen in der Kodierung des FACS zu bestimmen. Die Darstellung durch AU, die Bestimmung der Start und Endpunkte sowie der Dauer des Übergangs zwischen ihnen, soll es ermöglichen beliebigen Umwandlungen zwischen ihnen zu erreichen. Innerhalb einer Animation soll ein schwarzer Bildschirm als ISI (Inter Stimulus Intervall) eingefügt werden können. Die Anwendung unterstützt zwei Funktionen. Zum einen können beliebige Gesichtsausdrücke durch Action Units kombiniert werden und dann prozentual eingestellt werden. Es soll aber ebenfalls möglich sein, Gesichtsausdrücke mit dem FACS zu kodieren und als Bild zu speichern oder als Keyframes in einer Animation zu verwenden.

1.3 Aufbau der Arbeit

Zunächst erfolgt die Vorstellung der psychologischen und technischen Grundlagen auf welcher diese Arbeit aufgebaut ist. Danach wird anhand eines hypothetischen Anwendungsbeispiels die Herleitung der Anforderungen und deren tabellarische Darstellung besprochen. Anschließend folgen Ausführungen zu Architektur Entscheidungen. Eine Liste der Anforderungen wurde erstellt in Anlehnung an ein Produkt Backlog und der Arbeit angehängt. Überlegungen, die nicht umgesetzt wurden, werden kurz erläutert. Die Implementierung der Anwendung wird dargestellt und auf die wichtigsten Komponenten und ihre Funktionalitäten eingegangen. In der folgenden Vorstellung der Anwendung wird die vorgesehene Nutzerführung und ein beispielhafter Ablauf präsentiert. An geeigneter Stelle wird auf den Erfüllungsgrad der Anforderungen eingegangen. Zuletzt folgt eine kritische Betrachtung der Aufgabenstellung, ihrer Erfüllung und Möglichkeiten zur Weiterentwicklung.

2. Grundlagen

2.1 Facial Action Coding System

Der Gesichtsausdruck eines Menschen ist maßgeblich durch die Aktivität der mimischen Muskulatur geprägt. Aber auch das Öffnen und Schließen der Augen und des Mundes sowie der Ausdruck der Augen, Falten und die Beschaffenheit der Haut erschaffen einen charakteristischen Ausdruck (Aumüller et al., 2020, S. 959). Anatomisch normal entwickelte Menschen haben somit ein ähnliches Repertoire an Ausdrücken ihres Gesichtes und diese können somit prinzipiell kategorisiert werden.

Da es kein universell vergleichbares System gab um Gesichtsausdrücke zu beschreiben, sondern nur Systeme zur Untersuchung von bestimmten Verhaltensmustern, wurde das Facial Action Coding System entwickelt, um dieses Problem zu lösen (Ekman & Friesen, 1976, S. 56–60). Das FACS ist somit ein systematischer Ansatz die möglichen Gesichtsbewegungen des Menschen zu messen und zu kategorisieren und ermöglicht eine nachvollziehbare Sprache über Beobachtungen im Gesicht des Menschen. Aus dem System entstand ein Handbuch, das Interessierten die Erkennung und Bewertung sowie die Nachstellung der menschlichen Ausdrücke des Gesichts beibringen soll (Ekman et al., 2002, S. 1) .

Das Handbuch des FACS unterliegt einem strengen Copyright, das es nicht ermöglicht Bilder oder Abbildungen daraus in dieser Arbeit zu benutzen.

Action Units

Die Veränderung des menschlichen Gesichtsausdrucks kann aufgrund der Aktivität der Gesichtsmuskulatur in sogenannte Action Units kategorisiert werden. Eine AU repräsentiert eine spezifische Veränderung im Gesichtsausdruck der beobachteten Person. Während viele Gesichtsausdrücke direkt auf die Aktivität eines bestimmten Muskels zurückgeführt werden konnten, stießen Ekman und Friesen auch auf Fälle, in denen die Differenzierung schwierig war. Sie beobachteten dass sich Gesichtsveränderungen entweder nur in der Intensität unterschieden oder mehrere Muskeln den Ausdruck erzeugen konnten (Ekman & Friesen, 1976, S. 64). Sie konnten ebenfalls unterschiedliche Action Units durch die Aktivierung eines einzigen Muskels erzeugen. Deshalb nahmen sie Abstand zur rein muskulären Beschreibung und definierten Action Units basierend auf dem beobachtbaren Ergebnis einer Gesichtsbewegung, nicht nur auf der zugrunde liegenden Muskeltätigkeit. Daher entschieden sie sich für die Bezeichnung Action Unit und nicht Muscle Unit.

Neben den AU existiert im FACS noch der Begriff des AD (Action Descriptor). Diese sind laut den Autoren ebenso wichtig, allerdings nicht so spezifisch definiert wie die AU (Ekman et al., 2002, S. 295,304,322). Ein Beispiel ist das Aufblähen der Backen beschrieben durch AD34, welches auch in der Standardkonfiguration dieser Arbeit enthalten ist oder das seitliche Bewegen des Unterkiefers in AD30. Daneben gibt es noch weitere Verhaltensweisen wie Schulterzucken oder Kauen sowie Bewegungen der Augen, des Kopfes und die Sichtbarkeit von Gesichtsbereichen, welche ebenfalls codiert werden, indem das Präfix weggelassen wird. Die Zahl 70 bezeichnet beispielsweise, dass die Augen nicht sichtbar sind. Die zuvor genannten Bewegungen werden mit vorangestelltem „M“ kodiert, was von ihrer Klassifizierung als „Miscellaneous Actions“ hergeleitet wird und Bewegungen bezeichnet, die ihren Ursprung nicht unmittelbar in der Gesichtsmuskulatur haben.

In dieser Arbeit wird primär die Verwendung der AU in der Anwendung demonstriert, daher wird der Begriff AU verallgemeinert und bezeichnet auch die soeben erläuterten Begriffe. Ebenfalls soll mit AU sowohl Singular als auch Plural bezeichnet werden. Die meisten der Bewegungsmuster des FACS können ebenfalls implementiert werden. Beispielsweise ist die Seitwärtsbewegung des Unterkiefers in den verwendeten Blendshapes bereits enthalten und auch die Augen und das Mesh können räumlich verändert werden, allerdings nicht die Zunge.

Nomenklatur

Das FACS ordnet jeder AU eine Ausprägungsstärke, A,B,C,D,E zu ab welcher sie mit dem entsprechenden Buchstaben bewertet wird (Ekman et al., 2002, S. 8).

Scoring:

A: „trace of the action“

B: „slight evidence“

C: „marked or pronounced“

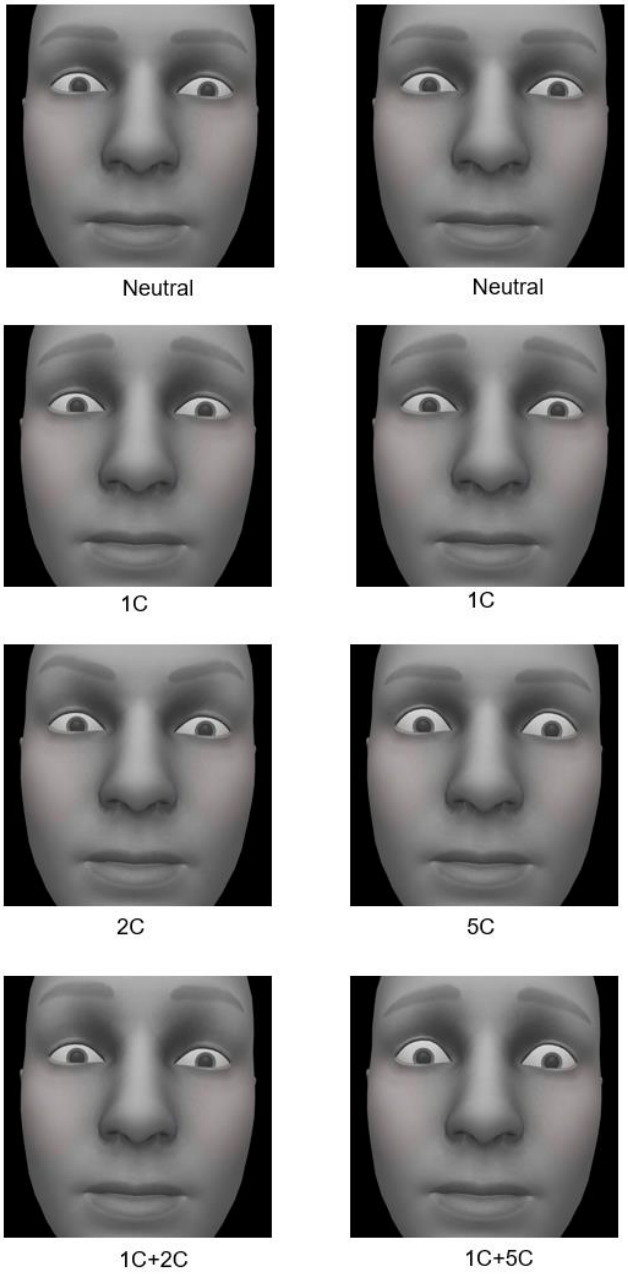
D: „severe or extreme“

E: „maximum evidence”

Bei dieser Einteilung ist zu beachten, dass es sich nicht um eine gleichmäßige Intensitätsverteilung handelt, sondern die Bereiche C und D, in welcher sich auch der Großteil der AU Bewegungen abspielt, einen größeren Bereich abdecken als A,B und E (Ekman et al., 2002, S. 8). Wie oben bereits erwähnt, richtet sich die entwickelte Anwendung an Personen, welche mit dem FACS vertraut sind. Für das Verständnis des FACS genügt diese grobe Einteilung. Auf die Folgen dieses Bewertungssystems wird im Anwendungsteil noch explizit eingegangen. Die einzelnen Action Units werden im FACS als AU n bezeichnet wobei n eine willkürliche Zuordnung einer Zahl ist, welche keiner Systematik unterliegt (Ekman et al., 2002, S. 6). Laut den Autoren sollen auch die wörtlichen Zuweisungen nicht auswendig gelernt werden, sondern die Nummern und ihre Bedeutung. Beispielsweise ist der wörtliche Eintrag der AU 12 „Action Unit 12 – Lip Corner Puller“ (Ekman et al., 2002, S. 178) und soll in Kombination mit dem Scoring Wert C kurz als 12C bezeichnet werden. Das Auftreten mehrerer AU wird durch ein „+“ gekennzeichnet. Hierbei ist jedoch zu beachten, dass die einzelnen Bestandteile einer Kombination 1+2+ … + n unabhängig voneinander, sich nicht überlagernd, sich gegenseitig beeinflussend und verstärkend oder abschwächend oder als Gesamtbild kategorisiert werden können (Ekman et al., 2002, S. 11;12). Im FACS wird zu jeder AU eine Beschreibung dieser Kombinationen sowie eine Erläuterung signifikanter Kombinationen geboten. Zum Verständnis dieser Arbeit ist es ausreichend das „+“ im intuitiv angenommenen additiven Sinne zu verstehen, wobei 1+2 als, zu AU1 tritt AU2 hinzu und 1A + 2C bedeutet eine AU 1 „Action Unit 1 – Inner Brow Raiser“ welcher gerade wahrnehmbar ist ,wird durch „Action Unit 2 – Outer Brow Raiser“ mit einer deutlich erkennbaren Ausprägung ergänzt (Ekman et al., 2002, S. 21;23;54).

Dabei unterscheiden sich die Kombination 1C + 2C in diesem Beispiel sehr deutlich von der Bewertung der 1C und 2C in Isolation. Zum Vergleich sei hier noch die “ Action Unit 5 – Upper Lid Raiser“ erwähnt welche in Stärke C in Kombination mit AU1 als nicht kombinierte AU gewertet werden kann (Ekman et al., 2002, S. 27). Dieser Sachverhalt wird in Abbildung 1 verdeutlicht.

Abb. 1 – Kombination von Action Units



Quelle: Eigene Darstellung

Laterale Bewertung

AU können einseitig oder beidseitig auftreten (Ekman et al., 2002, S. 370). Beidseitig ist die Aktion, wenn die der AU zu Grunde liegenden Muskeln auf beiden Seiten aktiviert werden. Einseitig ist sie, wenn ersichtlich ist, dass Aktivität auf der einen Seite aber nicht auf der anderen Seite vorliegt. Wenn der Fall einer unilateralen Bewegung gegeben ist, wird diese mit L für links und R für rechts an die Codierung der AU angefügt. Beispielsweise wäre R2, die einseitig gehobene rechte Augenbraue von Leonard Nimoy als Mister Spock in der Serie Star Trek, eine unilaterale Bewegung. Die auf den obigen Abbildungen ersichtliche Kombination 1C+5C eine bilaterale Bewegung der inneren Brauenheber und der oberen Lidheber.

Asymmetrische Bewertung

Die asymmetrische Bewertung ist eine Alternative zu der lateralen Bewertung und nicht für alle AU möglich (Ekman et al., 2002, S. 370). Ebenso ist die Auswertung auf Standbildern auf Grund der Veränderung einer Handlung zwischen Beginn und höchster Ausprägung problematisch, da hier nicht beurteilt werden kann an welchem Punkt des Bewegungsablaufs die Asymmetrie auftrat. Es wird von den Autoren daher eine Bewertung auf Basis von Bewegungsabläufen vorausgesetzt. Die Bewertung folgt nun der Ausprägung am „apex“ der höchsten Stelle der Ausprägung und im Zeitfenster von 200ms am höchsten Punkt der Ausprägung. Es erfolgt dann eine eigene Codierung mit individuellen Präfixen je nach Ausprägungsstärke beider Seiten zueinander (Ekman et al., 2002, S. 372). Da diese Kodierung in dieser Anwendung zwar zur späteren Implementierung berücksichtigt ist, in der derzeitigen Version aber keine Rolle spielt, wird auf ausführliche Erläuterung verzichtet.

2.2 Adaptation Aftereffects

Da in der exemplarischen Anwendung der Software eine Studie zu Adaptationseffekten beschrieben wird, erfolgt eine kurze Erklärung dieser Effekte. Adaptation in der wissenschaftlichen Untersuchung von sensorischen und wahrnehmungsbezogenen Phänomenen bezieht sich auf die Reaktion eines Organismus oder dessen sensorischen Rezeptors auf anhaltende Exposition gegenüber einem Umweltstimulus (Palumbo et al., 2017, S. 1). Dabei ist die Adaptation von sensorischen Rezeptoren gleichbedeutend mit einer reduzierten elektrischen Antwort des Rezeptors nach längerer Stimulation. Bei den Aftereffects handelt es sich um Veränderungen in der Wahrnehmung eines Reizes nach längerer Exposition gegenüber einem vorangegangenen Stimulus, dem Adaptor. Aftereffects zeigen sich in der veränderten Wahrnehmung des Teststimulus im Vergleich zu seiner Wahrnehmung, wenn er isoliert, ohne Adaption, präsentiert wird.

2.3 Polygon Mesh

Zur Darstellung von Gesichtern und komplexen Oberflächen im Allgemeinen wird in der Computergrafik häufig die Darstellung durch Polygone, die miteinander verbunden sind, im speziellen Dreiecke und Vierecke, zurückgegriffen. Diese Entwicklung nahm ihren Anfang in der Arbeit von Gouraud zur Darstellungen gekrümmter Oberflächen im Jahr 1971 (Gouraud, 1971, S. 1). Das führte in der Arbeit von Perke bereits zu einem deutlich erkennbaren menschlichen Gesicht bestehend aus 250 Polygonen (Parke, 1972, S. 451). Ein Mesh bezeichnet eben diese vernetzte Struktur von Polygonen. In der heutigen Computergrafik wird das Mesh, welches ein Gesicht darstellt, auch als Face Mesh bezeichnet. Die Berechnung mittels planaren Polygonen erfolgt sehr schnell, da sie nur aus linearen Gleichungen bestehen, die schnell zu berechnen sind (Nischwitz et al., 2019, S. 145). Das in dieser Arbeit verwendete Face Mesh ist in Three.js, im Beispiel Teil bereits enthalten und unterliegt der MIT Lizenz, die eine freie Verwendung gewährt (three.js n.d.-b). Als Urheber ist www.bannaflak.com genannt, das Mesh wurde in einer App für Apple Anwender verwendet. Dieses Mesh orientiert sich, obwohl nicht explizit genannt, anscheinend an dem ARKit Standard von Apple, der zur

Entwicklung von AR (Augmented Reality) genutzt wird. Die Bezeichnungen sind zwar geringfügig anders, so ist die Bezeichnung für die linke Augenbraue browDownLeft bei Apple und browDown_L bei three.js (Apple Inc., n.d.-a). Für die Umsetzung in dieser Arbeit spielt aber die genaue Bezeichnung nur eine untergeordnete Rolle, da eine universelle Verwendung von Face Meshes angestrebt wird und somit eine namentliche Nennung durch den Nutzer nicht vorgesehen ist.

2.4 Blendshapes

Blendshapes stellen eine Linearkombination von Zielpositionen eines Face Mesh dar, diese werden in Three.js Morph Targets genannt. Sie werden von Hand modelliert und orientieren sich in ihrem Design meist an dem FACS (Anjyo, 2016, S. 1,2). Sie dienen unter anderem auch einer sinnvollen Führung des Designers, um unnatürliche Verformungen des Meshes zu verhindern (Lewis et al., 2014, S. 202). Sie sind ein häufig angewandtes Mittel der Computergrafik, um realistische Gesichtsausdrücke darzustellen. Dazu werden die Zielpositionen der Blendshapes oder Morphtargets in den Keyframes der Animation definiert und die Bilderfolge zwischen ihnen, der Übergang, algorithmisch berechnet.

Da der Ausdruck des menschlichen Gesichtes von mehr abhängt als der Verformung der Oberflächenstruktur geht aus der Anatomie des Gesichts hervor. So beschreiben Aumüller et al. den Tonus der mimischen Muskulatur als den primären Faktor zur Formung von Gesichtsausdrücken. Allerdings spielen auch die Ausprägung von Falten, vor allem an Nase und Kinn, der Augenausdruck, sowie die Beschaffenheit der Haut eine Rolle (Aumüller et al., 2020, S. 959). Auch im FACS werden bestimmte Terme eingeführt, welche dieser individuellen Ausprägung der Haut Rechnung tragen beispielsweise Erhebungen, Falten, Wölbungen und weitere (Ekman et al., 2002, S. 2–4).

Blendshapes werden definitionsgemäß als kontinuierlicher Übergang von Ausgangs- zu Zielzuständen aufgefasst, welche vorher festgelegt werden müssen. Damit können sie nur nach ihrer oben beschriebenen immanenten Fähigkeit zur Verformung eingesetzt werden.

Gegenüber dem FACS-System, welches die Darstellung durch Blendshapes erheblich inspiriert hat, sind Blendshapes nicht als ein festes System zu sehen das Wert auf Vergleichbarkeit legt. Ein Set von Blendshapes wird von einem Designer erstellt, anhand der Struktur des Meshes und individueller Annahmen. So unterscheiden sich Blendshape Sets beispielsweise erheblich in der Anzahl der verwendeten Blendshapes (Lewis et al., 2014, S. 209). Das hier verwendete Blendshape System besteht aus 52 Stück, wobei das Zungensystem nicht verwendet wird, da das Modell keine Zunge enthält.

2.5 Three.js

Der OpenGL (Open Graphics Library) Standard, welcher inzwischen in der Version 4.6 vorliegt und durch die Khronos Group verwaltet wird, ist ein plattform- und sprachübergreifende API (Application Programming Interface) zum Rendern von 2D- und 3D-Vektorgrafiken (The Khronos Group Inc., 2022). Er ermöglicht die Interaktion mit einer GPU (Graphics processing unit) zur hardwarebeschleu-

nigten Grafikdarstellung. WebGL (Web Graphics Library) ist eine JavaScript Library, welche die Nutzung des OpenGL Standards in Webbrowsern ermöglicht und damit Bildverarbeitung- oder Physikberechnungen mit einer vergleichbaren Hardwarebeschleunigung wie durch OpenGL in Desktopanwendungen im Webbrowser ermöglicht (WebGL Public Wiki, 2016). Three.js bietet eine Abstraktion, die es einfacher macht, mit WebGL zu arbeiten, da viele der komplexeren Details hinter den Kulissen von Three.js gehandhabt werden. WebGL ist ein low-level, also sehr hardwarenahes, System das auf die Grundkomponenten der Grafikdarstellung, Punkte, Linien und Dreiecke, spezialisiert ist. Three.js nimmt dem Entwickler die kleinteilige Erstellung von Animationen aus diesen Grundbausteinen ab. Beispielsweise können Geometrien und Oberflächen in ihrer Gesamtheit angesprochen werden. Damit ist es möglich auf einer Ebene in der die Grundelemente zu beispielsweise Texturen, 3D Formen oder Lichtquellen als Objekte mit Methoden abstrahiert sind zu arbeiten. In Three.js kann zwar auch mittels der OpenGL Shading Language, GLSL, welche von der Khronos Group zur Shader Programmierung angeboten wird, Einfluss auf die Rendering Pipeline genommen werden, allerdings ist in dieser Arbeit die Klasse THREE.WebGLRenderer vollkommen ausreichend. Folgend werden die Grundlegenden Komponenten von Three.js dargestellt welche zur Entwicklung einer Applikation essenziell sind.

Scene

Die Grundlage der gesamten Darstellung ist die Klasse THREE.Scene. Eine Scene Instanz ist ein Container, welcher die zu zeichnenden Objekte , Lichtquellen und Kameras aufnimmt und bestimmt damit was wo gerendert wird (three.js, n.d.). Die Scene besitzt eine Baumstruktur in welcher die Objekte und ihre Beziehung zueinander als Knoten in diesem Baum vorliegen. Dabei ist die Scene die Wurzel des Baumes und definiert grundlegende Eigenschaften, wie beispielsweise den Hintergrund. Die hierarchisch nachfolgenden Knoten sind in einer Eltern-Kind-Beziehung organisiert. Damit werden die Orientierung und Position der Knoten relativ zu ihren Eltern abgebildet. Diese Datenstruktur gibt dem Renderer vor, welche Objekte in welchen Beziehungen zueinander auf die Leinwand gezeichnet werden sollen.

Renderer

Die Klasse THREE.WebGLRenderer, kurz Renderer genannt, ist eine Kernkomponente des Three.js Frameworks (three.js, n.d.-a)Der Renderer nimmt alle Objekte, die im Scenegraph gespeichert sind und errechnet aus den räumlichen Positionen, der Lichtquelle und den 3D Objekten die 2D Darstellung aus der Kameraperspektive, welche auf der Canvas angezeigt wird. Diese Canvas wird als Element in die DOM-Struktur der HTML-Seite integriert (three.js, n.d.-c).

Animation System

Grundlegend besteht eine Animation in three.js aus einem Clip, welcher sich aus Keyframe Tracks zusammensetzt (three.js, n.d.-a). Es existieren sechs Arten von Keyframe Tracks, welche jeweils einen Identifier, ein Array mit den Zeitwerten und ein Array mit den Werten der zu ändernden Objekteigenschaften sowie optional die Interpolationsmethode enthalten. Hierbei besteht der Identifier aus dem Namen des Objekts und der zu ändernden Eigenschaft. Durch die Transitionszeiten, in Kombination mit der gewählten Interpolationsmethode, wird festgelegt wie die Zwischenbilder zwischen zwei Keyframes berechnet werden. Aus der Dauer dieser Berechnung und der Darstellungsfähigkeit des Browsers ergibt sich dann auch die Framerate.

Die Keyframe Tracks werden in einem AnimationClip Objekt zusammengefasst. Üblicherweise umfasst ein Clip ein semantisch abgegrenztes Verhalten. Beispielsweise bestimmt ein Clip das Drehen des Armes eines Models und ein anderer das Bewegen der Finger. Die Animationsclips können mit einem Model importiert werden, wenn sie in anderen Programmen, beispielsweise Blender erstellt wurden und in dem Objekt gespeichert sind. So ist in dem hier verwendeten Mesh bereits eine Animation gespeichert, die beim Laden des Beispiels aus der Three.js Bibliothek automatisch abgespielt wird. Es können aber auch wie in dieser Arbeit, Clips aus eigens definierten Keyframe Tracks dynamisch zur Laufzeit erstellt werden.

Die AnimationClips werden an einen AnimationMixer übergeben, er übernimmt die Wiedergabesteuerung der Animationen, das Mischen mehrerer Clips oder das Überblenden ineinander. Dabei wird selten direkt mit dem AnimationMixer interagiert, sondern eine Animation Action verwendet, welcher auf einem Mixer definiert wird. Diese AnimationAction bietet dem Programmierer zahlreiche Methoden zur Steuerung unterschiedlicher Animationen und deren Eigenschaften wie Fading-Effekte, Starten und Stoppen und das Verhalten am Ende der Animation.

Dateiformat des Face Mesh

Das Graphics Language Transmission Format, kurz glTF, ist ein, von der Khronos Group entwickeltes, kompaktes und effizientes Format für die Übertragung und das Rendering von 3D-Assets, es ist Hersteller- und Plattformenunabhängig und damit durch unterschiedliche Anwendungen und Umgebungen nutzbar(The Khronos Group Inc., 2021). Es ist möglich eine vollständige Szenendarstellung mit Knoten, ihren Beziehungen, Materialien, Kameras sowie Meshes und Animationen zu speichern. Dazu wird eine JSON-Repräsentation zur Beschreibung der Szenenstruktur angelegt und die eigentlichen 3D-Daten, wie Geometrie und Texturen in separaten Binärdateien gespeichert. Das Mesh dieser Anwendung liegt im glb-Format vor, welches die binäre Repräsentation des glTF darstellt und entwickelt wurde, um Ladezeiten weiter zu optimieren. Das ebenfalls von der Khronos Group entwickelte KTX-Format ist ein Containerformat für Texturen, das effiziente Speicherung und Nutzung über verschiedene Plattformen und APIs hinweg ermöglicht (The Khronos Group Inc., 2024). KTX-Dateien müssen decodiert werden, um die enthaltenen Texturen in Grafikanwendungen nutzen zu

können. Ein KTX Loader, wie in three.js enthalten, liest die Texturen aus der KTX-Datei und initialisiert damit die Texturenobjekte im Grafikspeicher, damit stehen sie WebGL zur Verfügung. Das hier verwendete Mesh ist zudem mit „meshoptim“ optimiert worden, welches genutzt wird um die Rendering-Effizienz zu verbessern (*Meshoptimizer*, n.d.).

In Abbildung 19 im Anhang wird dieses Zusammenspiel der Komponenten übersichtlich dargestellt.

2.6 Webtechnologien

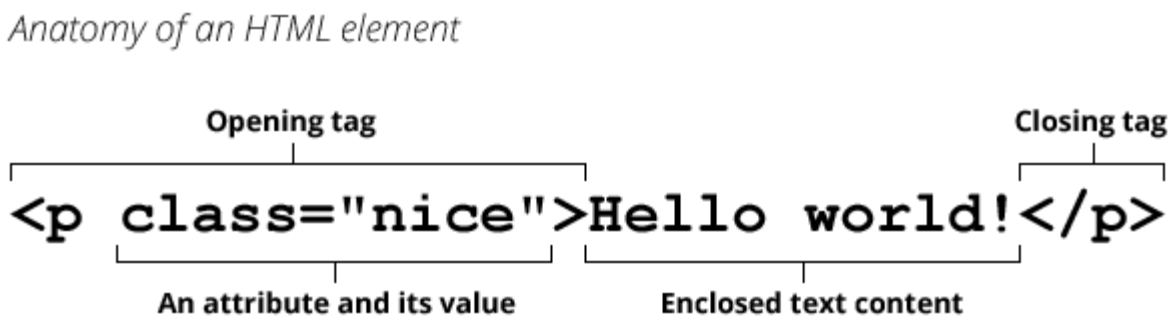
HTML und CSS

Zur Darstellung von Informationen und Medien im Browser hat sich HTML (Hypertext Markup Language) in der aktuellen Version HTML5 zum Quasi Standard des Internets herausgebildet, welcher durch die WHATWG (Web Hypertext Application Technology Working Group) Community verwaltet wird und liegt aktuell in der Version 5 vor, was aber eher als Beschreibung der aktuellsten Version verstanden werden sollte und nicht als eindeutige Versionierung (WHATWG, 2024). Daher wird in dieser Arbeit HTML5 und HTML gleichbedeutend verwendet. Vereinfacht gesagt, wird HTML verwendet um die semantische Struktur und die Inhalte einer Webseite zu beschreiben und CSS (Cascading Style Sheets) um die Gestaltung und Formatierung der Webseite fest zu legen (Bühler et al., 2023, S. 5).

Da die Webtechnologie, die hinter der Darstellung und Funktion einer Webseite steht für diese Arbeit essenziell ist, soll anhand des HTML-Standards, welche durch die oben bereits genannte WHATWIG Community verwaltet werden, ein kurzer Überblick gegeben werden.

Der Kern des HTML-Dokuments sind die Elemente, aus denen sich der gesamte Kontext der Seite aufbaut, das <HTML> Tag steht dabei an oberster Stelle und beinhaltet das <Head> und <Body> Tag, die die Grundstruktur darstellen. Diese Elemente bestehen aus einem öffnenden Tag und einem schließenden Tag, die weiter unten erläutert werden. Das Element enthält Attribute und ihre Werte sowie bei manchen Elementen einen Text-Inhalt. Es können weitere Elemente mit weiteren Kind-Elementen enthalten sein. Um diese ganzen Bestandteile miteinander zu einem Gesamtbild zu verbinden, das optisch ansprechend und übersichtlich ist, kann mittels CSS die Größe, Anordnung, Farbe und das Verhalten bei Änderung der Größe festgelegt werden. Üblicherweise wird das CSS-Design in einem Stylesheet festgelegt, welches in einer eigenen Datei liegt und durch das HTML hinzugeladen wird. Dieses wird dabei von oben nach unten abgelaufen, sodass Styles auch überschrieben werden können oder auf mehreren Klassen angewendet werden. Die HTML-Elemente können über ihre individuellen Bezeichner wie Klasse und ID aber auch über ihr Tag angesprochen werden, je nachdem ob individuelle Elemente oder Gruppen gesucht werden. Der grundlegende Aufbau ist in der folgenden Darstellung von Mozilla gut ersichtlich.

Abb. 2 Aufbau eines HTML-Elements



Quelle: MDN Web Docs nach CC-BY-SAv2.5 Lizenz, https://developer.mozilla.org/en-US/docs/Glossary/HTML

DOM

Das DOM (Document Object Model) ist eine Programmierschnittstelle welche die dynamische Veränderung der Struktur, Gestaltung und der Inhalte einer Webseite ermöglicht (MDN, 2023a). Das DOM ist dabei als Baumstruktur aufgebaut, welche aus Kanten und Knoten besteht. Über DOM-Methoden ist es möglich durch Programmiersprachen, meist JavaScript, auf diese Struktur zuzugreifen. Es ist nun möglich an der obigen Abbildung den Text „Hello World!“ durch JavaScript zu verändern, indem das DOM nach dem Klassennamen „nice“ durchsucht wird und der Text Inhalt der gefundenen Knoten geändert wird. Es kann aber auch dem Tag ein neues Attribut „id “ hinzugefügt werden, was eine eindeutige Identifizierung des Elements gegenüber den allgemeineren Klassennamen ermöglicht. So können während der Laufzeit die Struktur und die Elemente der HTML-Seite verändert, Elemente hinzugefügt oder entfernt werden.

Eventlistener

Eine wichtige Eigenschaft von HTML-Elementen besteht in der Möglichkeit Eventlistener an den Elementen anzubringen, welche auf bestimmte Ereignisse reagieren und deren Verarbeitung definieren. Damit ist es möglich eine effiziente Interaktion mit dem Nutzer zu modellieren. Im Kontext des DOM stellt sich die Verwendung, in vereinfachter Form, wie folgt dar. Ein Event Objekt wird durch ein Ereignis erzeugt, beispielsweise eine Netzwerkaktivität oder eine Nutzerinteraktion. Der Nutzer klickt auf einen Button und löst damit ein Event aus welches den gedrückten Knopf als Event Target gespeichert hat. Jetzt beginnt die erste Phase des Event-Lebenszyklus, die „Capturing_Phase“. Diese Phase beginnt nicht an dem ereignisauslösenden Knoten des DOM, sondern an dessen Wurzel und bewegt sich absteigend in Richtung des Zielknotens. Diese Phase nennt sich die „Target_Phase“ und aktiviert die an dem Ziel registrierten Eventlistener, wenn sie an dem Ziel Knoten in den Status „at_Target“ übergeht. Danach erfolgt die „Bubbling_Phase“, in welcher sich das Event zurück an die Wurzel bewegt. Während dieser Phase können weitere Eventlistener ausgelöst werden. Um einen Nutzen aus der DOM-Struktur und der Event Behandlung zu ziehen ist eine Programmlogik notwendig, was HTML nicht bieten kann.

JavaScript

ECMAScript 2025, welches die aktuellste Version darstellt, ist eine Standardisierung der Programmiersprache JavaScript, die von Netscape 1995 entwickelt wurde (Haverbeke, 2024, S. 6)So wichtig die Standardisierung und Vereinheitlichung der Sprache zur Anwendung in allen Webbrowsern unabhängig vom Hersteller auch ist, sei dieser Zusammenhang nur der Vollständigkeit halber erwähnt und spielt in dieser Arbeit keine Rolle. Es wird nur der Begriff JavaScript verwendet. Die Programmiersprache JavaScript wird größtenteils in der Programmierung browserbasierter Anwendungen und Webseiten verwendet auch wenn es möglich ist sie auch außerhalb des Browsers zu verwenden (Haverbeke, 2024, S. 7). Sie ist auch die unbestrittene Sprache der Wahl in diesem Bereich, weshalb sie auch in dieser Arbeit verwendet wird. Erst durch JS wird es möglich umfassend mit dem DOM zu interagieren, es können fast alle Elemente angepasst, erzeugt und gelöscht werden (Haverbeke, 2024, S. 219). Durch das Tag <script> ist es möglich auf eine JS-Datei zu verweisen, welche hinzugeladen werden soll anstelle den Code direkt in die Seite zu integrieren. Aus Sicherheitsaspekten wird JS in einer Sandbox Umgebung durch den Browser interpretiert (Haverbeke, 2024, S. 212). Einige Folgen dieses Umstands werden im Implementationsteil noch thematisiert.

Als Datenstruktur hat sich JSON (JavaScript Object Notation) etabliert, da es leicht für Maschinen zu verarbeiten und für Menschen zu lesen und zu schreiben ist. Aus diesem Grund wird es auch in dieser Anwendung zur Speicherung der Konfiguration verwendet.

Webpack

Webpack ist ein Open Source Tool das von seiner Community gepflegt und von open collective finanziert wird (*Webpack Concepts*, n.d.). Es ist ein nützliches Werkzeug in der modernen Webentwicklung, das als statischer Modulbundler fungiert. Abhängigkeiten in einem Projekt werden analysiert und die erforderlichen Modulbestandteile und Dateien in kompakte „Bündel“ zusammengefasst, die im Browser ausgeführt werden können. Daher besteht keine Notwendigkeit mehr, viele separate Dateien zu laden oder Abhängigkeiten zu warten, welche für die Anwendung nicht relevant sind. Hinsichtlich der Browserkompatibilität unterstützt Webpack alle Browser, die ES5-konform sind. Für darunterliegende Versionen kann eine Kompatibilität hergestellt werden. Da Webpack zahlreiche Datentypen unterstützt, kann es in dieser Arbeit verwendet werden, um alle Abhängigkeiten zu vereinen. Dies kommt der Performance und Einfachheit der Anwendung, welche in dieser Arbeit angestrebt wird, entgegen.

3. Anforderungsanalyse

3.1 Anwendungsbeispiel

Zur Veranschaulichung und als Ausgangspunkt für die Ermittlung der Anforderungen an das System ist es hilfreich ein konkretes Szenario zu erstellen, an welchem die Nutzung und der Umfang des zu erstellenden Systems verdeutlicht wird (Rupp, 2021, S. 162). Da die Anwendung zur Unterstützung von psychologischen Untersuchungen dienen soll, werden die Anforderungen exemplarisch aus der

Studie von de la Rosa et al. (De La Rosa et al., 2013) hergeleitet und ein fiktiver Studienaufbau erstellt.

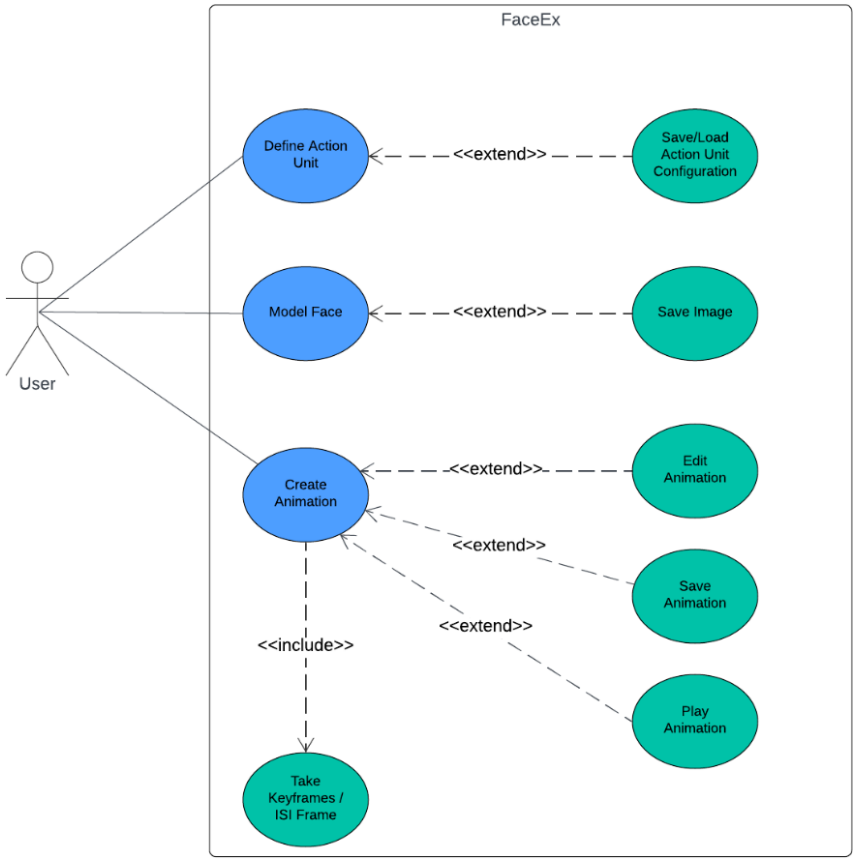
Im Rahmen einer Abschlussarbeit soll die Adaption von Personen an die Darstellung von menschlichen Emotionen untersucht werden. Hierzu werden zum einen Videos von Gesichtsausdrücken mit festgelegten prozentualen Ausprägungsstärken und ISI zwischen den Reizen benötigt. Zum anderen sollen Standbilder von Ausdrücken mit FACS kodierten Ausprägungen erstellt werden.

Da der Studienleiter Erfahrung in der Anwendung des FACS hat und sehr genaue Vorstellungen bestimmter Scoring Werte, sollen diese neu konfiguriert werden. Dazu wird mit Hilfe der Blendshape Steuerung ermittelt, welche Blendshapes in welcher Kombination am besten auf die FACS AU passen. Es gibt Blendshapes welche genau den Action Units entsprechen. Diese können belassen werden und nur die Scoring Werte müssen manuell angepasst werden (Oliver & Amengual Alcover, 2020, S. 5). AU, die nicht direkt mit Blendshapes korrelieren oder erst aus der Kombination entstehen, werden manuell definiert. Hierzu ist die Blendshape Steuerung zu benutzen, die die einzelnen Blendshapes direkt beeinflusst und so die Auswirkungen darstellt. Wenn die Konfiguration erkannt wurde, müssen noch die einzelnen Anteile in einem Scoring Wert festgelegt werden. Nach Abschluss dieser Tätigkeiten sind die AU bereit für die Verwendung, der Studienleiter stellt die gewünschten Konfigurationen ein, begutachtet sie im Vorschaufenster und speichert die Bilder ab.

Für die Videos wird das Vorgehen leicht abgewandelt. Da die prozentuale Ausprägung nicht in dem 5-stufigen Scoring System des FACS dargestellt werden soll, muss hier entschieden werden, welche AU an dem Zielausdruck beteiligt sind und diese zu einer kombinierten AU, Expression genannt, zusammengefasst werden. Stehen alle gewünschten Übergangsbereiche fest kann damit begonnen werden die Animation zu erstellen. Dafür werden die Keyframes erstellt und im Vorschaufenster betrachtet und die Transitionszeit bestimmt. Durch die Vorschaufunktion kann hier der optimale Übergang ermittelt werden. Das ist notwendig, weil sich durch die Interpolation nicht genau sagen lässt, wie das Ergebnis aussehen wird. Beispielsweise ist bei Zwischenzuständen nicht gewährleistet, dass das Ergebnis genauso aussieht, als wenn ein direkter Übergang in 3 Sekunden von 10 Prozent auf 80 Prozent erfolgt oder in 1 Sekunde von 10 auf 30 und dann in 2 Sekunden von 30 auf 80 Prozent. An den benötigten Stellen wird ein ISI für die notwendige Zeit eingefügt. Nach Abschluss oder während des Prozesses kann das Resultat im Vorschaufenster geprüft werden. Ist die Sequenz zufriedenstellend, wird die Animation aufgezeichnet und gespeichert.

In Form eines Use Case Diagramms sind die identifizierten Kernfunktionalitäten und ihre Beziehungen nachfolgend dargestellt.

Abb. 3 Use Case Diagramm



Quelle: Eigene Darstellung

3.2 Anforderungsanalyse

Zur Ermittlung der Anforderungen wurden die Erwartungen an die Anwendung zunächst als Stichpunkte notiert und anschließend durch die Anforderungsschablone von Rupp zu aussagekräftigen Sätzen umformuliert (Rupp, 2021, S. 361). Hierbei wird zwischen drei Typen unterschieden. Der Typ selbständige Systemaktivität charakterisiert sich durch das eigenständige Starten und Durchführen eines Prozesses. Beim Typ Benutzerinteraktion stellt der Betrachtungsgegenstand einem externen Akteur eine Interaktionsmöglichkeit zur Verfügung. Die Schnittstellenanforderungen charakterisiert Prozesse in Abhängigkeit eines Dritten wobei es sich dabei nicht um eine Person handelt und wird daher als Schnittstellenanforderung bezeichnet (Rupp, 2021, S. 363).

Da zur Erstellung der Anwendung ein iteratives Vorgehensmodell verwendet wird, ist die Liste an Anforderungen nach jedem Zyklus angepasst und verbessert worden. Dargestellt ist nun die resultierende Liste des letzten Zyklus. Unverzichtbare Kernfunktionen werden durch „muss“ gekennzeichnet. Die mit „sollte“ beschriebenen Anforderungen sind Features, die nicht zwingend notwendig, aber wünschenswert sind. Durch den Begriff „wird“, wird kenntlich gemacht, welche Aspekte im Rahmen dieser Arbeit nicht umgesetzt werden können aber in einer zukünftigen Version der Software enthalten sein sollten. Zusätzlich erfolgt eine Einteilung zwischen funktionalen Anforderungen und nicht

funktionalen Anforderungen. Hierbei wird der Unterschied im Sinne von Rupp als alles was nicht direkt in die technische Systemfunktion passt festgelegt (Rupp, 2021, S. 244).Somit sollen alle Anforderungen, deren Erfüllung nicht einfach mit ja oder nein beantwortet werden können oder sich die Antwort nicht auf eine spezielle implementierte Funktionalität bezieht als nicht funktionale Anforderungen bezeichnet werden.

Das Ergebnis der Anforderungsanalyse sowie der Erfüllungsgrad nach der letzten Iteration ist in der angehängten Tabelle 3 dargestellt.

4. Architektur

Die Anwendung FaceEx wird in einer SPA (Single-Page-Applikationen) Architektur umgesetzt, welche eine Anwendung bezeichnet die aus einer einzelnen HTML Seite besteht und ihre Inhalte mittels JavaScript dynamisch anpasst (Ackermann, 2023, S. 402). Ackermann empfiehlt dafür die etablierten Bibliotheken wie React zu verwenden, allerdings wird in dieser Arbeit auf Grund des beschränkten Umfangs und der gezielten Reduktion von Abhängigkeiten darauf verzichtet. Dieses Vorgehen bietet auch den Vorteil das der erstellte Programmcode vollständig anpassbar ist ohne Kenntnisse in einem dieser Frameworks zu besitzen. Auch ist es nicht ohne weiteres möglich Three.js in diese Bibliotheken zu integrieren ohne eine weitere Abhängigkeit, wie beispielsweise „react-three-fiber“ zur Nutzung von three.js in React, hinzuzufügen.

Da der Großteil der Anwendung aus Nutzerinteraktion mit einer SPA besteht, wird der Ansatz des Event-Driven-Programming als Paradigma gewählt. Das Event-Driven-Programming zeichnet sich durch den Fokus auf Benutzereingaben oder andere Triggerereignisse, welche den Programmfluss steuern, aus. Dieses Paradigma unterstützt die interaktive Nutzung dieser Anwendung und bietet eine gute Wartbarkeit der Software, da die Komponenten in Eventhandlern und den zugehörigen Handlerfunktionen gekapselt sind und so leicht angepasst werden können. Wie oben bereits erwähnt unterstützt HTML mit JS das Eventhandling in umfassender Weise. Die Anwendung stellt ein Fat Client Modell dar, bei welchem der Server keine Rechenleistung zur Verfügung stellt, sondern die Berechnungen durch den Client erfolgen. Daher ist auch nach Erhalt der Anwendungsdaten keine Verbindung mit dem Server mehr nötig. Das bietet auch den Vorteil, dass die teils aufwendigen grafischen Berechnungen der Animation bei mehreren Nutzern nicht durch eine Infrastruktur auf- und abskaliert werden müssen. Um dieses Modell umzusetzen, ist ein Framework notwendig welches die teils erheblichen Berechnungen, die eine Animation eines Gesichtes darstellt, effizient auf der Hardware des Nutzers umsetzt. Es wurde das Framework three.js verwendet da hier nicht nur die effiziente Hardwarenutzung und hohe Abstraktion der Geometrien, sondern auch ein Mesh bereits enthalten ist.

Da three.js ein auf JavaScript basierendes Framework ist und diese Sprache eine hervorragende Unterstützung der Webfunktionalität bietet, wird der Programmcode in JavaScript erstellt werden.

Die „lil gui“ die zum Anpassen der Blendshapes, wie in den Beispielprojekten von three.js, verwendet wird, soll hier ebenfalls genutzt werden und dient dem Ausprobieren der Blendshapes. Allerdings wird sie nicht direkt in der Seite verfügbar sein, sondern über eine Schaltfläche verfügbar gemacht, da die Einstellung der Gesichtsausdrücke über die Blendshape Steuerung nicht die gewünschte Nutzerführung darstellt.

4.1 Datenstrukturen

In der Anwendung werden drei Datenstrukturen verwendet, um den Programmfluss und die Konfiguration abzubilden. Zum einen wird der Zustand der Animation in einem Objekt „Animation“ mit einem Keyframe Array gespeichert werden. Das Array soll die notwendigen Daten zur Erstellung der Animation sowie das Thumbnail für den Keyframe Container enthalten. Zwei JS-Objekte im Controller Skript spiegeln die Konfiguration der AU sowie der Expressions wider und werden im JSON-Format als Konfigurationsdatei gespeichert. Ein Vorteil an der Nutzung von JSON ist, dass diese Dateien direkt in die Anwendung geladen und zu Objekten geparst werden können. Ihre einfache Struktur kann auch vom Menschen schnell gelesen und in der Datei angepasst oder durch andere Programme erzeugt werden können. Zum anderen kann mit einem JSON-Schema auf einfache Weise die Struktur der enthaltenen Objekte festgelegt und mittels eines Validators überprüft werden. Da es sich hier um eine kleine Anwendung handelt wird auf den Validator jema.js zurückgegriffen, der eine sehr leichtgewichtige Lösung bietet. Da die Scoring Werte über die JSON-Datei geändert werden, ist es zweckmäßig diese auf Korrektheit zu prüfen, bevor ihr Inhalt in die Website integriert wird. Somit kann sich der Nutzer sicher sein, dass seine Änderungen zu einer weiterhin gültigen JSON-Datei geführt haben. Mit jema.js ist es nicht nur möglich die Validierung durchzuführen, sondern auch eine genaue Erklärung auf der Konsole auszugeben, warum eine Datei nicht gültig ist.

Die AU, die über Schieberegler in den FACS Scoring Werten eingestellt werden können, werden in der Grundkonfiguration in einem festgelegten Bereich von 0 bis 1 in 0,2 Schritten gespeichert. Es wird aber möglich sein, dieses Array in der Konfigurationsdatei im Bereich von -2 bis 2 festzulegen. Dieser Bereich wird auch für die direkte Blendshape Steuerung mittels der „lil gui“ gewählt. Durch Versuche im Rahmen der Entwicklung hat sich gezeigt, dass einige Kombinationen zu guten Ergebnissen führen, wenn der Bereich zwischen 0 und 1 verlassen wird. Daher soll dieses Verhalten ermöglicht werden.

Da es möglich sein soll Gesichtsausdrücke aus AU in einem kontinuierlichen Übergang darzustellen, werden neben den AU auch Expression genannte Objekte definiert, die aus zuvor festgelegten AU bestehen. Die AU, aus denen die Expression besteht, werden jeweils gleichzeitig stufenlos im Bereich von 0 bis 1 aktiviert und stellen somit eine prozentuale Ausprägung eines Gesichtsausdrucks zur Verfügung. Diese werden ebenfalls über Schieberegler gesteuert.

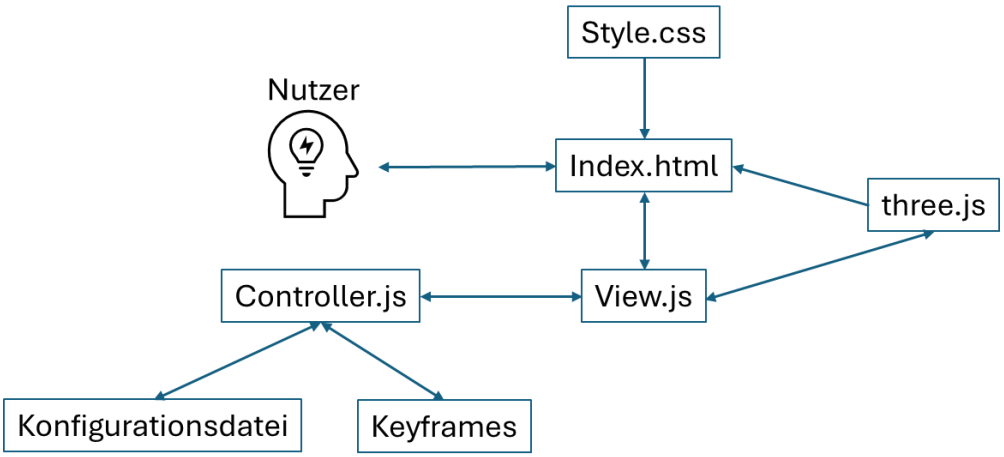
Es wird ein Keyframe Container angelegt, da es notwendig ist beim Erstellen einer Animation ein optisches Feedback zu erhalten, ohne die Animation abspielen zu müssen. Daher soll auch ein

Thumbnail angezeigt werden. Ebenso ist es notwendig, einzelne Keyframes nach der Erstellung abzuändern oder zu entfernen.

4.2 Programmstruktur

Es soll in grober Anlehnung an das MVC (Model View Controller) Modell eine Struktur aus zwei JavaScript Dateien genutzt werden. In einem Controller Skript sollen die Operationen auf und mit den Datenstrukturen sowie deren Prüfung und Verwaltung geregelt werden. In einem zweiten Skript werden die Interaktion und die grafische Darstellung der Anwendung gesteuert. Diese Einteilung hat weniger funktionale Aspekte, sondern dient einer sinnvollen Strukturierung, um die Lesbarkeit und Wartbarkeit des Codes zu gewährleisten.

Abb. 4 Systemstruktur



Quelle: Eigene Darstellung

Der direkte Zugriff auf die Blendshapes wird mittels der „lil gui“ möglich sein, allerdings hinter einer Schaltfläche verborgen, da die vorgesehene Nutzerführung die Blendshape Steuerung nur für das Erstellen der AU vorsieht.

Zur Darstellung der Kernfunktionalitäten, der Animationserstellung und eines Keyframe Containers, der den Stand der Animation widerspiegelt, wurden die Abbildungen 17 und 18 erstellt. Diese befinden sich im Anhang unter A und B.

4.3 Release

Die Anwendung wird auf Github als Repository mit allen Programmdateien von jema.js und three.js gehostet werden. Zusätzlich wird ein Release mittels Webpack erstellt, welcher nur die zur Ausführung notwendigen Programmdateien enthält. Daher genügt sie auch der Definition der Single Page Application, welche eine sehr einfache und installationsfreie Nutzung erlaubt, da sämtliche Operationen im Browsercache stattfinden werden und nur durch Nutzerinteraktion auf die Festplatte des Rechners zugegriffen werden muss (Ackermann, 2023, S. 402). Die Zugriffe auf die Grafikhardware, werden über Three.js mit WebGL gesteuert und müssen daher nicht gesondert betrachtet werden.

Da die Anwendung im Umfang dieser Arbeit nicht voll ausprogrammiert werden kann, sondern den Charakter eines Prototyps bzw. eines Entwicklungszustandes innerhalb eines iterativen Entwicklungsprozesses darstellt, müssen einige Architekturentscheidungen unberücksichtigt bleiben. Diese werden nachfolgend kurz erläutert.

Es wurde in dieser Anwendung kein Sicherheitskonzept angewendet. Im Rahmen dieser Arbeit, die keine vollständige Anwendung zum Ziel hat, erscheint es vertretbar, auf die Absicherung der Website zu verzichten. Es ist notwendig, in einer zukünftigen Version, die auf einem öffentlichen Server gehostet wird, bestimmte Sicherheitsvorkehrungen zu treffen. Da der Sever hier keine Verarbeitung von Daten vornimmt, sondern nur den Code zur Verfügung stellt, ist das Risiko auf Serverseite als gering anzusehen. Allerdings kann durch eine mangelnde Serverkonfiguration eine Änderung der Daten vorgenommen werden, welche dann zu unerwünschtem Verhalten auf dem Rechner des Nutzers führen kann. Eine Lösung wäre mittels Checksums die übermittelten Daten auf Integrität zu prüfen. Um zu verhindern das beispielsweise durch die Deserialisierung der JSON Dateien Schadcode zur Ausführung kommt. Weiterhin kann auch durch Formularfelder JS Code in die Anwendung gelangen oder Sicherheitslücken in den verwendeten Bibliotheken ausgenutzt werden. Daher wird in einer zukünftigen Version ein Grundsicherungskonzept anhand der OWASP (Open Worldwide Application Security Project) Top Ten Sicherheitsrisiken umgesetzt, das einen Großteil der Bedrohungen berücksichtigt(OWASP Foundation, 2021). Hierdurch soll Anforderung A3.4 erfüllt werden.

Ebenso notwendig wie die Sicherheit ist ein gutes Testkonzept und die automatisierte Durchführung der Tests in zukünftigen Versionen. Während im aktuellen Zustand noch gut mit manuellen Tests gearbeitet werden kann, sollte vor der Erweiterung eine feste Teststruktur implementiert werden. Damit können Änderungen am Quellcode auf einfache Weise geprüft werden und somit sichergestellt werden, dass die Anwendung nach Änderungen weiterhin funktioniert.

5. Implementierung

5.1. Vorgehensmodell

Zur Implementierung der Anwendung wurde ein agiles Vorgehensmodell gewählt, bei dem die identifizierten Anforderungen iterativ zu einer funktionierenden Software zusammengefügt wurden. So wurde sukzessive ein weiterer Teil der Anforderungen in jedem Sprint erstellt, auftretende Probleme erneut in den nächsten Sprint zur Lösung übernommen sowie die Anforderungen angepasst, erweitert oder gestrichen. Der große Vorteil an dieser Herangehensweise ist, dass am Ende jedes Durchgangs ein funktionierendes Stück Software entsteht, welches auf Funktionalität und Erfüllung der Anforderungen getestet werden kann. Zur klaren sprachlichen Trennung der Begriffe werden HTML-Elemente in ihrer öffnenden Tag Form dargestellt und benannt <tag> und Funktionen mit dem Funktionsnamen und angehängten () Klammern sowie eventuellen Parametern beschrieben. Zudem wird die übliche Bezeichnung von Funktionen auf Instanzen einer Klasse als Methoden und Funktionen ohne zugehöriges Objekt als Funktion bezeichnet. Zur besseren Orientierung wird das Skript oder

das Framework, bei der Nennung von Funktionen und Objekten, durch einen Punkt getrennt vorangestellt beispielsweise *Skript.Objekt* oder *Skript.funktion().* Das ist jedoch nicht im Sinne einer Klassenzugehörigkeit zu verstehen. Außerdem wird durch die kursive Hervorhebung eine Zugehörigkeit zum Code der Anwendung ausgedrückt. Da es nicht möglich ist auf alle Aspekte der Programmierung einzugehen, wird der Fokus auf die Kernfunktionalitäten gelegt.

5.2 Werkzeuge

Zum Erstellen der Programmdateien wurde Visual Studio Code, kurz VSCode ein Texteditor, der für unterschiedlichste Programmiersprachen und Funktionen genutzt werden kann, verwendet. Durch eine Vielzahl von verfügbaren Plugins kann der Funktionsumfang erweitert werden und ermöglicht das effiziente Erstellen der benötigten HTML, CSS, JS und JSON-Dateien. Durch das integrierte IntelliSense ist ein schnelles Schreiben des Codes möglich, da durch Code Vorschläge und automatische Fehlerkorrektur viele aufwändige Arbeiten erleichtert werden. Der integrierte Live-Server hostet die HTML-Datei auf einem lokalen Server und aktualisiert die Seite automatisch mit jeder gespeicherten Änderung. Somit kann jede Änderung direkt begutachtet werden.

Zum Hosten der erstellten Programmdateien wird auf die Versionsverwaltungssoftware Git und den Hosting Dienst GitHub zurückgegriffen. Hierbei wird das lokale Repository auf den GitHub Server geladen und ist öffentlich zum Download verfügbar.

Als Browser zum Testen der Anwendung wurden Firefox in der Version 126.0.1 und Chrome in der Version 125.0.6422.141 verwendet. Beide Browser wurden auf einem 64-Bit System mit Windows 11 einem i7 Prozessor der elften Generation, 16GB Arbeitsspeicher, ohne dedizierte GPU ausgeführt. Über den in beiden Browsern verfügbaren Entwicklermodus können umfassende Debugging Tools genutzt werden, was für den Entwicklungsvorgang notwendig ist.

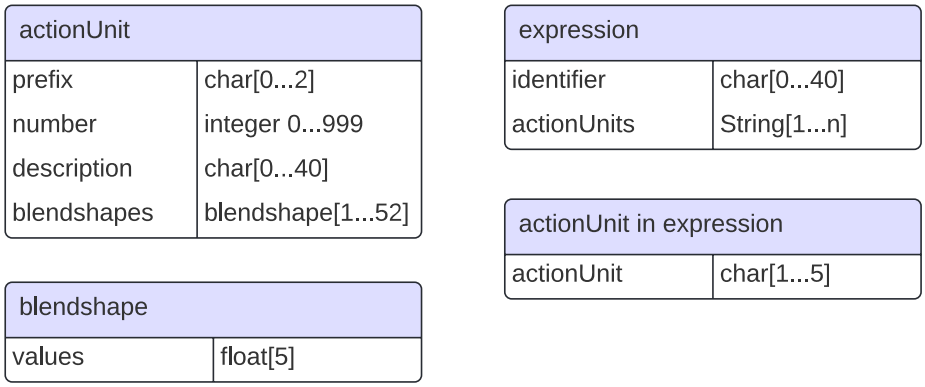
5.3 Konfigurationsdatei

Struktur

Eine AU, in der Konfiguration *Controller.actionUnits* genannt, wurde allgemein als Objektbezeichnung gewählt und ist nicht mit der oben ausgeführten Definition einer AU des FACS gleichgesetzt. Somit können auch Objekte vom Typ AD oder andere erstellt und auf der technischen Ebene als Action Unit behandelt werden. Hierfür ist die Eigenschaft „Prefix“ optional gehalten. Eine Nummer ist notwendig und maximal dreistellig. Die Beschreibung ist optional und die enthaltenen Blendshapes müssen mindestens ein Element und maximal die Anzahl der in dem verwendeten Mesh möglichen Blendshapes enthalten, wobei durch das verwendete FACS Scoring System ein Array mit genau fünf Float Werten verpflichtend ist. Durch diese individuelle Zuordnung zu jedem Blendshape ist es möglich, auch laterale oder asymmetrische Bewegungen abzubilden. Zu der später zu implementierenden Darstellung asymmetrischer Bewegungen, soll es möglich sein den oben beschriebenen Apex der Bewegung festzulegen und mittels der Zeitsteuerung der Animation so eine asymmetrische AU zu generieren. Die Werte des Arrays sind zwischen minus zwei und zwei erlaubt da es auch

Konfigurationen gibt in bei denen dieser Bereich sinnvolle Ergebnisse liefert. Die Expressions als zweites Objekt der Konfiguration dienen der gröberen Anpassung von Gesichtsausdrücken, die sich an einer linearen Skala zwischen den Blendshape Werten null und eins orientieren. Daher sind diese freier gestaltet und verlangen lediglich einen actionUnits Eintrag, der aber zwischen eins im Falle einer AU ohne Präfix im einstelligen Bereich und einer AU mit Präfix und drei Stellen liegen soll. Zur anschaulichen Darstellung sind die Datentypen in Anlehnung an ein UML-Klassendiagramm mit dem Datentyp, den Arrayklammern und den drei Punkten als Bereich dargestellt. Eine Zahl in den Arrayklammern steht für die Arraygröße im Sinn von stark typisierten Sprachen wie C++. JavaScript, als schwach typisierte Sprache, unterstützt diese Aufschlüsselung nicht. Als Überblick ist diese Darstellung zweckmäßig und wird durch das nach dem Vorbild von JSON Schema (www.json-schema.org) verwendete Schema im Controller Skript in Zeile 394 -462 abgebildet.

Abb.5 Struktur der JSON-Konfigurationsdatei



Quelle: Eigene Darstellung

Anfangskonfiguration

Um eine sinnvolle Anfangskonfiguration zu erstellen, müssen zum einen die Zuweisung von Blendshapes an die entsprechenden AU vorgenommen werden und zum anderen die Zuweisung der AU zu den Expressions genannten kombinierten Ausdrücken, die im momentanen Zustand die sechs Grundemotionen nach Ekman sein sollen (Ekman & Friesen, 1975, S. 14). Da es keine direkte Zuordnung zwischen Emotionen und AU-Kombinationen im FACS gibt, sondern eher Rahmenbedingungen und mögliche Beteiligungen genannt werden, wurde durch den Autor anhand von Beispielen der Seite iMotions.com (Bryn Farnsworth, 2022) sowie der Arbeit von Bansal und Ghayoumi zur Analyse von Gesichtsausdrücken (Ghayoumi & Bansal, n.d.) die Konfiguration der AU festgelegt. In der Arbeit von Oliver und Alcover (Oliver & Amengual Alcover, 2020, S. 5) wurden AU identifiziert, welche mit ihrem Blendshape Set ohne Anpassung korrespondieren. Es wird ein anderes Set als das in ARKit enthaltene verwendete, aber als Orientierung wurden einige AU mit der Annahme übernommen, dass die AU-Beschreibungen mit den Beschreibungen in Blendshape Sets korrelieren könnten. Da der Nutzer die Konfiguration umfassend anpassen kann, wird mit der Grundkonfiguration nur ein Vorschlag gemacht, wie die Zuordnung aussehen kann. Hierzu wurden in den verfügba-

ren Quellen nach Zuordnungen von Blendshapes zu AU gesucht und aus getroffenen Annahmen die im Anhang befindlichen Tabellen 1 und 2 erstellt. Trotz großer Sorgfalt bei der Erstellung, ist die Zuordnung subjektiv und nicht durch einen FACS-Kodierungsexperten validiert.

5.4 Hintergrundfunktionen

Startup

Sobald die Seite initialisiert ist und alle Code Dateien im Speicher vorliegen bzw. dem Browser bekannt sind, beginnt die Verarbeitung des Skriptes. Hier werden zunächst die notwendigen Instanzen für three.js initialisiert. Da hier die Standardvorgehensweise der Beispiele in three.js durchgeführt wird, soll nur ein kurzer Überblick über die wichtigsten Funktionen gegeben werden. Dieser Abschnitt bezieht sich auf die Codezeilen 43 bis 63 im View.js Skript. Wie oben bereits erläutert, spiegelt sich die Interaktion mit dem Animationssystem auf dem DOM-Element des Renderers, der wiederum den Scenegraph verarbeitet ab. Dazu wird eine Instanz der Klasse *THREE.WebGLRenderer* erstellt und ein Tone Mapping, das zur Darstellung von HDR-Inhalten in begrenzteren Farbräumen dient, hinzugefügt. Auch wenn sich hierdurch keine Vorteile in der momentanen Texturdarstellung des Meshes ergeben, ist es doch wichtig für photorealistische Texturen, die auch geladen werden können. Anschließend wird das DOM-Element des Renderers in die Seite integriert, indem dem Canvas Container dieses Element zugewiesen wird. Durch die Festlegung des *THREE.PMREMGenerators* und der *RoomEnvironment.environment* wird die physikalische Berechnung der Szene definiert. Die *OrbitControls.controls* sind für die flüssige Bewegung der Kamera um das gerenderte Objekt, hier das Face Mesh verantwortlich. Dadurch wird es möglich das Gesicht in unterschiedlichen Winkeln aufzunehmen. Da sich nicht das Objekt, sondern die Kamera bewegt, wird dazu die Kamera an den Konstruktor übergeben. Die Kamera erstellt im Zusammenspiel mit dem Renderer aus dem Scenegraph das Bild. Durch die Integration der *THREE.OrbitControls* werden die Anforderungen A2.19 und A2.22 erfüllt.

Face Mesh

Als nächstes wird in einer asynchronen Funktion das eigentliche Mesh geladen. Die asynchrone Verarbeitung mit der Rückgabe eines *Promise* hat den Vorteil das während der Ausführung auf eventuelle Fehler reagiert werden kann sowie die Nutzung des *await* Keywords, um die Ausführung anzuhalten, bis das Mesh vollständig geladen ist. Obwohl keine Fehlerbehandlung implementiert ist, wird dieses Verhalten beibehalten, da es die Verständlichkeit des Codes erhöht die Standardvorgehensweise eines Frameworks zu nutzen. Das Laden des Mesh sowie die Decodierung und Optimierung der Texturen wurde aus dem Beispielteil des three.js Frameworks übernommen und wird daher nur kurz erläutert. Der in den Grundlagen beschriebene Kompressions- und Optimierungsvorgang wird nun im Wesentlichen umgekehrt. Der folgende Abschnitt bezieht sich auf die Zeilen 66 – 83 der View.js. Es wird zunächst durch den Konstruktor der *Promise* Klasse eine Funktion übergeben, welche bei Aufruf zur Ausführung kommt und der Rückgabewert als ein Platzhalter für einen zukünftigen

Wert angesehen werden kann (Haverbeke, 2018, S. 185). Ein KTX2 Loader dient der Dekompression der Texturen und nutzt dafür den three.js Basis Transcoder. Das erstellte Objekt wird dem GLTF Loader, welcher für das Laden von glTF und glb Dateien zu Nutzen ist übergeben und anschließend das Mesh von dem angegeben Pfad geladen. In der Release Version werden diese Pfade angepasst und die notwendigen Dateien dem Programmcode beiliegen, da alle Ressourcen in dieser Version enthalten sein sollen. Anstelle nun mittels resolve und reject die Erfüllung des Promise zu verifizieren, wird das Mesh direkt in die Scene eingeladen. Das hat zur Folge das bei nicht Erfüllung des Promise die Anwendung weiter ausgeführt wird aber ohne geladenes Mesh. Auf Grund des Umfanges dieser Arbeit wird auf eine Fehlerbehandlung verzichtet da diese zu weit führen würde. Durch Aufruf der Funktion *Controller.setMeshProperties (head.morphTargetInfluences, head.morphTargetDictionary)* werden die Blendshapes bzw. Morphtargets, an den Controller übergeben. Dieses Vorgehen soll auch bei unterschiedlichen Meshes dafür sorgen, dass die jeweils enthaltenen Blendshapes im Controller Skript vorliegen, wenn die Konvention von three.js eingehalten werden. Es müssen ein Dictionary mit der Zuweisung der Morph Target Namen an ihre Nummer in einem Morph Target Array vorliegen. Die Anforderungen A1.1 und A1.2 werden erfüllt.

Animation Loop

In einer three.js Anwendung und jeder Anwendung, die auf einer Canvas ein Bild zeichnet, muss eine Kommunikation zwischen dem Browser, der für die Darstellung der gesamten DOM-Struktur und der Zuweisung von Ressourcen verantwortlich ist und der Anwendung, die eine Animation auf der Leinwand darstellen will, implementiert werden. Hierzu wird ab Codezeile 690 der View.js eine für three.js typische Funktion *View.animate()* Funktion, definiert.

Durch die *requestAnimationFrame()* DOM-Funktion wird dem Browser mitgeteilt, dass eine Animation abgespielt werden soll und eine Funktion übergeben, die beim nächsten *repaint* des Browsers aufgerufen werden soll (W3Schools, n.d.). Die Funktion ist rekursiv aufgebaut, ruft sich also selbst bei jedem *repaint* wieder auf und bildet so den Animation Loop. Zur Synchronisierung wird dem Mixer in jedem Durchlauf die Zeitdifferenz zwischen den Frames mitgeteilt, der daraufhin seine globale Zeitrechnung aktualisiert. Zuletzt wird der *THREE.WebGLRenderer* Instanz mitgeteilt auf welchen Objekten das Rendering für das nächste Frame durchgeführt wird. Diese Funktionalität wird inital gestartet und bedarf in dieser Anwendung keiner weiteren Anpassungen, da die Steuerung der Animation über die Instanz des *THREE.AnimationMixer* gesteuert wird. Das Starten des Loop geschieht in Zeile 111 der View.js.

Laden der Konfiguration

Bevor die Slider Container mit Elementen aus der Konfigurationsdatei befüllt werden können, muss sichergestellt sein, dass die *configDefault.json* Datei geladen ist und eine gültige Struktur besitzt. Dazu wird die Funktion *Controller.initialyLoadAUConfig()* in Zeile 87 der View.js, welche ihrerseits mit einem booleschen Wert anzeigt, dass der Vorgang erfolgreich abgeschlossen wurde, aufgerufen.

Diese Funktion ist in Zeile 286 - 312 im Controller Skript definiert. Hierzu wird die *fetch()* Methode der Fetch API verwendet. Diese Schnittstelle ist für den Zugriff auf Ressourcen entwickelt, um die teils sehr komplexen Vorgänge in eine einfach zu nutzende Form zu abstrahieren (WHATWG, n.d.). Durch den Aufruf der Methode wird ein Request, nach der durch die übergebene URL spezifizierte Ressource, gesendet, welcher mit einem *response* Objekt beantwortet wird. Dieses kann auf korrekte Erfüllung mit dem *response.ok* Attribut geprüft werden. Danach wird der Inhalt der Anfrage durch das *Controller.configSchema* in Zeile 394 - 462 der Controller.js auf Gültigkeit geprüft und bei Erfüllung ein *True* zurückgegeben. Das Schema Objekt des jema.js Validators ist im Controller Skript definiert und nutzt die *Schema.validate()* Methode, um die übergebenen JSON-Daten auf Gültigkeit zu prüfen. Sollte die Datei dem Schema nicht genügen werden durch einen Aufruf der *Schema.errors()* Methode auf dem Schema Objekt die Fehler in der Validierung zurückgegeben. Anschließend werden diese auf der Konsole als *error* Meldung ausgegeben. Sollte bereits das Laden der Datei nicht möglich sein wird mittels eines *try/catch* Blocks dieser Fehler abgefangen, um das Programm fortzuführen. Allerdings werden in diesem Fall die Slider Container nicht befüllt und zur Nutzung müssen die AU manuell hinzugefügt werden. Sollte ein Validierungsfehler in der JSON-Datei vorliegen, wird dem Nutzer die Möglichkeit gegeben mittels des *confirm* Dialogs des Browsers, der die Antwort in einen booleschen Wert wandelt, die Daten trotzdem zu laden. Dieses Vorgehen wurde gewählt, da es durch Kodierungsunterschiede in der Datei, wenn als Speicherformat nicht UTF-8 gewählt wurde, zu einer falschen Validierung trotz formal richtigen Einträgen kommen kann. Wenn der Fetch Vorgang abgeschlossen ist, wird im Programmfluss die Variable *View.succsefullyLoaded* ausgewertet und der Slider Container befüllt. Die Anforderung 2.9 ist somit umgesetzt und A2.28 und A2.29 wurden vorbereitet.

5.5. HTML-Dokument

Der Aufruf der Seite index.html, durch einen lokalen oder Webserver zur Verfügung gestellt, initialisiert das Laden des Stylesheets style.css sowie des JavaScript Einstiegspunkts View.js. Die Skripte nehmen ihrerseits ebenfalls die notwendigen Importe vor. Zur Nutzung in einem Browser ist es notwendig eine Import Map zu erstellen, da ein Browser darauf angewiesen ist eine URL für die benötigten Ressourcen zu verwenden und Browser keinen direkten Zugriff auf das Dateisystem haben und die relativen Pfade nicht auflösen können. (Bradley, n.d.). Dieses Problem besteht in der Webpack Release Version zwar nicht, da hier alle Abhängigkeiten aufgelöst und in der resultierenden bundle.js gespeichert wurden, aber um auch die ungepackte Anwendung auszuführen, ist dieses Vorgehen notwendig. Die Anforderung A1.2 ist somit erfüllt.

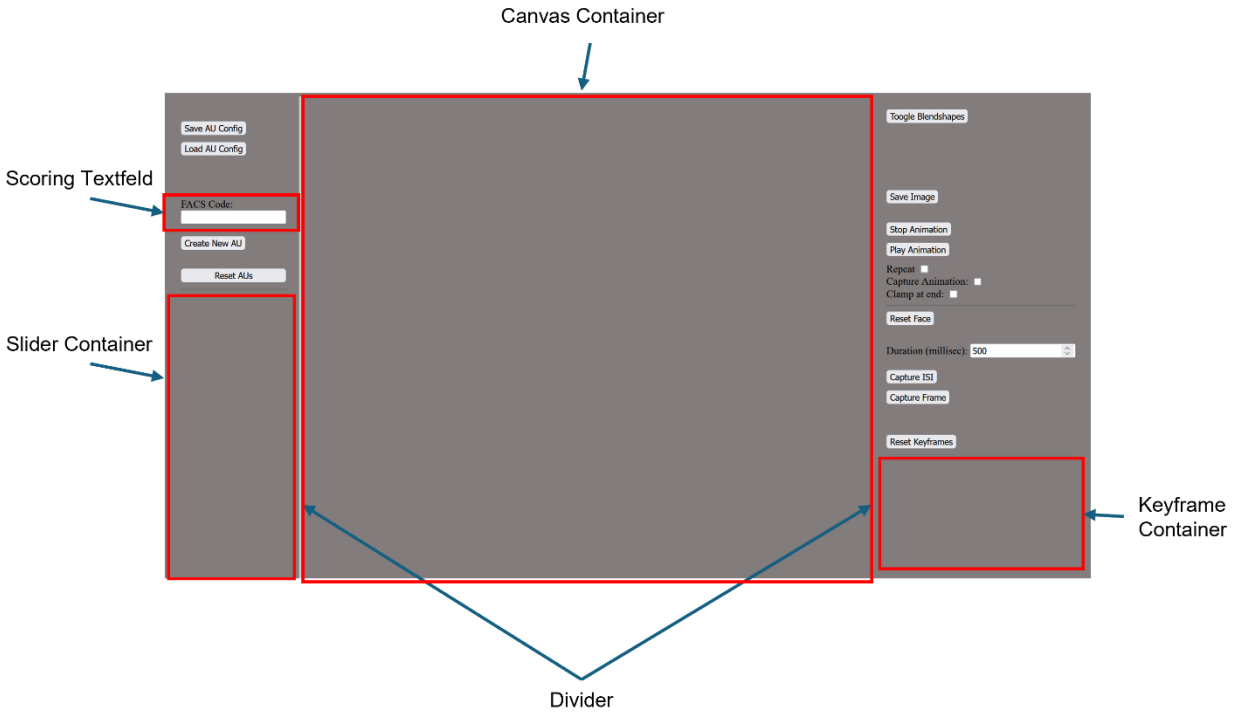
Die dynamischen Anteile des Layouts, der slider-container und der image-container werden zur Laufzeit befüllt, da erst hier feststeht, welche Elemente und wie viele erstellt werden müssen. Sie sind daher im HTML-Dokument nur als Platzhalter vorhanden, durch ein <*hr*> Tag, das als Kontext-Trennung fungiert, von den sie umschließenden Containern getrennt. Streng genommen sind alle

drei Columns dynamisch, da durch die Anpassung ihrer Style Eigenschaften seitens der Divider ihre Größe dynamisch geändert wird. Dieser Umstand wird in der Betrachtung vernachlässigt.

Zur Strukturierung der Oberfläche wird der <body> Tag zunächst in mehrere Bereiche aufgeteilt, die als Grundstruktur dienen. Ein Element wird in drei Teilbereiche eingeteilt (left, middle und right), sowie die statischen Kontrollelemente festgelegt. Die dynamischen Anteile der Seite werden mit HTML „id“ Attributen belegt, um sie über das DOM einfach verfügbar zu machen. Ohne die Programmlogik werden diese Bereiche als leere Elemente angezeigt, wie in nachfolgender Abbildung dargestellt. Zusätzlich befinden sich zwei Divider Elemente zwischen den Columns, um die Größe dynamisch anzupassen. Somit besteht die Seite aus statischen und dynamischen Elementen, die anschließend näher erläutert werden.

5.5.1 Dynamische Elemente des HTML-Dokuments

Abb. 6 Dynamische Elemente



Quelle: Eigene Darstellung

Canvas Container

Die Canvas, die vom Renderer als DOM-Element bereitgestellt wird, wird dem Canvas Container, der sich in der mittleren Column befindet, zugewiesen. Das geschieht in Zeile 48 der View.js. Die Scrollbar der Canvas zentriert sich direkt nach dem Laden sowie bei Größenänderung des Containers, um ein übersichtliches Bild zu präsentieren. Somit muss bei Änderung der Größe des Fensters nicht erst manuell angepasst werden, sondern das Gesicht bleibt stets zentriert. Hierzu wird das Verhalten in der Funktion *View.centerScrollbars()* in den Zeilen 517 - 527 der View.js definiert und beim Laden der Seite sowie der erneuten Berechnung des Rendering Bereichs aufgerufen.

Eine erneute Berechnung des Rendering Bereichs wird durch Größenänderung des Browserfensters oder der Nutzung der Divider ausgelöst. Hierzu wurde ein Eventlistener an dem *window* Element der Seite registriert, der dadurch die *View.resizeRendererToDisplaySize()* Funktion aufruft sobald ein *resize* Event ausgelöst wird. Diese Funktion ist in den Zeilen 499 - 514 des View Skripts definiert. Sie dient zur Erhaltung der Darstellung bei Änderung der Fenstergröße. Somit kann bei beliebiger Fenstergröße stets ein optisch Ansprechender Fokus auf das Gesicht gelegt werden und die Darstellung behält das richtige Aspect Ratio.

Die Darstellung des Face Mesh auf der Canvas wird wie oben erläutert durch den Animation Loop gesteuert und liegt vollständig bei three.js. Alle besprochenen Anpassungen betreffen nur den Container, welcher das DOM-Element des Renderers enthält und im Falle des Rendering Bereichs die Anpassung der Kamera. Die Anforderung A2.14 konnte somit teilweise erfüllt werden.

Slider Container

Da die beiden Funktionen *createSlidersForActionUnits()* und *createExpressionSliders()* funktional sehr ähnlich sind, wird das Erzeugen beider Slider Container anhand des Containers für die AU Steuerung erläutert und hier auch nur die Teile, welche nicht zur HTML Generierung verwendet werden, da diese ein Standardvorgehen darstellen und nicht zum Verständnis der Applikation beitragen.

Die nachfolgende Erläuterung bezieht sich auf die Codezeilen 353 - 408 in der View.js. Die Funktion *View.createSlidersForActionUnits()* erzeugt bei jedem Aufruf ein Mapping der aktuellen Konfiguration, die in dem Objekt *Controller.actionUnits* vorliegt. Hierzu wird das Objekt vom Controller angefragt. Nach dem Löschen des aktuellen Inhalts des Containers wird basierend auf den Werten der vom Controller übergebenen aktuellen Werte das Erscheinungsbild neu aufgebaut. Hierzu wird über *Controller.actionUnits* iteriert und ein * * mit einem *<input>* Kindelement für jede AU erstellt, das mit einem Eventlistener auf den Blendshape Objekten des Mesh verknüpft wird. Dazu werden vom Controller die aktuell zu dieser AU zugewiesenen Blendshapes abgefragt und Eventlistener registriert, die eine Änderung des Blendshape Wertes an die Funktion *View.changeBlendshapValue()* weiterreichen. Diese Funktion ist als das Interface des geladen Meshs gedacht und soll Zugriffe auf die Geometrie an einem zentralen Ort bündeln, um Konflikte beim Zugriff zu vermeiden. Hiervon wird nur in der *View.resetFace()* Funktion abgewichen da hier eine zentrale Funktion zum Setzen aller Blendshapes auf null gewünscht ist. Hierbei wird durch ein Ereignis, das Bewegen des Sliders, ein Event ausgelöst, das in den verbundenen Handlerfunktionen die Änderung der Einflüsse auf das Mesh steuert. Dabei sind die Slider Werte der AU auf sechs Stufen beschränkt, die fünf Scoring Werte des FACS und die Position null. Die kombinierten AU sind auf den Prozentbereich zwischen null und eins festgelegt. Dieses Vorgehen wurde gewählt, um dem distinkten Charakter des FACS gerecht zu werden und gleichzeitig eine kontinuierliche Verteilung zu ermöglichen. Dem Nutzer ist es somit ermöglicht, die Ausprägungen des FACS auf das Face Mesh zu übertragen, damit ist Anforderung A2.13 teilweise erfüllt.

Scoring Textfeld

Die nachfolgend beschriebenen Codezeilen sind 421 – 436 in der View.js Datei und erfüllen Anforderung A2.5. Zum Aktualisieren des Textfeldes wird an die Elemente der CSS-Klasse *au-slider* ein Eventlistener angehängt, der die Veränderung der Werte an den Slidern zu dem Textfeld weiterleitet. Hierzu wird, ähnlich dem obigen Abschnitt über den Slider Container, in dessen Eventlistener am *<input>* Element der Slider, die Funktion *View.updateAUTextField()* aufgerufen. Wenn der Slider nicht auf der null Position ist, wird aus den möglichen Scoring Werten der entsprechende ausgewählt und die AU-Nummer des Sliders in einen String formatiert, der eine Codierung nach FACS darstellt. Mit dem „+“ wird wie in den Grundlagen erläutert die Kombination der AU abgebildet. Das *<input>* Textfeld ist mit dem Attribut *readonly* erweitert, damit es als reine Anzeige fungieren kann.

Divider

Die Veränderung der Content Divider wurde nach dem Vorbild von phuoc.ng (Nguyen, 2023) erstellt und an die Anwendung angepasst. Die *View.makeResizableDivider(dividerId)* befindet sich in Zeile 467 - 497 in der View.js. Es wird ein Eventlistener auf den Divider Elementen registriert der auf das Drücken und Halten des Mausknopfes reagiert und bei Erreichen des Eventtargets wie in den Grundlagen beschrieben ein Event an den Handler übergibt. Das *e* benannte Eventelement enthält in seinem Attribut *clientX* die X Koordinate an welchem der *MouseEvent* ausgelöst wurde und damit die initiale Position der Berechnung auf der X-Achse des Bildschirms. Das CSS-Element *ew-resize* wird verwendet, um dem Nutzer die Rückmeldung über die nun beweglichen Divider zu geben, indem es den Cursor zu dem bekannten Doppelpfeil ändert. Da es zwei Divider Elemente *dividerLeft* und *dividerRight* gibt, muss in der Berechnung unterschieden werden auf welchem Divider die Operation ausgeführt wird. Mittels des Bedingungsoperators „?“ wird geprüft um welche der zwei Möglichkeiten es sich handelt und vom initiale X Wert entsprechend addiert oder subtrahiert. Wenn der Mausknopf wieder losgelassen wird, werden die Eventlistener entfernt da sie sonst nach dem Verschieben immer noch registriert wären und sich somit aufsummieren könnten was die Performanz beeinflussen würden. Dieses Vorgehen trägt zur Erfüllung der A2.14 bei.

Keyframe Container

Im Gegensatz zur Erstellung von Einzelbildern, die den aktuellen Zustand der Leinwand darstellen der sowohl über die direkte Blendshape Steuerung, eine Aufnahme während der Animation als auch über die Slider verändert werden kann, ist für die Animation eine strengere Struktur vorgesehen, um eine reibungslose Verarbeitung zu gewährleisten. Wie bereits im Architekturteil beschrieben spiegelt der Container den Zustand des Animation Objekts wider. Es wurde eine Callback Funktion für die Buttons *View.captureFrameButton()* und *View.captureISIButton()* angelegt, welche jeweils die Funktion *View.createThumbnail(keyframeType)* mit individuellem Übergabewert, *keyframeType*, aufrufen. In dieser Funktion wird der Keyframe an den Controller übergeben und auf eine Bestätigung gewartet. Ist der betreffende Keyframe in einem gültigen Format, wird das durch den Controller bestätigt

und der Container aktualisiert. Wenn das Format nicht gültig ist, was beispielsweise über eine Texteingabe im Duration Feld möglich ist, da der Number Constraint des HTML-Elements *<input>* das nicht kontrolliert, wird eine Meldung ausgegeben und der Container als auch die Animation verbleiben in dem vorherigen Status.

Abb.7 Aktivierung aller Blendshapes

Als Indikator für ein ISI wurde ein Array der Blendshapes gewählt das mit Einsen befüllt ist, da wie in Abbildung 7 ersichtlich, das Resultat keinen Nutzen erfüllt. Da dieser Ausdruck nicht in dem Keyframes Container angezeigt werden soll, wurde hierzu ein schwarzer Bildschirm in den Abmessungen 64 mal 64 Pixel erstellt und der Applikation beigelegt. Dieses Bild wird im Fall eines ISI-Frames als Thumbnail dem Container hinzugefügt.

Quelle: Eigene Darstellung - Screenshot aus der Anwendung

Das *Controller.animation* Objekt verwaltet die Liste an Keyframes in der nachfolgend dargestellten Struktur.

Abb.8 Struktur des Keyframes Array

```
let animation = { keyframes: [] };
    const keyframe =
    {
        position,
        duration,
        influences,
        thumbnail
    }
```

Quelle: Eigene Darstellung aus der Controller.js Datei

Durch die *View.createThumbnail(keyframeType)* Handlerfunktion wird zunächst der dem Typ des Frames entsprechende Thumbnail erstellt und anschließend der Keyframe an den Controller weitergereicht und auf eine Validierung gewartet. Wenn die Validierung erfolgreich ist, wird der Frame der *Controller.animation* hinzugefügt und der Keyframe Container aktualisiert. Bei fehlerhaftem Eintrag wird der Validierungsfehler mit der *Schema.errors()* Funktion eine Meldung auf der Konsole ausgegeben und zusätzlich wird der Nutzer über die *alert* Methode des Browsers hierüber informiert.

Es werden die Einträge *duration* und *influences* für die Erstellung der Animation benötigt. Das Thumbnail stellt die Konfiguration des Gesichtes in dem Keyframe Container dar. Die dem Keyframe angehängten Buttons „Edit“ und „Remove“ ermöglichen die nachträgliche Bearbeitung der Animation. Durch die *View.changeThumbnail(position)* ab Zeile 205 der View.js wird bei Anklicken des Buttons „Edit“ ein neuer Keyframe über die *Controller.changeKeyframe(position, newDuration = null,*

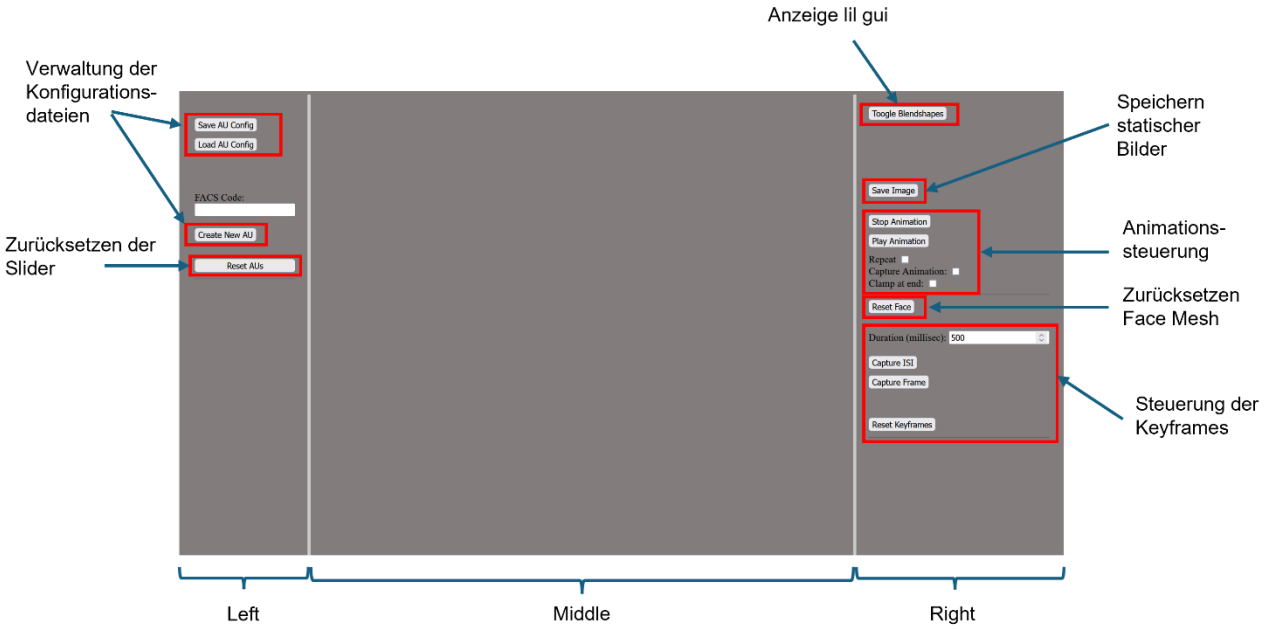
newInfluences = null, newThumbnail = null) die Attribute des Keyframes im Animation Objekt aktualisiert. Hierfür wird nur auf gültige Felder geprüft und die Position, wie an die Funktion übergeben, verwendet, da diese ein unveränderlicher Bestandteil eines Keyframes ist der nur durch die Remove Methode geändert wird, welche das gesamte *Controller.Animation* Objekt neu aufbaut und somit sicherstellt, dass die Position den aktuellen Zustand widerspiegelt. Hierbei dienen die Default Werte, die auf „null“ gesetzt sind, der Sicherstellung das ein Wert verfügbar ist. Auf eine Prüfung der Wertebereiche wurde in diesem Entwicklungszustand verzichtet.

Zum Entfernen eines Keyframes wird direkt die *Controller.removeKeyframe(position)* in Zeile 157 – 171 des Controller Skripts aufgerufen, da hier keine Attribute notwendig sind, welche von der Oberfläche abgefragt werden müssen, sondern nur die im Thumbnail selbst gespeicherte Position, die nicht zu ändern ist. Die Funktion baut, nach erfolgter Prüfung auf den gültigen Wertebereich des Position Parameters das Animation Objekt neu auf. Im Anschluss an beide Operationen ist, durch die den Eventlistenern der Buttons angehängte Handlerfunktion, mit Aufruf der *updateThumbnail-Container()* eine konsistente Darstellung gewährleistet. Dies geschieht, indem der Thumbnail Container anhand des neuen Animation Objekts neu erstellt wird.

In diesem Abschnitt wurde die Erfüllung der Anforderungen A2.1, A2.2, A2.16 und A2.24 gezeigt. Das Laden einer Animation in A2.15 und A2.12 sollen umgesetzt werden, indem das *Controller.animation* Objekt als JSON-Datei gespeichert und geladen werden kann. A2.21 zielt darauf ab, dass bei Druck auf einen Thumbnail das Mesh auf der Canvas den Ausdruck des angeklickten Thumbnails annimmt und wird ebenfalls auf eine zukünftige Version verschoben. Eine erweiterte Fehlerkorrektur wird die Umsetzung der A2.28 und A2.29 unterstützen.

5.5.2 Statische Elemente des HTML-Dokuments

Abb.9 Statische HTML-Elemente



Quelle: Eigene Darstellung

Direkte Blendshape Steuerung

Der direkte Zugriff auf die Blendshapes erfolgt über die „lil gui“. Dazu wird ein GUI-Objekt erstellt und die Attribute des Mesh übergeben, die durch die Steuerung kontrolliert werden sollen. Hier sind das die Steuerung der Blendshapes, in three.js Morphtargets genannt, dieser Teil wurde aus der Beispiel Implementierung von three.js übernommen, sowie die Farbe des Hintergrundes und die Darstellung der Geometrie als Wireframe. Das wird durch die Funktion *View.createGUIForModel(),* die im Startup des Programms aufgerufen wird, erfüllt. Die gui wird zunächst mit der Methode *hide()* des GUI Objekts nicht sichtbar gehalten da zur Nutzung der App meist keine Interaktion mit den Blendshapes notwendig ist. Durch den „Toggle Blendshapes“ Button wird die Oberfläche Sichtbar geschaltet und kann über den Pfeil ausgeklappt werden. Auf die Bedienung wird im Anwendungsteil noch eingegangen.

Verwalten der Konfigurationsdateien

Da sich die Steuerung der Gesichtsausdrücke durch die Slider widerspiegeln soll, welche die Konfigurationsobjekte abbilden, muss eine gültige Konfigurationsdatei geladen werden, um die Darstellung zu ermöglichen. Hierzu wurde das im Anhang ersichtliche Schema einer JSON-Datei erstellt, das nach dem Einlesen der Konfigurationsdatei validiert werden muss. Da das Laden einer Konfigurationsdatei im Wesentlichen wie das initiale Laden der Konfiguration abläuft, wird hier auf eine genaue Erläuterung verzichtet. Die Funktionalität ist in der Controller.js in Zeile 270 - 284 definiert.

Beim Speichern der aktuellen Konfigurationsobjekte ist es notwendig die UTF-8 Kodierung zu verwenden, da ansonsten das Schema als ungültig validiert werden kann, obwohl die offensichtlichen

Voraussetzungen erfüllt sind. Die Objekte werden ohne Validierungsschema direkt in eine JSON-Datei geschrieben, da es durch die in einer zukünftigen Version implementierte Validierung, beim Hinzufügen von Einträgen in die Konfigurationsobjekte, actionUnits und Expressions nicht möglich sein soll ungültige Objekte hinzuzufügen. Das eigentliche Speichern erfolgt hier ähnlich dem Speichern von Thumbnails durch die Generierung eines Inline-Elements und anschließende Zuweisung an ein HTML-Element, auf dem ein Click Event ausgelöst wird, der anschließend den Download Dialog des Browsers öffnet. Hier zeigen sich die bereits erwähnten Vorteile in der Kombination von JSON und JS da die Objekte im Code durch die *stringify()* Methode der JSON API direkt in Text umgewandelt werden und sich abspeichern lassen.

Rücksetzen der Slider

Zum Rücksetzen der Schieberegler wird ein Button erstellt, der sich innerhalb der Begrenzungen des Containers bis zum Rand ausbreitet. Dazu wird die „inline“ Eigenschaft der „left-column“ im Stylesheet überschrieben und die Ausbreitung bis zum Rand des Containers festgelegt. Somit wird das Element, das häufig verwendet wird, besser sichtbar und komfortabler zu bedienen. An diesem Button ist ein Eventhandler registriert und über die Funktion *View.resetSliders()* die AU die Expressions, dass Face Mesh sowie das Textfeld in den Initialzustand versetzt. Dabei werden die Slider nicht geändert, sondern neu initialisiert, wie in der Beschreibung des Slider Containers erläutert. Auch das Rücksetzen des Mesh folgt der Beschreibung zum Zurücksetzen des Face Mesh. Für das Leeren des Textfeldes mit der Kodierung des angezeigten Ausdrucks wurde die Funktion *View.clearFACSCodeTextfield()* definiert, die den Wert auf den leeren String zurücksetzt. Das ist notwendig, da wie im Abschnitt zum Textfeld beschrieben, Eventhandler nur durch eine Änderung der Slider die Update Funktion aufrufen und somit bis zur nächsten Bewegung eines Sliders noch der letzte Wert dargestellt werden würde.

Speichern statischer Bilder

Zum Erstellen der Standbilder dient der Button „Save Image“, der die Funktion *View.saveCroppedImage()* über einen Eventlistener aufruft. Das Bild soll an den Rändern gekürzt werden, da eine Darstellung der gesamten Canvas den Gesichtsausdruck zu klein darstellt. Die Werte um die die Darstellung abgeschnitten wird, wurden durch Ausprobieren festgelegt und sollen in einer zukünftigen Version vom Nutzer bestimmt werden können. Die Funktion erstellt als erstes eine Kopie der aktuell dargestellten Leinwand. Aus dieser wird dann der Bereich, in dem sich das Gesicht befindet, ausgeschnitten. Da die Scrollbars der mittleren Column die Leinwand als Ganzes verschieben befindet sich das Gesicht immer in der Mitte, auch wenn es an den Bildrändern sichtbar ist, lediglich die Größe die durch das Heran- und Herauszoomen geändert werden kann, hat somit einen Einfluss auf die Position in dem Screenshot. Die so erstellte Kopie der aktuellen Darstellung wird als Data URL im PNG-Format in die Seite integriert und über ein *href* Attribut in dem *<a>* Tag zum Download verfügbar gemacht. Das Vorgehen bietet den Vorteil, dass die Daten nicht auf dem Server, sondern auf dem Rechner des Nutzers gespeichert werden. Der so erstellte Downloadlink wird über das

HTML-Element <*link*>*,* dass zum Laden von Ressourcen verwendet werden kann, aufgerufen indem ein *click* Ereignis an dem <*link*> Element ausgelöst wird (MDN, 2024b). Daraufhin öffnet sich der Download Dialog des verwendeten Browsers und der Nutzer kann den Namen der Datei ändern, den Download bestätigen oder abbrechen. Als Vorschlag für den Dateinamen dient ein String der den FACS-Code aus dem Textfeld *codeFACSTextfield* der den Scoring Code mit dem Suffix „_cropped_image“ verbindet. Anforderung 2.7 ist somit erfüllt.

Zurücksetzen des Face Mesh

Der Button mit der id „resetAUButton“ ruft die Handlerfunktion *View.resetFace()* auf. In den Codezeilen 541 - 553 der View.js werden die 4 Geometrien des Gesichts direkt aufgerufen und jeweils das Attribute *visible* des Mesh auf *true* gesetzt. Dieses Vorgehen ist notwendig, da über die Checkbox „Clamp at end“ das Bild am Ende einer Animation in dem Zustand eines ISI-Frames verbleiben kann und ein schwarzer Bildschirm gezeigt wird. Danach erfolgt die Zuweisung des Wertes 0 an alle Blendshapes des Meshes, um den neutralen Gesichtsausdruck wiederherzustellen. Anforderung A2.6 wird hiermit teilweise erfüllt, da die Kameraposition nicht zurückgesetzt wird. Zur Nutzung ist es nicht zwingend erforderlich, da mit der Aktualisierung der Seite die Kamera wieder in die Ursprungsposition versetzt wird. Zukünftig soll das in die hier erläuterte Funktion integriert werden.

Steuerung der Keyframes

Der Image Container, der die Thumbnails der Keyframes enthält, wird ständig durch das Hinzufügen und Entfernen von Keyframes bzw. ISI-Frames aktualisiert. Die Übergangszeit zwischen den Frames wird durch den Nutzer mittels des „Duration“ Feldes festgelegt. Hierzu wurde das HTML-Element <*input*> in Zeile 70 der index.html so konfiguriert, dass durch die Pfeiltasten des Textfeldes in 100ms Schritten erhöht und reduziert werden kann. Zusätzlich wird ein Startwert von 500ms festgelegt. Es kann aber auch durch eine direkte Eingabe der Wert angepasst werden. Durch den Button „Reset Keyframes“ wird die Funktion *View.resetAnimation()* ab Zeile 606 der View.js aufgerufen und nach einer Bestätigung gefragt. Dies geschieht, um ungewolltem Löschen der Animation vorzubeugen, sollte der Nutzer versehentlich auf den Button drücken. Sollte das Löschen bestätigt werden, wird die *Controller. deleteKeyframes()* aufgerufen und setzt das *Controller.animation* Objekt zurück.

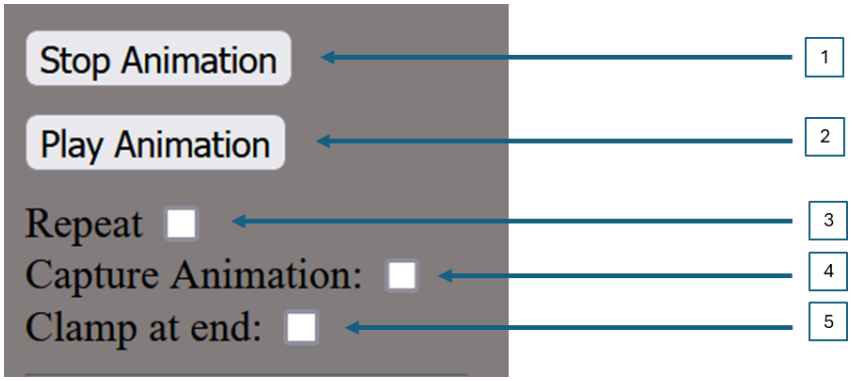
Animationenssteuerung

Zum Erstellen der Animationen wird das Objekt *Controller.animation* verwendet, welches das Array Keyframes enthält. Hier werden die Blendshape Werte, die Transitionszeit des Frames sowie das Thumbnail im Vorschaufenster abgelegt. Diese Struktur ermöglicht auch das Editieren und die Vorschau der Keyframes. Dabei wird das angezeigte Thumbnail in *Controller.animation* gespeichert.

Da das Erstellen der Animation die Kernfunktionalität der App darstellt, soll diese hier ausführlich erläutert werden. Animationen stellen sich in der Regel aus Keyframes und dem Übergang zwischen ihnen dar, was auch eine intuitive Annahme über eine Veränderung im Zeitverlauf widerspiegelt. Aus

diesem Grund wurde dieses bewährte System auch in dieser Anwendung gewählt, um dem Nutzer die Interaktion so intuitiv wie möglich zu präsentieren.

Abb.10 Animationssteuerelemente



Quelle: Eigene Darstellung

Die Schaltfläche 1 stoppt die aktuell laufende Animation. Hierzu wird die Funktion *View.stopAllAnimations* aufgerufen, die über den internen Cache Speicher des Mixers auf alle *THREE.AnimationAction* Objekte, welcher der Mixer gespeichert hat, die Methode *stop()* des *THREE.AnimationMixers* aufruft. Anschließend wird der Clip aus dem Cache des Mixers freigegeben. Dieses Verhalten ist sinnvoll, da in der Anwendung auch nur eine Animation abgespielt werden soll. Daher ist immer nur die aktuelle Animation, welche durch Schaltfläche 2 erzeugt, wird von Bedeutung. Hierdurch wird sichergestellt, dass immer der aktuelle Zustand des Keyframe Containers in der Animation widergespiegelt wird. Durch Schaltfläche 2 wird die Erzeugung eines *THREE.AnimationClip* Objekts durch die Controller Komponente angestoßen, die ihrerseits den Zustand des *Controller.animation* Objekts zu einem Animationsclip wandelt, dieser Vorgang wird im nächsten Abschnitt genauer erläutert. Sobald der Animationsclip erstellt wurde, wird durch den Mixer die Animation als Action definiert und abgespielt. Hierzu werden die Zustände der Checkboxen 3, 4, 5 abgefragt und in der Action verarbeitet. Dazu werden die Methoden der Action im Fall von 3 und 5 genutzt. Die beiden Methoden bewirken zum einen, dass das zu animierende Objekt im Falle der Klasseneigenschaft *clampWhenFinished* eines THREE.AnimationAction Objekts im Zustand des letzten Frames verharrt, wenn die Action abgeschlossen ist. Im Falle von Repeat können über die *setLoop()* Methode und die Übergabe der beiden Konstanten *THREE.LoopRepeat* und *THREE.LoopOnce,* eine einmalige oder eine Endloswiedergabe festgelegt werden. Die Checkbox 4 ruft die Funktion „recordCanvas“ auf. Die Funktion „View.recordCanvas“ basiert auf der „captureStream“ Methode, die das HTML-Element <canvas> anbietet. Die Implementierung folgt hier den Beispielen der MDN (MDN, 2024a).

Hierdurch wird der Inhalt der Leinwand zu einem MediaStream geparst, der somit der MediaStream API zugänglich wird. Ein MediaRecorder Objekt wird erstellt und der Stream als Parameter übergeben. Nun ist es möglich, über dieses im Programmcode „recorder“ genannte Objekt auf die Frames der Canvas zuzugreifen. Hierzu wird die Methode „ondataavailable“ des recorder Objekts genutzt, welche die einzelnen Frames in einem Array speichert. Sobald über den am Mixer registrierten

Eventlistener der Stopp des Streams signalisiert wird, wird das Array mit den Einzelframes zu einem Blob im Dateiformat „webm“ umgewandelt und dieser dann ähnlich dem Vorgehen beim Speichern der Standbilder zum Download angeboten. Da die entstandenen Dateien größer als einzelne Standbilder sind, wird hier der Browsercache nicht nur freigegeben, sondern die entstandenen Objekte explizit gelöscht. Dieser Abschnitt behandelte die Erfüllung der Anforderungen A2.3, A2.4, A2.18, A2.20.

Erstellen des Animation Clips

Die Funktion *Controller.createMorphAnimationClip()* in Zeile 197 - 266 der Controller.js, wird durch das JS-Keyword *export* dem View Skript verfügbar gemacht. Zunächst wird der aktuelle Zustand des *Controller.animation* Objekts ausgelesen und in den beiden Konstanten *duration* und *influences* gespeichert. Hierbei muss gesichert sein, dass diese beiden Arrays die gleiche Größe aufweisen, jeweils einen Zeitwert zwischen zwei Keyframes und den initialen Zeitwert von 0. Prinzipiell ist es durch die *Controller.addKeyframe()* Funktion nicht möglich, dass diese Struktur verletzt wird. Allerdings wird zur Sicherheit trotzdem eine Prüfung vorgenommen.

Es folgt die Behandlung der ISI-Frames. Da die Darstellung eines Influence Arrays, das aus Einsen besteht, kein gewünschter Ausdruck ist, soll nicht in diesen übergeblendet werden, sondern der vorherige Zustand beibehalten oder der nachfolgende vorgezogen werden. Dazu wird das Influences Array neu aufgebaut und ohne die ISI-Ausprägungen gespeichert, da diese zur Steuerung der Sichtbarkeit verwendet werden. Hierzu ist als erstes die Befüllung des Sichtbarkeitstracks wichtig. Es wird über das Influences Objekt iteriert und mittels der Erstellten Hilfsfunktion *isISIFrame()* geprüft, ob es sich um ein Array ausschließlich bestehend aus „1“ Einträgen handelt. Wenn dem so ist, wird dem Sichtbarkeitstrack der Wert *false* hinzugefügt, ansonsten *true* um das Mesh sichtbar zu halten.

Mittels der Template-Literals, die in JS dazu verwendet werden können einen Platzhalter dynamisch mit Werten zu befüllen und am Ende einen String herauszubekommen(MDN, 2023b), wird für jeden Blendshape Eintrag ein Track erzeugt. Dieser Track besteht aus dem erzeugten String, der zuerst die betroffene Komponente des Objekts enthält. Hier *mesh_2,* gefolgt von der dynamisch erstellten Bezeichnung der Eigenschaft der Komponente *morphTargetInfluences[${influence}]* und einem Array, welches die Werte des Attributes angibt und dem Zeitarray, das die Zeitpunkte vorgibt zu welchem die Komponente diesen Zustand annimmt. Der dazwischenliegende Bereich wird durch three.js, wie in den Grundlagen beschrieben, interpoliert. An dieser Stelle kann durch Auswahl der Interpolationsfunktionen und dem Hinzufügen eines Tweening Funktion die Anforderung A2.30 später implementiert werden.

Nun wird anhand des zuvor erstellten Arrays für die Sichtbarkeitszeiten ein Track für jeden Teil des Meshes erstellt. Nach Überprüfung, ob eine Änderung der Sichtbarkeit vorliegt, hierzu wird das Array nach einem vom Wert true abweichendem false durchsucht, werden zu den „visible“ Attributen der Objekte, Tracks erstellt. Dazu müssen in diesem Fall auch die unbeweglichen Teile, Augen und Ge-

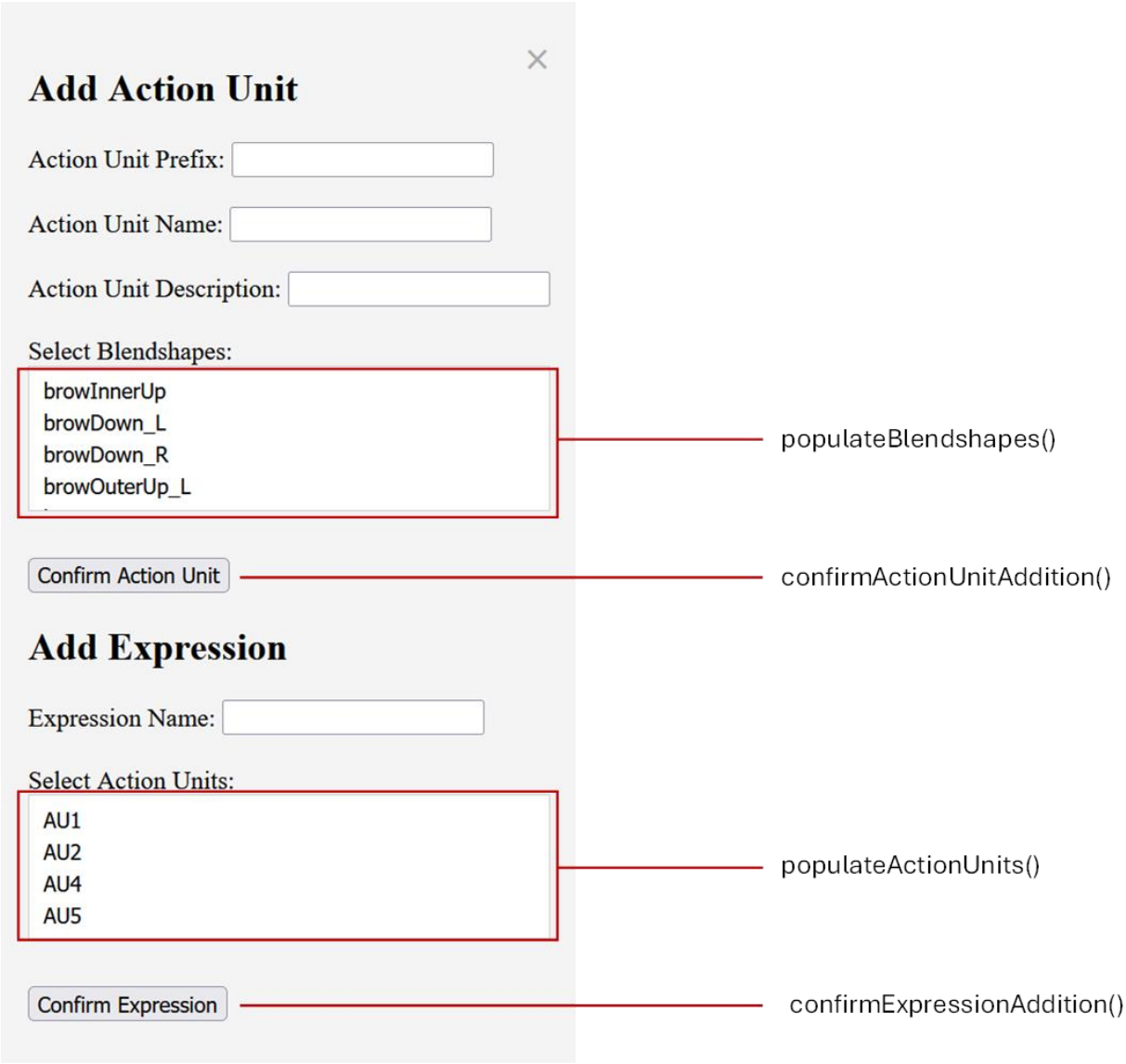
biss angesprochen werden, die in der normalen Nutzung keine Rolle spielen. Die Erstellung der Tracks folgt dem obigen Beispiel. Allerdings handelt es sich um *THREE.BooleanKeyframeTrack* Objekte, die zur Steuerung von Attributen mit Wahrheitswerten verwendet werden im Gegensatz zu den *THREE.NumberKeyframeTrack* Objekten, die für Attribute mit numerischen Werten verwendet werden. Zuletzt werden alle erstellten Tracks in dem tracks Array an den Konstruktor des THREE.AnimationClip Objekts übergeben, der daraus einen Animations Clip erstellt, den die Funktion zurückgibt. Durch die -1 im Konstruktor wird festgelegt, dass die Animationsdauer aus den Zeitwerten berechnet wird, der Name der Animation „CustomAnimation“ ist ein notwendiger Parameter, aber in dieser Anwendung bedeutungslos, da nur eine Animation gleichzeitig existiert.

5.5.3 Modales Fenster zu der Anpassung der Konfiguration

Ein Fenster, welches in Anlehnung an Bühler als modales Fenster bezeichnet wird, da es sich über die Oberfläche legt und den Nutzer zu einer Eingabe zwingt, wird zur Konfigurationsanpassung genutzt (Bühler et al., 2018, S. 35). Diese Definition wird nicht genau erfüllt aber der Name ist zweckmäßig, um das Fenster von dem übrigen Interface abzugrenzen. Da wie oben dargestellt eine exakte Struktur der Einträge in den Konfigurationsobjekten notwendig ist, wird zum einfachen Hinzufügen von AU und den Expressions der Nutzer durch diesen Prozess geleitet und dabei möglichst alle Fehleingaben verhindert.

Für die Darstellung wird nach dem Anklicken des Buttons Create New AU die Sichtbarkeit des Elements hergestellt und das Fenster über allen anderen Elementen angezeigt.

Abb.11 Fenster zur Anpassung der Konfiguration



Quelle: Eigene Darstellung

Durch die Funktionen *View.populateBlendshapes()* und *View.populateActionUnits()* wird der Inhalt der aktuellen AU-Konfiguration und alle Blendshapes die das geladene Model besitzt in die Auswahldialoge eingefügt. Hierzu wird der aktuelle Inhalt gelöscht und in das <*select multiple*> Element des betreffenden Containers jeweils ein <*option*> Element pro Blendshape bzw. AU erzeugt und eingefügt. Dabei ist *multiple* ein Attribut, welches das HTML-Element <*select*> erweitert, um das Auswählen mehrerer Elemente in dem durch <*select*> definierten Drop-Down Menü zu ermöglichen.

Auf die Textfelder wird nicht genauer eingegangen, da ihre Beschriftung und die Erläuterungen zu den Datenstrukturen in Abschnitt 5.3 und die Erläuterung im Anwendungsteil als Erklärung ausreichend sind. Es sei nur noch erwähnt, dass in einer späteren Version eine Formularprüfung vorgesehen ist, um fehlerhafte Nutzereingaben mit Fehlermeldungen abzufangen und damit zu den geplanten A2.28 und A2.29 beizutragen. Erfüllt wurden die Anforderungen A2.10 und A2.11.

5.6 Erfüllung der Anforderungen

Die nicht funktionalen Anforderungen lassen sich nicht direkt einem Implementationsteil zuordnen und ergeben sich aus Interaktionen und Architekturentscheidungen. Die Anforderungen A3.1, A3.2, A3.3 und A3.6 werden als erfüllt angenommen, mit den folgenden Begründungen. Der Quellcode wird öffentlich verfügbar auf GitHub gehostet und ist mit einer MIT Lizenz versehen, die keine Nutzungseinschränkungen enthält, selbes gilt für die Komponenten anderer Entwickler. Nach Ansicht des Autors ist die Oberfläche selbsterklärend und bis auf die Animationserstellung, welche das Verständnis von Keyframes voraussetzt, intuitiv nutzbar. Im Anwendungsteil werden die Erläuterungen zur Nutzung des Keyframe Containers ausreichend sein, um eine einfache Nutzung zu ermöglichen. Auf Grund der während der Entwicklung verwendeten Hardware und der Kompatibilität von aktuellen Webbrowsern zu den verwendeten Frameworks, sollte es möglich sein FaceEx auf einem durchschnittlichen PC mit aktuellem Webbrowser auszuführen.

In der angehängten Tabelle 3 werden die ermittelten Anforderungen und deren Erfüllung gegenübergestellt. An dieser Stelle soll nur eine kurze Erläuterung der als „teilweise Erfüllten“ und „geplanten“ Anforderungen gegeben werden, da diese im Diskussionsteil noch besprochen werden. Anforderungen, die sich erst in der Nutzung zeigen werden an geeigneter Stelle im Anwendungsteil erläutert.

Um vollen Nutzen aus dem Mapping eines Blendshape Systems auf das FACS zu ziehen, muss es möglich sein verschiedene Meshes zu laden wie in A1.3 geplant. Hierzu bietet three.js umfangreiche Möglichkeiten mittels Loadern und Decodern unterschiedliche Asset Formate zu laden. Um dieses Feature umzusetzen, wird ein Dialog Feld angestrebt, in dem der Nutzer ein Mesh auswählen kann und mittels des Datei Headers geprüft wird welche Loader Funktionalität genutzt werden soll. Durch die Übergabe der Influences als Dictionary und Array an das Controller Skript wird zumindest die Blendshape Anzahl momentan variabel gehalten, da aber einige fest programmierten Bereiche Mesh spezifisch sind, erfordert diese Erweiterung umfassende Änderungen. Die Kodierung asymmetrischer und lateraler Ausdrücke, wie in A2.25 und A2.31 beabsichtigt, wurden aktuell nur durch die Konfigurationsstruktur berücksichtigt. Für die oben genannte Beispiel-Studie zu den Adaptation Aftereffects ist die Umsetzung der Anforderungen A2.32, A2.33 und A2.34 wünschenswert, aber im Rahmen dieser Arbeit nicht machbar.

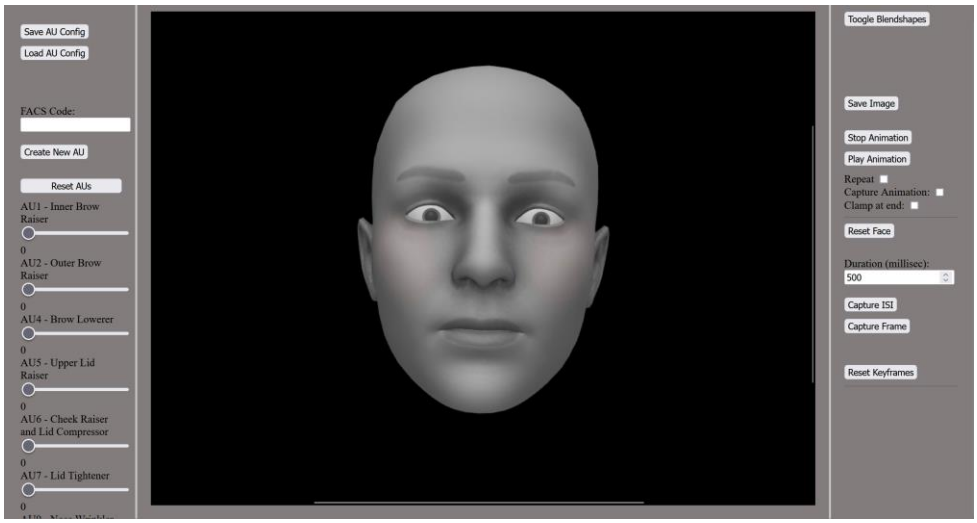
6. Die Anwendung FaceEx

Da das Repository im aktuellen Entwicklungsstand keine Dokumentation enthält, soll die Vorstellung der Anwendung diese Funktion auch teilweise erfüllen und erfolgt daher ausführlich. Parallel dazu wird auf die Erfüllung der oben erstellten Liste der Anforderung eingegangen und an geeigneter Stelle, ein Fazit der Erfüllung gezogen.

Laden der Anwendung

Da es sich um eine Webanwendung handelt, ist zur Ausführung ein Server notwendig. Es können also entweder die Programmdateien auf einem Webserver gehostet werden und die index.html als Einstiegspunkt gewählt werden oder über das Hosting über einen lokalen Server übernommen werden. Eine Erläuterung dazu findet sich in dem Repository zu dieser Arbeit auf Github. Wenn die Seite durch eine dieser Möglichkeiten im Browser verfügbar ist, zeigt sich die Seite wie in nachfolgender Abbildung. Damit werden die Anforderungen A1.1 und A1.2 erfüllt.

Abb.12 Startbildschirm



Quelle: Eigene Darstellung

Nach dem Laden kann direkt mit der Nutzung begonnen werden. Die Grundkonfiguration von AU und Expressions wird in dem Slider Bereich links unten angezeigt. Das Design ist schlicht und zweckmäßig und erfüllt die Anforderung A3.5 daher teilweise. Moderne Anwendungen legen großen Wert auf ein aktuelles Design und ansprechendes Aussehen mit vielen Farben, da sich kein Mehrwert für den Zweck der Arbeit ergibt wird darauf verzichtet.

Anpassen des Fensters

Durch Drücken und Halten der linken Maustaste wird ein Verschieben der beiden Divider am Rand der Leinwand ermöglicht. Das wird durch den, bei Druck angezeigten Doppelpfeil symbolisiert. Die Canvas zentriert sich mit der Anpassung der Größe des Leinwandbereichs sowie bei manueller Anpassung der Fenstergröße.

Drehen und Zoomen

Die in three.js enthaltenen *OrbitControls* sind ein einfacher Weg um die Kamera um Objekte zu bewegen. Durch Drücken der linken Maustaste und anschließendem Bewegen der Maus kann die Kamera um das Gesicht herumbewegt werden. Das Mausrad oder die Zoom Gesten eines Touchpads steuern den Zoom Faktor. Die veränderte Kameraposition wirkt sich auf die Animationswieder-

gabe aus, da das Rendering aus der aktuellen Kameraposition berechnet wird. Somit ist es möglich, Aufnahmen in beliebigen Blickwinkeln und Zoomstufen durchzuführen.

Erstellen einer Action Unit

Da die AU 23 nicht in der Konfiguration enthalten ist, soll an deren Erstellung der Umgang mit der Konfiguration gezeigt werden. Die AU23 bezieht sich auf das Anspannen der Lippen, wodurch der rote Teil schmaler erscheint. Die Lippen können sich nach innen rollen und fast unsichtbar werden oder bei geöffneten Lippen einen kleinen Vorsprung zeigen, wobei der Hauptvorgang das Straffen der Lippen ist (Ekman et al., 2002, S. 237). Diese Aktion kann auch kleine Falten oberhalb und unterhalb des roten Lippenbereichs sowie Wölbungen unter der unteren Lippe erzeugen und zu einem begrenzten Abflachen oder Runzeln des Kinns führen.

Ekman und Friesen definieren das Scoring folgendermaßen (Ekman et al., 2002, S. 237,238).

23 A: Eine leichte Anspannung der Lippen erkennbar.

23 B: Die Lippen sind leicht angespannt *und* die roten Anteile sind leicht verengt.

23 C: Die Verengungen in 23B haben beide den Wert „marked“ aber unter 23 D.

23 D: Die Verengungen in 23B haben beide den Wert „severe“ aber unter 23 E.

23 E: Die Verengungen in 23B haben beide ihr anatomisches Maximum erreicht.

So kann diese AU durch mehrere Kombinationen von Blendshapes dargestellt werden. Auf Grund der Benennungen der Blendshapes und Ekmans Beschreibung kommen die folgenden infrage.

MouthPress_L und MouthPress_R:
Könnten genutzt werden, um eine Anspannung der Lippen zu simulieren.

MouthLowerDown_L und MouthLowerDown_R:
Durch das Heben und Senken des Meshes um den Mund kann die Ausprägung des Verschmälerns der Lippen unterstützt werden.

MouthPucker:
Das Schürzen der Lippen kann einer unnatürlichen Verbreiterung entgegenwirken.

MouthStretch_L und MouthStretch_R:
Indem sie die Mundwinkel seitlich ziehen wird in dem Mesh auch eine Spannung von Nase und Kinn induziert.

MouthFrown_L und MouthFrown_R:
Stellen eine leichte Einrollbewegung der Lippenkanten dar.

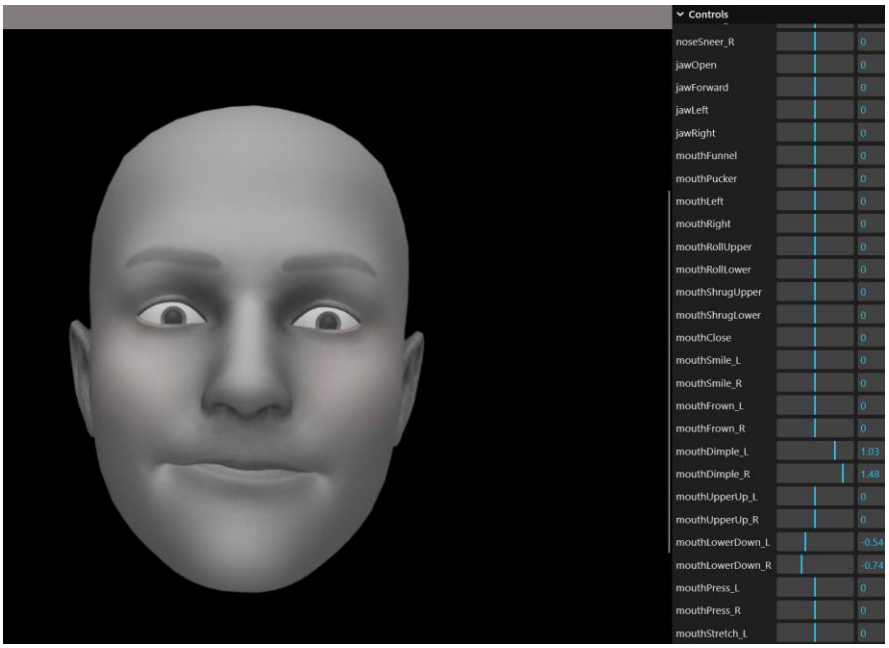
mouthDimple_L und mouthDimple_R:
Könnten die Spannung um die Mundwinkel simulieren.

mouthRollUpper und mouthRollLower:
Könnten verwendet werden, um das Einrollen der Lippen nach innen darzustellen.

Diese Liste ist eine Interpretation des Autors, welcher keine Schulung nach dem FACS besitzt und somit nur eine laienhafte Einschätzung geben kann. Allerdings eignet sich dieses Beispiel, um den Prozess der AU-Festlegung zu demonstrieren und gleichzeitig die Notwendigkeit einer Anpassung der Ausprägungsstärken darzustellen, sowie die Erarbeitung einer Action Unit.

Zunächst wird mittels der Schaltflache „Toggle Blendshapes“ das Blendshape Menü sichtbar gemacht und durch Klick auf das eingeklappte Menü geöffnet.

Abb.13 – Ausprobieren der Blendshapes



Quelle: Eigene Darstellung

Hier kann jetzt ausprobiert werden, wie sich verschiedene Einstellungen der Blendshapes auf das Mesh auswirkt. Sobald die gewünschten Blendshapes für jede Scoring-Stufe identifiziert sind, können diese zu einer AU zusammengefasst werden. In diesem Beispiel fiel die Entscheidung auf die folgenden Einstellungen.

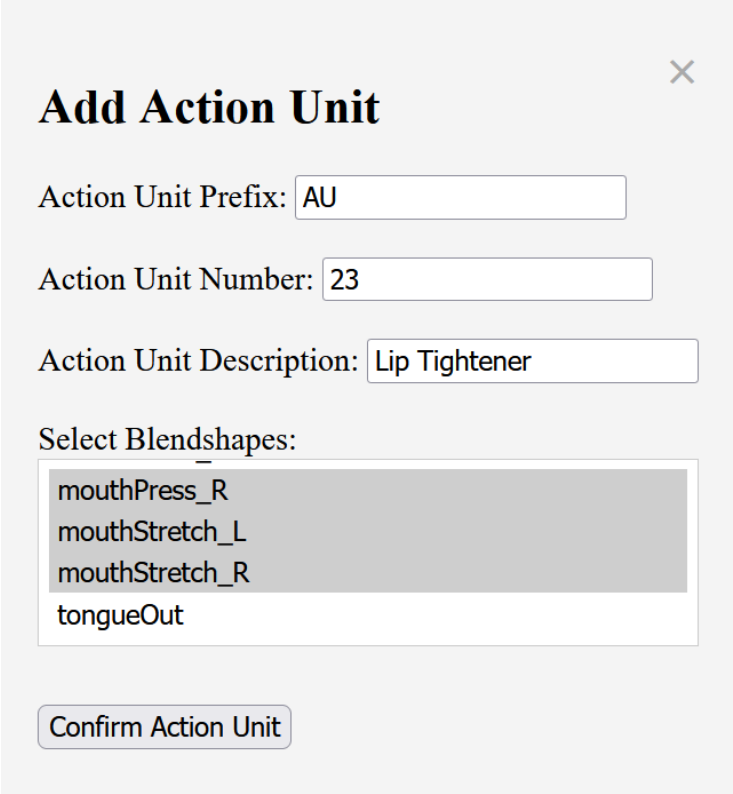
Tab.4 Scoring Werte der AU 23

	A	B	C	D	E
mouthRollUpper	0	0.35	0.35	0.5	0.5
mouthRollLower	0	0.35	0.35	0.5	0.5
mouthClose	0.1	0	0.1	0.1	0
mouthLowerDown_L	0	0	0	0	0.35
mouthLowerDown_R	0	0	0	0	0.35
mouthPress_L	0.15	0	0	0	0.8
mouthPress_R	0.15	0	0	0	0.8
mouthStretch_L	0	0.25	0.25	0.6	1
mouthStretch_R	0	0.25	0.25	0.6	1

Quelle: Eigene Darstellung

Über die Schaltfläche Create New AU öffnet sich der Dialog, um die neue Unit hinzuzufügen. Hier werden nun die Benennung eingetragen und die Blendshapes ausgewählt die an dem Ausdruck mitwirken sollen.

Abb.14 – Hinzufügen einer Action Unit

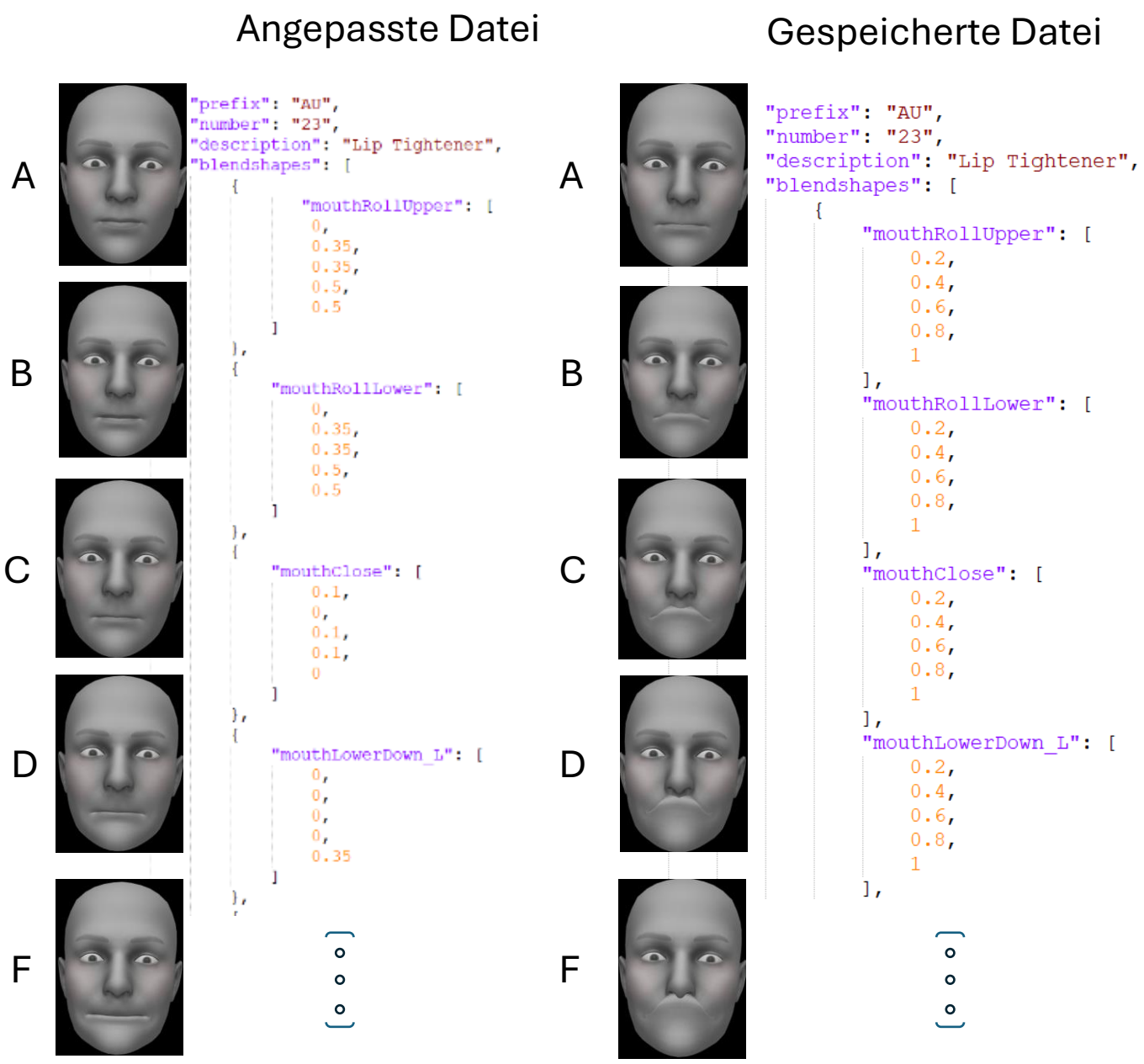


Quelle: Eigene Darstellung

Nach dem Speichern der neuen Action Unit ist diese im Slider Container direkt sichtbar, aber wurde noch nicht in einer Konfigurationsdatei gespeichert. Der neue Eintrag bleibt während der aktuellen Sitzung erhalten, ist aber nicht dauerhaft verfügbar. Es müssen außerdem noch die Einstellungen des Scorings vorgenommen werden, da eine neue Unit mit der Standardverteilung von 0 bis 1, in Schritten von 0.2 angelegt wird. Hierzu wird über die Schaltfläche „Save AU Config“ ein Download Dialog geöffnet und ein Name für die neue Konfiguration vergeben. Die Datei kann jetzt mit einem

beliebigen Texteditor geöffnet und bearbeitet werden. Wie in der folgenden Abbildung ersichtlich, ist das Ergebnis deutlich besser als die lineare Verteilung.

Abb.15 Anpassungen in der JSON-Datei



Quelle: Eigenen Darstellung

Nachdem die angepasste Datei gespeichert wurde, wird sie durch den Button „Load AU Config“ in das Programm geladen. Wenn dabei keine Fehlermeldung erscheint, ist kein Fehler beim Bearbeiten passiert. Sollte es zu einer ungültigen Datei bei der Bearbeitung gekommen sein, kann über die Entwicklerkonsole des Browsers der Output des Validierungsfehlers wie im Implementationsteil beschrieben nachgesehen werden. Es wird auch zusätzlich in einem Dialogfeld abgefragt, ob die Datei trotzdem geladen werden soll. Während Tests der Anwendung wurde festgestellt, dass es zu ungültigen Dateien kommen kann, was auf nicht sichtbare Zeichen in der Kodierung des Editors zurück-

zuführen ist. Daher erscheint es zweckmäßig diese Möglichkeit anzubieten. Für das Debugging ist es hilfreich, um diese Kodierungsfehler zu identifizieren. Die Anforderung A2.17 wird hiermit teilweise erfüllt, da eine Änderung der Scoring Arrays nicht in der Oberfläche möglich ist, in Zukunft aber möglich sein soll.

Expression hinzufügen

Zum Hinzufügen einer Expression verhält es sich ähnlich wie für die AU beschrieben. Nach Öffnen des Dialogfeldes wird ein Name für die zu erstellende Expression ausgewählt und die gewünschten AU aus der Liste ausgewählt. Da in der aktuellen Version nur ein Übergangsbereich von 0 bis 100 Prozent bei linearer Aktivierung der Blendshapes vorgesehen ist, kann die neu erstellte AU 23 zwar aus der Liste ausgewählt werden, besitzt dann aber die Standardverteilung der Blendshape Werte und kann somit nicht wie angepasst genutzt werden. Hier sind die Probleme die distinkte Zuweisung von Werten im FACS und der gewünschte lineare Übergang. Dieses Problem kann in einer zukünftigen Version gelöst werden und wird im Diskussionsteil noch erläutert.

Erstellen von Standbildern

Das Erstellen von Standbildern funktioniert sehr intuitiv und wird nur kurz beschrieben. Sobald ein Ausdruck des Gesichts erstellt wurde, wird durch einen Klick auf die Schaltfläche „Save Image“ ein Download Dialog geöffnet und sofern der Ausdruck über die AU Slider erstellt wurde, ein Vorschlag für den Dateinamen aus dem FACS-Code und dem Anhang „_cropped_image“ angeboten, der durch Bestätigung den Download in das gewählte Verzeichnis startet. Es ist hierbei egal welcher Weg zur Erstellung verwendet wurde. Es kann durch die direkte Steuerung der Blendshapes, die AU Slider oder die Expression Slider entstanden sein, da das Bild den aktuellen Zustand der Canvas abbildet.

Erstellen von Keyframes

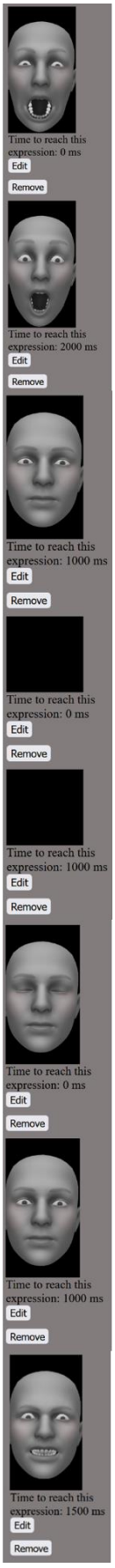
Eine Animation wird bis zu ihrem Abspielen in dem Keyframe Container, der rechts unten in der Anwendung angezeigt wird, als eine Folge von Keyframes gespeichert, die durch ein Thumbnail repräsentiert werden. Zum Hinzufügen eines Keyframes in den Container ist das Vorgehen ähnlich dem für Standbilder. Es ist auch hier egal wie der Ausdruck zustande kam. Durch Drücken der Schaltfläche „Capture Frame“ wird die Konfiguration der Blendshapes, welche auf der Leinwand gerade angezeigt werden, in dem Keyframe gespeichert. Zusätzlich wird der Wert des „Duration“ Textfeldes sowie ein Screenshot des Ausdrucks ebenfalls abgespeichert und in der Keyframe Vorschau angezeigt. Durch Drücken der Schaltfläche „Capture ISI“ wird ein ISI-Frame hinzugefügt, was an dem schwarzen Thumbnail erkennbar ist. Einfügen eines ISI heißt nicht 500ms ISI-Zeitraum, sondern, dass der zuletzt erreichte Ausdruck für die angezeigte Zeit gehalten wird bis auf den schwarzen Bildschirm umgeschaltet wird und die nächste Transitionszeit beginnt. Der schwarze Bildschirm repräsentiert die Zeit, die auf schwarz geschaltet ist, also von einem ISI-Frame zum nächsten und die Zeit von ISI zu Keyframe die Zeit die für die Veränderung von schwarz auf den nächsten Frame. Hierbei ist zu beachten, dass das Gesicht nach dem Verlassen der ISI-Zeit im Zustand neut-

ral, also alle Blendshapes auf 0, konfiguriert ist, Somit muss, wenn ein anderer Zustand gewünscht ist, ein Frame erstellt werden welches direkt auf das ISI folgt, um von hier aus zu starten.

Das erste Frame der Animation wird, unabhängig der eingestellten Zeit, immer auf 0 gesetzt, da es zum ersten Frame keine Transitionszeit geben kann. Die Zeit, in welcher der Übergang zwischen Keyframes interpoliert wird, hat einen direkten Einfluss auf die Geschwindigkeit. Wie in den Grundlagen erläutert, ist die Animation von Blendshapes nur eine Linearkombination von Anfangs- und Zielzustandsübergängen der Geometrie. Deshalb ändert sich nichts an den geometrischen Zuständen aber die Zeit bzw. die Anzahl der interpolierten Zwischenbilder, die die Verformung benötigt. Aus diesem Grund erscheint die Animation in Zeitlupe zu laufen, wenn der Wert zu groß gewählt ist und bei zu kleinem Übergang als Zeitraffer. Das muss bei der Erstellung beachtet werden.

Es wichtig zu verstehen, dass die gesamte Animation für three.js eine durchgehende Bewegung darstellt und nicht aus einzelnen Stücken. Hierfür müssen einzelne Clips erstellt und gespeichert werden, wofür ISI-Frames an Anfang und Ende sinnvoll sind. Es folgt eine Erläuterung mit den Keyframes der Beispielanimation, die auch dem GitHub Repository in dem Ordner Release beiliegen.

Abb. 16 Keyframe Sequenz der Beispielanimation



Die Animation startet mit dem Ausdruck „Fear“ in 100-prozentiger Ausprägung. Da der Ausdruck in dem ersten Frame gespeichert ist wird damit die Animation begonnen und sofort mit der Transition zum zweiten Frame, der eine 100-prozentige Ausprägung der Expression Surprise darstellt, gestartet. Für den Übergang werden 2000ms zwischen den Frames interpoliert, was der eingetragene Wert in dem Durations Textfeld ist, der bei der Erstellung übernommen wurde. Zur besseren Verständlichkeit wurde ein Label hinzugefügt, welches den Nutzer mit „Time to reach this expression“ daran erinnert, dass es sich um Übergänge handelt und die Zeit zwischen zwei Frames durch die Anwendung berechnet wird. Nach der Rückkehr zum neutralen Gesichtsausdruck, in 1000ms, wird ein ISI-Frame eingefügt. Das geschieht durch die Schaltfläche „Capture ISI“ mit der Einstellung der Zeit von 0ms wird der schwarze Bildschirm sofort nach Erreichen des Zielzustandes des vorherigen Frames eingeblendet und bleibt 1000ms sichtbar. Da die Animation mit einem neutralen Gesicht und geschlossen Augen weiterlaufen soll, wurde die AU45 auf Ausprägung E gestellt und die Zeit auf 0ms zur Aufnahme des Keyframe. Dann wird analog dem vorherigen Vorgehen in 1000ms zum neutralen Gesicht übergeblendet und die Animation mündet schließlich in der durch das FACS kodierten Kombination 4E+5E+7C+10C+14C+16C, welche durch eine Aktivierung der Checkbox „Clamp when finished“ beibehalten wird. Durch Anklicken der „Play Animation“ Buttons wird die Wiedergabe gestartet, zum Aufzeichnen muss die Checkbox „Capture Animation“ aktiviert sein. Nach Ende wird dem Nutzer dann ein Downloadfenster seines Browsers präsentiert, in dem der Download bestätigt werden muss. Wenn die Checkbox „Repeat“ aktiviert ist, wird die Animation in einer Endlosschleife wiedergegeben. In diesem Modus ist die Checkbox „Capture Animation“ ohne Funktion, da die Aufnahme von Wiederholungen nicht vorgesehen ist. Das Starten der Wiedergabe durch „Play Animation“ und das Unterbrechen mit „Stop Animation“ ist intuitiv. Ebenso verhält es sich mit der „Remove“ Schaltfläche, welche jedem Keyframe angeheftet ist. Die „Edit“ Schaltfläche bedarf allerdings der Erläuterung, dass der Frame mit dem Anklicken den aktuell eingestellten Zustand der Leinwand und den Wert des Duration Feldes an die gewählte Position einfügt und somit den betreffenden Keyframe an seiner Position ersetzt.

Quelle: Eigene Darstellung

6.1 Ähnliche Anwendungen

Im Rahmen der Recherche zu dieser Arbeit konnte keine Anwendung gefunden werden die dem Nutzer eine direkte Darstellung von FACS kodierten Gesichtsausdrücken und deren Übergang ineinander, im Browser ermöglicht. Für die Erstellung von Animationen als Desktop Anwendung sind insbesondere Blender und Metahumans zu nennen.

Da Blender die OpenGL benutzt, ist grundsätzlich alles, was mit FaceEx modelliert und animiert werden kann auch in Blender möglich, da wie in den Grundlagen beschrieben, WebGL eine Teilmenge von OpenGL ist. Allerdings ist die Einarbeitung in Blender schwierig und langwierig und für das schnelle Erstellen von Animationen weniger geeignet. Blender selbst beschreibt auf seiner Seite die Oberfläche als „einschüchternd“ für Anfänger (Blender, 2024).

Metahumans ist eine Erweiterung der Unreal Engine und ermöglicht das Erstellen photorealistischer Charaktere und integriert Motion Capture, um diese zu bewegen. Die Bedienung und Lernkurve sind aber nochmals anspruchsvoller als bei Blender. Es ist notwendig die Unreal Engine zu installieren und ein EPIC Games Konto zu erstellen, um das Metahumans Plugin zu verwenden und durch Quixel Bridge den erstellten Metahuman aus der Cloud Anwendung MetaHumans Creator herunterzuladen (Epic Games, Inc., n.d.). Zur Feinsteuerung der Gesichtsausdrücke ist ein weiteres Plugin notwendig „MetaHumans Faceial Rig“. Allein dieser Prozess kann so lange dauern wie FaceEx für die Erstellung mehrerer Animationen benötigt.

Eine Nutzung der beiden Anwendungen im Browser ist weder sinnvoll noch beabsichtigt. Es sind Multifunktions-Werkzeuge, die allen Anforderungen an die Erstellung virtueller Inhalte gerecht werden wollen. Sie können umfangreich angepasst und erweitert werden und stellen hohe Anforderungen an die Hardware, um Ergebnisse höchster Qualität zu erzeugen.

Zuletzt sei noch auf die Produkte von iMotions hingewiesen, die auch als Inspiration zu den Konfigurationen benutzt wurden. Sie dienen aber zu Zwecken der Analyse und nicht dem Erstellen definierter FACS Animationen (iMotions, n.d.).

7 Praktischer Einsatz

Wie bereits in der Anforderunganalyse dargestellt, ist als Primärer Einsatzbereich des Prototyps die Untersuchung von Adaptionseffekten an präsentierte Stimuli. Dieses Anwendungsfeld, repräsentiert durch die exemplarische Studie in Abschnitt 3, wird durch die Anwendung erfüllt. Es ist möglich ein Set von definierten Emotionsabbildungen als Bilder zu erzeugen und diese zu speichern. Durch die Anzeige der Kodierung und einer angepassten Konfigurationsdatei ist es möglich, diese Bilder vergleichbar und reproduzierbar zu verwenden. Die Animationsteuerung bietet die Möglichkeit prozentuale Übergänge auf definierten Blendshapes, repräsentiert durch ihre AU, darzustellen. Ebenfalls kann von einem FACS kodierten Zustand in einen anderen übergeblendet werden. Somit ist es möglich, ein Experiment zur Untersuchung von Adaptation Aftereffects durchzuführen. Hierzu können beispielsweise Ausprägungen von Emotionen durch die Scoring Skala erstellt werden die Pro-

banden an einer Stufe adaptiert werden und danach Bilder präsentiert werden, die eine andere Scoring Stufe anzeigen. Somit wären Experimente denkbar die eine Sequenz von Veränderungen der Ausprägung präsentieren und Rückschlüsse aus der Adaptionsstärke in den unterschiedlichen Stufenkombinationen messen.

Es greifen aber viele Einsatzfelder auf die Nutzung von standardisierten Gesichtsausdrücken zurück. Beispielsweise werden in der Autismus-Forschung Bilder zur Erkennung und Imitation von Emotionen genutzt. Beispielsweise wird in der Studie von Drimalla et.al. das Erkennen und das imitieren von Emotionen bei Menschen mit Autismus untersucht (Drimalla et al., 2021). Hierzu wurden Bilder von Schauspielern präsentiert, die eine der sechs Basisemotionen darstellen, welche anschließend imitiert werden sollten und die Qualität der Imitation mittels OpenPose anhand der FACS Action Units quantifiziert wurden. Die Autoren schlussfolgerten das Menschen mit Autismus von einem Training zur Imitation von Emotionen, die spontane Nachahmung im sozialen Kontext verbessern und dadurch die Emotionserkennung fördern könnte (Drimalla et al., 2021, S. 13). Hier könnte die Anwendung als einfache kostenlose Trainingsmöglichkeit genutzt werden. Es existiert in three.js die Möglichkeit eine Webcam auf das Media Pipe Framework von Google zu mappen und damit die Auswertung per Webcam zu ermöglichen. Somit wäre eine Anpassung und damit ein Trainingstool für betroffene Personen denkbar.

8. Diskussion

Es konnte in dieser Arbeit gezeigt werden, dass es möglich ist, ein leicht zu bedienendes und ressourcenschonendes Werkzeug zu erstellen, um Animationen und statische Bilder von Gesichtsausdrücken für Forschung und Lehre zu erzeugen. Hierbei wurde die Kodierung des FACS zur Steuerung der Ausdrucksstärke sowie der Definition der Bewegungen in den anvisierten Teilbereichen erfolgreich umgesetzt. Es wurden Vorüberlegungen ausgearbeitet weitere Bereiche des FACS in das Interface der Ausdrucksmodellierung zu integrieren. Der Quellcode ist öffentlich einsehbar und durch die verwendete MIT Lizenz und die Lizensierung der beteiligten Frameworks kann der Programmcode ohne jede Einschränkung genutzt werden. Somit erfüllt die Anwendung das eingangs gesetzte Ziel des Open Access. Die Nutzerführung ist übersichtlich strukturiert und bereits so gestaltet, dass das Risiko für Fehleingaben durch Nutzer sehr gering ist. Dem Nutzer ist es leicht möglich den Überblick in der Anwendung zu behalten. Dies wurde erreicht indem nur eine Oberfläche ohne tiefere Menüstrukturen zur Interaktion verwendet und eine Validierung der Konfiguration erstellt wurde. Trotzdem wird eine hohe Flexibilität bei der Nutzung geboten da jede beliebige Kombination von Blendshapes zu einer AU zusammengefasst, angepasst und verwendet werden kann. Für einfache Basisanwendungsfälle die keine tiefere Darstellung des FACS, wie die Miscellaneous Actions benötigen ist durch die Anfangskonfiguration eine Anwendung entstanden, die nach dem Laden der Seite sofort benutzt werden kann. Durch die Anzeige der Beschreibungen der AU, wie in den Grundlagen beschrieben von Ekman und Friesen zwar nicht gewünscht, aber für Anfänger nützlich ist, kann auch ohne Kenntnisse des FACS mit der Arbeit begonnen werden. Die Expressions als Repräs-

sentation der sechs Grundemotionen nach Ekman und Friesen stellen eine einfache Abbildung dar, die auch für einfache Anwendungen ausreichend erscheint. Der verwendete Rechner zur Erprobung der Anwendung wird als guter Durchschnittsrechner angenommen. Somit ist FaceEx für den durchschnittlichen Nutzer in guter Performanz nutzbar. Der Ressourcenaufwand für einen Server ist als äußerst gering einzustufen da die Release Version kleiner als 2MB ist und keine Serverseitige Rechenleistung benötigt wird. Somit kann diese Anwendung auf jeder Hochschul-Website platziert werden, um sie den Studenten zur Verfügung zu stellen. Sollte durch das Hinzufügen photorealistischer Texturen und einer höheren Polygon-Anzahl die Größe ansteigen, ist die Bereitstellung weiterhin möglich. Allerdings könnten die Hardware des Nutzers und die Möglichkeiten des WebGL Standards hier nicht mehr ausreichen. Ein Interface, das als Basis die Kodierung des FACS nutzt hat Vorteile gegenüber Blendshape basierter Steuerung. So ist ein Blendshape Set ein individuelles System, das zwar oft durch das FACS inspiriert aber eben doch ein auf die Anwendung und den Annahmen des Designers beschränktes System. Der große Vorteil liegt in der Vergleichbarkeit des FACS. Animationen und die Bewegungen eines Face Mesh können auf vielfältige Weisen erstellt werden und es ist schwierig die Vergleichbarkeit unter den Systemen herzustellen. In FaceEx ist es durch das Mapping der Blendshapes des jeweiligen Modells möglich, sich auf die vergleichbare Bewertung des FACS auch mit unterschiedlichen Meshes, Texturen und Blendshape Sets, auf ein System zur Kategorisierung zu verständigen. Das ist ein Vorgehen welches dem eingangs erwähnten Gedankens des FACS als universelles messbares und wissenschaftlich vergleichbares System entspricht. Motion Capture Anwendungen besitzen eine höhere Qualität, vor allem wenn sie durch geübte Schauspieler Emotionen darstellen. Allerdings ist das ein nicht zu unterschätzender Aufwand, da die erstellten Daten entweder einen vorher festgelegten Verlauf haben, der dann aufgezeichnet wird, oder eine Nachbearbeitung notwendig ist, in welcher ein Computermodell zur Erstellung der Morph Animationen genutzt wird. Hier liegt der Vorteil der Anwendung eindeutig der möglichen Umsetzungsgeschwindigkeit.

Dennoch ist das FACS kein vollständig vergleichbares System, da viele individuelle Interpretation in den AU liegen. Allerdings sind die Anweisungen des Scorings sowie zur Reproduktion wie beispielsweise das Verschwinden des roten sichtbaren Teils der Lippen zum Übergang in die nächste Stufe, schon ein hoher Präzisionsgrad. Dies wurde durch die Erstellung der AU 23 gezeigt. Es ergibt sich durch das unmenschliche Aussehen des aktuellen Modells sowie der steif und symmetrisch wirkenden Verformung ein künstlicher Eindruck. Während der Weg über die Darstellung eines trainierten Menschen als Gesamtresultat gesehen werden kann, in dem eine Emotion natürlich in die andere überblendet, ist die Anwendung auf den Übergang distinkter Zustände ausgelegt. Das hat zur Folge, dass durch Ausprobieren eine flüssige Bewegung zwischen unterschiedlichen Keyframes bestimmt werden muss. Wenn ein Set an Emotionen durch einen Menschen einmal aufgenommen und gelabelt ist, kann es auch ganz oder in Teilsequenzen wiederverwendet werden. Damit können bereits erstellte Medien öffentlich zugänglich gemacht werden, was sie für viele unterschiedliche Exprimente verfügbar macht. Somit ist die Flexibilität von FaceEx oft nicht notwendig und es können natürliche

Aufnahmen verwendet werden, wie es in vielen Studien auch der Fall ist. JS und auch three.js weisen, aber auch technische Limitationen auf. So kann zwar eine Animation, die in Blender oder Unreal erstellt wurde, durch three.js im Webbrowser wiedergegeben werden, allerdings ist der WebGL Standard nicht dafür ausgelegt photorealistische Anwendungen in Echtzeit zu rendern. Dafür haben die Anwendungen meist Lösungen durch welche die Grafikkarte entsprechend genutzt werden kann. Ebenso erfolgt die Aufzeichnung der Animation nicht in Echtzeit. Three.js unterstützt die OpenGL Shading Language, mit der eine effizientere Nutzung der Hardware möglich ist. Deren Anwendung überschreitet allerdings den Umfang dieser Arbeit. Es müsste in diesem Zusammenhang auch über eine Berechnung auf Serverseite nachgedacht werden, um die Nutzerhardware zu entlasten. In der entwickelten Anwendung ist die Erstellung von Animationen, als Aufzeichnung einer Canvas während der Widergabe realisiert. Somit ist es nicht einfach möglich, eine Animation mit definierter FPS-Zahl zu erstellen. Im Gegensatz zu beispielsweise Blender, wo das Rendering nicht in Echtzeit durchgeführt wird, ist FaceEx darauf angewiesen, dass der Rechner des Anwenders in der Lage ist mit einer ausreichenden Framezahl zu rendern, um eine flüssige Animation abzuspeichern. Im Rahmen der Arbeit konnte die komfortable Konfiguration der AU und Emotionen nicht vollständig umgesetzt werden. Das Scoring Array kann zwar vollständig angepasst werden, allerdings ist dafür eine direkte Interaktion mit der Konfigurationsdatei notwendig. Dies kann den Nutzer abschrecken und zu Fehlern führen. Diese werden zwar festgestellt und rückgemeldet, allerdings ist das nicht im Sinne der intuitiven Nutzung.

Da der aktuelle Zustand der Software nur einen Schritt in einem Entwicklungsprozess darstellt, soll noch auf die Weiterentwicklungsmöglichkeiten eingegangen werden. Das Ziel der vollständigen Abbildung des FACS wurde offensichtlich nicht erreicht, war aber im Rahmen dieser Arbeit auch nicht möglich und gewollt. Es sollen noch laterale und asymmetrische Bewertung von Ausdrücken ermöglicht werden. Hierzu kann das Scoring Array mit seinen fünf Stufen für jedes Blendshape genutzt werden. Bei Blendshapes, die einen linken und einen rechten Bestandteil haben, kann direkt die laterale Ausprägung eingestellt werden, indem ein Blendshape an einer Scoring Stufe einen hohen Wert und das korrespondierende eine entsprechend niedrigeren hat. Da in einer AU auch mehrere Blendshapes verwendet werden können, ist es auch möglich nicht beidseitige Blendshapes durch Hinzufügen von entgegenwirkender Aktivierung zu modellieren. Beispielsweise kann die Stirnpartie als gesamtes verformt und durch die Blendshapes des Auges, welche bilateral sind, an einer Seite eine entgegengesetzte Bewegung eingestellt werden. Für die asymmetrische Bewertung, welche durch einen Höhepunkt der Bewegung, einen Apex besitzt ist vorgesehen den Apex auf die Scoring Stufe E zu legen und somit das Anschwellen in dem von Ekman und Friesen empfohlenen Ausprägungzeitraum zu definieren. Hier kann durch das einfache Ausprobieren der ideale Zeitpunkt bestimmt werden und die Animation so mit sehr kleinen Schritten zu diesem Apex geführt werden. Schwieriger wird die Modellierung der verdeckten Bereiche, welche eine präfixfreie Kodierung haben. Hier könnte beispielsweise eine schwarze Ebene vor die Kamera gelegt werden, die nur den entsprechenden Bereich als sichtbar anzeigt. Die Kodierung könnte ähnlich dem ISI-Array mit einem

Wert gesteuert werden, der keine sinnvolle Darstellung repräsentiert. Beispielsweise könnten die verdeckten Bereiche mit 10 kodiert werden und dann hierauf wie bereits in der Erstellung der Animation umgesetzt reagiert und die Ebene eingefügt werden. Hier ist das *THREE.Object3d* sehr nützlich da Blendshapes als räumliche Koordinaten vorliegen, die die Position der Ebene steuern können, insofern sie auf einer Geraden im Projektionsfeld der Kamera liegen. Das Problem des unnatürlichen Aussehens, kann ebenfalls mittels der vielfältigen Möglichkeiten von three.js gelöst werden, indem eine photorealistische Textur geladen werden kann. Hierzu ist entweder eine KTX2 kodierte Textur zu verwenden oder mittels einer Anpassung der Funktion zum Laden des Models andere Texturen einzusetzen. Hierdurch wäre auch die Darstellung von Falten und Wölbungen der Haut möglich, welche durch das Morphen der Targets mitbewegt werden. Auf Grund der Flexibilität der Anwendung, welche nicht auf die Anzahl oder Namen der Blendshapes beschränkt sein soll, könnte auch mittels eines neuen Modells mit geringen Anpassungen die Anzahl der Blendshapes geändert werden. Ein Aspekt, der in der Entwicklung nicht berücksichtigt wurde, aber wünschenswert wäre, ist die Generierung von fertigen HTML-Seiten für Experimente. Im gegenwärtigen Zustand müssen die erstellten Medien noch in eine Form überführt werden, in welcher sie den Testpersonen präsentiert werden können. Hierzu bietet sich die Erzeugung einer HTML-Seite an, die auch einen Fragebogen enthält und über eine Datenstruktur zur Speicherung der Antworten verfügt. Ein responsives Design ist ein wichtiges Ziel, das in einer Weiterentwicklung unbedingt bedacht werden sollte. Da Nutzer an den unterschiedlichsten Endgeräten arbeiten, ist ein optisch Ansprechendes Design auf unterschiedlichen Bildschirmgrößen und Eingabemethoden wünschenswert.

Zusammenfassend ist festzuhalten, dass das Ziel der Arbeit erfüllt werden konnte und wichtige Erkenntnisse für zukünftige Entwicklungen zu diesem Thema gewonnen werden konnten.

Literaturverzeichnis

Ackermann, P. (2023). Fullstack-Entwicklung: Das Handbuch für Webentwickler (2., aktualisierte und erweiterte Auflage). Rheinwerk Verlag.

Anjyo, K. (2016). Blendshape Facial Animation. In B. Müller, S. I. Wolf, G.-P. Brueggemann, Z. Deng, A. McIntosh, F. Miller, & W. S. Selbie (Hrsg.), Handbook of Human Motion (S. 1–11). Springer International Publishing. https://doi.org/10.1007/978-3-319-30808-1_2-1

Apple Inc. (n.d.-a). ARFaceAnchor.BlendShapeLocation. Apple Developer Documentation. https://developer.apple.com/documentation/arkit/arfaceanchor/blendshapelocation

Apple Inc. (n. d.-b). ARKit. https://developer.apple.com/documentation/arkit

Aumüller, G., Aust, G., Conrad, A., Engele, J., Kirsch, J., Maio, G., Mayerhofer, A., Mense, S., Reißig, D., Salvetter, J., Schmidt, W., Schmitz, F., Schulte, E., Spanel-Borowski, K., Wennemuth, G., Wolff, W., Wurzinger, L. J., & Zilch, H.-G. (2020). Duale Reihe Anatomie (5. Aufl., S. b-007-170976). Georg Thieme Verlag. https://doi.org/10.1055/b-007-170976

Blender. (2024, Januar 15). Einführung—Blender Manual. https://docs.blender.org/manual/de/4.0/getting_started/about/introduction.html#key-features

Bradley, S. (n. d.). Module Specifiers Versus Relative Import References—Three.js Tutorials. https://sbcode.net/threejs/module-specifiers/

Bryn Farnsworth. (2022, Oktober 18). Facial Action Coding System (FACS)—A Visual Guidebook—iMotions. https://imotions.com/blog/learning/research-fundamentals/facial-action-coding-system/

Bühler, P., Schlaich, P., & Sinner, D. (2018). Webtechnologien: JavaScript – PHP – Datenbank. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-54730-4

Bühler, P., Schlaich, P., & Sinner, D. (2023). HTML und CSS: Semantik - Design - Responsive Layouts. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-66663-0

De La Rosa, S., Giese, M., Bulthoff, H. H., & Curio, C. (2013). The contribution of different cues of facial movement to the emotional facial expression adaptation aftereffect. Journal of Vision, 13(1), 23–23. https://doi.org/10.1167/13.1.23

Drimalla, H., Baskow, I., Behnia, B., Roepke, S., & Dziobek, I. (2021). Imitation and recognition of facial emotions in autism: A computer vision approach. Molecular Autism, 12(1), 27. https://doi.org/10.1186/s13229-021-00430-0

Ekman, P., & Friesen, W. V. (1975). Unmasking the Face: A Guide to Recognizing Emotions From Facial Expressions. https://api.semanticscholar.org/CorpusID:141285976

Ekman, P., & Friesen, W. V. (1976). Measuring facial movement. Environmental Psychology and Nonverbal Behavior, 1(1), 56–75. https://doi.org/10.1007/BF01115465

Ekman, P., Friesen, W. V., & Hager, J. C. (2002). Facial action coding system: The manual. Research Nexus.

Epic Games, Inc. (n. d.). Wie erschaffen Sie digitale Menschen? Unreal Engine. https://www.unrealengine.com/de/explainers/digital-humans/how-do-you-create-a-digital-human

Ghayoumi, M., & Bansal, A. K. (n. d.). Unifying Geometric Features and Facial Action Units for Improved Performance of Facial Expression Analysis. Signal Processing. https://doi.org/10.48550/arXiv.1606.00822

git-scm.com. (n. d.). Git. https://git-scm.com/

Gouraud, H. (1971). Computer display of curved surfaces. The University of Utah.

Haverbeke, M. (2024). Eloquent JavaScript (4. Aufl.).No Starch Press. https://eloquentjavascript.net/Eloquent_JavaScript.pdf

iMotions. (n. d.). Facial Expression Analysis—Emotion Detection Software. https://imotions.com/products/imotions-lab/modules/fea-facial-expression-analysis/

Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., & Deng, Z. (2014). Practice and Theory of Blendshape Facial Models. Eurographics 2014 - State of the Art Reports, 20 pages. https://doi.org/10.2312/EGST.20141042

MDN. (2023a, November 29). Introduction to the DOM. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

MDN. (2023b, Dezember 3). Template literals (Template strings)—JavaScript | MDN. Template Literals. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

MDN. (2024a, Januar 15). Using the MediaStream Recording API - Web APIs | MDN. Using the MediaStream Recording API. https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API/Using_the_MediaStream_Recording_API

MDN. (2024b, Mai 2).The External Resource Link element - HTML: HyperText Markup Language. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link

MDN. (2024c, Mai 31). <script type="importmap"> - HTML: HyperText Markup Language | MDN. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script/type/importmap

meshoptimizer. (n. d.). Meshoptimizer, http://meshoptimizer.org/

Microsoft. (n. d.). IntelliSense in Visual Studio Code. https://code.visualstudio.com/docs/editor/intellisense

Nguyen, P. (2023, October 31). Create resizable split views—Phuoc Nguyen. https://phuoc.ng/collection/reactdrag-drop/create-resizable-split-views/

Nischwitz, A., Fischer, M., Haberäcker, P., & Socher, G. (2019). Computergrafik: Band I des Standardwerks Computergrafik und Bildverarbeitung. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-25384-4

Oliver, M. M., & Amengual Alcover, E. (2020). UIBVFED: Virtual facial expression dataset. PLOS ONE, 15(4), e0231266. https://doi.org/10.1371/journal.pone.0231266

OWASP Foundation. (2021). OWASP Top Ten. https://owasp.org/www-project-top-ten/

Palumbo, R., D'Ascenzo, S., & Tommasi, L. (2017). Editorial: High-Level Adaptation and Aftereffects. Frontiers in Psychology, 8. https://doi.org/10.3389/fpsyg.2017.00217

Parent, R. (2012). Computer animation: Algorithms and techniques (Third edition). Elsevier, Morgan Kaufmann is an imprint of Elsevier.

Parke, F. I. (1972). Computer generated animation of faces. Proceedings of the ACM Annual Conference on - ACM'72, 1, 451. https://doi.org/10.1145/800193.569955

Rupp, C. (2021). Requirements-Engineering und -Management: Das Handbuch für Anforderungen in jeder Situation (7., aktualisierte und erweiterte Auflage). Hanser.

The Khronos Group Inc. (2021, Oktober 11). glTF™ 2.0 Specification. glTF™ 2.0 Specification. https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html

The Khronos Group Inc. (2022, Mai 5). OpenGL 4.6 (Core Profile)—May 5, 2022. https://registry.khronos.org/OpenGL/specs/gl/glspec46.core.pdf

The Khronos Group Inc. (2024, Februar 20). KTX File Format Specification. https://registry.khronos.org/KTX/specs/2.0/ktxspec.v2.html

three.js. (n. d.).Animation system. https://threejs.org/docs/#manual/en/introduction/Animation-system

three.js. (n. d.-b). Fundamentals. Fundamentals. https://threejs.org/manual/en/fundamentals.html

three.js. (n. d.-c). Loader – three.js docs. https://threejs.org/docs/#api/en/loaders/Loader

three.js. (n. d.-d). Object3D – three.js docs. https://threejs.org/docs/#api/en/core/Object3D

three.js. (n. d.-e). Scene. https://threejs.org/docs/?q=scene#api/en/scenes/Scene

three.js. (n. d.-f). WebGLRenderer. https://threejs.org/docs/#api/en/renderers/WebGLRenderer.domElement

three.js (2024).three.js. https://github.com/mrdoob/three.js/blob/dev/LICENSE

Unicode, Inc. (2017, Juli 24). What is Unicode?
https://www.unicode.org/standard/WhatIsUnicode.html

W3Schools. (n. d.). Window requestAnimationFrame() Method.
https://www.w3schools.com/jsref/met_win_requestanimationframe.asp

WebGL Public Wiki. (2016, Februar 22). WebGL and OpenGL - WebGL Public Wiki.
https://www.khronos.org/webgl/wiki/WebGL_and_OpenGL

Webpack Concepts. (n. d.). Webpack. https://webpack.js.org/concepts/

WHATWG. (n. d.). Fetch Standard. https://fetch.spec.whatwg.org/

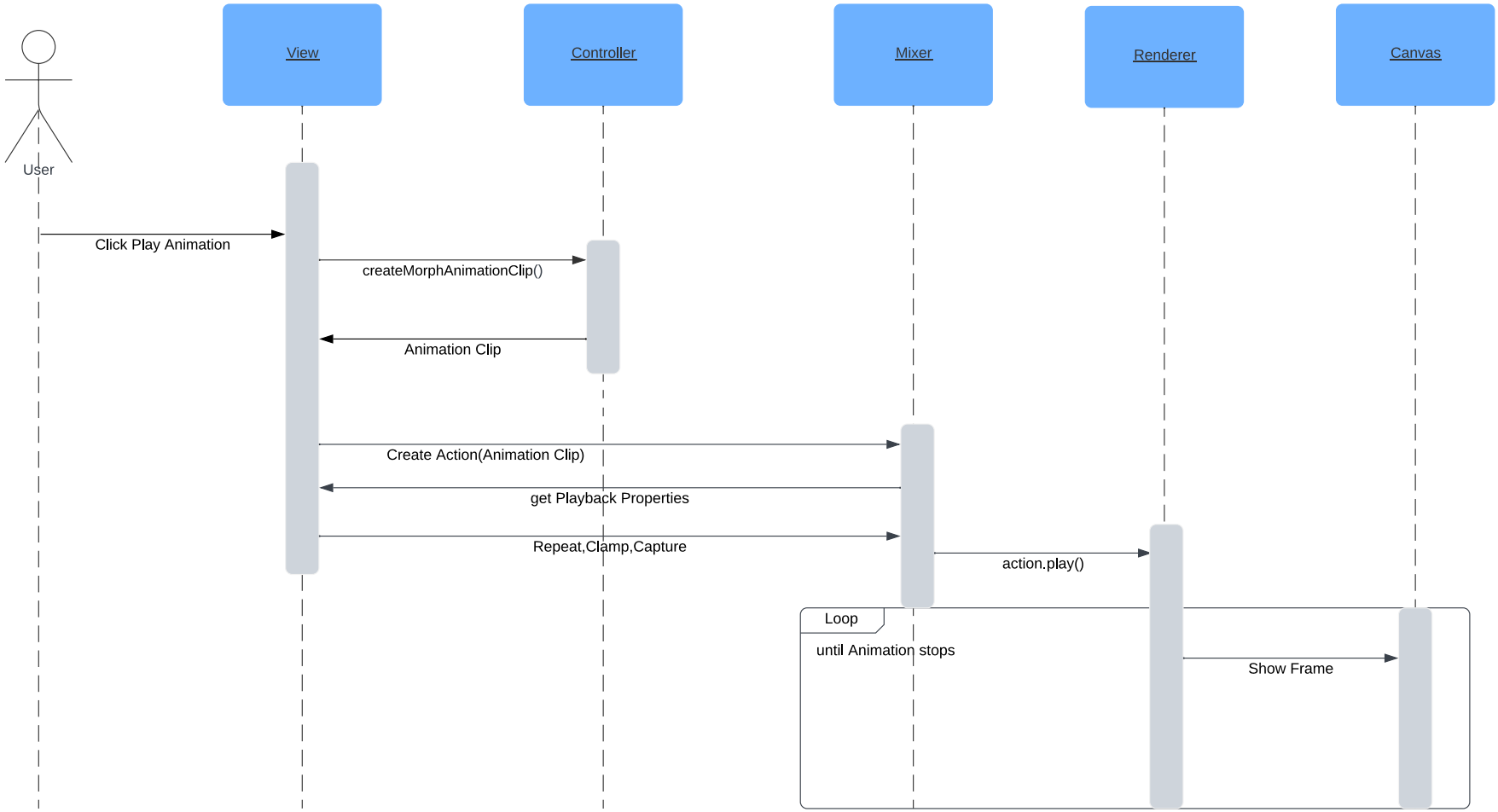
WHATWG. (2024, Juni 21). HTML Standard.
https://html.spec.whatwg.org/multipage/#toc-introduction

Anhangsverzeichnis

Anhang

Anhang A: Sequenzdiagramm zum Abspielen von Animation

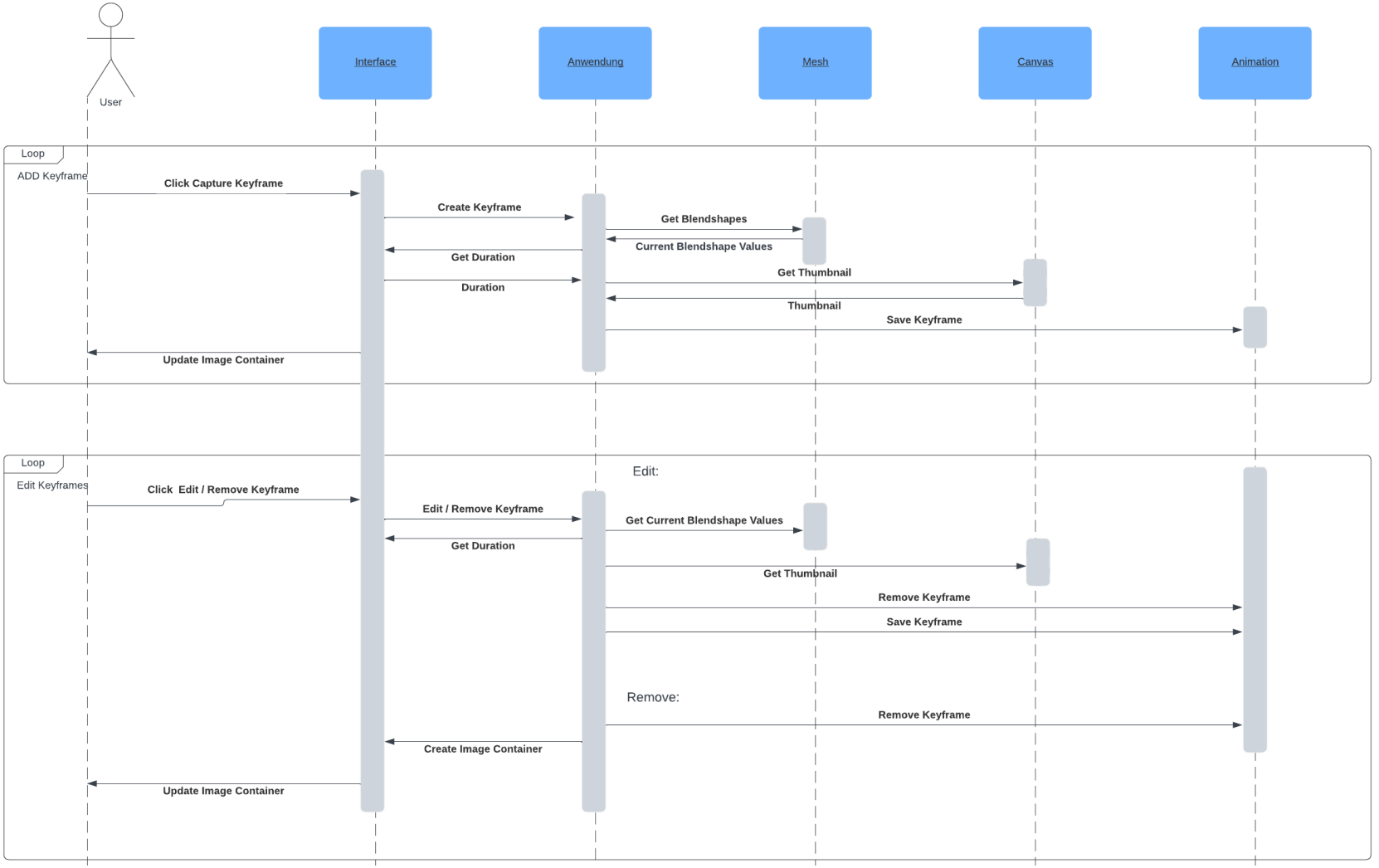
Abb.17 Sequenzdiagramm zum Abspielen von Animation



Quelle: Eigene Darstellung

Anhang B: Sequenzdiagramm der Operation des Keyframe Arrays

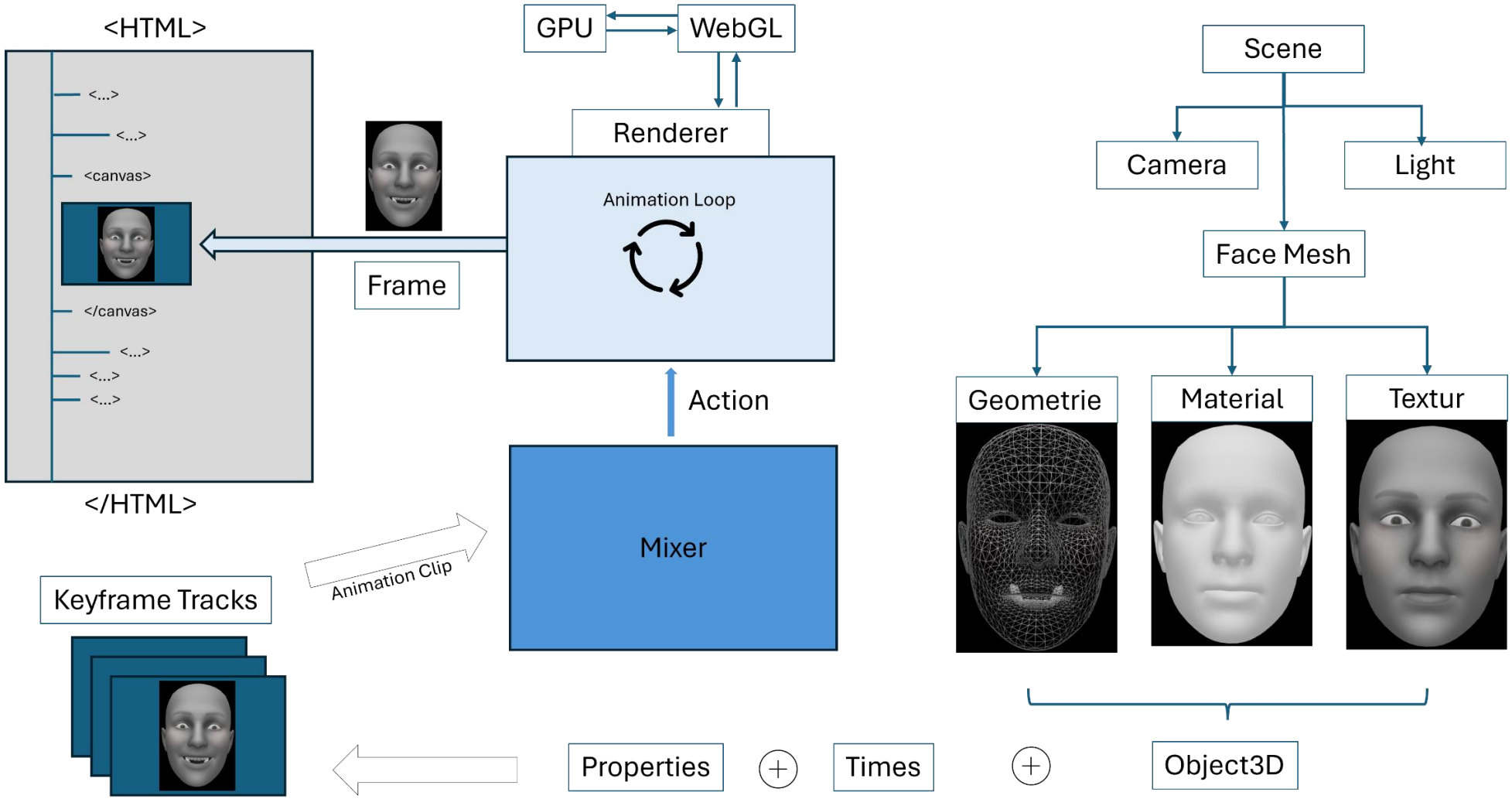
Abb.18 Sequenzdiagramm der Operation des Keyframe Arrays



Quelle: Eigene Darstellung

Anhang C: Übersicht Animationssystem three.js

Abb.19 Übersicht Animationssystem three.js



Quelle: Eigene Darstellung

Anhang D: Zuordnung der Blendshapes

Tab. 1 Zuordnung der Blendshapes

FACS Manual 2002		FaceEx		UIBVFED	iMotions	
Action Unit	Beschreibung	Action Unit / Action Descriptor	Blendshapes	AU Nummer	Action Unit	Beschreibung
1	Inner Brow Raiser	AU1	browInnerUp	-	1	Inner Brow Raiser
2	Outer Brow Raiser	AU2	browOuterUp_L, browOuterUp_R	2	2	Outer Brow Raiser (unilateral, right side)
4	Brow Lowerer	AU4	browDown_L, browDown_R	-	4	Brow Lowerer
5	Upper Lid Raiser	AU5	eyeWide_L, eyeWide_R	5	5	Upper Lid Raiser
6	Cheek Raiser	AU6	cheekSquint_L, cheekSquint_R	-	6	Cheek Raiser
7	Lid Tightener	AU7	eyeSquint_L, eyeSquint_R	-	7	Lid Tightener
9	Nose Wrinkler	AU9	noseSneer_L, noseSneer_R	-	9 (also shows slight AU4 and AU10)	Nose Wrinkler
10	Upper Lid Raiser	AU10	mouthUpperUp_L, mouthUpperUp_R	-	10 (also shows slight AU25)	Upper Lip Raiser
11	Nasolabial Fold Deepener	-	-	11	11	Nasolabial Deepener
12	Lip Corner Puller	AU12	mouthSmile_L, mouthSmile_R	12	12	Lip Corner Puller
13	Cheek Puffer	-	-	-	13	Cheek Puffer
14	Dimpler	AU14	mouthDimple_L, mouthDimple_R	14	14	Dimpler
15	Lip Corner Depressor	AU15	mouthFrown_L, mouthFrown_R	-	15	Lip Corner Depressor
16	Lower Lip Depressor	AU16	mouthLowerDown_L, mouthLowerDown_R	16	16 (with AU25)	Lower Lip Depressor
17	Chin Raiser	AU17	mouthShrugUpper, mouthShrugLower	17	17	Chin Raiser
18	Lip Puckerer	AU18	mouthPucker	18	18 (with slight AU22 and AU25)	Lip Puckerer
20	Lip Stretcher	AU20	mouthStretch_L, mouthStretch_R	-	20	Lip stretcher
22	Lip Funneler	AU22	mouthFunnel	22	22 (with AU25)	Lip Funneler
23	Lip Tightener	-	-	-	23	Lip Tightener
24	Lip Pressor	AU24	mouthPress_L, mouthPress_R	-	24	Lip Pressor
25	Lips Part	-	-	-	25	Lips part
26	Jaw Drop	AU26	jawOpen	-	26 (with AU25)	Jaw Drop
27	Mouth Stretch	-	-	27	27	Mouth Stretch
28	Lip Suck	AU28	mouthRollUpper, mouthRollLower	-	28 (with AU26)	Lip Suck
AD 34	Puff	AD34	cheekPuff	-	-	-
in aktueller Version nicht enthalten		AU44	eyeSquint_L, eyeSquint_R	44	44	Squint
45	Blink - Optional	AU45	eyeBlink_L, eyeBlink_R	-	45	Blink

Quelle: Eigene Darstellung nach (Bryn Farnsworth, 2022; Ekman et al., 2002; Oliver & Amengual Alcover, 2020)

Anhang E: Zuordnung der AU zu den Emotionen

Tab. 2 Zuordnung der AU zu den Emotionen

Emotion	iMotions	Bansal / Ghayoumi	FaceEx
Surprise	1 + 2 + 5 + 26	1, 2, 5,15,16, 20, 26	1 + 2 + 5 + 26
Fear	1 + 2 + 4 + 5 + 7 + 20 + 26	1, 2, 4, 5,15,20, 26	1 + 2 + 4 + 5 + 7 + 20 + 26
Disgust	9 + 15 + 16	2, 4, 9, 15, 17	2 + 9 + 15 + 16
Anger	4 + 5 + 7 + 23	2, 4, 7, 9,10, 20, 26	4 + 5 + 7
Happiness	6 + 12	1, 6,12,14	1 + 6 + 12 + 14
Sadness	1 + 4 + 15	1, 4,15, 23	1 + 4 + 15

Quelle: Eigene Darstellung nach (Bryn Farnsworth, 2022; Ghayoumi & Bansal, n.d. S.260)

Anhang F: Anforderungen

Tab. 3 Anforderungen

Funktionale Anforderungen		Status
Selbsttätige Systemaktivitäten		
A 1.1	Das System muss ein Mesh laden können.	erfüllt
A 1.2	Das System muss Abhängigkeiten nachladen können.	erfüllt
A 1.3	Das System wird unterschiedliche Meshes laden können.	geplant
Benutzungsinteraktionen		
A 2.1	Das System muss Nutzern die Möglichkeit bieten, Keyframes mit Transitionszeiten zu erstellen	erfüllt
A 2.2	Das System muss Nutzern die Möglichkeit bieten, Inter-Stimuli Intervalle einzufügen	erfüllt
A 2.3	Das System muss Nutzern die Möglichkeit bieten, Animationen als Videodatei zu speichern.	erfüllt
A 2.4	Das System muss Nutzern die Möglichkeit bieten, Animationen abzuspielen.	erfüllt
A 2.5	Das System muss Nutzern die Möglichkeit bieten, den aktuellen Gesichtsausdruck als FACS-Kodierung anzuzeigen.	erfüllt
A 2.6	Das System muss Nutzern die Möglichkeit bieten, das Mesh in einen neutralen Zustand zurückzusetzen.	teilweise erfüllt
A 2.7	Das System muss Nutzern die Möglichkeit bieten, den aktuellen Gesichtsausdruck als Bild zu speichern.	erfüllt
A 2.8	Das System muss Nutzern die Möglichkeit bieten, Konfigurationen zu speichern.	erfüllt
A 2.9	Das System muss Nutzern die Möglichkeit bieten, Konfigurationen zu laden.	erfüllt
A 2.10	Das System muss Nutzern die Fähigkeit bieten, AUs hinzuzufügen.	erfüllt
A 2.11	Das System muss Nutzern die Fähigkeit bieten, Expressions hinzuzufügen.	erfüllt
A 2.12	Das System sollte Nutzern die Möglichkeit bieten, Animationen zu speichern.	nicht erfüllt
A 2.13	Das System sollte Nutzern die Möglichkeit bieten, die Ausprägung der Action Units einzustellen.	teilweise erfüllt
A 2.14	Die Oberfläche sollte eine responsives Design haben	teilweise erfüllt
A 2.15	Das System sollte Nutzern die Möglichkeit bieten, Animationen zu laden.	nicht erfüllt
A 2.16	Das System sollte Nutzern die Möglichkeit bieten, einzelne Keyframes der Animationen und deren Zeitwerte zu ändern.	erfüllt
A 2.17	Das System sollte Nutzern die Fähigkeit bieten, die Ausprägungsstärke der AUs zu bearbeiten.	teilweise erfüllt
A 2.18	Das System sollte Nutzern die Möglichkeit bieten, Animationen in einer Endlosschleife abzuspielen	erfüllt
A 2.19	Das System sollte Nutzern die Möglichkeit bieten, den aktuellen Gesichtsausdruck aus unterschiedlichen Blickwinkeln anzuzeigen.	erfüllt
A 2.20	Das System sollte Nutzern die Möglichkeit bieten, die Animation aus unterschiedlichen Blickwinkeln aufzuzeichnen.	erfüllt
A 2.21	Das System sollte Nutzern die Möglichkeit bieten, den Gesichtsausdruck eines Keyframes im Vorschaufenster anzuzeigen.	nicht erfüllt
A 2.22	Das System sollte Nutzern die Möglichkeit bieten, das Gesicht heran- und herauszuzoomen	erfüllt
A 2.23	Das System sollte Nutzern die Möglichkeit bieten, das Gesicht im Raum zu bewegen	nicht erfüllt
A 2.24	Das System sollte Thumbnails im herangezoomten Zustand anzeigen.	erfüllt
A 2.25	Das System wird Nutzern die Möglichkeit bieten, asymetrische Ausdrücke zu kodieren	geplant
A 2.26	Das System wird Nutzern die Möglichkeit bieten, Meshes mit beliebiger Blendshape Konfiguration zu laden	geplant
A 2.27	Das System wird Nutzern die Möglichkeit bieten, Animation textbasiert zu erstellen	geplant
A 2.28	Das System wird Nutzereingaben auf Korrektheit prüfen	geplant
A 2.29	Das System wird Datenstrukturen auf Korrektheit prüfen	geplant
A 2.30	Das System wird Transitionen durch Tweening anpassbar machen	geplant
A 2.31	Das System wird Nutzern die Möglichkeit bieten, laterale Ausdrücke zu kodieren.	geplant
A 2.32	Das System wird Nutzern die Möglichkeit bieten, Kopfbewegungen darzustellen.	geplant
A 2.33	Das System wird Nutzern die Möglichkeit bieten, Augenbewegungen darzustellen.	geplant
A 2.34	Das System wird Nutzern die Möglichkeit bieten, Signaltöne in das ISI einzufügen.	geplant
Nicht funktionale Anforderungen		
A 3.1	Die Anwendung muss Nutzern die Möglichkeit bieten, den Quellcode einzusehen und zu ändern	erfüllt
A 3.2	Die Anwendung muss Nutzern die Möglichkeit bieten, den Quellcode unbeschränkt zu nutzen	erfüllt
A 3.3	Die Anwendung muss Nutzern die Möglichkeit bieten, mit der Oberfläche intuitiv zu agieren	erfüllt
A 3.4	Das System wird gegen Angriffe geschützt sein	geplant
A 3.5	Die Oberfläche wird ein optisch ansprechendes Design besitzen	teilweise erfüllt
A 3.6	Die Anwendung muss fähig sein in allen gängigen Webbrowsern ausgeführt zu werden	erfüllt

Quelle: Eigene Darstellung

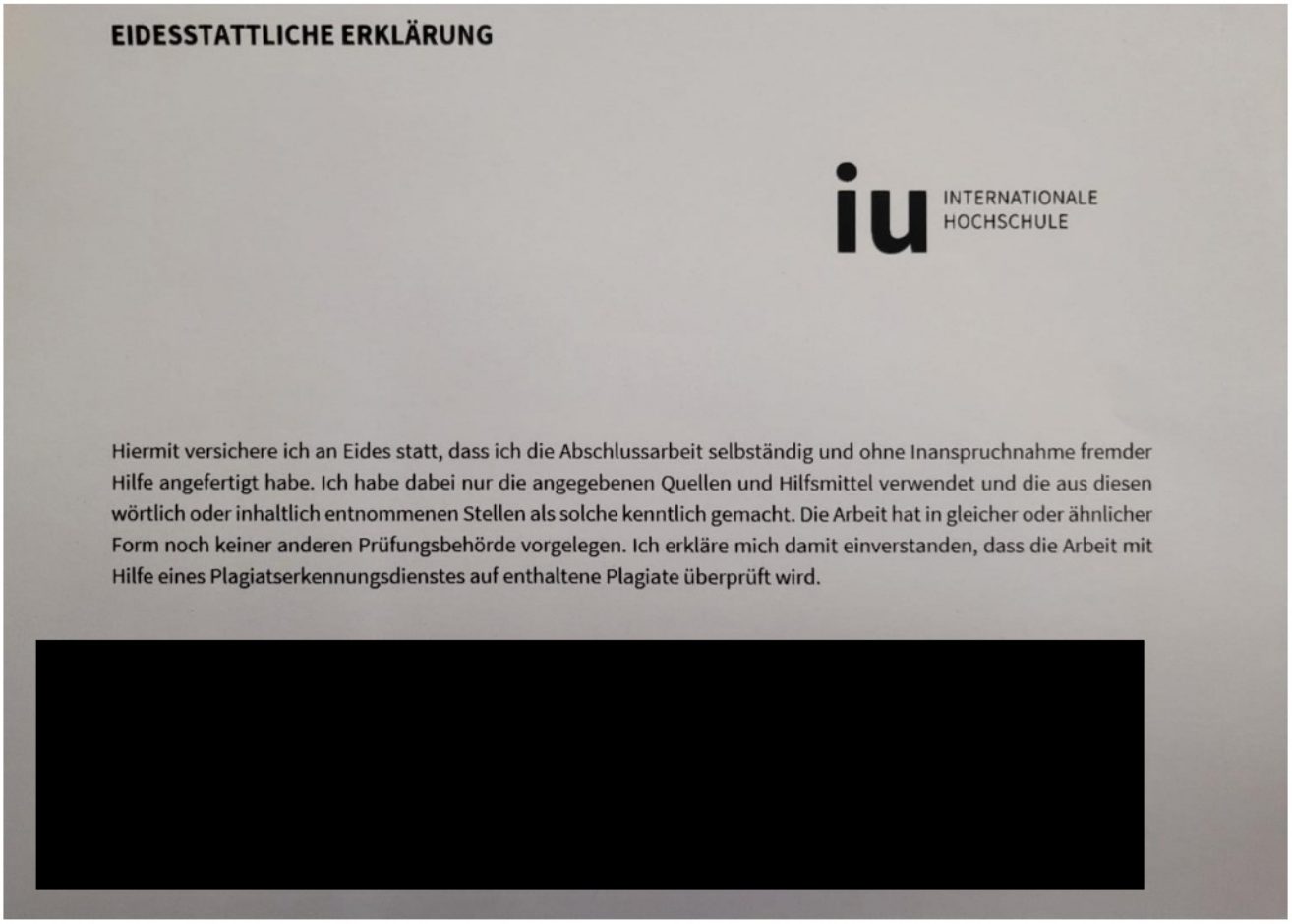
Glossar

Begriff	**Definition**
Agile Softwareentwicklung	Ein iterativer und inkrementeller Ansatz zur Softwareentwicklung, festgehalten in dem agilen Manifest.
Animation	Laut Nieschwitz et al. wird unter einer Animation die Veränderung einer Szene über die Zeit verstanden welche dabei häufig der Augenpunkt und die Lichtquelle sind oder wie in dieser Arbeit die Gestalt des Objekts mit den zugehörigen Schattenwürfen (Nischwitz et al., 2019, S. 561). Dazu wird, um den Aufwand zur Erstellung zu bewältigen die Bewegungsablaufe nicht zu jedem Zeitpunkt definiert, sondern zu bestimmten Zeitpunkten in einem Keyframe festgehalten und der Übergang zwischen den Bildern algorithmisch Interpoliert oder Approximiert (Parent, 2012, S. 62).
ARKit	Ein Framework von Apple um Hardware-Sensorfunktionen, in Augmented Reality-Apps und -Spiele zu integrieren (Apple Inc., n.d.).
Aspect Ratio	Das Verhältnis der Breite zur Höhe eines Bildschirms oder Bildes.
Asynchron (Async)	Rechenoperationen, die unabhängig und parallel zu anderen Operationen ausgeführt werden.
BASE64	Kodierungsverfahren zu Umwandlung von Binärdaten in eine ASCII Repräsentation.
Binary Large Object (BLOB)	Datenformat das große Mengen binärer Daten speichert, z.B. Videos. Es besitzt keine Metainformationen und muss auf Anwendungsebene verwaltet werden.
Browser Repaint	Aktualisierung der Website durch den Browser bei Änderungen der angezeigten Inhalte.
Git	Git ist ein freies und quelloffenes, verteiltes Versionskontrollsystem, das dafür entwickelt wurde, kleine bis große Projekte effizient zu verwalten (git-scm.com, n.d.).
Import Map	Ermöglicht die Steuerung, wie der Browser Modulspezifizierer beim Import von JavaScript-Modulen auflöst(MDN, 2024c).
IntelliSense	IntelliSense ist ein allgemeiner Begriff für verschiedene Code-Bearbeitungsfunktionen in VSCode, beispielsweise Code-Vervollständigung oder Informationen zu Code Teilen (Microsoft, n.d.).
JSON Schema	Die Definition der vorgegebenen Struktur einer JSON-Datei.
MIT Lizenz	Ein Term zur Lizensierung für Produkte die in der Regel umfassende Rechte einräumt. Die Bedingung ist meistens nur die Gewährung derselben Rechte bei Nutzung der lizensierten Software.

Model-View-Controller Modell (MVC)	Ein Software-Architekturmuster zur Trennung von Datenmodell, Benutzeroberfläche und Steuerlogik.
Parsen	Das Analysieren und Umwandeln von Daten aus einem Format in ein anderes.
Plugin	Programmcode der eine bestehende Software um zusätzliche Funktionen erweitert.
Projektions Matrix	Eine Matrix, welche die 2D Projektion der Szene darstellt.
React	Bibliothek zur Erstellung von Benutzeroberflächen (Ackermann, 2023, S. 404)
Rendering	Der Prozess des Renderings ist ein hochkomplexer Prozess welcher in vielen Schritten die Generierung von synthetischen Bildern zum Ziel hat (Nischwitz et al., 2019, S. 50).
THREE.Geometrie	Repräsentation der Vertex Daten einer geometrischen Form (three.js, n.d.-b).
THREE. Loader	Basisklasse für das Laden von Assets (three.js, n.d.-c).
THREE.Material	Klasse, die zum Zeichnen der Geometrien verwendet wird, einschließlich Eigenschaften wie Farbe und Glanz(three.js, n.d.-b).
THREE. Object3D	Die grundlegende Klasse in Three.js, die für fast alle 3D-Objekte verwendet wird und bietet zahlreiche Attribute und Methoden zur räumlichen Manipulation der Objekte(three.js, n.d.-d).
Thumbnail	Ein kleines Vorschaubild einer größeren Grafik.
Transitionszeit	Der Übergang von einem Zustand in einen anderen. In dieser Arbeit definiert die Transitionszeit den Übergang von einer Blendshape Konfiguration bzw. einem Gesichtsausdruck zu einem anderen.
try/catch Block	Bezeichnet eine Kontrollstruktur, die verwendet wird, um Fehler oder Ausnahmen zu behandeln, die während der Ausführung eines Programms auftreten.
Tweening	Technik zur Berechnung von Zwischenbildern in einer Animation.
Uniform Resource Locator (URL)	Die eindeutige Adresse einer Ressource.
UTF-8	Stark vereinfacht gesagt ordnet das Unicode-Transformationsformat jedem Zeichen eine eindeutigen 8 Bitfolge zu (Unicode, Inc., 2017).
Viewport Browser	Sichtbare Bereich einer Website im Browserfenster.
WebRTC API	Programmierschnittstelle, die Webbrowsern und mobilen Apps, Echtzeitkommunikation ermöglicht.

Wireframe	Ein Wireframe oder Drahtgittermodell ist eine Darstellungsform von 3D-Objekten, bei der nur die Kanten der Polygone angezeigt werden (Nischwitz et al., 2019, S. 167).

Eidesstattliche Erklärung



EIDESSTATTLICHE ERKLÄRUNG

iu INTERNATIONALE HOCHSCHULE

Hiermit versichere ich an Eides statt, dass ich die Abschlussarbeit selbständig und ohne Inanspruchnahme fremder Hilfe angefertigt habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Ich erkläre mich damit einverstanden, dass die Arbeit mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft wird.

[redacted]