# Eine normale Website zu einer 3D Website ergänzen

*Berufsmaturitätsarbeit zum Oberthema:*
*«Zeichen der Zeit»*

**Yanick Christen, Jeffry Dahinden**
**BIN18a**

*Berufsmaturitätsschule Zürich*

*Technik, Architektur, Life Sciences*

*Begleitende Lehrperson: Willi Felchlin*

*Abgabetermin: 29.11.2021*

# Abstract

In dieser Arbeit beschäftigen wir uns mit der Frage, welche Herausforderungen beim Erstellen einer 3-dimensionalen (3D) Webseite entstehen.

Um dies zu untersuchen, wird eine eigene 3D Webseite erstellt. Diese wird mithilfe einer Library namens Three.js und einem Open Source Tool namens Blender realisiert. Zuerst werden die notwendigen Grundkenntnisse durch die Maturanden angeeignet, bevor man mit dem Programmieren und Implementieren beginnt. Die dabei aufkommenden Herausforderungen / Schwierigkeiten werden im Rahmen der Arbeit entsprechend dokumentiert.

Dabei erkennt man, dass das Realisieren eines 3D Projektes sehr zeitaufwändig ist und ein grosses Wissen und viel praktische Erfahrung essenziell ist, um eine «brauchbare» 3D Webseite erstellen zu können. Wenn sich jedoch eine Firma von der Konkurrenz abheben möchte, macht es durchaus Sinn, diesen Aufwand zu betreiben, auch wenn die Erstellung viel Zeit in Anspruch nimmt und die Kosten dabei deutlich höher ausfallen können.

# Inhalt

# 1. Einleitung

Viele Konzerne investieren Zeit und Geld, ihre Webseiten möglichst kreativ, modern, dynamisch und imposant für den Benutzer zu gestalten, um sich auf dem Markt abheben und von der Konkurrenz differenzieren zu können. Dabei ist in letzter Zeit vermehrt die Idee von 3-dimensionalen (3D) Webseiten entstanden, welche genau diese Bedürfnisse erfüllen sollen. Aus diesem Grunde passt diese Arbeit auch gut zum Themenbereich «Zeichen der Zeit». Unsere Arbeit beschäftigt sich somit mit dem Thema «3D Webseiten» kreieren, Schwierigkeiten erkennen und diese dokumentieren.

Die Leitfrage in unserer Arbeit ist: «Welche Herausforderungen entstehen beim Erstellen einer 3-dimensionalen Webseite?». Dabei lassen wir uns von der Hypothese leiten, dass wir dies auch ohne besondere Vorkenntnisse schaffen, uns jedoch mit unterschiedlichen Schwierigkeiten konfrontiert sehen, welche wir im Rahmen dieser Projektarbeit erkennen, lösen und dokumentieren wollen. Zusätzlich werden wir die Umsetzung entlang verschiedener Meilensteine auf einer Zeitlinie darstellen. Die Meilensteine sind kurze Videos, welche Jeffry Dahinden für seine eigene Firma erstellt hat.

Wir haben erkannt, dass die Visualisierung via unterschiedlicher 3D Optionen faszinierende Möglichkeiten bietet und uns sehr viel Spass macht. Dazu gehören insbesondere Foto- und Videografie, Animationen, Zeichnungen und Modelle. In einer Webseite kann man all dies zusammenfügen und für das Internet und somit für die ganze Welt öffentlich machen. In unserem Beruf als Informatiker haben wir beide bereits einen Bezug zur Erstellung von Webseiten, die 3D Welt ist aber Neuland für uns.

Im Projekt wird eine JavaScript Library namens Three.js verwendet, welche bei der Erstellung von 3D Webseiten unterstützen soll. Ausserdem wird ein Open Source Tool namens Blender benutzt, welches zur Modellierung von Figuren und Objekten dient.

Da das Programmieren einer solcher Webseite ein gewisses Grundwissen voraussetzt, wird in dieser Arbeit in den Kapiteln 2 und 3 zuerst eine Einführung und Erklärung zu den einzelnen Komponenten gegeben, bevor dann im Kapitel 4 die Implementierung beschrieben wird. Zusätzlich findet man im Kapitel 5 ein Glossar für verschiedene Begriffe, bevor dann im Kapitel 6 das Schlusswort folgt.

# 2. Was ist Three.js?

In diesem Kapitel werden wir versuchen, die Grundlagen von Three.js zu erläutern. Eine 3D Webseite zu erstellen ist sehr aufwändig und benötigt oft ein grosses Team und mehrere Monate Aufwand. Einige grosse Firmen haben bereits 3D Webseiten realisiert, wie zum Beispiel SpaceX, Google, Gucci und noch andere mehr. Damit auch wir eine 3D Webseite kreieren können, hat Yanick Christen einen Three.js Kurs absolviert, welcher von Bruno Simon, einem französischen Programmierer und Freelancer, erstellt wurde.

## 2.1 Three.js und WebGl Theorie

Doch was ist Three.js überhaupt? Three.js ist eine 3-dimensionale Library für Javascript. Erstellt wurde sie von Ricardo Cabello. An Three.js arbeiten aktuell über 100 verschiedene Personen, welche versuchen, die Library stehts zu verbessern. Mit Three.js kann man als Programmierer 3-dimensionale Erfahrungen für den Browser kreieren. Die Library Three.js baut auf WebGl (Web Graphics Library) auf. Doch was ist WebGl überhaupt?

WebGl ist eine Javascript API. Die grundsätzliche Idee ist, dass man mit WebGl Dreiecke zeichnen kann und diese dann möglichst schnell in einem Browser «renderen» kann. Diese Dreiecke können verschiedene Formen und Grössen haben. Um alles schön darzustellen, zeichnet man diese Dreiecke in einem Canvas.

Selbst mit WebGl zu programmieren, wäre aber sehr schwierig. Wir wollen darum in dieser BerufsMaturitätsArbeit (BMA) nicht zu weit in dieses Thema hineinschauen. Bereits um ein einzelnes Dreieck zu erstellen, bräuchte man einen Code von über 100 Zeilen (vgl. Turtorialspoint 2021).

Genau hier kommt die Library Three.js ins Spiel. Mit Three.js kann man eine Figur erstellen, mit der die zuvor genannten Dreiecke automatisch generiert werden.

## 2.1 Basic Scene

Um eine einfache Scene zu erstellen braucht man ein index.html File. Zusätzlich braucht man noch ein script.js File. Für was diese zwei Dateien gut sind, kann man im Kapitel 4.2.1. «Benötigte Klassen» sehen. Auf der offiziellen Webseite kann man danach die Three.js Library herunterladen.

## 2.2 Kameras

Ein wichtiges Element von Three.js ist die Kamera. Der Benutzer der Webseite sieht alle Inhalte durch eine Kamera, ohne diese Kamera würde man nichts sehen. In Three.js gibt es viele Kameras, welche alle von der abstrakten Kamera-Klasse erben. Alle Kameras haben andere Funktionen und sind für verschiedene Anwendungsfälle einsetzbar, wie z.B. Kameras, welche Augen simulieren oder auch 360 Grad Kameras (vlg Three.js 2021c).

In unserer BMA haben wir die «Perspektive-Kamera» benutzt. Diese Kamera ist die am meisten benutzte Kamera. Mit dieser Kamera hat man eine 3D Sicht was bedeutet, dass Objekte, welche in der Distanz sind, kleiner angezeigt werden. Dieser Kamera kann man viele Argumente mitgeben, wie z.B wo sie ist, wo sie hinschauen soll, wie weit die Kamera sehen kann, ab wann die Kamera sehen kann und noch viele mehr (vgl Greggman 2021).

## 2.3 Meshes

Vereinfacht gesagt ist ein Mesh eine Figur mit einem Material. Um ein Mesh zu erstellen, braucht man eine Geometrie und ein Material. Um Objekte in Three.js zu erstellen, braucht man ebenfalls Geometrien. Diese Geometrien bestehen aus Eckpunkten (Punkte in einem Raum) und aus Flächen. Mit einem Material kann man dann dieser Geometrie eine Textur oder eine Farbe geben. Man kann sich das wie folgt vorstellen: ein Tisch kann aus verschiedenen Holzarten, Stein, Keramik oder auch aus Glas bestehen. Die Figur des Tisches ist dann die Geometrie und z.B. Eichenholz wäre das Material. Ein Mesh wäre dann der gesamte Tisch.

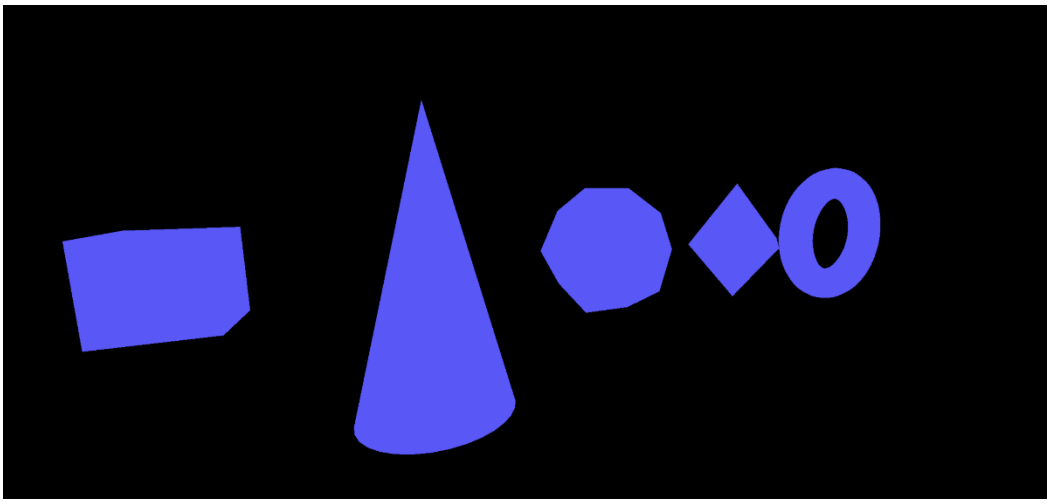

*Abbildung 1: Meshes in Three.js*

## 2.4 Lichter

Damit man in der realen Welt etwas sehen kann, braucht man ein Licht. Dieses Licht kann z.B. die Sonne, ein Feuer, eine Taschenlampe oder auch ein Glühwürmchen sein.

In Three.js ist dieses Prinzip ähnlich. Ohne ein Licht kann man nichts sehen. Darum hat Three.js einige verschiedene Lichter, welche man benutzen kann.

| Nr | Name | Variablen | Beschreibung |
|---|---|---|---|
| **1** | Ambient Light | Bei diesem Licht kann man die Farbe und die Stärke des Lichtes setzen. | Dieses Licht, kommt von jeder Richtung und wird damit benutzt, jedes Objekt und jeder Fläche gleich hell zu beleuchten. |
| **2** | Directional Light | Bei diesem Licht kann man die Farbe, die Stärke und die Position des Lichtes setzen. | Man kann sich dieses Licht wie die Sonne vorstellen. Es strahl wie das Ambient Light von überall, jedoch nicht von jeder Richtung, sondern die Lichtstrahle sind immer parallel zueinander. |
| **3** | Hemisphere Light | Bei diesem Licht kann man zwei Farben und die Stärke setzen. | Dieses Licht ist sehr ähnlich zu dem Ambient Light, mit dem Unterschied, dass man zwei Lichter setzt. Dabei scheint das eine von oben und das andere von unten. Diese Lichtart, gibt es in der Natur nicht, zum Programmieren kann man dieses Licht jedoch gut einsetzten. |
| **4** | Point Light | Bei diesem Licht kann man die Farbe, die Stärke und die Position setzen. Ausserdem kann man definieren, wie stark das Licht abnimmt. Dies ist dafür gedacht, dass ein Objekt, welches weiter vom Licht entfernt ist, weniger hell erscheint als das Nähere. | Dieses Licht ist einfach ein Punkt, welcher in alle Richtungen abstrahlt. |

| | | | |
|---|---|---|---|
| **5** | rectArea Light | Bei diesem Licht kann man die Farbe, die Stärke, die Länge und die Höhe setzen. Ausserdem kann man noch die Position bestimmen und in welche Richtung dieses Licht scheinen soll. | Dieses Licht soll einen Scheinwerfer imitieren Man kann sagen, wie gross die Fläche sein soll und auch von wo dieser Schweinwerfer Licht abgeben soll. rectArea kommt von rectangle Area zu Deutsch: Rechteckige Fläche. |
| **6** | spotLight | Bei diesem Licht kann man sehr viele Variablen setzen. Z.B. die Farbe, die Stärke, die Distanz, die Richtung, die Position und noch einige mehr. | Das spotLight ist einfach eine normale Taschenlampe. |

## 2.5 Physik

Damit etwas in der realen Welt funktioniert, ist Physik ein fundamentaler und wichtiger Teil. Auf einer 3D Webseite ist dies nicht so wichtig. Vieles kann man mit Animationen auch ohne Physik lösen. Doch Menschen sehen gerne, wie sich Objekte realistisch bewegen und miteinander interagieren. In Three.js gibt es an sich keine Physik, welche man einbinden kann. Jedoch kann man eine andere Library verwenden. Die Library welche wir hier vorstellen möchten, heisst Cannon.js. Aber auch mit Cannon.js ist es nicht gerade einfach Physik zu implementieren. Das Prinzip wird hier kurz erklärt:

Normalerweise haben wir unsere Three.js Welt, mit allen Objekten und Figuren. Um Physik zu implementieren, müssen wir eine zweite, nicht sichtbare Welt erstellen. In dieser zweiten Welt muss man alle Objekte und Figuren an der gleichen Stelle wie in der «realen» Welt platzieren. Dank Cannon.js wird dann in dieser Welt Physik angewendet. Man muss dann aber auch noch jedes Mal in der Three.js die Objekte gleich platzieren.

# 3. Blender

## 3.1 Was ist Blender?

Blender ist eine kostenlose Software, mit welcher man Körper modellieren, textieren und animieren kann. Zusätzlich kann man Physik Simulation Tools für diverse Effekte brauchen wie z.B. Rauch, Feuer, Partikel und Wasser. Blender ist mit verschiedenen Software-Tools verbunden wie z.B. After Effects oder Unity, um so auch das Animieren oder das Programmieren von Videospielen einfacher zu ermöglichen. Zudem ist es Open Source und man kann beliebige Änderungen vornehmen.

Blender Foundation wurde im Jahr 2002 gegründet und ist eine unabhängige, gemeinnütze Organisation. Die Vision von Blender ist, dass man die Freiheit hat, 3D Inhalte zu erstellen mit freiem Zugang zu den Märkten. Das Ziel ist es, die beste Open Source Applikation zu sein, um 3D Inhalte zu modellieren. Blender unterstützt alle 3D Formate: Modellierung, Rigging, Animation, Simulation, Rendering, Compositing, Motion Tracking und sogar Videobearbeitung und Spieleerstellung. Profis nutzen die Blender-API, um Skripte einzusetzen und die Anwendungen individuell anpassen zu können, um spezielle Tools zu schreiben. Blender läuft auf Linux, Windows- und Macintosh-Betriebssystemen. Es verwendet die Benutzeroberfläche OpenGL, damit das Verwenden der Software einheitlich ist.

## 3.2 Verwendungszweck

Blender verwendet man für Modellierung, Rigging, Animation, Simulation, Rendering, Compositing, Motion Tracking und sogar Videobearbeitung und Spieleerstellung. Mit Rigging ist gemeint, dass man bei einem Personenkörper z.B. Knochen hinzufügt und den Körper dann physikalisch korrekt bewegen kann. Compositing ist die Farb-/Materialbearbeitung von dem 3D Modell gemeint. So kann man z.B. bei einem Tisch, das Material Holz hinzufügen, welches sehr realistisch wirk. Bei Websites muss man schauen, dass man ein Framework benutzt, welches Blender enthält, wie z.B. Three.js.

## 3.3 Grundsätzliche Befehle/Shortcuts

Folgende Shortcuts sind sehr relevant, um schnell mit Blender arbeiten zu können:

G → Bewegen

R → Rotieren

S → Skalieren

E → Ausstossen

Shift + R → Schneiden

Ctrl + J → Zusammenführen

Tap → von Edit Mode zu Object Mode oder umgekehrt

3 → Fläche auswählen

2 → Kante auswählen

Z → auswählen von Aussehen (solides Aussehen, materielles Aussehen etc.)

# 4. Implementierung

In diesem Kapitel zeigen wir, wie wir vorgegangen sind und wie unsere Fortschritte ausgesehen haben. Um unser Vorgehen und unsere Schwierigkeiten zu veranschaulichen, haben wir zusätzlich noch ein Video gedreht. Den Link zu dem Video findet man im Kapitel 6.

## 4.1 Vorgehen Blender

Jeffry Dahinden hat zuallererst ein Prototyp modelliert. Am Anfang waren wir sehr unsicher, wie es genau aussehen sollte. Wir schwankten zwischen Räume bauen, welche dich führen oder einfach eine Strasse zu modellieren, bei welcher man nur geradeaus laufen muss. Schlussendlich haben wir uns für die Strasse entschieden. Die hat folgendermassen ausgesehen:

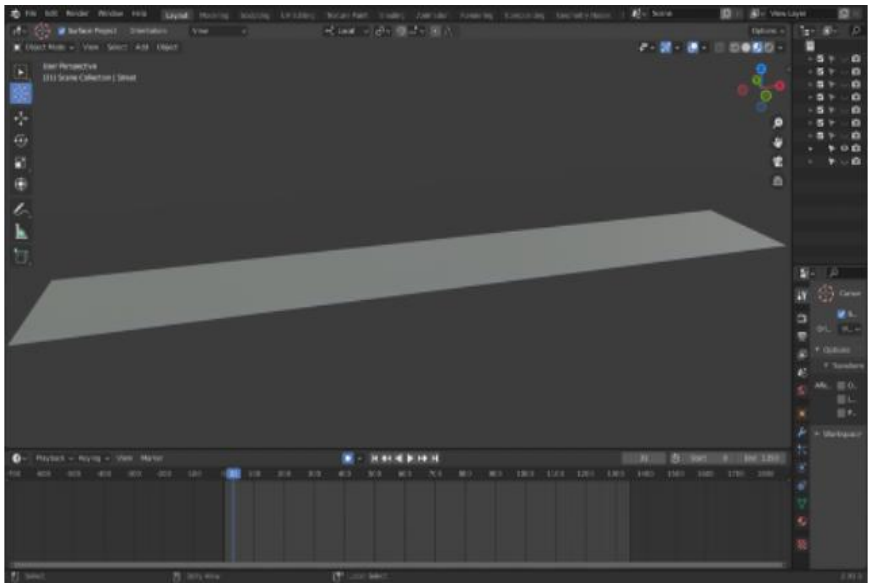

*Abbildung 2: Strasse - Geomtrie*

Leider konnten wir keine Textur hinzufügen, damit es aussieht wie eine echte Strasse. Aus diesem Grund haben wir in Photoshop eine Strasse gezeichnet. Mit Hilfe von YouTube sind wir auf dieses Ergebnis gekommen:

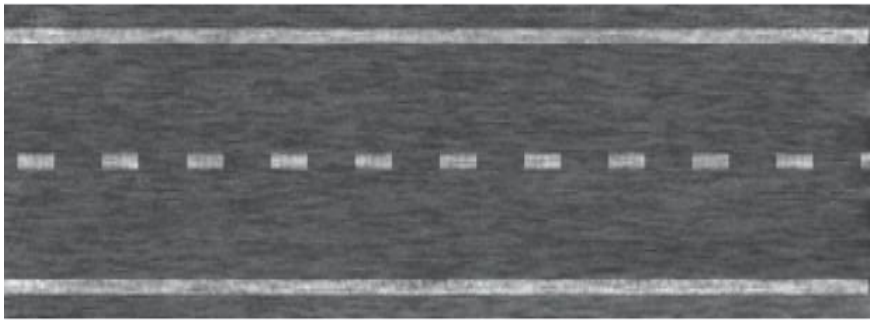

*Abbildung 3: Strasse - Textur*

Nachdem wir die Strasse in Blender hinzugefügt und es dann richtig auf den Boden positioniert haben, fingen wir mit dem ersten Meilenstein an. Es geht um das erste Autovideo, welches Jeffry Dahinden anfangs Jahr 2021 produziert hat. Aus diesem Grund modellierten wir ein Auto. Da es nicht sehr einfach ist, ein Comic-Realistisches-Auto zu modellieren, haben wir wieder Hilfe aus dem Internet geholt. Jetzt ist es ein roter Sportwagen. Neben dem Auto sieht man noch eine Sony Kamera, welche das Auto fotografiert. Wenn man nahe zum Bildschirm kommt, wird das Video, angezeigt.

*Abbildung 4: Erster Meilenstein*

Beim Meilenstein zwei geht es um das erste Autovideo, welches durch das Anschreiben auf Instagram durchgeführt wurde. Wie beim ersten Meilenstein kann man auf den grossen Bildschirm klicken und das produzierte Video wird angezeigt. Daneben sieht man meinen Arbeitstisch mit meinem gebogenen Bildschirm, den ich besitze. Es war sehr schwierig, den Bildschirm mit dieser Form zu modellieren. Ich brauchte Hilfe aus dem Internet und schlussendlich konnte ich es durch eine YouTube Erklärung erstellen. Man hat viele Modifikatoren gebraucht, was für mich sehr neu war.

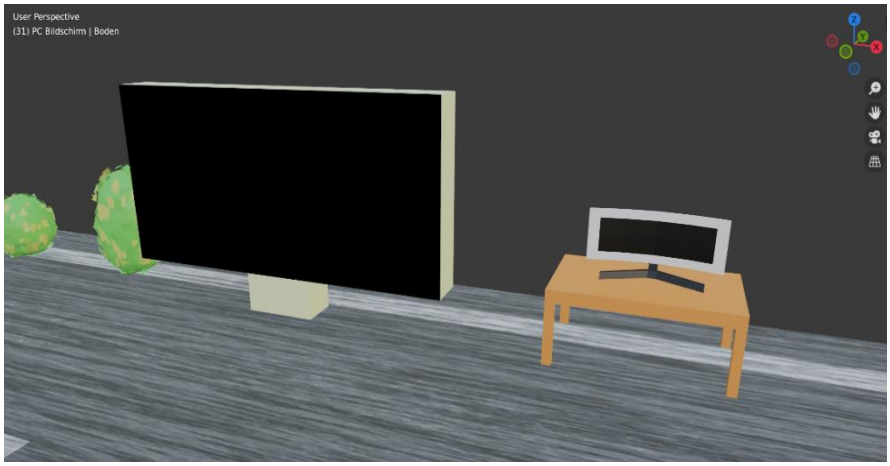


*Abbildung 5: Zweiter Meilenstein*

Bei dem Meilenstein drei geht es um ein Projekt, welches Jeffry mit einer Influencerin hatte. Wir haben aus diesem Grund ein Handy modelliert und in das Handy ein Video hinzugefügt. Darin wird kurz gezeigt, für welche Influencerin Jeffry das Video gemacht hat und wo sie das hochgeladen hat. Das Video haben wir selbst mit der Bildschirmaufnahme auf dem Handy erstellt. Sobald in die Nähe zu dem grossen Bildschirm kommt, wird das produzierte Video angezeigt.


*Abbildung 6: Dritter Meilenstein*

Bei dem vierten Meilenstein geht es um den Auftrag mit einem Lamborghini Huracan. Hier kann man nur den Bildschirm anklicken und das Video anschauen.

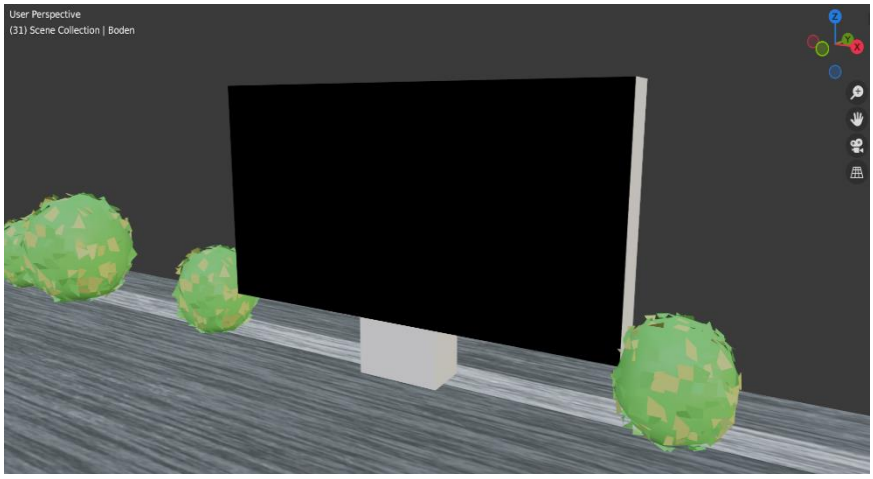


*Abbildung 7: Vierter Meilenstein*

Beim Meilenstein fünf geht um das erste Projekt von Jeffry, welches mit einer FPV Drohne gefilmt wurde. Da auch Bilder vorhanden sind, haben wir zwei Schilder modelliert und Bilder hinzugefügt. Die FPV Drohne wurde sehr detailliert gebaut, denn man sieht die Propeller, die Batterie und die GoPro. Da wir bei diesem Meilenstein schon vertieftes Wissen mitbrachten, haben wir es ohne Hilfe aus dem Internet modellieren können. Diese Drohne haben wir zusätzlich animiert. Dafür mussten wir jedoch jeden Frame einzeln animieren. Das war eine sehr kreative und aufwändige Aufgabe.

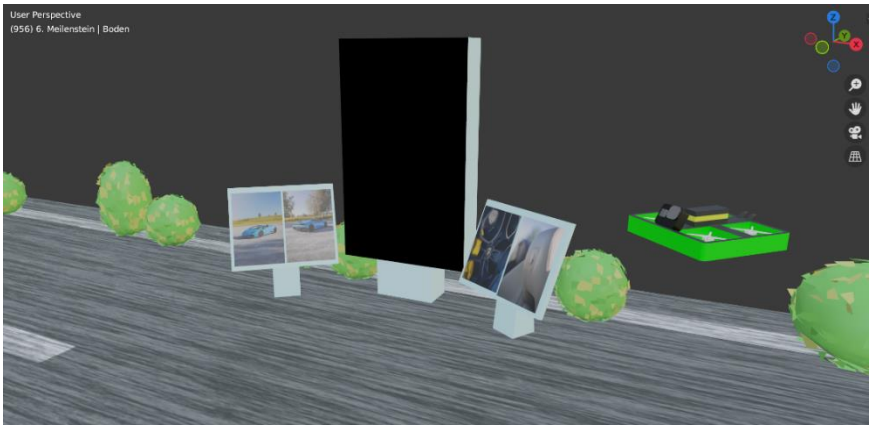


*Abbildung 8: Fünfter Meilenstein*

Das ist der letzte Meilenstein. Bei diesem geht es darum, dass Jeffry das erste Mal ein Aftermovie im Club erstellt und Bilder geschossen hat. Da es um Clubs geht, haben wir eine Tanzfläche mit einer Discokugel modelliert. Bei der Discokugel mussten wir Modifikationen hinzufügen und modellieren, damit es genau aussieht wie eine echte Discokugel. Bei der Tanzfläche mussten wir mehrere Flächen hinzufügen, gleichmässig schneiden und richtig färben. Wenn man auf den linken Bildschirm klickt, erscheint das Aftermovie. Rechts sieht man zwei Bilder, welche im Club gemacht wurden.


*Abbildung 9: Sechster Meilenstein*

## 4.2 Vorgehen Three.js

In diesem Kapitel erfährt man, wie wir in unserer Projektarbeit vorgegangen sind. Man kann die Fortschritte anhand von Bildern sehen. Ebenfalls beschreiben wir Probleme und Schwierigkeiten, die wir in dieser Zeit hatten.

### 4.2.1 Vorbereitung und Start der Arbeit

Die Vorbereitungen zu unserem Projekt begannen schon einige Wochen bevor wir wussten, welches Thema wir für die BMA nehmen werden, da Yanick Christen einen Three.js Kurs begann. Nach der Projektvereinbarung ging die Vorbereitung aber erst richtig los. Die Schulung beinhaltet 39 Kurse und insgesamt 45 Stunden reines Video Material. Die Schulung war ein vorzeige Kurs, in welchem man mit Bruno Simon gleich mitprogrammieren konnte. In diesem Kurs hat Yanick Christen sehr viel gelernt und konnte kleine Szenen erstellen. (vgl. Simon 2019)

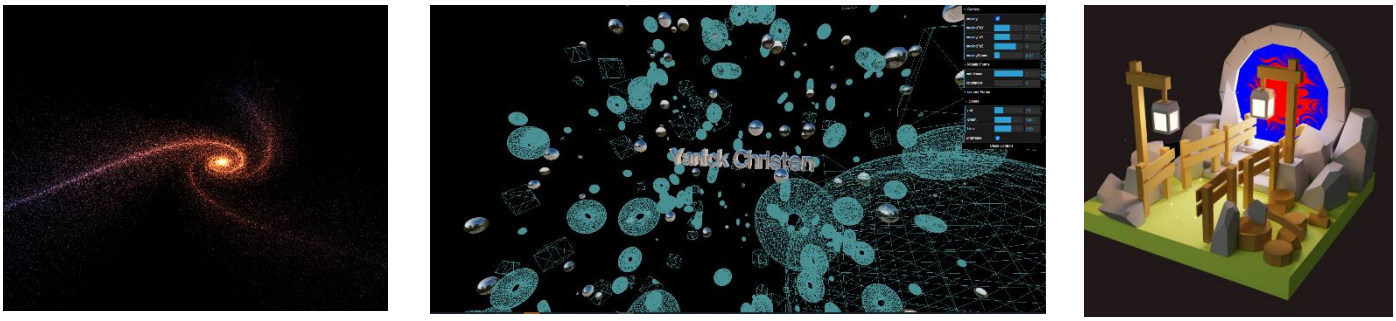


*Abbildung 10: Three.js – Kurs*

Besonders beim Portal, hatten wir grossen Respekt, dass wir es zeitlich nicht schaffen könnten. Um dieses Portal zu erstellen, musste man die Scene in Blender erstellen, diese Scene exportieren, dies dann in Three.js einfügen und zum Schluss musste man noch Details hinzufügen und kleine Anpassungen machen, bis man dieses Ergebnis hatte. Insgesamt gab dies über 6 Stunden Aufwand und alles was existierte, war dieses statische Portal. Es gab keine Funktionen oder sonst etwas, was man damit machen konnte. Zudem war dies nur möglich in diesen 6 Stunden, da man alles eins zu eins dem Video nachmachen konnte.

Ausser dem Video haben wir noch einige Youtube Videos angesehen, welche die Grundlagen von Blender beschrieb.

Bevor wir mit unserem eigentlichen Projekt begonnen haben, erstellten wir noch ein Testversuch, in welchem wir ohne grossen Plan ausprobierten, was mit unserem Wissen etwa möglich ist. Wie man sehen kann, hatten wir hier schon die Vorstellung von Bildschirmen und ein bewegbares Auto, welches wir später mit einer Person auswechselten.

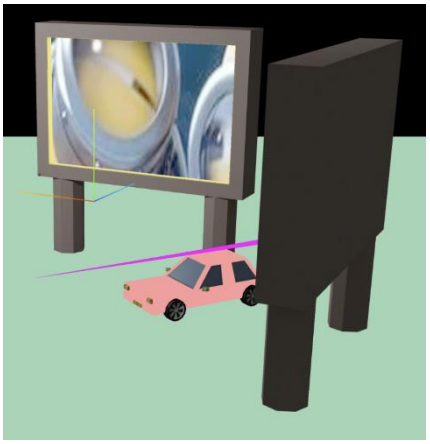

*Abbildung 11: Test-Projekt*

Das Projekt zu starten, war dank dem Kurs keine grosse Schwierigkeit mehr. Zu Beginn hatten wir einfach eine schwarze Scene mit einer statischen Kamera.

### 4.2.2 Implementierung des Charakters

Begonnen haben wir mit einem Charakter, welchen wir ebenfalls zuerst in Blender modellieren mussten. Dazu haben wir auf Youtube ein Turtorial gefunden, welches dies schön erklärte. Da wir einem Video folgen konnten, war dies nicht sehr schwierig. Ausserdem gaben wir dem Charakter ein Skelett mit Knochen, welches bei der Animierung hilfreich ist. Zudem gaben wir der Figur auch gleich drei Animationen: Rennen, springen und stehen. Das Implementieren der Figur gab uns kleine Probleme, wie z.B. dass das Material nicht mitgenommen wurde, oder auch uns keine Animationen anzeigt wurde. Nach einiger Zeit konnten wir diese Probleme jedoch lösen und somit hatten wir eine Figur in unserem Projekt.

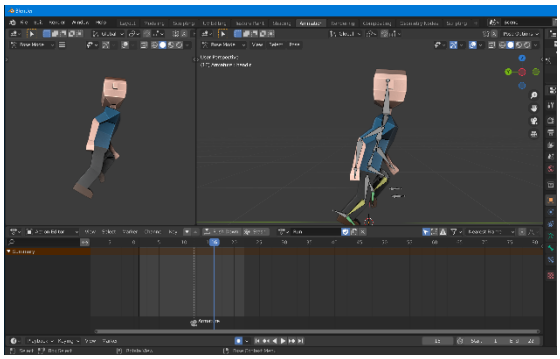

*Abbildung 12: Charakter*

Danach haben wir versucht, dass sich die Figur auch auf Tastendruck bewegen kann. Da wir bereits JavaScript Kenntnisse haben, war es nicht sehr schwierig, mit einem Tastenklick einen Event zu starten. Damit konnten wir die Figur immer an eine neue Position schicken. Jedoch war es ein

schwieriges Unterfangen, die Person so bewegen zu können, wie wir es gerne hätten. Alle Schwierigkeiten findet man ebenfalls in unserem Erklärungsvideo.

**Keine stockenden Bewegungen:**

Zu Beginn konnten wir die Person bewegen, jedoch mussten wir die eine Taste immer wieder drücken, damit unser Charakter vorwärts kommt. Dadurch stoppte die Person immer wieder kurz und das ganze sah sehr unschön aus. Dies konnten wir nach einiger Zeit lösen, indem wir geschaut haben, welche Taste gerade gedrückt ist.

**Gleichzeitig nach vorne laufen und die Richtung ändern:**

JavaScript kann sich jeweils nur auf eine Taste konzentrieren, was heisst, dass man nicht gleichzeitig nach vorne laufen und die Richtung ändern konnte. Um dieses Problem zu lösen haben wir im Internet eine schlaue Lösung gefunden, die den Status der Taste in einem Objekt speichert. In welche Richtung muss die Person gehen? Da sich die Person in einem Koordinatensystem bewegt, ist es schwierig zu sagen, die Person soll nach links gehen.

**Animationen:**

Der Charakter sollte stehts nur eine Animation zu einem bestimmten Zeitpunkt haben. Sobald die Person beginnt zu laufen, muss die steh Animation angehalten werden.

### 4.2.3 Implementierung der Strasse / der Meilensteine

Während wir die Funktionalitäten des Projektes programmierten, konnten wir gleichzeitig in Blender alle Meilensteile erstellen. Wie wir die Meilensteile implementiert haben, kann man im Kapitel 4.1 «Vorgehen in Blender» nachlesen. Diese konnten wir dann immer implementieren. Hier sieht man ein paar Fortschritte der Meilensteine. Bei der Implementierung gab es zwar kleinere Schwierigkeiten, jedoch keine schwerwiegende Stolpersteine.

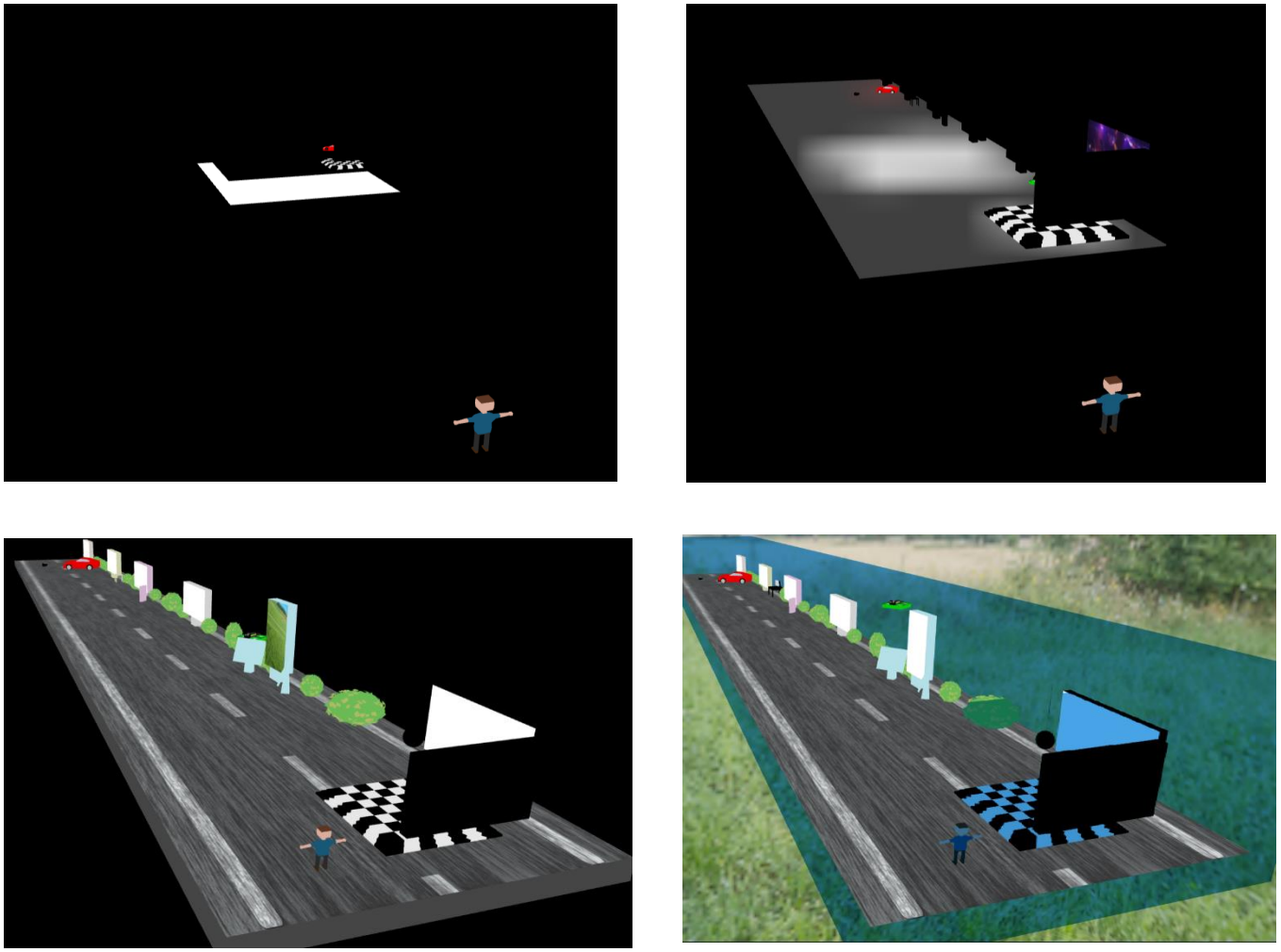

*Abbildung 13: Fortschritte, der Scenen*

### 4.2.4 Kamera

Wir wollten, dass die Kamera dem Charakter folgen kann, wie man es bei einem Third Person Spiel kennt. Mit unserer Kamera hatten wir zu Beginn viele Probleme. Darum haben wir im Internet geschaut, wie man so eine Third Person programmieren kann. Wir fanden auch ein Video, doch dieses half uns nicht weiter. Also haben wir selbst versucht unsere Schwierigkeiten zu lösen. wie man in unserem Erklärungsvideo sehen kann, war die Kamera stockend, die Kamera schaute zu direkt auf die Person, sie schaute nicht unserem Charakter nach und einmal, bewegte die Kamera sogar die Person, was für grosse Verwirrung sorgte. Mit der Zeit konnten wir aber alle Schwierigkeiten überwinden. Zum Schluss versuchten wir noch die Perspektive der Kamera auf Tastendruck zu wechseln, sowie auch auf Tastendruck zwischen der folgenden Kamera und der statischen Kamera zu wechseln. Dies zu lösen war eine richtige Knacknuss. Lange haben wir nicht

gewusst, dass wir das sogenannte Orbit Controls nicht benutzen dürfen, wenn wir erreichen möchten, dass die Kamera in eine bestimme Richtung schauen soll. Doch schlussendlich konnten wir auch diese Hürde überwinden.

### 4.2.5 Lade Fenster

Da mit der Zeit unsere Meilenstein Datei sehr gross wurde, musste die Webseite beim Starten jeweils recht lange laden, worauf man die Person zuerst in der Luft stehen sah. Glücklicherweise gab es im Three.js Kurs eine Lektion, welche genau dies behandelte. Darum hatten wir keine grossen Mühen oder Probleme, eine Lösung zu finden.

### 4.2.6 Code Strukturierung

Lange hatten wir unseren ganzen Code in einem File. Dies funktionierte zwar, jedoch war es sehr mühsam die einzelnen Code Elemente zu finden, wenn wir etwas bearbeiten mussten. Das störte beim Programmierungsprozess und auch das schlechte Gewissen eines Programmierers. Darum haben wir am Ende der Projektwoche einen Tag damit verbracht, den ineffizienten langen Code in mehrere Dateien zu strukturieren und Klassen daraus zu machen. Mehr zu den einzelnen Klassen findet man im Kapitel 4.2 «Klassen», in welchem beschrieben wird, was die einzelnen Files / Klassen machen.

### 4.2.7 Implementierung des Soldaten

Nachdem wir bereits mit einem grossen Teil unseres Programmes fertig waren, wollten wir unsere Figur verschönern. Zuerst wollten wir wieder selbst eine Figur in Blender modellieren, jedoch wussten wir, dass uns für dieses Vorhaben die Zeit nicht reichen würde. Ausserdem wollten wir in unserer Arbeit herausfinden, wie schwierig es ist, eine 3D Webseiten zu programmieren. Interessant wäre für uns, den Unterschied zu eruieren, wenn man eine eigene Figur macht oder eine externe Figur implementiert. Hierfür haben wir eine Figur von der offiziellen Three.js Webseite verwendet, welche jedoch ursprünglich von der Seite Mixamo kommt.

## Unterschied von normaler Figur

Da wir beide keine grossen Erfahrungen in Blender haben und auch nicht Tage in einen Charakter investieren können, haben wir uns entschieden, einen simplen Charakter mit einfachen Animationen zu erstellen. Wir wollten jedoch schauen, ob es auch Alternativen geben würde, wenn man sich nicht so gut in Blender auskennt. Dabei sind wir auf die Seite Mixamo.com gestossen. Auf dieser Webseite kann man aus vielen verschiedenen Figuren und vielen Animationen eine Passende auswählen, diese herunterladen und sie dann relativ einfach in einem Projekt verwenden. Das positive daran ist, dass eine solche Figur unglaublich detailliert gemacht

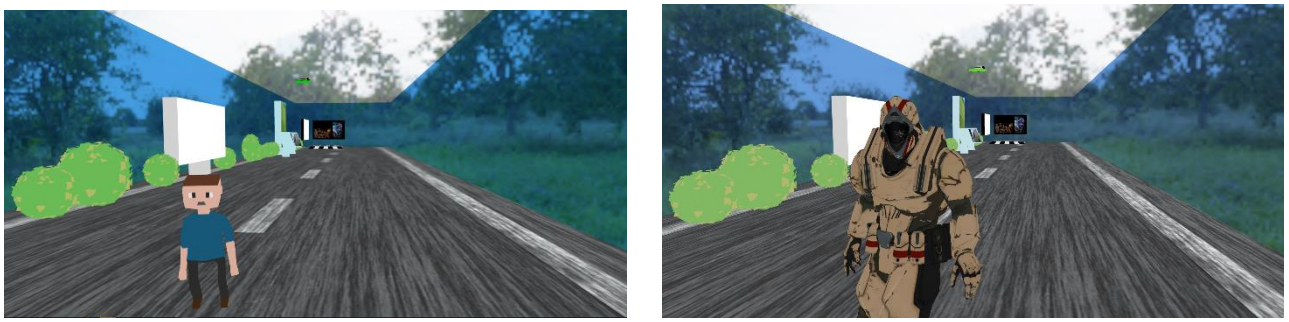

*Abbildung 14: Unterschied der beiden Figuren*

wurde und sehr realistisch aussieht. Auch die Animationen laufen perfekt. Als Kontra könnte man argumentieren, dass die Freiheit eingeschränkt wird. Jedoch kann man die Figur in Blender nach Belieben verändern und muss sich daran nicht an den Style von Mixamo's Figuren halten. Da unsere Webseite aber kein Grossprojekt ist, haben wir uns entschieden, dass sich der Aufwand nicht lohnt, diese Figuren zu ändern und haben sie ohne Anpassungen übernommen.

## Three.js anstatt Mixamo

Auf Mixamo gibt es viele Figuren, welche man verwenden kann. Wir haben unsere Figure nicht direkt von Mixamo heruntergeladen (Mixamo 2021), sondern von einem Beispiel auf der offiziellen Three.js Webseite. (three.js 2021b)

Für das hatten wir mehrere Gründe. Zum einen existieren auf diesem Beispiel bereits drei Animationen, nämlich stehend, laufend und rennend, welche wir mühsam zusammenfügen müssten. Zum anderen ist das File auf Three.js bereits im richtigen Format. Wir können das Format neu konvertieren, dies wäre aber aufwändig. Als letzten Punkt hat das File von Mixamo eine Grösse von 6424 Kilobyte, im Gegensatz zum File von Three.js mit nur 2110 Kilobyte also rund dreimal kleiner. Dies hilft dabei, dass die Webseite weniger lange laden muss.

**Rechte:**

Sobald man etwas Externes für ein eigenes Projekt übernimmt, muss man natürlich schauen, ob man dies überhaupt darf. In unserem Falle ist Mixamo für jeden kostenlos, der wie wir, eine Adobe ID hat. Man muss dazu auch kein Abonnement zu Creative Cloud haben. Jedoch darf man die Webseite nicht für eine Fima benützen, warum wir eine eigene Figur erstellen müssten, sobald wir die Webseite auf der Firma von Jeffry aufschalten würden (Mixamo 2021).

# 4.3 Klassen

JavaScript wird als eine «Multi-Paradigma» Sprache bezeichnet. Das heisst, dass die Sprache sowohl objektorientierte Programmierung als auch funktionale Programmierung unterstützt. (vgl. Dr. Derek 2021)

Für den grössten Teil wird JavaScript als funktionale Programmiersprache verwendet. Da jedoch unser gesamtes Projekt auf JavaScript aufbaut, haben wir uns entschieden, objektorientierte Programmierung zu verwenden. Der Grund war, dass man so den Code auf mehrere Dateien aufteilen und auch einfach auf die einzelnen Dateien zugreifen konnte.

Hier geben wir einen Einblick, was die einzelnen Files oder auch Klassen für Funktionen haben. Da wir in unserem Projekt über 800 Zeilen Code geschrieben haben, werden wir nicht auf jede Methode

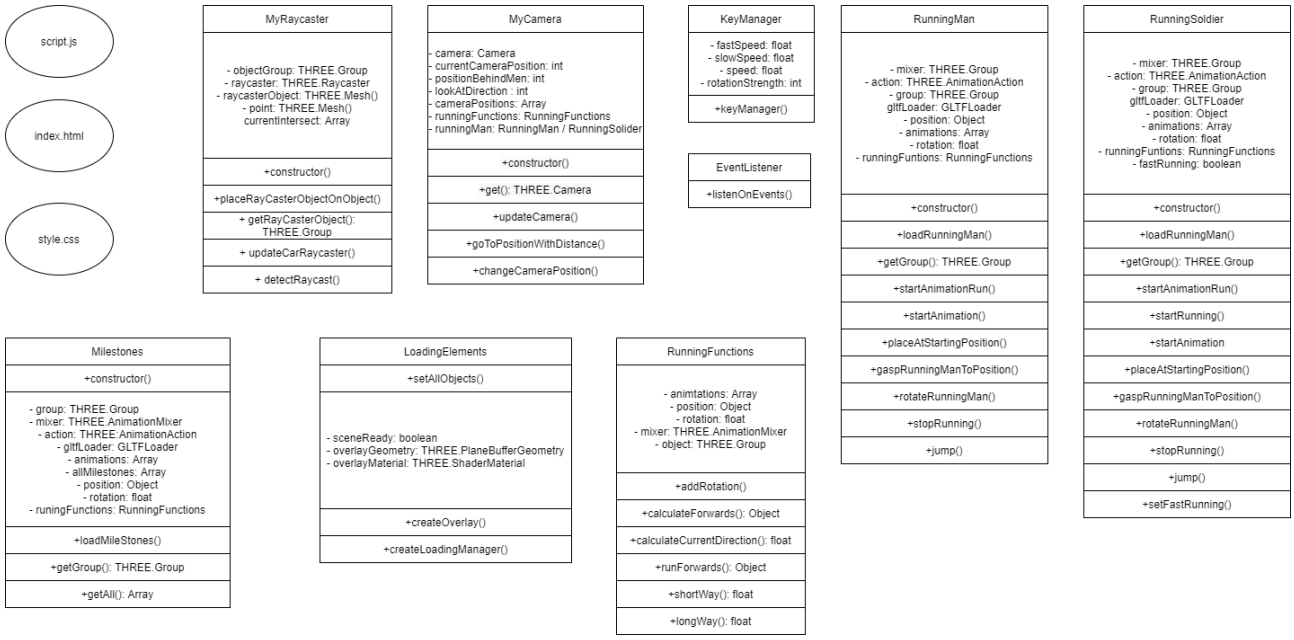


*Abbildung 15: Klassendiagram*

und auf jedes Objekt eingehen, da dies nach den Rahmen sprengen würde:

## 4.3.1 Benötigte Klassen

**Script.js**

Dieses File kann man als Haupt- oder Startfile unseres Projektes ansehen. Sobald man mit dem Link auf unsere Webseite geht, wird dieses File gestartet. Von diesem File aus ruft man jedes andere File

auf und verwendet dies direkt oder auch indirekt. Gleich zu Beginn des Files instanziieren wir alle anderen Klassen, sowie auch die Scene, das Licht und unseren «Draco Loader». Zudem wird der Hintergrund gesetzt. Damit man unsere Meshes auf dem Bildschirm sehen kann, geben wir diese unserer Scene auch gleich mit. Was ebenfalls ein wichtiger Teil ist, ist der Renderer, welchen wir hier setzten. Die wichtigste Funktion, welche wir in diesem File haben, ist die «tick» Funktion, die bei jedem neuen Frame aufgerufen wird. Diese «tick» Funktion wird dafür benutzt, um die Kamera, unsere rennende Person, die ganze Strasse, den Raycaster und der Renderer zu aktualisieren.

Ausserdem haben wir Funktionen, welche von den anderen Klassen aufgerufen werden. Diese Funktionen werden mit dem Schlüsselwort «export» angegeben, womit man der Funktion sagen kann, dass sie extern aufgerufen wird. Einerseits haben wir eine Funktion namens «resize», welche aufgerufen wird, sobald sich das Browser Fenster verändert. Dort werden die Kamera und der Renderer neu gesetzt. Andererseits haben wir die «changeOrbitControls» Funktion, welche die Kamera so setzt, dass sie entweder statisch ist oder dem Spieler folgen kann.

**Index.html**

Das index.html File ist eine Datei welche man kennt, wenn man bereits eine Webseite geschrieben hat. Wenn man eine Webseite schreibt, sollte man im Wurzelverzeichnis oder auch Stammordner (engl.: root directory) eine Datei namens index.html anlegen. Dieses File wird vom Browser immer als Erstes geladen (vgl. Haunschild 2017, 43).

In unserem Fall wird in der Datei: «webpack.common.js» auch bereits angegeben, dass die script.js und die style.css Dateien ebenfalls bereits mitgeladen werden sollen, sobald der Browser auf die URL anspricht.

**Style.css**

In dem Style.css File, haben wir den einzelnen Elementen, wie zum Beispiel dem Ladebalken oder auch dem Informations Block, Gestaltungsanweisungen gegeben.

### 4.3.2 Klassen für die Blender Objekte

**MileStones**

Unsere grössten Elemente holen wir in der “MileStones” Klasse, zu Deutsch Meilensteine. Darin sind die Strasse, alle Bildschirme, alle Büsche, das Auto und noch einige andere Figuren enthalten. Die Klasse an sich ist aber recht simpel aufgebaut. Alles was in dieser Klasse passiert, ist das Laden der Blender Datei. In der Klasse bestimmen wir selbst noch die Grösse und die Rotation. Ausserdem lassen wir einfach alle Animationen laufen, welche in dem File enthalten sind.

**RunningMan**

In der «RunningMan» Klasse implementieren wir unsere selbst gemachte Figur in Three.js. Wie auch in der «MileStones» Klasse setzen wir die Rotation und Grösse. Zusätzlich platzieren wir den Charakter auch noch an einer bestimmen Position. Da unser Charakter nicht wie unsere Meilensteile einfach statisch an einem Ort stehen, sondern sich ständig bewegt und rotiert, haben wir noch einige weitere Methoden implementiert. Ausserdem muss die Person je nach ihrem Zustand eine andere Animation abspielen. Auch dies regeln wir in dieser Klasse. Die Person kann rennen, stehen / aktives stehen (auch wenn die Figur steht, bewegt sie sich ein wenig), rotieren und springen. Die Sprung Animation sieht jedoch nicht sehr gut aus.

**RunningSoldier**

Die «RunningSoldier» und die «RunningMan» Klasse sind sich sehr ähnlich. Der Unterschied ist, dass bei dem RunningSoldier die externe Militärs Figur implementiert wird. Diese sieht um einiges besser aus und auch die Animationen wirken viel schöner.

Wir haben uns dazu entschieden, dass wir beide Klassen lassen, weil man damit mit nur einer Zeile Code die Person wechseln kann. Ausserdem müssen wir, sobald wir die Webseite als Firma verwenden, wieder die alte Person nehmen oder eine neue Person modellieren.

### 4.3.3 Übrige Klassen

**EventListener**

In JavaScript existieren viel verschiedene Events, welche man mit dem Befehl «addEventListener» abhören kann. In unserem Projekt schauen wir kontinuierlich auf einige diese Events und führen danach bestimmte Befehle aus.

| Nr. | Auslöser | Beschreibung |
|---|---|---|
| 1 | Fester Grösse wurde verändert | Sobald, die Fenstergrösse geändert wurde, wird die Methode «resize» aufgerufen, womit die Kamera und der Renderer neu konfiguriert werden. |
| 2 | Taste wurde gedrückt | Sobald eine Taste gedrückt wird, wird unser «KeyManager» ausgeführt, mit der Taste, welche gedrückt wurde. Je nachdem, welche Taste gedrückt wurde, wird im «KeyManager» etwas anderes ausgeführt. Als Beispiel, wechselt die Position der Kamera, sobald man auf die Taste «c» für «change» drückt. |
| 3 | Taste wird dauernd gedrückt | Der Unterschied zu dem vorherigen Auslöser ist, dass diese Funktion nicht nur einmal beim Drücken der Taste ausgelöst wird, sondern die Funktion wird kontinuierlich ausgeführt. Dieser Auslöser benötigen wir, damit sich die Figur weiterbewegt, wenn man zum Beispiel die Taste «W» gedrückt hält. |
| 4 | Taste wird nicht mehr gedrückt | In JavaScript kann man sogar sehen, sobald man eine Taste loslässt. Dieser Auslöser benutzen wir, um das Laufen oder Drehen, der Figur zu beenden. |

**KeyManager**

Die meisten Events, welche wir in unserer «EventListener» Klasse abhören, gehen zu der «KeyManager» Klasse. In unserer Klasse «EventListener» haben wir eine Funktion, welche den identischen Namen trägt. Einfach gesagt wird der «KeyManager» immer mit einer Taste aufgerufen. Je nachdem, welche Taste es ist, werden unterschiedliche Befehle ausgeführt.

| Nr. | Tasten Name | Beschreibung |
|---|---|---|
| 1 | «w» und Pfeil nach oben | Der Charakter bewegt sich nach vorne und eine Lauf Animation wird abgespielt. |
| 2 | «a» und Pfeil nach Links | Der Charakter rotiert sich im Gegenuhrzeigersinn. |
| 3 | «d» und Pfeil nach rechts | Der Charakter rotiert sich im Uhrzeigersinn. |
| 3 | «s» und Pfeil nach unten | Der Charakter bewegt sich nach hinten und eine Lauf Animation wird abgespielt. |
| 4 | «o» | Die Kamera folgt nicht mehr dem Charakter, sondern wird zu einer statischen Kamera. Falls dies bereits der Fall ist, passiert das Gegenteil, womit die Kamera wieder dem Charakter folgt. |
| 5 | «c» | Die Kamera ändert die Sichtweise. Als erstes, folgt die Kamera der Person auf der linken Seite der Strasse. Als zweites, befindet sich die Kamera gleich hinter der Figur. Danach bewegt sich die Sicht des Benutzers auf die rechte Strassenseite. Zum Schluss hat man noch eine Frontansicht von dem Charakter. Danach kommt wieder die erste Ansicht und der Zyklus wiederholt sich. |

| | | |
|---|---|---|
| 6 | «shift» | Diese Taste wird in Video Spiele häufig dazu verwendet, die Person schneller laufen zu lassen, warum bei uns das Gleiche der Fall ist. Wenn man also gleichzeitig «shift» und «w» drückt, ändert sich die Animation vom Laufen zu Rennen und man ist sichtlich schneller unterwegs. |

**MyCamera**

Ein wichtiges Element in unserer 3-dimensionalen Webseite ist die Kamera. Die Kamera dient als unsere Augen und ist dafür da, dass der Benutzer auch die Webseite anschauen und geniessen kann. Leicht verwechselbar ist die Bezeichnung unserer Klasse «MyCamera» und der echten Three.js Kamera. Unsere Klasse beinhaltet zum einen die Three.js Kamera. Zum anderen ist in dieser Klasse aber auch definiert, welche Position die Kamera gerade hat, wie weit sie von der Figur entfernt sein soll und in welche Richtung die Kamera schaut.

Damit man auf die normale Three.js Kamera in unserer Klasse zugreifen kann, haben wir eine «get» Methode verwendet, welche genau diese Kamera zurückgibt. Da die Kamera immer der Person folgen soll, müssen wir die Kamera stehts aktualisieren, wobei wir der Kamera immer eine neue Position relativ zu der Person geben und diese dann dort hinbewegen. Des Weiteren haben wir auch programmiert, dass beim Sichtwechsel der Kamera, die Kamera auch wirklich seine Position ändert.

**MyRaycaster**

In der «MyRaycaster» Klasse muss man, wie auch in der «MyCamera» Klasse zwischen dem Three.js Raycaster und unserer Klasse unterscheiden. Raycaster ins Deutsche übersetzt heisst Strahler. Man kann sich diesen Raycaster als einen unsichtbaren und unendlich langen Strahl vorstellen, bei welchem man sagen kann, wo der Strahl entspringt und in welche Richtung der Strahl zeigt.

Diesen Strahl kann uns sagen, durch welche Objekte er hindurchschiesst oder auch einfach nur berührt. Wir wollen diese Eigenschaft dafür nutzen, um alle Objekte zu erkennen, welche unsere

Figur berührt. Dafür setzten wir den Anfangspunkt des Strahles in die Figur und die Richtungen des Strahles schaut jeweils in die gleiche Richtung wie unsere Person.

Da man diesen Strahl jedoch nicht sieht, ist es schwierig zu testen, ob der Strahl richtig platziert ist und auch in die richtige Richtung schaut. Darum kreieren wir in dieser Klasse einen Kegel, welchen wir dann sehr lange ziehen. Diesen Kegel soll auf der gleichen Position sein und in die gleiche

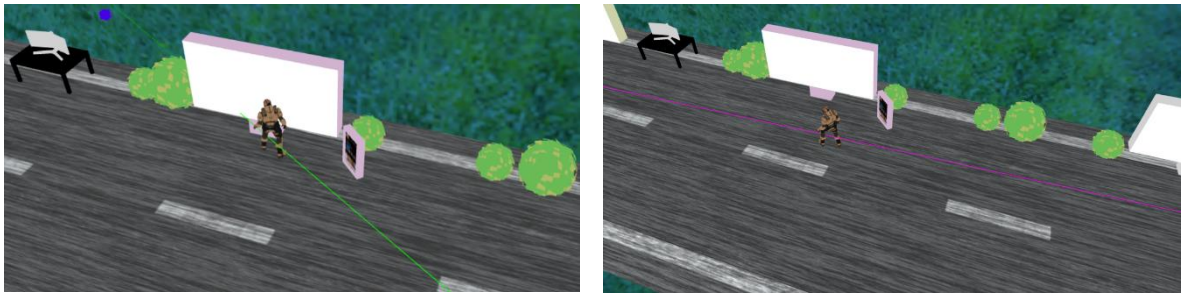

*Abbildung 16: Raycaster in sichtbarer Form*

Richtung schauen, wie der Strahl. Dieser Kegel sollte grün werden, sobald unser Strahl und damit auch unser Charakter ein anderes Objekt berührt.

**LoadingElements**

In dieser Klasse geht es um den Ladebalken, welchen man zuerst zu Gesicht bekommt, sobald man auf unsere Webseite geht. Dieser Ladebalken ist wichtig, da zum Beispiel unsere Figur schneller lädt als die Strasse. Ansonsten würde man zu Beginn nur die Figur frei in der Luft sehen, bis die Strasse erscheint.

Im Prinzip haben wir mittels Html, Css und Photoshop den Ladebalken, Legende und einen schwarzen Hintergrund gestaltet. Sobald alles geladen wurde, lassen wir diese Komponenten verschwinden, damit unser eigentliches Projekt zum Vorschein kommt. Um diese Komponenten verschwinden zu lassen, haben wir eigene Shaders erstellt, damit wir die Kapazität verringern können.

In Three.js existiert zum Glück einen Lademanager, welcher erkennt, sobald alles geladen ist. In den Entwickler-Tools kann man schön sehen, was alles geladen wurde, wie gross die einzelnen Files sind und wie lange das Laden gedauert hat.

Gut sehen kann man bei diesem Abbild die Grösse der Strasse (blau markiert), welche mit Abstand am längsten laden musste und auch das grösste File ist.

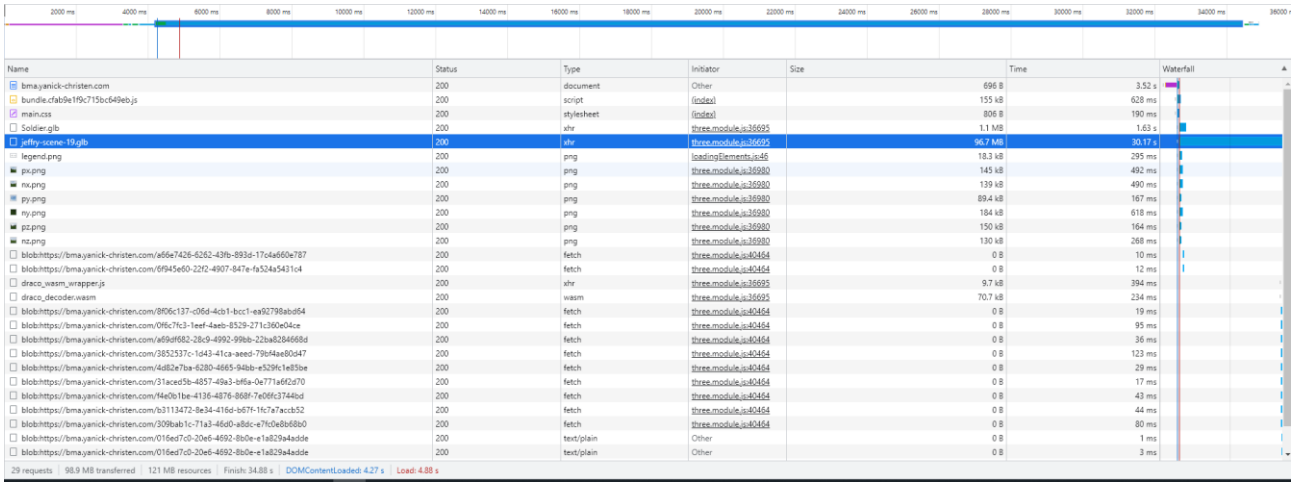


*Abbildung 17: Dev-Tools von Chrome:*

## RunningFunctions

Da wir mehrere Klassen für eine Person benutzen, ist es einfacher, die Logik des Fortbewegens und das Rotieren der Figur in einer separaten Klasse zu implementieren. In dieser Klasse geben wir Anweisungen, um wie viel sich die Person in X und Y Richtung bewegen soll. Man kann sich das so vorstellen, dass die Person von sich aus ein wenig nach links laufen will (violetter Pfeil). Wir berechnen dann, wie weit die Person in Richtung X laufen soll und ob dies positiv oder negativ ist. Das Gleiche machen wir auch bei der Y-Koordinate. Auf die Lösung sind wir nur mit ständigem probieren und testen gekommen.

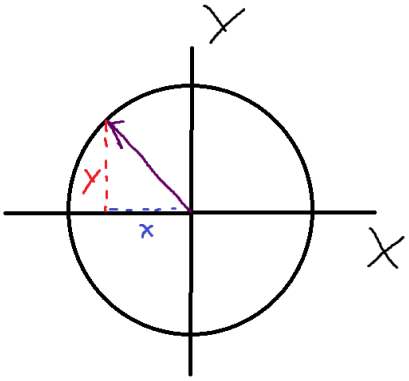


*Abbildung 18: Einheitskreis*

# 5. Schlusswort

Mit unserer Arbeit wollten wir herausfinden, welche Herausforderungen auf uns zukommen, wenn wir eine 3D Website erstellen, welche eine Timeline enthält. Zudem wollten wir wissen, ob es uns gelingt, ein solches Produkt mit Blender und Three.js zu erstellen.

Unser Projekt hat uns gezeigt, wie viele Probleme bei der Erstellung einer solchen 3D Website entstehen und dass viele Arbeiten sehr zweitaufwändig sind. Zum Beispiel ist das Modellieren von detailreichen Körpern sehr umfangreich und das Programmieren mit Three.js sehr komplex. Die grössten Herausforderungen entstanden vor allem, als wir unsere speziellen Anforderungen rund um das Thema der Kameraperspektiven programmieren wollten. Dabei haben wir viel im Internet recherchiert, um unsere Ideen zu implementieren und die auftretenden Probleme zu lösen.

Wir konnten unsere Absichten bez. Timeline mit den jeweiligen Meilensteinen erfolgreich umsetzen. Zudem sind auch die schwierigeren Anforderungen, wie z.B. einen Charakter selbst zu bewegen oder die FPV Drohne zu animieren, gelungen. Wir haben sehr viel über Blender und Three.js gelernt. Das war auch ein grosses Ziel, welches wir zu Beginn hatten.

Das konkrete Projekt könnte man weiter untersuchen, indem man deutlich komplexere Anforderungen umsetzen würde, wie z.B. mit der Schaffung einer Mehrspielerumgebung oder eines Kollisionsmanagements. Ohne zusätzliches Know-How wäre aber eine Umsetzung nicht realistisch.

Gleichzeitig empfehlen wir Firmen, für welche der Webauftritt eine wichtige Visitenkarte darstellt, den Weg in Richtung 3D zu suchen. Dabei lohnt es sich aufgrund der gemachten Erfahrungen, auf ein spezialisiertes 3D Team zurückzugreifen, damit der Auftritt innert vernünftiger Zeit und fairem Kostenrahmen umgesetzt werden kann.

Innerhalb unserer Arbeit haben ausser unserer 3D Webseite noch zwei Videos gemacht, welche unser Produkt zeigen und unsere Schwierigkeiten erklären. Um zu unserem Projekt, sowie auch den Videos zu gelangen, haben wir diese hier verlinkt.

**3D Webseite / Produkt:** https://bma.yanick-christen.com/

**Video zur Erklärung unseres Projektes:** https://youtu.be/NpZOwYWEqnY

**Video zur veranschaulichen unserer Schwierigkeiten:**

# 6. Glossar

| Name | Beschreibung |
|---|---|
| **Abstrakte Klasse** | Eine abstrakte Klasse kann man nicht wie eine normale Klasse instanziieren. Diese Klasse ist dafür gedacht, um anderen Klassen Vorgaben zu geben. In diesem Beispiel könnte eine abstrakte Klasse Gebäude heissen. Jedes Gebäude braucht eine Türe und darum auch eine Methode, welche die Türe implementiert. |
| **API** | API steht für Application Programming Iinterface und ist dafür gedacht, dass zwei Applikationen miteinander reden können. |
| **Browser** | Ein Browser ist ein Tool wie beispielsweise Chrome oder Firefox, mit welchen man das Internet durchsuchen kann. «Ein Browser - das ist Englisch und bedeutet so viel wie "Stöberer" - ist ein Programm, mit dem Websites korrekt angezeigt werden.» (Predan-Hallabrin 2019) |
| **Canvas** | Canvas ist ein tag, welchen man in Html definieren kann. Diesen tag kann dazu verwendet werden Figuren zu zeichnen. Um Diese Figuren aber wirklich zeichnen zu können, brauch es JavaScript. |
| **Css** | Css oder lang Cascading Style Sheets wird dafür benutzt, um die Inhalte von HTML schön zu gestalten. Dies kann auf viele Weisen getan werden. Zum Beispiel könnte man dem «h1» tag eine Hintergrundfarbe geben. |
| **Deploy** | Um ein Produkt online zu stellen, muss man sie Deployen. Auf Deutsch heisst deploy einfach einsetzen oder aufstellen. |
| **Draco Loader** | Mit einem Draco Loader, kann man files laden, welche im Draco Format sind. Dieses Format bekommen man, wenn man eine Draco compression macht. Dabei werden die Dateien wie z.B von Blender ein wenig kleiner. |
| **Framework** | Ein Framework ist wie auch schon eine Library eine Ansammlung von Funktionen, mit dem Unterschied, dass man bei einem Framework nicht entscheiden kann, wo und wann man das Framework benutzt, da das ganze Programm in diesem Framework sein muss. Wie der Name dies bereits erzählt, ist es ein Rahmen, in welchem man Arbeiten kann. (vgl Wozniewicz 2019) |
| **Frame** | Ein Browser ist ein Tool wie beispielsweise Chrome oder Firefox, mit welchen man das Internet durchsuchen kann. «Ein Browser - das ist Englisch und |

| | |
|---|---|
| | bedeutet so viel wie "Stöberer" - ist ein Programm, mit dem Websites korrekt angezeigt werden.» (Predan-Hallabrin 2019) |
| **Funktion** | Eine Funktion ist ein Stück Code, welches benannt wurde. Diese Funktionen kann man einfach mit dem Namen aufrufen. Eine Funktion ist dazu da, das Programm in Blöcke zu strukturieren. Ausserdem kann man so Redundanzen verhindern. |
| **Funktionale Programmierung** | Eine Funktionale Programmierung besteht hauptsächlich aus einfachen Funktionen. Der Code ist einfach zu lesen und zu testen. Darum beginnen viele Programmierer mit der Funktionalen Programmierung. |
| **Html** | Html oder lang Hypertext Markup Language, ist dazu gemacht, Inhalte auf Webseiten anzuzeigen. Ein Html Projekt kann man erstellen, in dem man ein File mit der Endung Html kreiert. Dieses File kann man danach in einem Browser öffnen. In Html wird alles in sogenannten Tags hineingeschrieben. |
| **Instanziieren** | Sobald man dieses Haus jetzt bauen wollte, könne man diese Klasse instanziieren. Dadurch bekommt man ein Objekt mit dem Typ Haus und schon hat man ein Haus. Um Eine Tür einzubauen, könnte man jetzt einfach die Methode Türe_einbauen aufrufen. |
| **JavaScript** | JavaScript ist neben Html und Css die dritte Grundkomponente für das Web. JavaScript ist eine Programmiersprache, welche für die Logik in einer Webseite zuständig ist. 90% der Webseiten wären ohne JavaScript statisch und mit einem normalen PDF vergleichbar. (vgl Megida 2021). Für JavaScript gibt es sehr viele Libraries und Frameworks, welche die Grundfunktionen von JavaScript erweitern. |
| **Klasse** | Eine Klasse kann man sich als einen Bauplan vorstellen. Als Beispiel: Wenn man ein Haus bauen möchte, hätte man die Klasse Haus, was die Bauanleitung des Hauses widerspiegelt. In dieser Bauanleitungen ist dann z.B. beschrieben, wie man eine Türe einbaut. Solche Instruktionen, werden als Methoden abgebildet als hätte man somit in der Klasse Haus die Methode Türe_einbauen. |
| **Library** | Eine Library oder auch Bibliothek ist eine Ansammlung von Funktionen. Sie helfen dabei Probleme einfacher zu lösen. Man kann in einem Programm frei |

| | |
|---|---|
| | entscheiden, wo und wann man die Funktionen einer Library benutzten kann. |
| **Methode** | Eine Methode ist sehr ähnlich wie eine Funktion. Der Unterschied ist, dass eine Methode immer ein Objekt braucht, um aufgerufen zu werden. In unserem Beispiel kann man keine Tür einbauen, ohne ein Haus zu haben. Also muss die Methode Türe_einbauen auf ein Haus aufgerufen werden. Anders wie in diesem Beispiel, sind Methoden meistens auf Englisch und werden darum klein geschrieben. Eine korrekte Namensgebung wäre also: «implementDoor» |
| **Objekt Orientiere Programmierung** | Die Objekt Orientiere Programmierung ist eine klassenbasierte Programmierung. Alles besteht aus Objekten, welche man übergibt und verarbeitet. Die Daten in den Objekten sind gekapselt, was heisst, dass sie nicht wissen, was in der «Aussenwelt» also im Programm sonst passiert. Damit ist Zugriff sehr kontrolliert. |
| **Orbit Controls** | Orbit Controls ist eine Library, die man verwenden kann, um seine Kamera zu kontrollieren. Damit kann man mit der Maus die Kamera bewegen. |
| **Rendering** | Rendering zu Deutsch Wiedergabe oder Übertragung, ist ein Prozess, um Bilder oder Modelle zu einem realistischen Bild zu übertragen. Dieses Geschehen passiert bei Three.js, mehrere Male in der Sekunde. |
| **Scene** | Um ein Produkt online zu stellen, muss man sie Deployen. Auf Deutsch heisst deploy einfach einsetzen oder aufstellen. |
| **Shader** | Ein Shader ist ein kleines Programm, welche die Scenes während dem rendern manipuliert.<br>«Sie werden verwendet, um jeden Scheitelpunkt einer Geometrie zu positionieren und jeder sichtbare Pixel dieser Geometrie einzufärben.» (Simon 2021). |

| | |
|---|---|
| **Tag** | Ein tag ist ein Schlüsselwort welches in kleiner/grösser-als Zeichen geschrieben wird. Dies kann danach ein Browser lesen und verstehen. Ein Beispiel ist: <h1> Hallo </h1>, was zu einem Titel wird, in welchem «Hallo» steht. (vgl Hayes 2020) |
| **Third Person** | Thrid Person ist eine Sicht, welche man sich so vorstellen kann, als würde eine andere Person dem gespielten Charakter von einer Distanz folgen und ihn dabei Filmen. Das Pendant dazu wäre First Person. |

# 7. Quellenverzeichnis

Abdobe (14 September 2021) How do I get access to Mixamo? URL: https://helpx.adobe.com/creative-cloud/faq/mixamo-faq.html (zuletzt besucht am 20.11.2021)

Bosnjak, Dusan (5. August. 2018) What is three.js? URL: https://medium.com/@pailhead011/what-is-three-js-7a03d84d9489 (besucht am 18.11.2020)

Dev Simon (16.11.2020) Simple Third Person Camera (using Three.js/JavaScript) https://www.youtube.com/watch?v=UuNPHOJ_V5o

Dillion Megida (29. March 2021) What is JavaScript? A Definition of the JS Programming Language URL: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/ (besucht am 24.11.2021)

Domainssaubillig (2021) Objektorientierte vs. funktionale Programmierung im Web URL: https://www.domainssaubillig.de/blog/blog-artikel/items/objektorientierte-vs-funktionale-programmierung-im-web.html (zuletzt besuch 24.11.2021)

Dr. Derek, Austin (2020 - 2021). Warum JavaScript eine "Multi-Paradigma" -Sprache ist URL: https://ichi.pro/de/was-sind-javascript-programmierparadigmen-69204631281627 (besucht am 16.11.2021)

Greggman (July 2021) Three.js Cameras URL: https://three.jsfundamentals.org/three.js/lessons/three.js-cameras.html (zuletzt besucht am 29.10.2021)

Haunschild, Marc Webseiten erstellen und veröffentlichen. Aufbau der Ordnerstruktur. Bodenheim 2017

Hayes Adam (13. November 2020) HyperText Markup Language URL: https://www.investopedia.com/terms/h/html.asp (besucht am 24.11.2021)

Mixamo (2021) mixamo Characters URL: https://www.mixamo.com/#/?page=1&type=Character (besucht am 20.11.2020)

Predan-Hallabrin Beatrice (31.08.2019) Was ist ein Browser? Einfach erklkärt. URL: https://praxistipps.chip.de/was-ist-ein-browser-einfach-erklaert_41369

Simon, Bruno (2019) The ultimate Three.js course URL: https://three.js-journey.com/ (zuletzt besucht am 18.11.2021)
Simon Bruno (2021) https://threejs-journey.com/lessons/27 (zuletzt besuch 24.11.2021)

Three.js (29 Oktober 2021a) Three.js Offizielle Webseite URL: https://three.js.org/ (zuletzt besucht am 18.11.2021)

Three.js (29 Oktober 2021b) Three.js examples URL: https://three.js.org/examples/#webgl_animation_skinning_blending (zuletzt besucht am 20.11.2021)

Three.js (29.Oktober 2021c) Camera URL: https://three.js.org/docs/?q=camera#api/en/cameras/Camera (Zuletzt besucht am 29.10.2021)

Turtorialspoint (2021) WebGL – Drawing a Triangle URL: https://www.tutorialspoint.com/webgl/webgl_drawing_a_triangle.htm (zuletzt besucht am 29.10.2021)
Wozniewicz Brandon (1. Februar 2019) The Difference Between a Framework and a Library URL: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/ (besucht am 24.11.2021)

# 8. Abbildungsverzeichnis

Alle Bilder, welche wir in dieser Arbeit verwenden, sind selber erstellte Bilder und benötigen darum keinen Verweis auf eine URL.

# 9. Anhang