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 bei dem Erstellen einer 3-dimensionalen Webseite entstehen.

Um dies zu untersuchen, wird eine eigene 3-dimensionale Webseite erstellt. Dies wird Mithilfe von einer Library namens Three.js und einem open Source Tool namens Blender realisiert. Zuerst werden Grundkenntnisse angeeignet, woraufhin mit dem Programmieren und Implementieren begonnen wird. Die Schwierigkeiten, welche dabei aufkommen werden, dokumentiert.

Im Ergebnis sieht man, dass das Realisieren eines 3D Projektes sehr zeitaufwändig ist und man ein grosses Wissen und viel praktische Erfahrung haben muss, um eine brauchbare 3D Webseite erstellen zu können Man erkennt, dass eine solche Website im Markt Sinn macht, sofern es richtig eingesetzt wird, da man sich von der Konkurrenz damit deutlich abheben kann. Der grösste Nachteil ist, dass es sehr viel Zeit in Anspruch nimmt und dadurch auch die Kosten erheblich erhöht werden.

Bei diesem Produkt wird herausgefunden, welche Schwierigkeiten aufkommen, wenn man eine 3D-Website erstellt.

Jeffry's (BMA Partner) Abstract:

Es wird mit Blender gearbeitet, um 3D Körper zu modellieren und zu animieren. Three.js wird benutzt, um die 3D Körper in die Website zu implementieren, die ganze Logik hinter dem Projekt zu konstruieren und online darzustellen. Aus dem Grund haben wir die Arbeit so aufgeteilt, dass jemand mit Blender arbeitet und jemand mit Three.js. Da wir jedoch keine Erfahrungen in diesen Bereichen hatten, haben wir mit Hilfe von Kursen, Kollegen und dem Internet, Schritt für Schritt das Framework sowie Blender erlernt.

Es zeigt sich, dass das Projekt sehr zeitaufwändig ist und man viel wissen muss, bevor man eine brauchbare 3D Website erstellen kann. Wir erkennen, dass eine solche Website im Markt Sinn macht, sofern es richtig eingesetzt wird, da man sich von der Konkurrenz damit deutlich abheben kann. Der grösste Nachteil ist, dass es sehr viel Zeit in Anspruch nimmt und dadurch auch die Kosten erheblich erhöht werden.

Inhalt

1. Einleitung

Unsere Arbeit beschäftigt sich mit dem Thema 3-dimensionale Webseiten zu kreieren. Herausfinden wollen wir, welche Schwierigkeiten dabei entstehen. Viele Konzerne investieren viel Zeit und Geld damit, ihre Webseite möglichst gestalterisch und imposant für den Benutzer zu Gestalten. Das Design von Webseiten verändert sich ständig und geht mit der Zeit, woher die Verbindung zu dem Themenbereich «Zeichen der Zeit» kommt.

Die Leitfrage in unserer Arbeit ist, welche Herausforderungen bei dem Erstellen einer 3-dimensionalen Webseite entstehen. Die Hypothese ist, eine solche Webseite erstellen zu können, welches anhand von unserem Projekt überprüft wird. In unserer Umsetzung werden wir die verschiedenen Meilensteine, anhand einer Zeitlinie darstellen. Die Meilensteine sind kurze Videos, welche Jeffry Dahinden für seine eigene Firma erstellt hat.

Die ständige Entwicklung, des Visuellen Aspektes, fasziniert uns sehr. Dazu gehören insbesondere Foto- und Videografie, Animationen, Zeichnungen und Modelle. In einer Webseite kann man all dies zusammenfügen und öffentlich für das Internet und somit für die ganze Welt machen. In unserem Beruf als Informatiker haben wir auch beide einen Bezug zu der Erstellung von Webseiten. Jedoch ist der Aspekt, des 3-dimensionalem für uns beide neu.

In dem Projekt wird eine JavaScript Library namens Three.js, welches bei der Erstellung von 3D Webseiten unterstützen soll, verwendet. Ausserdem wird ein Open Source Tool namens Blender benutzt, was dazu dient Figuren und Objekte zu Modellieren.

Da das Programmieren einer solcher Webseite ein gewisses Grundwissen vorgibt, wird in dieser Arbeit zuerst eine Erklärung, zu den einzelnen Komponenten gegeben, welche in unserer BMA verwendet wurden. Im Kapitel 4 wird beschrieben, was unser Vorgehen war, sowie auch die Erklärung der einzelnen Komponenten.

Jeffry's Einleitung :
Mit unserer Arbeit wollten wir herausfinden, welche Herausforderungen kommen, wenn wir eine 3D Website erstellen, welches eine Timeline erhält. Zudem wollten wir wissen, ob es uns geling so ein Produkt mit Blender und Three.js zu erstellen.

Unser Produkt hat uns gezeigt, wie viele Probleme eine solche 3D Website hervorhebt und welche Arbeiten sehr zweitaufwändig und komplex sind. Zum Beispiel ist das Modellieren von Detailreichen Körpern sehr zeitaufwändig, während dem das Programmieren mit Three.js sehr komplex ist. Die grössten Herausforderungen kamen vor allem, als wir unsere speziellen Anforderungen wie z.B. Kamera perspektiven einstellen, programmieren wollten. Vieles ist verbunden mit, im Internet nachschauen, implementieren und Problem lösen.

Wir finden unser Produkt ist gut gelungen. Wir konnten es umsetzten, dass eine Timeline besteht mit den einzelnen Meilensteinen. Zudem sind auch die schwierigeren Anforderungen, wie z.B., dass man einen Charakter selbst bewegen kann oder dass die FPV Drohne animiert ist, vorhanden auf der 3D Webseite. Wir beide haben sehr viel über Blender und Three.js gelernt. Das war auch ein grosses Ziel, welches wir am Anfang vom Projekt hatten.

Das Thema könnte man weiter untersuchen, indem man sehr schwierige Anforderungen umsetzt. Ein Beispiel wäre, dass wenn man auf der Website auf eine Plattform laufen würde, dass dann ein Popup kommt mit Bildern und Texten. So müsste man auf der Webseite einen Koordinaten Scanner einbauen, der das durchgehend prüft. Man könnte unser Produkt auch optimieren. Der Ladescreen z.B. ladet momentan sehr lange.

2. Was ist Three.js?

In diesem Kapitel werden wir versuchen, Ihnen 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 eine 3d Webseite realisiert wie zum Beispiel SpaceX, Google, Gucci und noch viele mehr. Damit auch wir so eine 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 aber gerade 100 verschiedene Leute und 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 ein Canvas.

Selbst mit WebGl zu programmieren, wäre aber sehr schwierig. Wir wollen darum in dieser 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, wobei diese 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, welches in Three.js existiert 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 Camera Klasse erben. Alle Kameras haben andere Funktionen und sind für die verschiedensten Anwendungsfälle benutzbar. Wie zum Beispiel 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 benutze, Kamera. Mit dieser Kamera hat man eine 3D Sicht was bedeutet, dass Objekte, welcher 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

(Bild von allen Geometrys)

Einfach gesagt ist ein Mesh einfach 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 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. Der Figur des Tisches ist dann die Geometrie und z.B Eichenholz wäre das Material. Ein Mesh wäre dann der gesamte Tisch.

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 setzten.	Dieses Licht, kommt von jeder Richtung und wird damit benutzt, jedes Objekt and jeder Fläche gleich hell zu Beleuchten.
2	Directional Light	Bei diesem Licht kann man die Farbe, die Stärke und die Position des Lichtes setzten.	Man kann sich dieses Licht wie die Sonne vorstellen. Es strahl wie auch 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 setzten.	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. Dieses Beispiel, 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 setzten. Ausserdem kann man definieren, wie stark das Licht abnimmt. Dies ist dafür gedacht, dass ein Objekt, welche weiter weg vom Licht entfernt ist, weniger hell erscheint wird als das andere.	Dieses Licht ist einfach ein Punkt, welcher in alle Richtungen Licht abgibt.
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 hin scheinen soll.	Dieses Licht, soll einen Scheinwerfer imitieren, man kann sagen, wie gross diese 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 Spot Licht 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 Leute 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 euch hier vorstellen möchten heisst Cannon.js. Aber auch mit Cannon.js ist es nicht gerade einfach Physik zu implementieren. Wir werden euch kurz das Prinzip dahinter erklären:

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

3. Blender

2.1 Was ist Blender?

Blender ist eine kostenlose Software, bei welchem man Körper modellieren, textieren und animieren kann. Zusätzlich kann man Physik Simulation Tools brauchen wie Rauch, Feuer, Partikel und Wasser. Blender ist mit verschiedenen Softwares verbunden wie z.B. After Effects oder Unity, um so auch das Animieren oder das Programmieren von Videospielen einfacher ist. Zudem ist es Open Source und man kann kleine bis grosse 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 die gesamte 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 dem Linux, Windows und Macintosh Betriebssystemen. Es verwendet die Benutzeroberfläche OpenGL, damit das Verwenden der Software einheitlich ist.

2.2 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 nah der Realität steht.
Bei Websites muss man schauen, dass man ein Framework benutz, welches Blender erhält. Three.js, welches wir auch brauche, ist z.B. eines welches oft gebraucht wird und auch professionell.

2.3 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 werden zeigen wir, wie wir vorgegangen sind und wie unsere Fortschritte ausgesehen haben.

3.1 Vorgehen Blender

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

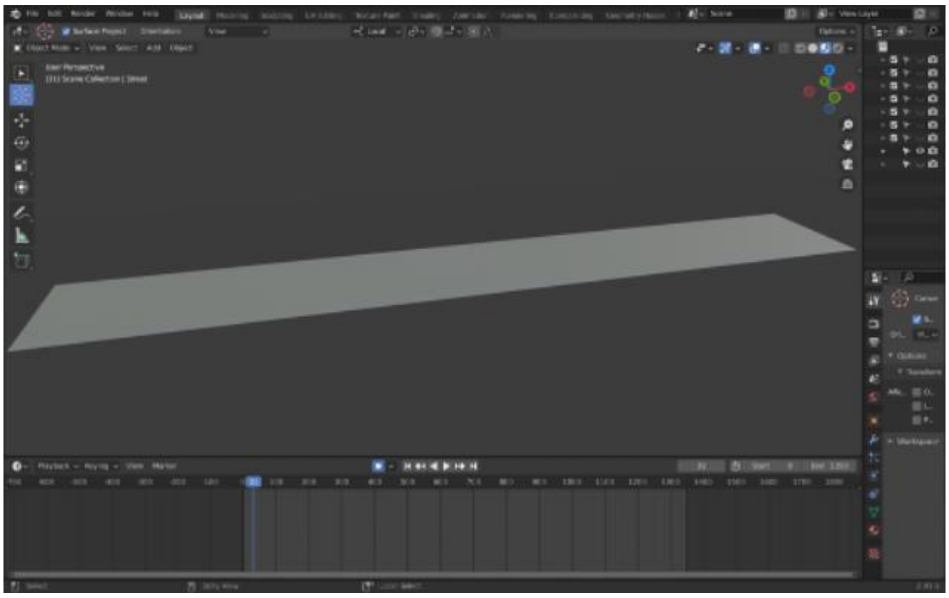


Abbildung 1: Strasse - Geomtrie

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

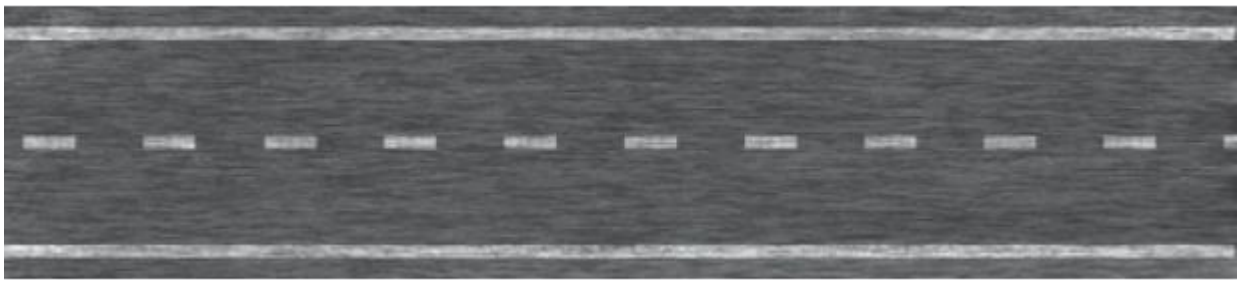


Abbildung 2: Strasse - Textur

Nachdem wir die Strasse in Blender hinzugefügt haben und es dann richtig auf den Boden positionierten, 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 sieht man noch eine Sony Kamera welches das Auto fotografiert. Wenn man auf den Bildschirm drückt, wir das Video, welches Jeffry produziert hat, angezeigt

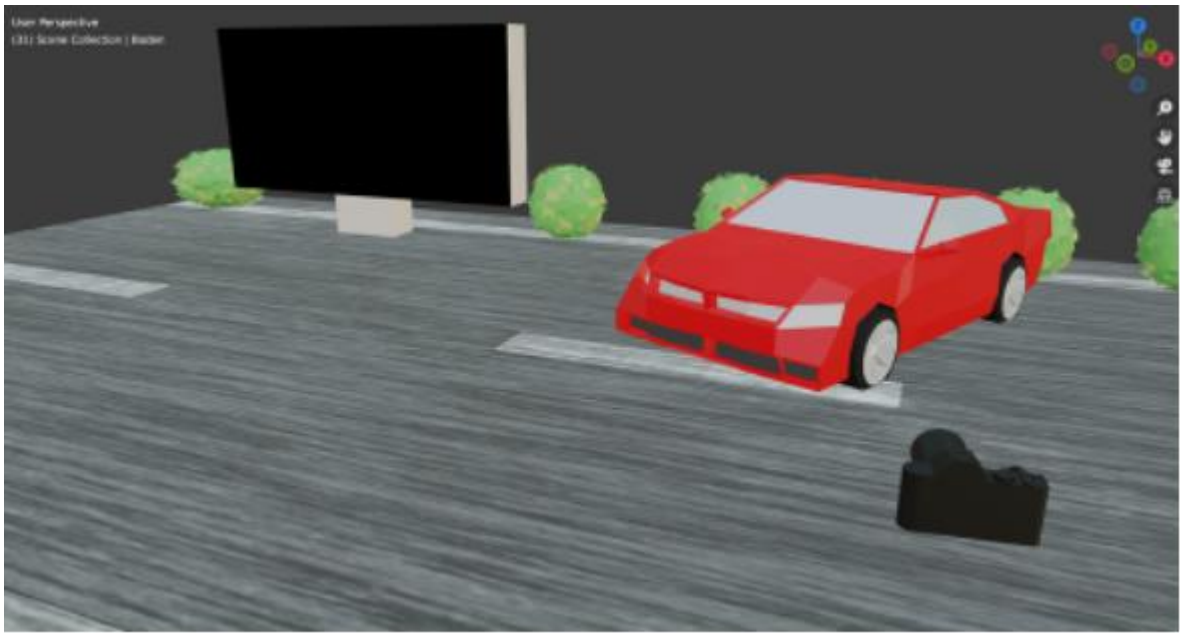


Abbildung 3: Erster Meilenstein

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. Neben dem sieht man mein Arbeitstisch mit meinem gebogenen Bildschirm, den ich besitze. Es war sehr schwierig den Bildschirm wegen der Form zu modellieren. Ich brauchte Hilfe aus dem Internet und Schlussendlich konnte ich es durch ein YouTube Video erstellen. Man hat viele Modifikatoren gebraucht, welches für mich sehr neu war.

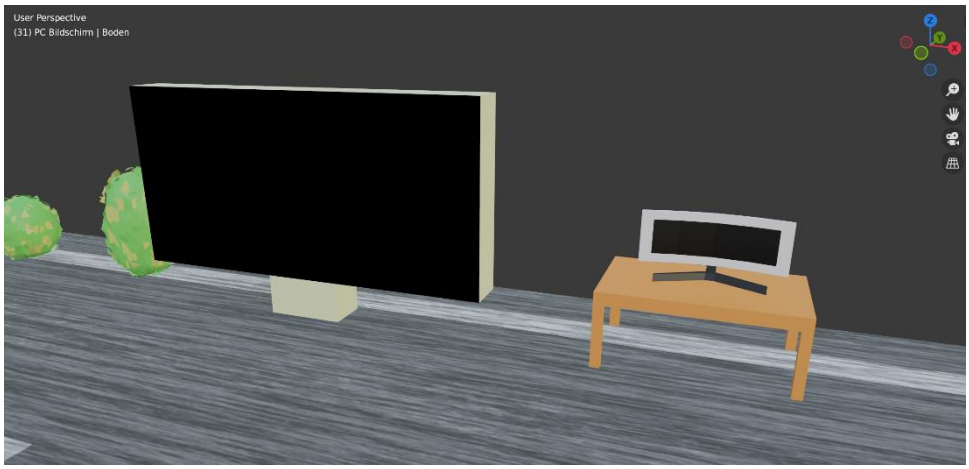


Abbildung 4: Zweiter Meilenstein

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. In dem Video 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. Wenn man auf den grossen Bildschirm klickt, dann wird ebenso das prduzierte Video angezeigt.

Abbildung 5: 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.

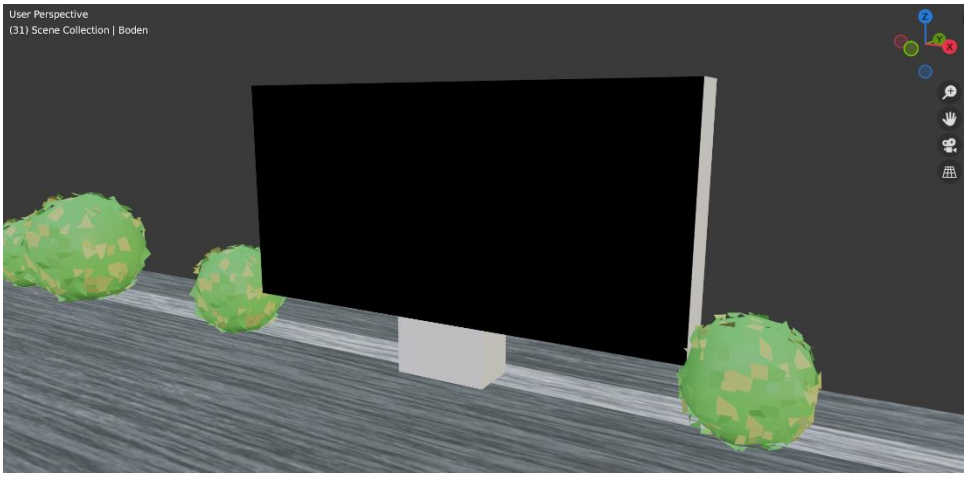


Abbildung 6: Vierter Meilenstein

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 an diesem Meilenstein schon ein gutes Knowhow hatten, haben wir es ohne Hilfe vom Internet modellieren können. Diese Drohne haben wir zusätzlich animiert. Die Dafür mussten wir jeden Frame einzeln animieren. Das war eine sehr kreative und aufwändige Aufgabe.

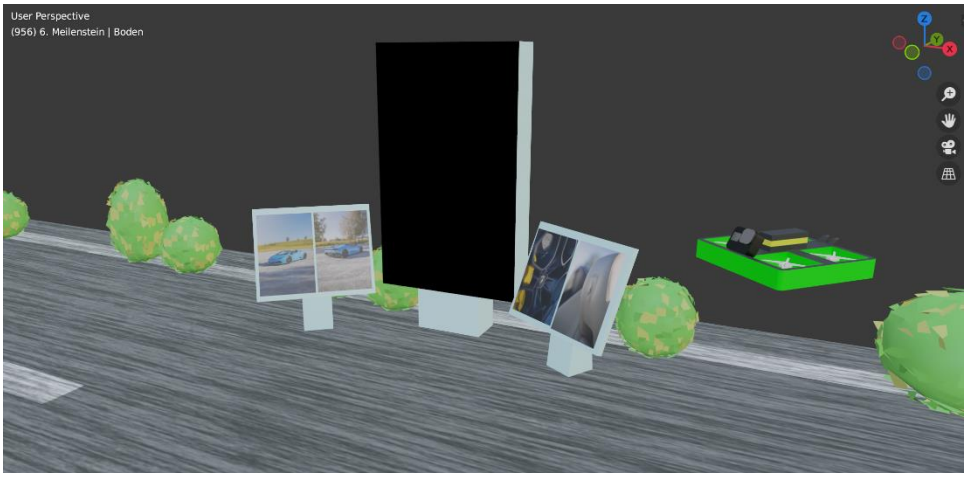


Abbildung 7: Fünfter Meilenstein

Das ist der letzte Meilenstein. Bei dem geht es darum, dass Jeffry das erste Mal ein Aftermovie im Club erstellt hat 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 geschossen worden sind.

Abbildung 8: Sechster Meilenstein

3.2 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 auch Probleme und Schwierigkeiten, die wir in dieser Zeit hatten.

3.2.1 Vorbereitung und Start der Arbeit

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

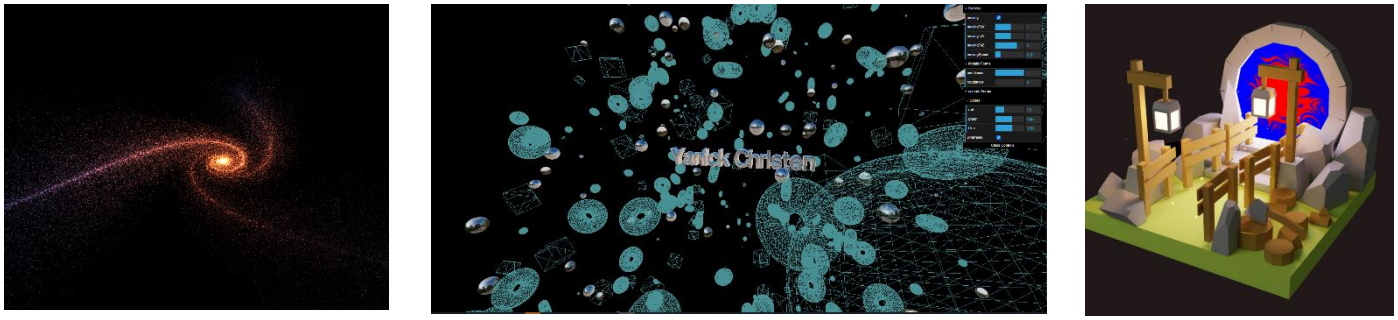


Abbildung 9: Three.js – Kurs

Besonders bei dem Portal, hatte ich jedoch ein wenig Bange, dass wir es zeitlich nicht schaffen. 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 erkennte.

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.

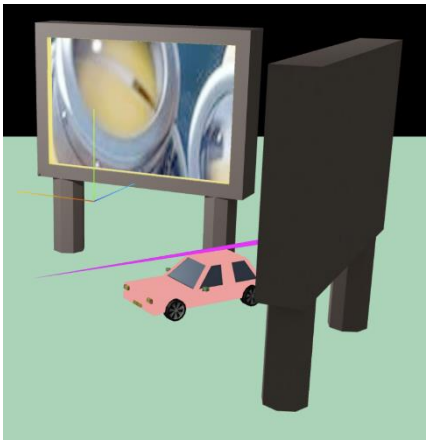


Abbildung 10: Test-Projekt

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

3.2.2 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 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 unserem Projekt.

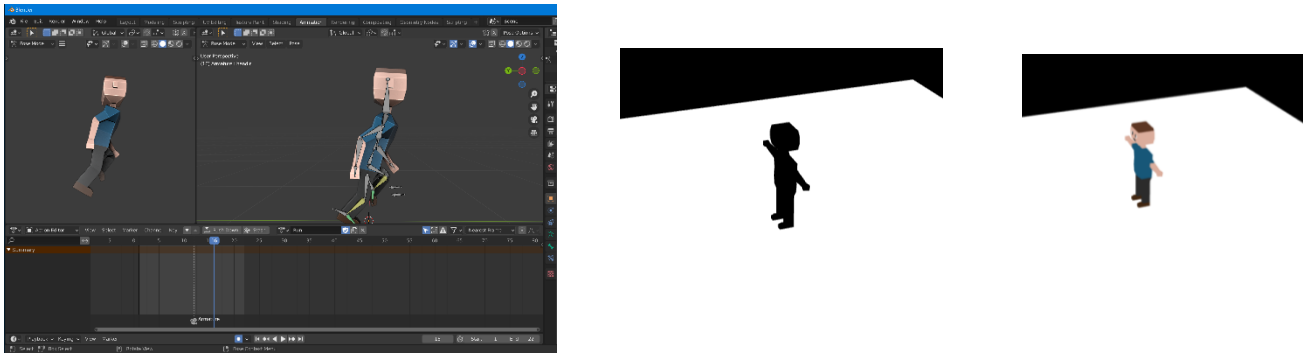


Abbildung 11: Charakter

Danach haben wir versucht, dass die Figur sich auch auf Tastendruck bewegen kann. Da wir bereits JavaScript Kenntnisse haben, war es nicht sehr schwierig bei einem Tasten klick einen Event zu starten. Damit konnten wir die Figur immer an eine neue Position schicken. Jedoch war es ein schwieriges Empfangen, die Person so bewegen zu können, wie wir es gerne hätten. Alle Schwierigkeiten kann 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 vorwärtskommt. Dadurch stoppte die Person immer wieder kurz und das ganze sah sehr unschön aus. Dies konnten wir nach einiger Zeit lösen, indem wir darauf gehört haben, welche Taste gerade gedrückt ist.

Gleichzeitig nach vorne laufen und die Richtung ändern:
JavaScript kann jeweils sich 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, meiner Meinung nach, schlaue Lösung gefunden, die den State der Taste in einem Objekt speichert.
In welche Richtung muss die Person gehen?
Da sich die Person in eine Koordinate bewegt, ist es schwierig einem zu sagen, die Person soll nach links gehen. Zuerst muss man berechnen, wo Links relativ zu dieser Person ist.

Animationen:
Sobald die Person anfangt zu Laufen, muss die andere Animation aufhören und danach soll die Lauf Animation starten. Das Umgekehrte gilt, wenn die Person aufhört zu laufen.

3.2.3 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 dem Implementieren gab es kleinere Schwierigkeiten, jedoch nichts Erwähnens Wertes.

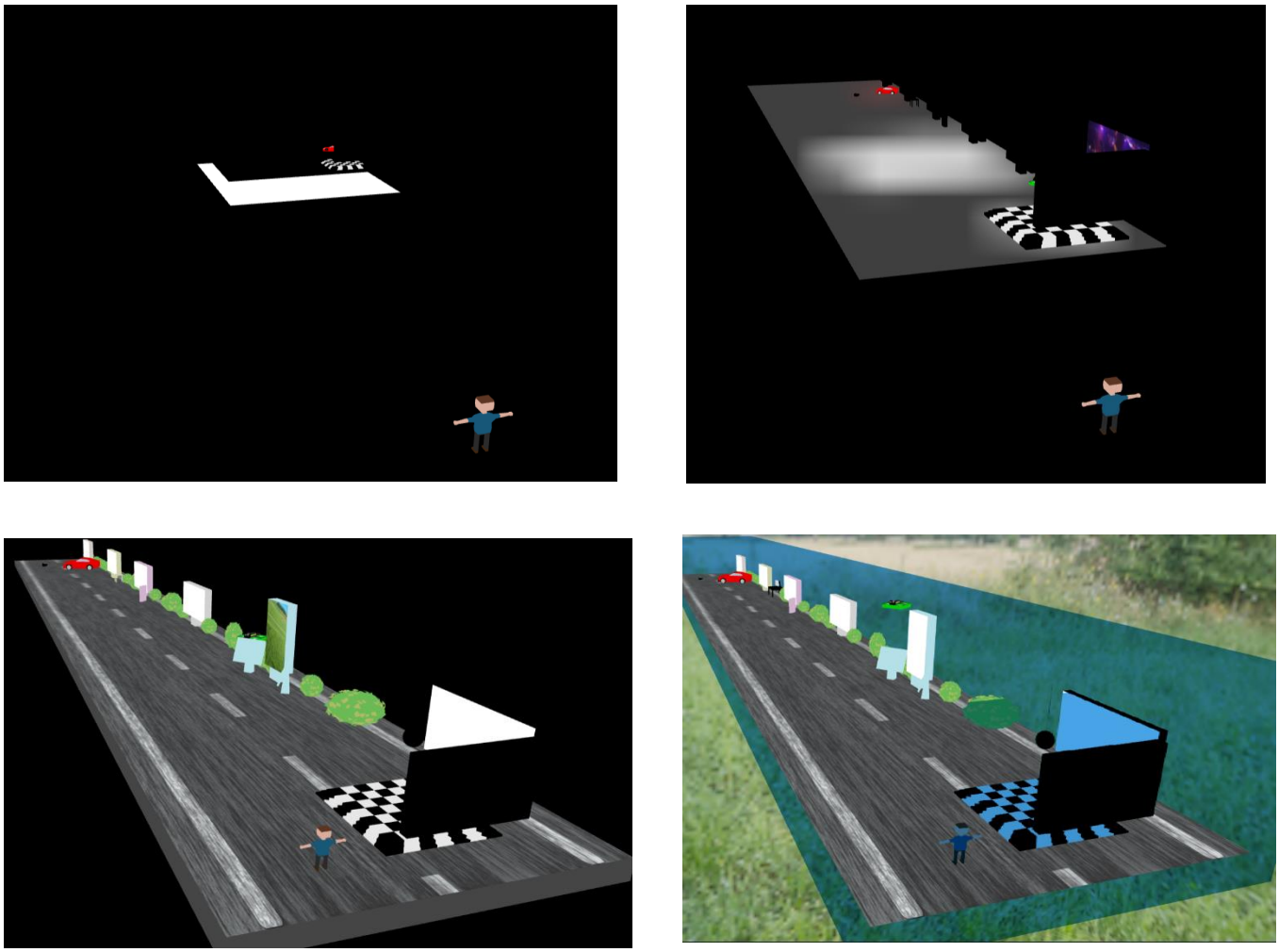


Abbildung 12: Fortschritte, der Scenen

3.2.4 Kamera

Wir wollten, dass die Kamera dem Charakter folgen kann, wie man es bei einem Thrid 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 konnte. Wir fanden auch ein Video, doch dieses half uns nicht weiter. Also haben wir selbst versucht unsere Schwierigkeiten zu lösen. Wir man in unserem Erklärungs-video 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 switchen. Dies zu Lösen war recht schwierig. Lange haben wir nicht gewusst, dass das sogenannte Orbit Controls nicht benutzen dürfen, wenn wir wollen, dass die Kamera an eine Bestimme Position schaut. Doch schlussendlich konnten wir auch diese Hürde überwinden.

3.2.5 Lade Fenster

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

3.2.6 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 dem schlechten 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.

3.2.7 Implementierung des Soldaten

Nachdem wir bereits mit einem grossen Teil unseres Programmes fertig waren, wollten wir unsere Figur verschönern. Zuerst wollten wir wieder selber eine Figur in Blender modellieren, jedoch wussten wir, dass dies uns viel Zeit kosten würde. Ausserdem wollten wir in unserer Arbeit schauen, wie schwierig es ist, eine 3D Webseiten zu Programmieren und so könnten wir schauen, was der Unterschied ist, wenn man eine eigene Figur macht, oder eine externe Figur implementiert. Hierfür haben wir eine Figur von der Offiziellen Three.js Webseite ist, 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 passenden auswählen, diese herunterladen und sie dann relativ einfach in einem Projekt verwenden. Das Positive daran ist, dass eine solche Figur unglaublich detailliert gemacht wurde und sehr realistisch aussieht. Auf die Animationen laufen perfekt. Als Kontra könnte man argumentieren, dass die Freiheit eingeschränkt wird. Jedoch kann an die Figur in Blender nach Belieben verändern und muss sich daran nicht an den style von Mixamo’s Figuren halten. Da unsere Webseite aber nicht in einem allzu grosse Rahmen ist, haben wir uns entschieden, dass sich der Aufwand nicht lohnt, diese Figuren zu abändern und haben die Figur so genommen, wie sie ist.

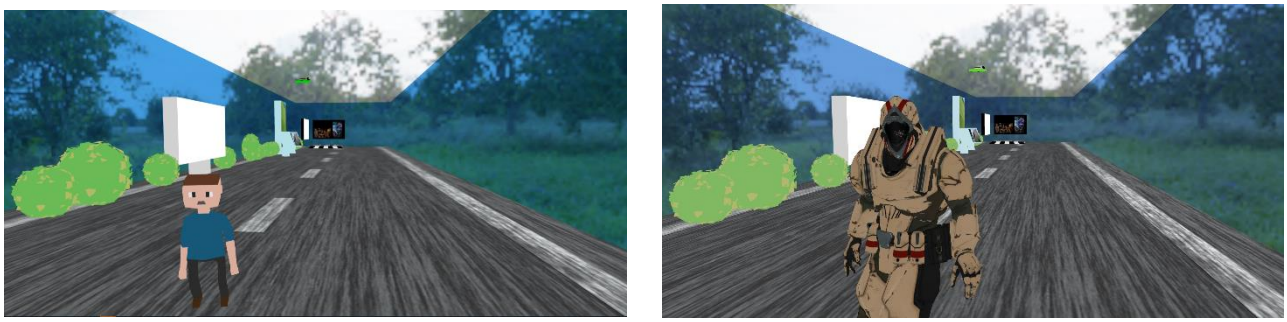


Abbildung 13: Unterschied der beiden Figuren

Three.js anstatt Mixamo

Gefunden haben wir die Figur von 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 einten existieren auf diesem Beispiel bereits drei Animationenm 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 um konvertieren, dies wäre aber ein wenig aufwändig. Als letzten Punkt, hat das File von Mixamo eine Grösse von 6424 Kilo Byte, wobei das File von Three.js nur 2110 Kilo Byte hat, also ist es dreimal kleiner. Dies hilft dabei, dass die Webseite weniger lange laden muss.

Rechte:

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

3.3 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 entscheiden. Objektorientierte Programmierung zu verwenden. Der Grund war, dass man so den Code auf mehrere Dateien teilen und auch einfach auf die einzelnen Dateien zugreifen konnte.

Hier werden wir einen Einblick darauf geben, was die einzelnen Files oder auch Klassen für Funktionen hat. Da wir in unserem Projekt über 800 Zeilen Code geschrieben haben, werden wir nicht auf jede Methode und auf jedes Objekt eingehen, da dies unserer Meinung nach den Ramen sprengen würde.

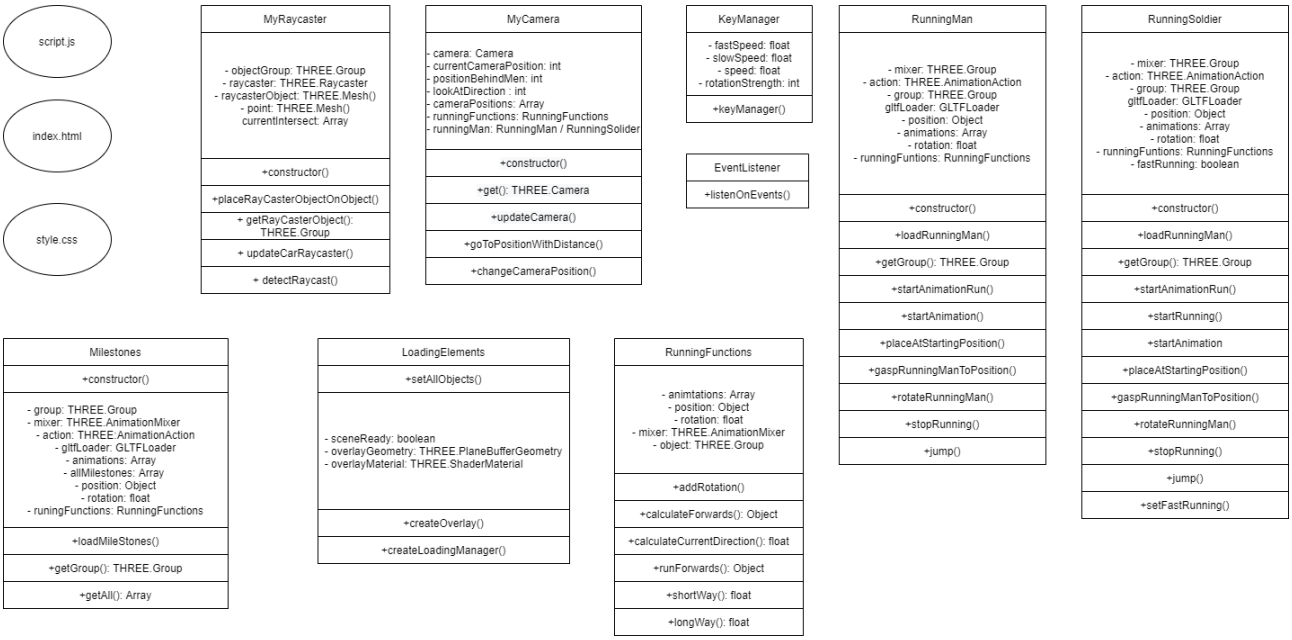


Abbildung 14: Klassendiagram

3.3.1 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 auch unseren «Draco Loader». Zudem wird der Hintergrund gesetzt. Damit man auch unsere Meshes auf dem Bildschirm sehen können, geben wir diese unserer Scene auch gleich mit. Was ebenfalls ein wichtiger teil ist, ist der Renderer, erklärt im Kapitel 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. Zum einten haben wir eine Funktion namens «resize», welche aufgerufen wir, sobald sich das Browser Fenster verändert. Dort werden die Kamera und der Renderer neu gesetzt. Die andere Funktion, welche wir haben, ist 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 welcher jeder, der bereits eine Webseite geschrieben hat kennt. Wenn man eine Webseite schreibt, sollte man im Wurzelverzeichnis oder auch Stammordner (engl.: root directory) eine Datei namens index.html anlegen, da dieses File immer das erste ist, welches vom Browser geladen wird. (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.

3.3.2 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 drin. Die Klasse an sich ist aber recht simple aufgebaut. Alles was in dieser Klasse passiert, ist das Laden der Blender Datei. Wie man ein Blender File in Three.js implementiert, wird im Kapitel blablabla dokumentiert. 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 man ein Blender File in Three.js implementiert, wird im Kapitel blablabla dokumentiert. Wie auch in der «MileStones» Klasse, setzen wir die Rotation und Grösse. Zusätzlich Platzieren wir den Charakter aber 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 für das noch einige Methoden implementiert. Ausserdem muss die Person je nach seinem 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 neu die externe Militärs Figur implementiert wird. Diese sieht um einiges besser aus und auch die Animationen wirken viel schöner. Der Soldat kann nicht mehr spring, dafür gibt es aber einen Unterschied zum Laufen und Rennen.

Wir haben uns dazu entschieden, dass wir beide Klassen lassen, da man so sehr einfach 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.

3.3.3 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 Fenster Grösse geändert wurde, wird die Methode «resize» aufgerufen, womit sich 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 gerückt 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 ist heruntergedrückt	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 dich 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 von der Kamera. Als erstes, folgt die Kamera die Person 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 Strasse. 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 verwenden, 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 anaschauen und geniessen kann. Leicht verwechselbar ist die Bezeichnung unserer Klasse «MyCamera» und der echten Three.js Kamera. Unsere Klasse beinhaltet zum einten 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 auch in welche Richtung die Kamera schaut.

Damit man auch auf die normale Three.js Kamera in unserer Klasse zugreifen kann, haben wir eine «get» Methode, 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 in dieser Programmiert, dass bei der Sicht Wechsel der Kamera, die Kamera auch wirklich seine Position ändert.

MyRaycaster

In der «MyRaycaster» Klasse muss man, wie auch in der «MyCamera» Klasse, von dem Three.js Raycaster und unserer Klasse unterscheiden. Raycaster ins Deutsche übersetzt, heisst Strahler. Man kann man diesen Raycaster als einen unsichtbaren und unendlich langen Strahl, 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 auch einen Kegel, welchen wir dann sehr lange ziehen. Diesen Kegel soll auf der gleichen Position sein und in die gleiche Richtung schauen, wie der Strahl. Dieser Kegel sollte Grün werden, sobald unser Strahl und damit auch unser Charakter ein anderes Objekt berührt.

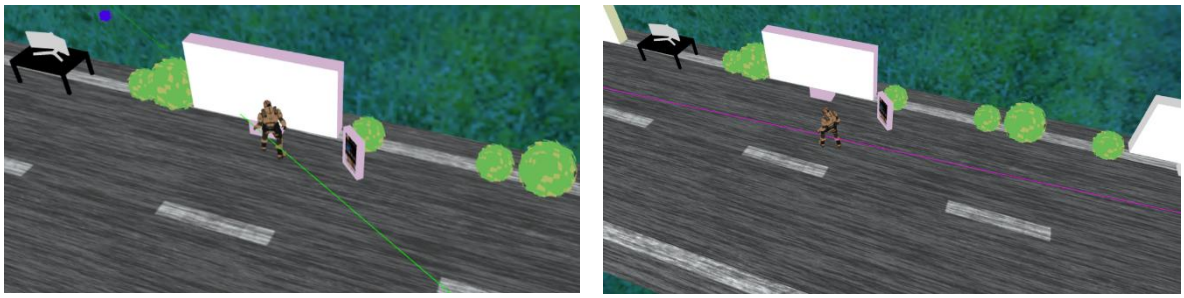


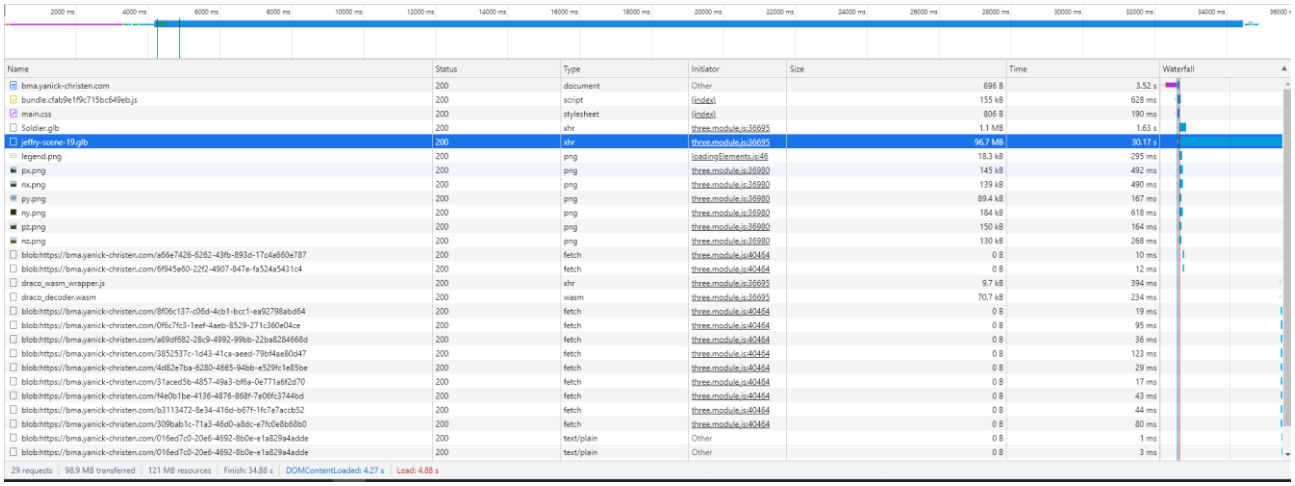
Abbildung 15: Raycaster in Sichtbarer Form

LoadingElements

In dieser Klasse geht es darum den Ladebalken, welchen man zuerst zu Gesicht bekommt, sobald man auf unsere Webseite geht. Dieser Ladebalken ist recht wichtig, da zum Beispiel unsere Figur schneller laden kann als die Strasse. Dies würde bedeuten, dass man zu Beginn die Figur, welche frei in der Luft steht, sehen würde und die Strasse nicht.

Im Prinzip haben wir mittels Html, Css und Photoshop den Lade-balken, 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 Opazität verringern können.

In Three.js existiert zum Glück einen Lade Manager, welcher uns sagen kann, 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.



Name	Status	Type	Initiator	Size	Time	Waterfall
bma.yanick-christen.com	200	document	Other	696 B	3.52 s	
bundle.cfab9e1f9c715bc649eb.js	200	script	(index)	155 kB	628 ms	
main.css	200	stylesheet	(index)	806 B	190 ms	
Soldier.glb	200	xhr	three.module.js:36695	1.1 MB	1.63 s	
jeffry-scene-19.glb	200	xhr	three.module.js:36695	96.7 MB	30.17 s	
legend.png	200	png	loadingElements.js:46	18.3 kB	295 ms	
px.png	200	png	three.module.js:36980	145 kB	492 ms	
nx.png	200	png	three.module.js:36980	139 kB	490 ms	
py.png	200	png	three.module.js:36980	89.4 kB	167 ms	
ny.png	200	png	three.module.js:36980	184 kB	618 ms	
pz.png	200	png	three.module.js:36980	150 kB	164 ms	
nz.png	200	png	three.module.js:36980	130 kB	268 ms	
blob:https://bma.yanick-christen.com/a66e7426-6262-43fb-893d-17c4a660e787	200	fetch	three.module.js:40464	0 B	10 ms	
blob:https://bma.yanick-christen.com/6f945e60-22f2-4907-847e-fa524a5431c4	200	fetch	three.module.js:40464	0 B	12 ms	
draco_wasm_wrapper.js	200	xhr	three.module.js:36695	9.7 kB	394 ms	
draco_decoder.wasm	200	wasm	three.module.js:36695	70.7 kB	234 ms	
blob:https://bma.yanick-christen.com/8f06c137-c06d-4cb1-bcc1-ea93798abd64	200	fetch	three.module.js:40464	0 B	19 ms	
blob:https://bma.yanick-christen.com/0f6c7fc3-1eef-4aeb-8529-271c360e04ce	200	fetch	three.module.js:40464	0 B	95 ms	
blob:https://bma.yanick-christen.com/a69df682-28c9-4992-99bb-22ba8284668d	200	fetch	three.module.js:40464	0 B	36 ms	
blob:https://bma.yanick-christen.com/3852537c-1d43-41ca-aeed-79bf4ae80d47	200	fetch	three.module.js:40464	0 B	123 ms	
blob:https://bma.yanick-christen.com/4d82e7ba-6280-4665-94bb-e529fc1e85be	200	fetch	three.module.js:40464	0 B	29 ms	
blob:https://bma.yanick-christen.com/31aced5b-4857-49e3-bf6a-0e771a6f2d70	200	fetch	three.module.js:40464	0 B	17 ms	
blob:https://bma.yanick-christen.com/f4e0b1be-4136-4876-868f-7e0f6fc3744bd	200	fetch	three.module.js:40464	0 B	43 ms	
blob:https://bma.yanick-christen.com/b3113472-8e34-416d-b67f-1fc7a7accb52	200	fetch	three.module.js:40464	0 B	44 ms	
blob:https://bma.yanick-christen.com/309bab1c-71a3-46d0-a8dc-e7fc0e8b68b0	200	fetch	three.module.js:40464	0 B	80 ms	
blob:https://bma.yanick-christen.com/016ed7c0-20e6-4692-8b0e-e1a829a4adde	200	text/plain	Other	0 B	1 ms	
blob:https://bma.yanick-christen.com/016ed7c0-20e6-4692-8b0e-e1a829a4adde	200	text/plain	Other	0 B	3 ms	

29 requests | 98.9 MB transferred | 121 MB resources | Finish: 34.88 s | DOMContentLoaded: 4.27 s | Load: 4.88 s

Abbildung 16: Dev-Tools von Chrome:

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.

RunningFunctions

Da wir mehrere Klassen für eine Person benutzen, ist es simpler, 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 viel die Person in die X Richtung 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.

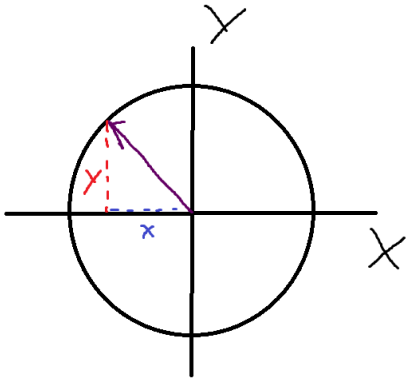


Abbildung 17: Einheitskreis

5. Glossar

Nr	Name	Beschreibung
1	Plane	
2	Orbit Controls	
3	Html	
4	Css	
5	JavaScript	
6	Third Person	
7	Library	
8	Browser	
9	API	
10	Renderen	

6. Schlusswort

Mit unserer Arbeit wollten wir herausfinden, welche Herausforderungen kommen, wenn wir eine 3D Website erstellen, welches eine Timeline erhält. Zudem wollten wir wissen, ob es uns geling so ein Produkt mit Blender und Three.js zu erstellen.

Unser Produkt hat uns gezeigt, wie viele Probleme eine solche 3D Website hervorhebt und welche Arbeiten sehr zweitaufwändig und komplex sind. Zum Beispiel ist das Modellieren von Detailreichen Körpern sehr zeitaufwändig, während dem das Programmieren mit Three.js sehr komplex ist. Die grössten Herausforderungen kamen vor allem, als wir unsere speziellen Anforderungen wie z.B. Kamera perspektiven einstellen, programmieren wollten. Vieles ist verbunden mit, im Internet nachschauen, implementieren und Problem lösen.

Wir finden unser Produkt ist gut gelungen. Wir konnten es umsetzten, dass eine Timeline besteht mit den einzelnen Meilensteinen. Zudem sind auch die schwierigeren Anforderungen, wie z.B., dass man einen Charakter selbst bewegen kann oder dass die FPV Drohne animiert ist, vorhanden auf der 3D Webseite. Wir beide haben sehr viel über Blender und Three.js gelernt. Das war auch ein grosses Ziel, welches wir am Anfang vom Projekt hatten.

Das Thema könnte man weiter untersuchen, indem man sehr schwierige Anforderungen umsetzt. Ein Beispiel wäre, dass wenn man auf der Website auf eine Plattform laufen würde, dass dann ein Popup kommt mit Bildern und Texten. So müsste man auf der Webseite einen Koordinaten Scanner einbauen, der das durchgehend prüft. Man könnte unser Produkt auch optimieren. Der Ladescreen z.B. ladet momentan sehr lange.

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

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

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

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

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)

8. Abbildungsverzeichnis

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

9. Anhang

4. TODOS

Wer Zeit hat

- DONE Test-BMA reinschauen
- DONE Präsi machen

Jeffry

- DONE Blender 6 Plattformen machen
- DONE Blender 6 Plattformen in Three.js einfügen
- https://www.youtube.com/watch?v=eBOcbYHexAM -modeling
- https://www.youtube.com/watch?v=XkiWBSSuxLw -rigge
- https://www.youtube.com/watch?v=yjjLD3h3yRc&t=660s -animiere
- DONE Büsche implementieren
- Details zu Scenen einfügen

Yanick

- Vorher
    - DONE Blender Person erstellen
    - DONE Blender Person riggen
    - DONE Blender Person animieren
- 25. Montag
- Vormittag
    - DONE Blender einfache Figur erstellen (15min)
    - DONE Three.js einfache Figur einfügen (15min)
    - DONE Blender einfache Figur materials geben (15min)
    - DONE Three.js Figur mit Materials einfügen (15min)
    - DONE Belnder figur mit uv's machen (15min)
    - DONE Three.js uv's Figur einfügen (15min)
    - DONE Figur einfügen wenns geklappt hat (30min)
- Nachmittag
    - DONE Figur bewegen. (30min)
    - DONE Figur in alle Richtungen bewegen (1.5h)
    - DONE Code aufteilen in mehere Klassen (2h)
- 26. Dienstag
- vormittag
    - Theorie von Blender aufschreiben (3h)
- nachmittag
    - Raycast einbauen (4h)
    - Wand einbauen (2h)
- 27. Mittwoch
    - Jeffrys sachen einfügen (1h)
    - Jeffrys sachen platzieren (2h)
- 28. Donnerstag
    - Präsentieren
    - Weitermachen
- 29. Freitag
    - Dokumentieren
- Kein Datum:
    -