

**3D Website**

Dokumentation

Version 1.0.0, 18.08.2021| Projektmitglieder: Yanick Christen, Jeffry Dahinden

# Abstract

# Inhalt

[Abstract 1](#_Toc88120809)

[Inhalt 2](#_Toc88120810)

[1 Einleitung 3](#_Toc88120811)

[2 Was ist Threejs? 4](#_Toc88120812)

[2.1 Threejs und WebGl Theorie 4](#_Toc88120813)

[2.1 Basic scene 4](#_Toc88120814)

[2.2 Kameras 4](#_Toc88120815)

[2.3. Meshes 5](#_Toc88120816)

[2.4. Lichter 5](#_Toc88120817)

[2.5. Physik 7](#_Toc88120818)

[3 Blender 8](#_Toc88120819)

[3.1 Was ist Blender? 8](#_Toc88120820)

[3.2 Verwendungszweck 8](#_Toc88120821)

[3.3 Grundsätzliche Befehle/Shortcuts 8](#_Toc88120822)

[3.4 Unser Abschlussprojekt Blenderfile 9](#_Toc88120823)

[3.5 Schwierigkeiten 10](#_Toc88120824)

[4 Implementierung 11](#_Toc88120825)

[4.1 Vorgehen 11](#_Toc88120826)

[4.2 Klassen 12](#_Toc88120827)

[4.2.1 Benötigte Klassen 12](#_Toc88120828)

[4.2.2 Blender Implementierung Klassen 13](#_Toc88120829)

[4.2.3 Übrige Klassen 14](#_Toc88120830)

[4.3 Schwierigkeiten 18](#_Toc88120831)

[5 Vorhaben 21](#_Toc88120832)

[6 Glossar 21](#_Toc88120833)

[7 Schlusswort 23](#_Toc88120834)

[8 Quellenverzeichnis 24](#_Toc88120835)

[4. Anhang 25](#_Toc88120836)

# Einleitung

Die BMA von uns ist „eine normale Website zu einer 3D Website ergänzen“. Wir setzten das um mit der Software Blender und dem Framework Three.js. Die Idee ist es, dass wir darstellen, welche erreichten Meilensteine Jeffry Dahinden bei seinem Videografie Business erreicht hat.

Yanick Christen war sehr interessiert an Three.js bzw. 3D Websites erstellen. Jeffry Dahinden war sehr interessiert an Blender bzw. 3D Körper erstellen und animieren. Da man bei Three.js Blender braucht, sind wir auf die Idee gekommen als Projektarbeit eine 3D Website zu erstellen. Die Motivation war auf jeden Fall, dass wir beide einen neuen Skill erlernen wollen für die Zukunft. Das ist auch der Grund, warum wir ein Produkt realisiert haben. Jeffry’s zusätzliche Motivation war, dass er dann neben der normalen Website, eine 3D Website hat, welches er für sein Unternehmen brauchen kann.

Unsere Hypothese lautet: Welche Herausforderungen gibt es, wenn man eine 3D Website erstellt, welches eine Timeline erhält?

Das war unsere Vorgehensweise, nachdem wir die Projektvereinbarung unterschrieben haben. Erst machten wir uns auf die Arbeit, die im Projekt gebrauchten Skills zu erlernen. Yanick Christen hat mit Hilfe eines Videokurses, Three.js gelernt und Jeffry Dahinden hat Blender durch Youtube und einen Bekannten erlernt. Der Anfang war für uns beide sehr hart, da wir sehr viel Stunden investieren mussten um die Basics zu können. Nachdem der Wissensstand bei beiden über den Basics war, erstellte Jeffry eine grobe Timeline für Yanick, damit er das Three.js Projekt aufsetzen kann. Nachdem hat Yanick programmiert und Jeffry hat immer wieder neue Versionen von der Timeline geschickt. So ging es weiter bis zum Schluss. Die Dokumentation wurde immer fortlaufend ergänzt.

# Was ist Threejs?

In diesem Kapitel werden wir versuchen, Ihnen die Grundlagen von Three.js einzuführen. Eine 3D Webseite zu erstellen ist sehr aufwändig und benötigt oft ein grosses Team mit mehreren Monaten an 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, ein französischer Programmierer und Freelancer, erstellt wurde.

## 2.1 Threejs und WebGl Theorie

Die Quellen, auf welche wir uns in diesem Kapitel beziehen sind (d

Doch was ist Three.js überhaupt? Threejs ist eine 3-dimensionale Library (Glossar) für Javascript (Glossar). Erstellt wurde sie von Ricardo Cabello aka. mrDoop (<https://mrdoob.com/> ). An Threejs arbeiten aber gerade 100 verschiedene Leute und versuchen, die Library stehts zu verbessern.). Mit Three js als Programmierer 3 Dimensionale Erfahrungen für den Browser (Glossar) kreieren kann. Die Library Threejs baut auf WebGl auf. Doch was ist WebGl überhaupt?

WebGl ist eine Javascript API (Glossar). Die Grundsätzliche idee ist, dass man mit WebGl Dreiecke zeichnen kann und diese dann möglichst schnell in einem Browser Renderen (Glossar) kann. Diese Dreiecke können verschiedene Formen und Grössen haben. Um alles schön darzustellen zeichnet man diese Dreiecke in ein Canvas (Glossar).

Selbst mit WebGl zu programmieren, wäre aber sehr schwierig. Wir wollen in dieser BMA nicht zu weit hineinschauen, was man alles braucht, nur damit man ein Dreieck erstellen könnte, jedoch bräuchte man bereits, um ein Einzelnen Dreieck zu erstellen über 100 Zeilen Code. (<https://www.tutorialspoint.com/webgl/webgl_drawing_a_triangle.htm> )

Genau hier kommt die Library Theejs ins Spiel. Mit Threejs kann man eine Figur erstellen, wobei diese Dreiecke automatisch generiert werden.

## 2.1 Basic scene

Um eine Sehr einfache scene (Glossar) zu estellen, brauch man zwei Dateien. Als erstes brauch man ein index.html (Glossar) File und danach brauch man noch ein script.js (Glossar) File. Auf der Offiziellen Webseite, kann man danach die Threejs Library herunterladen. (<https://threejs.org/> )

## 2.2 Kameras

Ein wichtiges Element, welches in Threejs existiert ist die Kamera. Der Benutzer der Webseite sieht alle Inhalte durch eine Kamera, warum ohne die Kamera man nichts sehen würde. In Threejs giebt es viele Kameras, welche alle von der Abstrakten(Glossar) Camera Klasse (Glossar) erben (Glossar). Alle Kameras haben andere Funktionen und sind für die verschiedensten Anwendungsfällen benutzbar. Wie zum Beispiel Kameras welche Augen simulieren, oder auch 360 Grad Kameras.<https://threejs.org/docs/?q=camera#api/en/cameras/Camera>

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 anzeigt 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.

<https://threejsfundamentals.org/threejs/lessons/threejs-cameras.html> July 2021 greggman

* 1. 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 Threejs zu erstellen, braucht man Geometrien. Diese Geometiren bestehen aus Eckpunkten (Punkte in einem Raum) und aus Flächen. Mit einem Material kann man dann diese 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.

* 1. Lichter

Damit man in der realen Welt etwas sehen kann, braucht man ein Licht. Dieses Licht kann 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 Licht | Bei diesem Licht kann man zwei Farben und die Stärke setzten. | Dieses Licht ist sehr ähnlich zu dem Ambient Licht, der unterschied ist, dass man eine Farbe, welche von oben scheinen soll und eine Farbe, welche von unten scheinen soll setzten. 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 ablässt. 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, welchen man setzen kann, welche dann 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. | Dieser 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. |

* 1. 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

# Blender

## 3.1 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.

## 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 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.

## 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

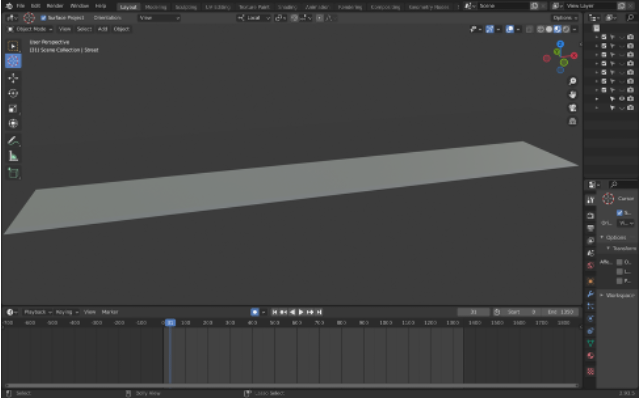
Z → auswählen von Aussehen (Solides Aussehen, Materielles Aussehen etc.)

# Implementierung

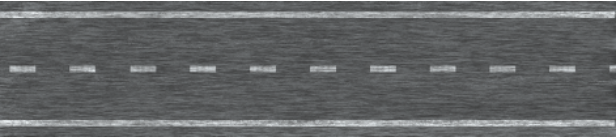
In diesem Kapitel werden wir euch zeigen, wie wir vorgegangen sind und wie unsere Fortschritte ausgesehen haben.

* 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:



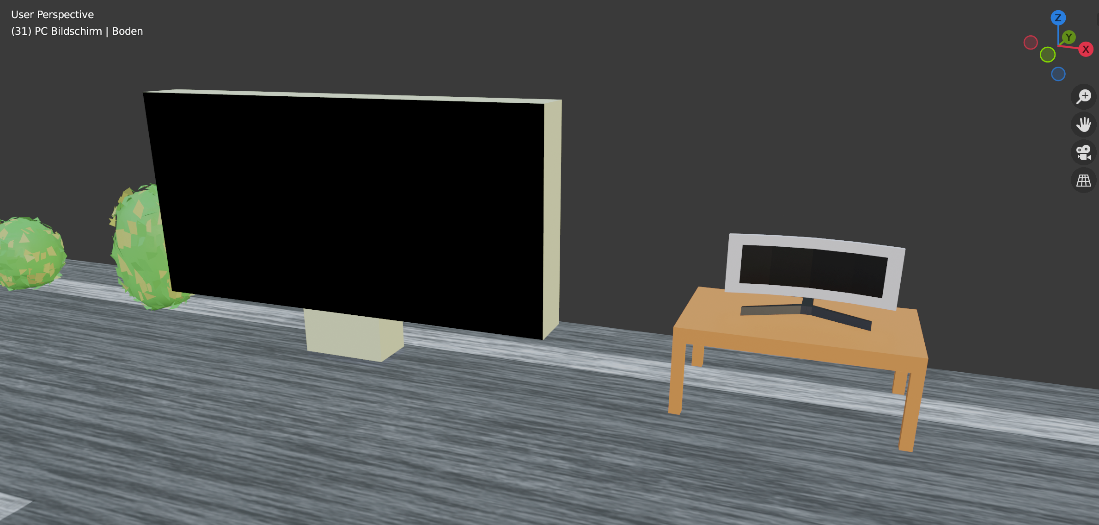
Leider konnte ich keine Textur hinzufügen, damit es aussieht wie eine Strasse. Aus diesem Grund haben wir in Photoshop eine Strasse gezeichnet. Mit Hilfe von YouTube sind wir auf dieses Ergebnis gekommen:



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.



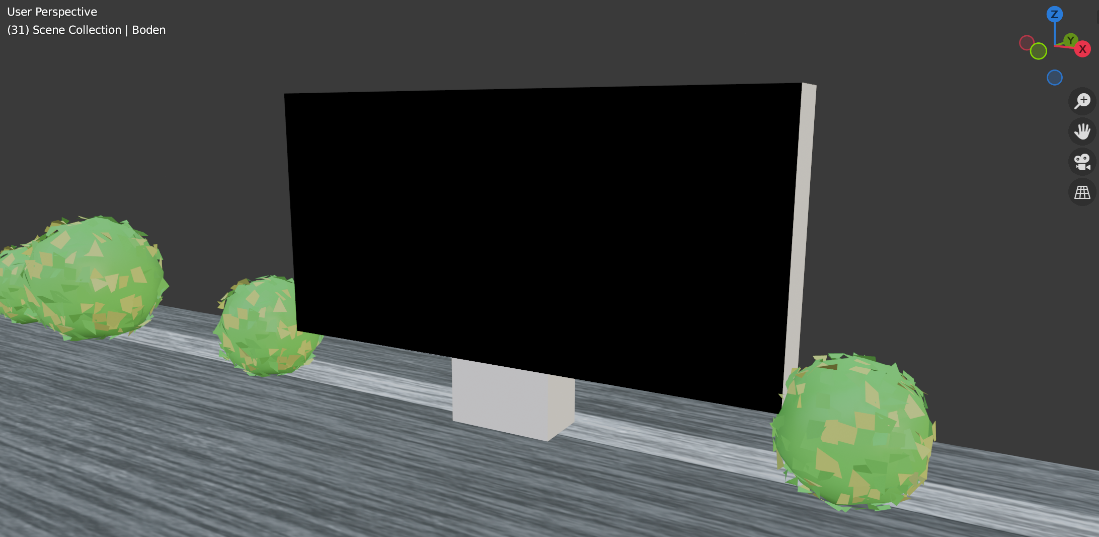
Beim Meilenstein Zwei geht es um das erste Autovideo, welches durch das Anschreiben auf Instagram durchgeführt wurde. Wie beim ersten Meilenstein, kann man auf den grossen Bildschirm klicken und das produzierte Video wird dann angezeigt. Neben dem sieht man mein Arbeitstisch mit meinem gebogenen Bildschirm, den ich besitze. Es war sehr schwierig den Bilschirm 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



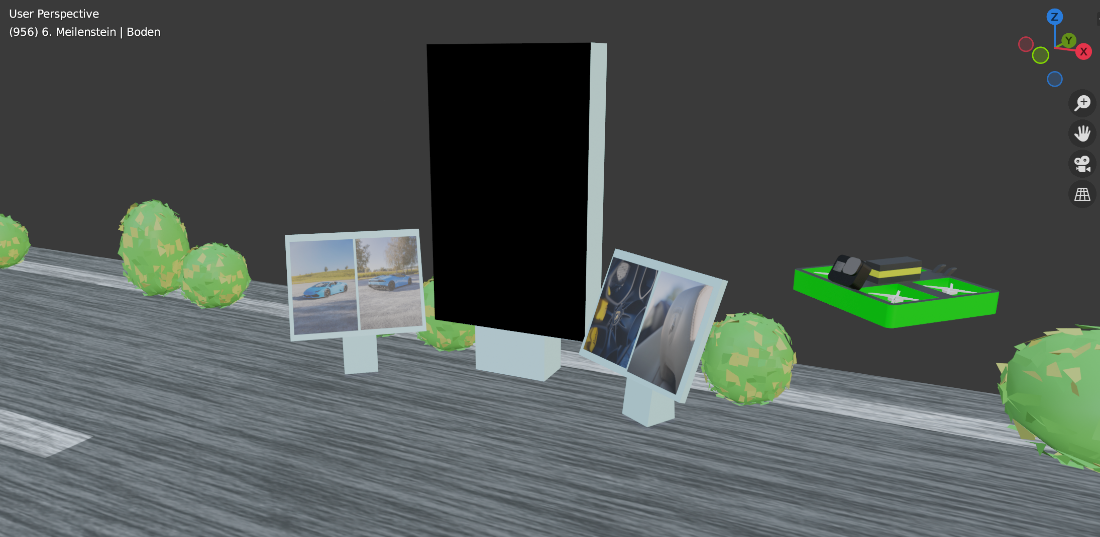
Bei dem Meilenstein Drei geht es um ein Projekt, welches Jeffry mit einer Influencerin hatte. Wir haben aus diesem Grund ein Handy modelliert und in das Handy ein Video hinzugefügt. In dem Video wird kurz gezeigt, für welche Influencerin ich das Video gemacht habe und wo sie das hochgeladen hat. Das Handyvideo habe ich selbst mit der Bildschirmaufnahme erstellt. Wenn man auf den grossen Bildschirm klickt, dann wird ebenso das Video angezeigt.



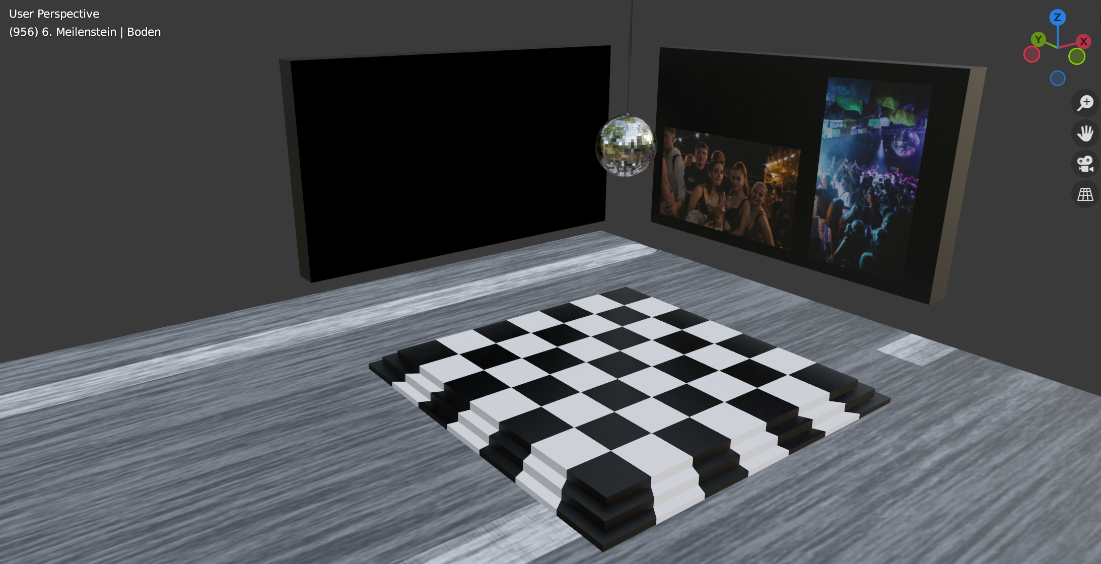
Bei dem Vierten Meilenstein geht es um den Auftrag mit einem Lamborghini Huracan. Hier kann man nur den Bildschirm anklicken und das Video anschauen.



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 modeliert. Die FPV Drohne wurde sehr detailliert gebaut, denn man sieht die Propeller, die Batterie und die GoPro. Ohne der Hilfe von dem Internet haben wir es modelliert. Bei diesem Meilenstein fliegt die Drohne sogar. Dafür mussten wir jeden Frame animieren. Das war eine sehr kreative und aufwändige Aufgabe.



Das ist der letzte Meilenstein. Bei dem habe ich das erste Mal ein Aftermovie im Club gemacht und zusätzlich noch Bilder geschossen. Da es um Clubs geht, habe ich eine Tanzfläche mit einer Discokugel modelliert. Es war sehr einfach zu modellieren, da wir nur einen Modifikator hinzufügen mussten und bisschen konfigurieren mussten. Hier natürlich auch mit der Hilfe von Youtube. Die Tanzfläche wurde selbst gebaut. Wenn man auf das linke Bild klickt, erscheint das Aftermovie. Rechts sieht man zwei Bilder vom Club.



## 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.

* + 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)

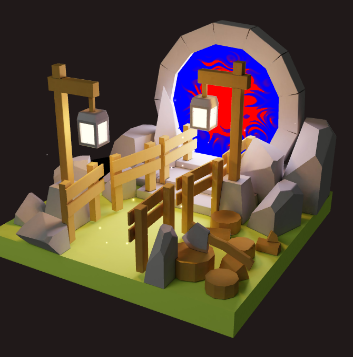
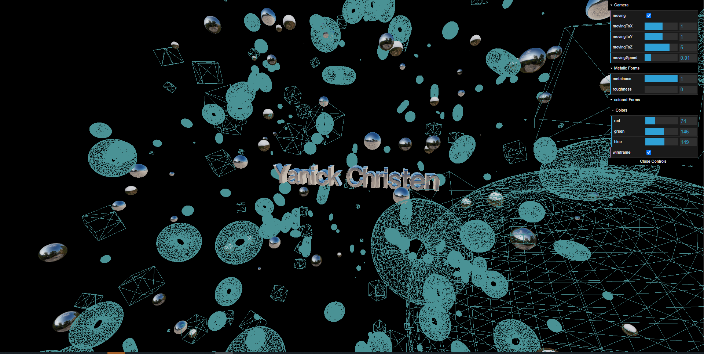
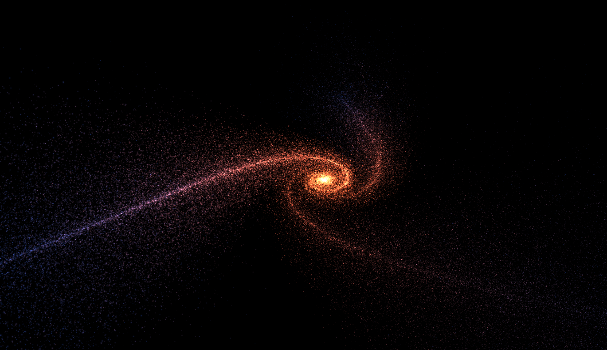


Abbildung : Three.js – Kurs

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.

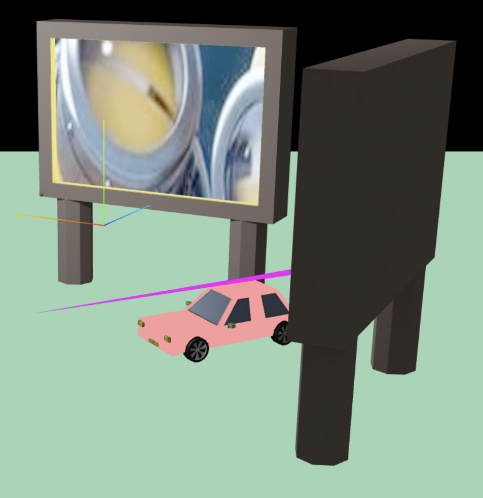


Abbildung : Test-Projekt

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.

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

* + 1. 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 und wir gaben der Figur auch gleich drei Animationen: Rennen, Springen und stehen. Das Implementieren der Figur gab uns kleine Probleme, wie z.B, Dass des Materials nicht mitgenommen wurden, oder auch dass es uns keine Animationen anzeigte. Nach einiger Zeit konnten wir diese Probleme jedoch lösen und somit hatten wir eine Figur in dem Projekt.

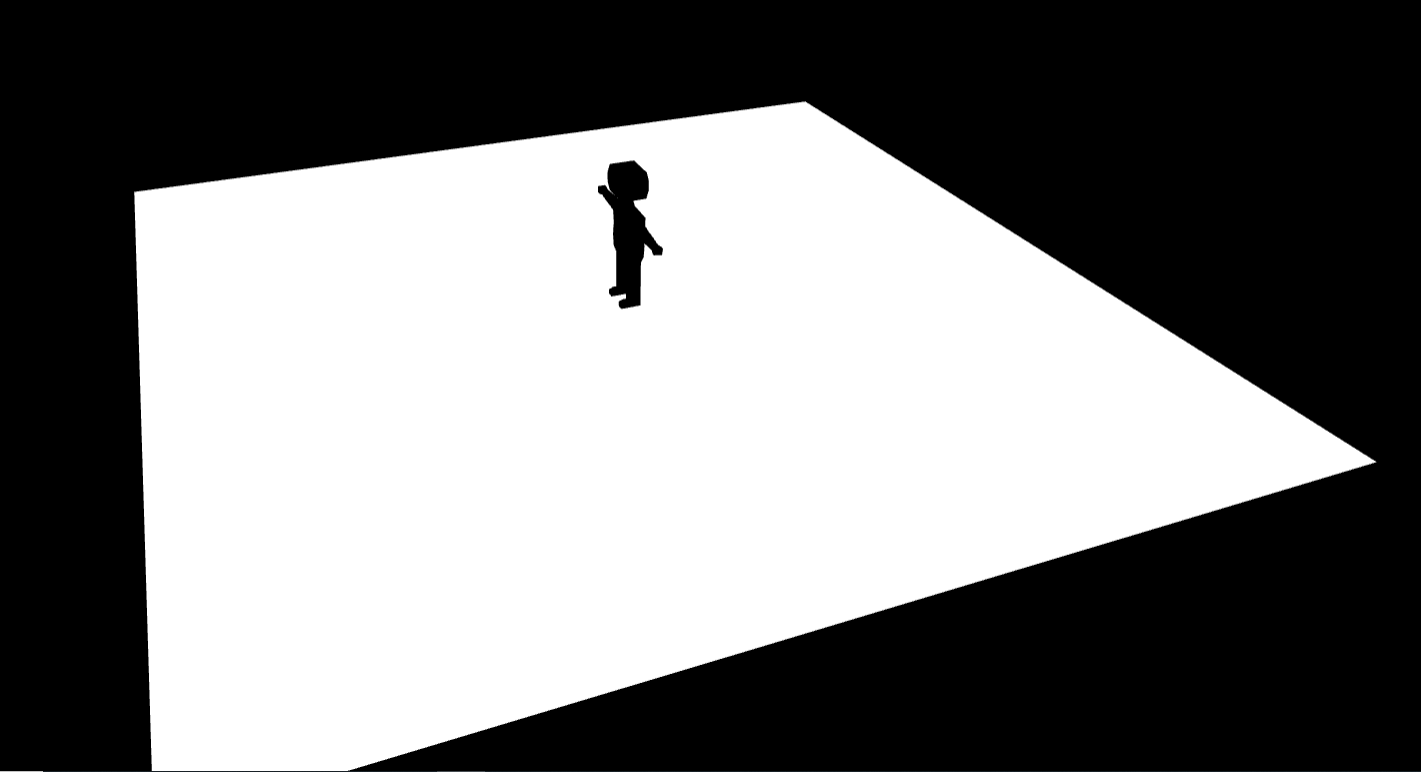
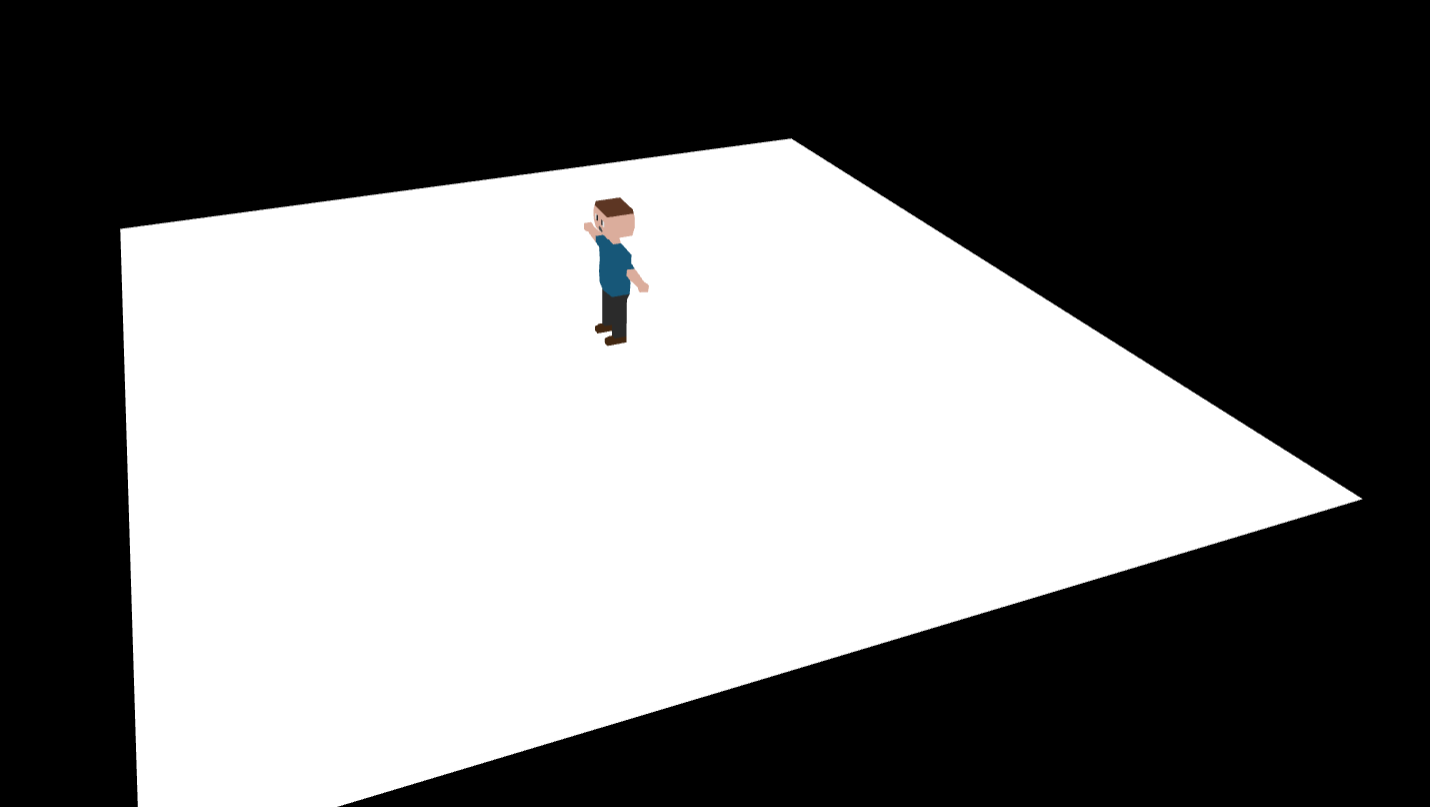
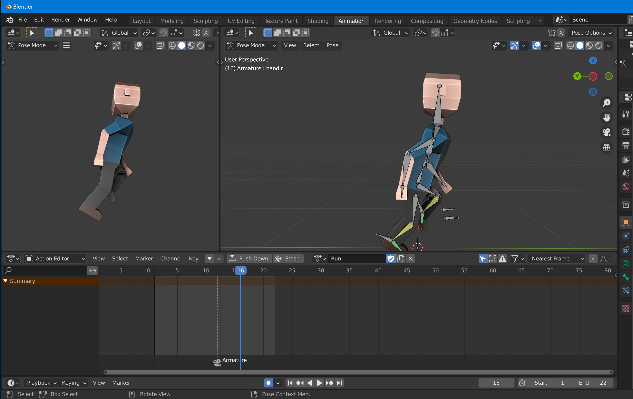


Abbildung : Charakter

Als nächsten, haben wir versucht, dass die Figur sich auch 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. Hier sind einige Probleme, welche wir überwinden mussten:

Keine Stockenden Bewegungen

Zu Begin konnten wir die Person bewegen, jedoch mussten wir die eine Taste immer wieder drücken, damit

Gleichzeitig nach vorne laufen und die Richtung ändern

In welche Richtung muss die Person gehen?

* + 1. Implementierung der Strasse / der Meilensteine

Während wir die Funktionalitäten des Projektes Programmierten, konnten wir gleichzeitig in Blender alle Figuren erstellen

* + 1. Kamera

Versuch mit der Third Person Kamera

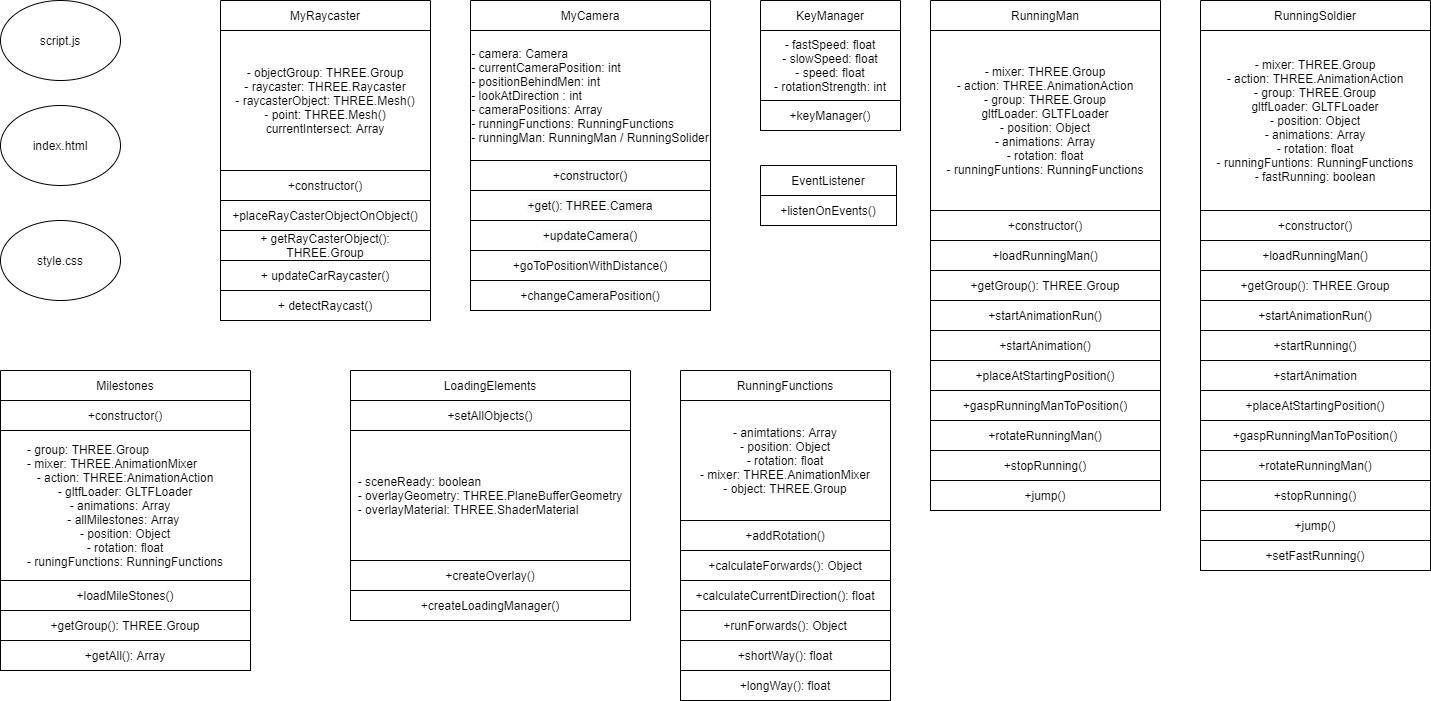
* + 1. Lade Fenster
    2. Code Strukturierung
    3. Implementierung des Soldaten

4.2

4.2 Klassen

JavaScript wird als eine «Multi-Paradigma» Sprache bezeichnet. Das heisst, dass die Sprache sowohl Objektorientierte Programmierung (Glossar), als auch Funktionale Programmierung (Glossar) 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.

4.2.1 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 (Glossar) wir alle anderen Klassen, sowie auch die Scene (Glossar), das Licht und auch unseren «Draco Loader» (Glossar). Zudem wird der Hintergrund gesetzt. Damit man auch unsere Meshes (Glossar) 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 (Glossar) 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 (Glossar) 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.

4.2.2 Blender Implementierung Klassen

**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.

4.2.3 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.

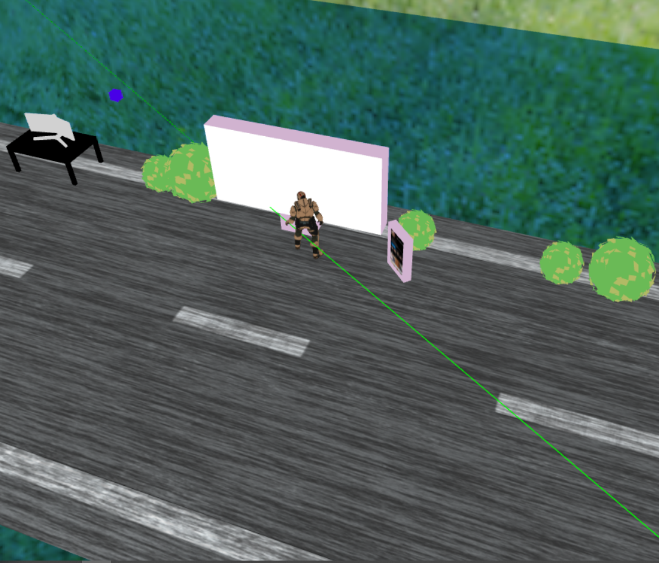
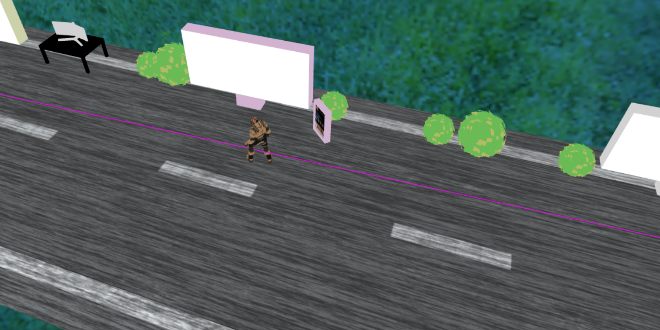


Abbildung : Grüner Raycaster

Abbildung : violetter RaycasterAbbildung : Grüner Raycaster

Abbildung : violetter Raycaster

Abbildung : Dev-Tools von Chrome:Abbildung : violetter Raycaster

**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 (Glossar) 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.

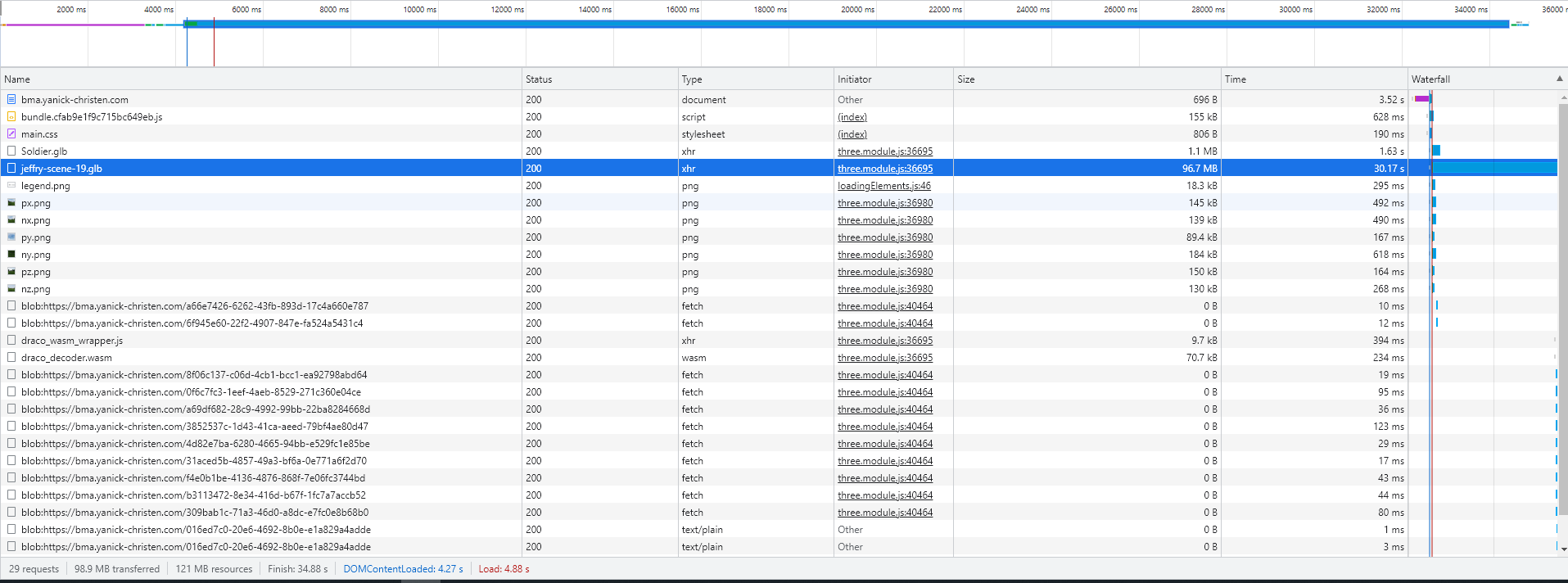
Gut sehen kann man bei diesem Abbild, die Grösse der Strasse (blau markiert), welche mit Abstand am längsten laden musste, und auch das grösste File ist.

Abbildung : Dev-Tools von Chrome:

Abbildung 11: EinheitskreisAbbildung : Dev-Tools von Chrome:

**RunningFunctions**

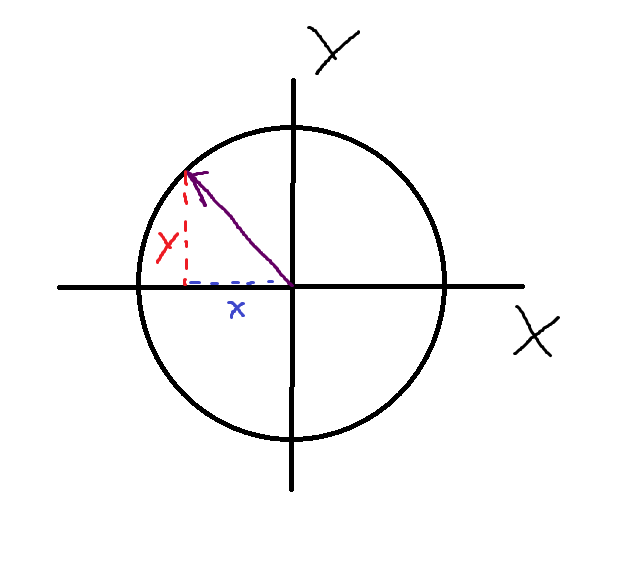
Da wir mehrere Klassen für eine Person benutzen, ist es simpler, die Logik des Fortbewegens und das Rotieren der Figur in einer separaten Klasse zu implementieren. In dieser Klasse geben wir Anweisungen, um wie viel sich die Person in X und Y Richtung Bewegen soll. Man kann sich das so vorstellen, dass 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 13: Einheitskreis

Abbildung 1: Three.js - GalaxieAbbildung 14: Einheitskreis

## 4.3 Schwierigkeiten

**Animierte Person von Blender in Threejs implementieren:**

**Das bewegen der Animierten Person**

Sobald wir die Animierte Person mit einer Taste auf dem Computer wie zum Beispiel die Taste “w” kommt es zu stockenden Bewegungen, die Person rennt nur von frame zu frame.

Wir wissen nicht, auf welche richtung die Person gerade schaut. Die Person soll bei dem Vorwärts laufen relativ zu der sicht vorwärtslaufen.

Man kann sich das wie folgt vorstellen: man hat zwei Personen, welche in die gleiche Richtung schauen. Jetzt dreht sich die zweite Person um 180 Grad. Jetzt ist das Rechts der zweiten Person das Links der ersten Person.

Um Dieses Problem zu lösen, sind wir auf eine Library namens “gsap” gestossen. Diese hilft, zwischen den Bewegungen eine schönere Transition zu machen. Perfekt ist es jedoch trozdem nicht.

Damit wir wissen, in welche Richtung die Person gehen soll, haben wir recht lange herum probiert und sind stück für stück weitergekommen. Die Lösung, welche wir jetzt haben, ist eine ziemlich komplizierte Lösung, bei welche wir die Rotation der Figur durch PI / 200 Rechnen und dann die ersten zwei Ziffern nach dem Komma nehmen. so bekommen wir eine zahl zwischen 0 und 1. Mit dieser Zahl können wir dann sagen um wie viel die Person in der X und auch in der Z Achse bewegen soll.

**Kamera Bewegungen**

Unsere Kamera soll unserem rennender Person von einer Distanz folgen. Dies soll natürlich ohne schwankungen passieren.

Dies zu Lösen war recht schwierig. zum einten haben wir lange nicht gewusst, dass das sogenannte Orbit Controlls nicht benutzen dürfen, wenn wir wollen, dass die Kamera an eine Bestimme Position schaut. Zum anderen, ist es ein sehr schwieries Unterfangen, die Kamera so zu bewegen, dass sie nicht die ganze Zeit stop und immer in einem Richtigen Abstand zu der Person ist.

**Deploy Error**

Einmal gab es einen Build (GLOSSAR) Fehler, als ich meine Webseite deployen (GLOSSAR) wollte. Solch einen Fehler hatte ich noch nicht und ich wusste auch nicht, wie ich diesen Fixen sollte. Dies hat mir sehr Angst gemacht.

Der Fehler war, dass es eine Klasse nicht findet. Also habe ich diese Klasse herausgenommen. Danach konnte ich auch wieder Deployen. Als ich dann die Klasse wieder hineinnahm, ging es plötzlich wieder. Was jetzt genau der Fehler war weiss ich nicht. Ich bin aber sehr froh, dass es jetzt funktioniert hat, da dies Fatal hätte sein können.

Rennender Soldat

**Third Person Kamera**

Als wir das erste Mal versucht hatten, mit der Kamera der Person zu folgen, stockte die Kamera sehr und machte immer nur kleine Schritte. Also haben wir nachgeforscht und ein Youtube Video gefunden, in welchem einer eine Thirth Person 3D Kamera implementiert.

**Figuren selber ersellen:**



Abbildung 15: Soldat Figur stehend

Klasse: RunningSoldier.js

In dieser Klasse wird das glb File von einem Soldaten geladen. Aufgebaut, ist diese Klasse gleich wie die runningMan.js Klasse. Diese zwei Klassen ähneln sich auch sehr, was dazu dient, dass man die Personen einfach auswechseln kann, ohne gross Code zu verändern. Die Methoden, ähneln sich sehr der runningMan.js Klasse. Der unterschied ist vor allen, dass die Animationen an anderen Positionen sind und auch unterschiedlich heissen. Ausserdem kann man beim Soldat unterscheiden, ob man normal laufen oder Rennen will, wobei bei dies bei unserem Mann nicht implementiert wurde. Springen, kann diese Figur dafür nicht.

**Rechte:**

Info: Skeletal Animation Blending (model from mixamo.com)

Die Rechte kann man hier nachlesen: (16.03.2020)<https://www.adobe.com/legal/terms.html>

<https://helpx.adobe.com/legal/legal-notices.html>

<https://helpx.adobe.com/creative-cloud/faq/mixamo-faq.html>

**Three.js anstatt Mixamo**

Wir haben unsere Figure nicht direkt von Mixamo heruntergeladen, sondern von einem Beispiel auf der offiziellen Three.js Webseite.<https://github.com/mrdoob/three.js/tree/master/examples/models/gltf>

Für das haben wir mehrere Gründe. Zum einten haben wir auf diesem Beispiel bereits drei Animationen, 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 mit nur einer Animation bereits 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.

**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 all zu grossen 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.

# 5 Glossar

|  |  |  |
| --- | --- | --- |
| Nr | Name | Beschreibung |
| 1 | Plane |  |
| 2 | Orbit Controls |  |
| 3 | Html |  |
| 4 | Css |  |
| 5 | JavaScript |  |
|  |  |  |

# 6 Schlusswort

# 7 Quellenverzeichnis

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

Cabello, Ricardo (29 Oktober 2021) Threejs Offizielle Webseite: <https://threejs.org/> (zuletzt besucht am 18.11.2021)

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)

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

Bosnjak, Dusan (5. August. 2018) What is three.js? <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>

**Abbildungsverzeichnis**

[Abbildung 5: Grüner Raycaster 20](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232560)

[Abbildung 5: violetter RaycasterAbbildung 5: Grüner Raycaster 20](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232561)

[Abbildung 5: violetter Raycaster 20](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232562)

[Abbildung 6: Dev-Tools von Chrome:Abbildung 5: violetter Raycaster 20](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232563)

[Abbildung 6: Dev-Tools von Chrome: 21](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232564)

[Abbildung 7: EinheitskreisAbbildung 6: Dev-Tools von Chrome: 21](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232565)

[Abbildung 7: Einheitskreis 21](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232566)

[Abbildung 1: Three.js - GalaxieAbbildung 7: Einheitskreis 21](https://migros-my.sharepoint.com/personal/yanick_christen_mgb_ch/Documents/BMADOKU19-11.docx#_Toc88232567)

[Abbildung 8: Soldat Figur stehend 24](#_Toc88232568)

# Anhang

1. TODOS

Wer Zeit hat

* DONE Test-BMA reinschauen
* DONE Präsi machen

Jeffry

* DONE Blender 6 Plattformen machen
* DONE Blender 6 Plattformen in Threejs 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 Threejs einfache Figur einfügen (15min)
  + DONE Blender einfache Figur materials geben (15min)
  + DONE Threejs Figur mit Materials einfügen (15min)
  + DONE Belnder figur mit uv’s machen (15min)
  + DONE Threejs 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: