HTL LEONDING

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

# ScribbleFight - Plattformbasiertes 2D-Brawl-Spiel mit Trainings-KI und Bilderkennung

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Medientechnik und Informatik


Eingereicht von:
Himmetsberger Jonas
Rafetseder Tobias
Weinzierl Ben

Betreuer:
Aistleitner Gerald

Projektpartner:
none


Leonding, April 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022 S. Schwammal & S. Schwammal

Zur Verbesserung der Lesbarkeit wurde in diesem Dokument auf eine geschlechtsneutrale Ausdrucksweise verzichtet. Alle verwendeten Formulierungen richten sich jedoch an beide Geschlechter.

# Abstract

Brief summary of our amazing work. In English. This is the only time we have to include a picture within the text. The picture should somehow represent your thesis. This is untypical for scientific work but required by the powers that are. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

# Zusammenfassung

Zusammenfassung unserer genialen Arbeit. Auf Deutsch. Das ist das einzige Mal, dass eine Grafik in den Textfluss eingebunden wird. Die gewählte Grafik soll irgendwie eure Arbeit repräsentieren. Das ist ungewöhnlich für eine wissenschaftliche Arbeit aber eine Anforderung der Obrigkeit. *Bitte auf keinen Fall mit der Zusammenfassung verwechseln, die den Abschluss der Arbeit bildet!* Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Abkürzungen

## 1.2 Autoren der Diplomarbeit

### 1.2.1 Himmetsberger Jonas

### 1.2.2 Rafetseder Tobias

### 1.2.3 Weinzierl Ben

# 2 Zieldefinition

## 2.1 Projektanlass

Die Möglichkeit seiner Kreativität freien Lauf zu lassen ist bei den meisten populären Online-Spielen sehr eingeschränkt, da man wenig Einfluss auf die Spielumgebung hat. Unsere Arbeit soll diesem Problem entgegenwirken. Der Spieler kann selbst entscheiden, wie die 2D-Spielumgebung auszusehen hat, indem er diese auf ein Blatt Papier zeichnet, welche dann via Bilderkennung als spielbare Welt aufbereitet wird.

## 2.2 Ziele

Bis zum Abgabetermin sollen folgende Ziele erreicht werden:

- Das Spiel soll als Browserspiel funktionsfähig sein.
- Für die Benutzer soll es möglich sein ihre eigenen Kampfumgebungen zu erschaffen.
- Das Spiel soll als online-Multiplayer "Player versus PlayerSpiel funktionieren.
- Ein eigener Modus, in welchem der Spieler als Singleplayer gegen eine funktionsfähige KI antreten kann, soll umgesetzt werden.

## 2.3 Aufgabenverteilung

Dadurch, dass die Diplomarbeit drei mitwirkende Schüler hat, wurde das Thema in drei ähnlich anspruchsvolle Teile unterteilt. Diese werden im Folgenden genauer erläutert:

### 2.3.1 Web-Game und Deployment [R]

Die wichtigsten Punkte, die im Bezug auf das Web-Game umzusetzen zu waren, sind:

- Die Spielphysik, also wie sich Spieler und Objekte verhalten

- Die Collisiondetection von Spielern mit der Umgebung und mit Objekten
- Die Hitregistration, falls ein Spieler von etwas getroffen wurde
- Ab wann ist das Spiel zu Ende, beziehungsweise wann hat jemand gewonnen

Das Deployment des Projekts in die Leocloud, ein Cloud-System der HTL-Leonding, soll mittels Kubernetes erfolgen.

### 2.3.2 Gamedesign und Lobby-System [B]

### 2.3.3 Aufbereitung der Spielumgebung und Künstliche Intelligenz [H]

Die Aufbereitung der Spielumgebung sollte folgende Funktionen erfüllen:

- Erkennung der Konturen in einer Live-View
- Aufnahme soll in brauchbare Daten umgewandelt werden

Folgende Forderungen waren an die KI gestellt:

- Die KI soll auf einem herausfordernden Niveau agieren
- Im Zuge der Forschung sollte ein Vergleich zwischen brauchbaren Lernalgorithmen gemacht werden

Im Laufe der Diplomarbeit und der damit zusammenhängenden Forschung änderte sich oft die Vorstellung darüber, wie das Endprodukt auszusehen hat.

## 2.4 Dokumente

# 3 Umfeldanalyse

## 3.1 Ähnliche Spiele

Es gibt schon viele "Player vs PlayerBrawlspiele. Doch solche Spiele, die man ohne Download im Browser miteinander spielen kann, findet man selten.

### 3.1.1 Super Smash Bros (SSM)

Super Smash Bros (SSB) sind eine Reihe von plattform-basierten Videospielen. Diese sind von Nintendo entwickelt und beinhalten die bekanntesten Charakteren des Unternehmens. Figuren, wie Super Mario oder Sonic, bekämpfen sich in einer Arena mit dem Ziel sich gegenseitig von einer Plattform zu stoßen. Was jedoch fehlt, ist die Plattformunabhängigkeit, da das Spiel nur auf Nintendo-Systemen funktioniert. Außerdem besteht eine Limitierung in der Auswahl von Spielumgebungen.

### 3.1.2 Brawlhalla

Brawlhalla ist ein von Blue Mammoth entwickeltes 2D-Kampfspiel, und wurde für alle gängigen Betriebssysteme entwickelt. Auch wie in Scribble-Fight ist es das Ziel, den Gegner von einer Plattform zu stoßen.

Der Nachteil hierbei ist, dass man sich vor dem Spielen einen Account erstellen und dann einen Download abschließen muss. Dazu kommt noch, dass man die Spielumgebung nie beeinflussen kann. Bei Scribble-Fight ist das nicht so.

### 3.1.3 Stick Fight: The Game

In dem Spiel Stick Fight kämpft man als Strichmännchen gegeneinander, die durch eine Ragdoll-Engine gesteuert werden. Auch wie bei schon bei vorher erwähnten Spielen, kann man aber nicht einfach plattformunabhängig im Browser gegeneinander antreten. Für die Spielekonsole Nintendo Switch zum Beispiel, gibt es das Spiel nicht. Hinzu


Abbildung 1: Brawlhalla Spielumgebung

kommt, dass man auch die Umgebung nicht selbst frei erstellen kann, so wie es bei Scribble-Fight möglich ist.

## 3.2 Ist-Zustand

Dadurch das es eine Diplomarbeit ist, fängt alles bei 0 an. Es gibt jedoch Frameworks, welche die Erarbeitung erleichtern. Zum Beispiel im Falle des Spiels, welches mittels Webtechnologien umgesetzt wird, kann p5.js verwendet werden, welches die Umsetzung eines Spiels erleichtert. Ein weiteres Beispiel ist die Objekterkennung für die Spielumgebung (Map). Diese wird von Open-CV übernommen.

# 4 Technologien

## 4.1 JavaScript [R]

Zuerst musste die Entscheidung gefällt werden, ob unser Projekt ein Standalone-Programm sein soll, oder im Browser zu erreichen ist. Weil wir aber wollten, dass es ein Party-Game werden soll, dass man ohne jeglichen Aufwand sofort mit Freunden spielen kann, haben wir uns für die Browser-Variante entschieden. Für mich war es eine leichte Entscheidung JavaScript zu verwenden, da die Programmiersprache genau für den Browser geeignet ist, und man damit nicht nur im Frontend, sondern auch im Backend programmieren kann.

### 4.1.1 p5.js / p5.play [R]

p5.js ist eine open-source JavaScript Library, die für Kreation von Spielen genutzt wird. p5.play ist eine Library für p5.js, mit der man visuelle Objekte managen kann. Außerdem beinhaltet es Features wie Animation-support, Kollisionserkennung, sowie aber auch Funktionen für Mouse und Tastatur Interaktionen. Es ist wichtig sich im Hinterkopf zu behalten, dass p5.play für barrierefreies und simples Programmieren gedacht ist, nicht für perfomantes. Es ist keine eigene Engine, und unterstütz auch keine 3D-Spiele.

#### 4.1.1.1 Einbindung

Der einfachste Weg P5.js einzubinden ist auf ein JavaScript File online zu verweisen.

```
1       <script
2       src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.js">
3       </script>
```

Man kann sich aber auch die P5.js Library lokal downloaden unter https://p5js.org/download/ Dann muss man nur noch auf das lokale File verweisen.

```
1       <script src="../p5.min.js"></script>
```

Jedoch muss man das Projekt dann auf einem lokalen Server (z.B. Node.js) hosten.

#### 4.1.1.2 Struktur eines P5.js Projekts

Die Struktur ist sehr simpel. Im Ganzen ist es nur ein `index.html` File und ein `sketch.js` File. In dem HTML File bindet man die P5 Library ein, und auch das sketch.js File.

```html
<!DOCTYPE html>
<html lang="">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Scribble Fight</title>
  <link rel="stylesheet" href="style.css">
  <script src="lib/p5.js"></script>
  <script src="sketch.js"></script>
</head>

<body>
  <main>
  </main>
</body>

</html>
```

Abbildung 2: Aufbau index.html

So kann man nun in dem sketch.js File die P5.js Methoden nutzen. Die wichtigsten Methoden sind die Setup- und die Draw-Methode. Die Setup-Methode wird vor der Draw-Methode aufgerufen um das Spiel zu initialisieren. (Es wird zum Beispiel ein Canvas erstellt). Wenn diese abgeschlossen ist, wird die Draw-Methode 60 mal in der Sekunde aufgerufen und updatet jedes mal den Bildschirm.

```javascript
function setup() {
    createCanvas(400, 400);
}

function draw() {
    background(220);
}
```

Abbildung 3: Simples sketch.js Beispiel

#### 4.1.1.3 P5.js vs Processing

p5.js ähnelt sich sehr stark mit Processing, eine Programmiersprache die man sich wie ein stark vereinfachte Version von Java vorstellen kann. Der Unterschied liegt darin, dass Java eine Umgebung, basierend auf der Java Programmiersprache ist, während p5.js eine Bibliothek, basierend auf der JavaScript Programmiersprache ist. Processing ist dafür geeignet, lokale Applikationen zu bauen, hingegen dazu kann p5.js nur im Browser ausgeführt werden.

p5.js ist also kurzgesagt ein direkter JavaScript Port für die Processing Programmiersprache.

Vorteile von p5.js:

- Man kann interaktive Programme entwickeln, die in jedem modernen Browser funktionieren (plattformunabhängig)
- Das Programm ist nicht nur lokal auf dem eigenen Gerät, was das Teilen sehr viel leichter macht
- Man hat die Option den p5.js Editor im Web zu verwenden: Überhaupt kein Aufwand, um loszuprogrammieren

Nachteile von p5.js:

- Ist langsamer beim Pixel manipulieren
- Ist kein Standalone-Programm, d.h. ein Browser wird benötigt

### 4.1.2 Node.js [R]

Node.js ist eine plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung, mit der Besonderheit, dass sie JavaScript-Code außerhalb eines Webbrowsers ausführen kann und wurde ursprünglich von Ryan Dahl 2009 entwickelt, einem Software-Entwickler aus San Diego, Kalifornien. Die Laufzeitumgebung wurde darauf spezialisiert, leicht skalierbare Server zu bauen.

In dem folgenden "Hello WorldBeispiel, können viele Verbindung gleichzeitig behandelt werden. Bei jeder Verbindung wird die Callback-Funktion ausgeführt, aber wenn keine Arbeit zu erledigen ist, schläft Node.js.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Abbildung 4: Sehr simpler Node.js Server

Man merkt den Unterschied zu den heutzutage weit verbreiteten Concurrency-Modellen, die mit OS-Threads arbeiten. Der Vorteil von Node.js hierbei ist, dass man sich keine Sorgen über dead-locking machen muss, da fast keine Node.js Funktion direkte I/O Operationen durchführt. Also ist der ganze Prozess so gut wie nie blockiert, ausser wenn synchrone Methoden der Node.js Standard Library benutzt wird.

#### 4.1.2.1 NPM

Neben Node an sich, ist NPM (Node Package Manager) das wichtigste Werkzeug für Node Applikationen. Mit NPM können alle Packages, die das Projekt benötigt, gefetched werden. Es ist möglich, alle Packages einzeln zu fetchen, jedoch benutzt man normalerweise ein package.json File. In diesem File stehen alle Dependencies für jedes JavaScript Package, das benötigt wird, sowie auch Meta-Daten zu dem Node Projekt. Erstellt wird dieses in dem man in das Verzeichnis navigiert, in dem man das Projekt haben will und den Befehl `npm init` ausführt.

```
{
  "name": "p5_backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rafetseder Tobias",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "socket.io": "^4.1.3"
  }
}
```

Abbildung 5: Package.json des Scribble-Fight Backends

#### 4.1.2.2 Express

Express ist das am meisten verbreiteteste Node Web Framework und ist auch die Basis für andere Node Web Frameworks. Die Hauptverantwortung von Express ist das Bereitstellen von Server-Logik wie zum Beispiel das Schreiben von Handlers für Requests mit unterschiedlichen Http Verbs auf unterschiedlichen URL Pfaden. Man kann Express mit dem Node Package Manager mit `npm install express` installieren, oder man befindet sich in einem Verzeichnis mit package.json File bei dem Express als Dependency hinzugefügt wurde, dann reicht `npm install`

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', function(req, res) {
  res.send('Hello World!')
});

app.listen(port, function() {
  console.log(`Example app listening on port ${port}!`)
});
```

Abbildung 6: Simpler Web Server mit Express

### 4.1.3 Socket IO [R]

Socket.IO ist eine Library, die eine bidirektionale Echt-Zeit Verbindung zwischen Server und Client ermöglicht. Es besteht aus

- Einem Node.js Server
- Einer JavaScript Client Library (Es bestehen auch einige andere Client Implementationen für Sprachen wie Java, C++, Python, etc.)

Socket.IO funktioniert so, dass der Client, falls möglich, eine WebSocket Verbindung mit dem Server herstellt. Ist keine Verbindung möglich, setzt der Client einen HTTP long polling Request ab.

```
const socket = io("http://localhost:3000");

socket.on("connect", () => {
  // Entweder mit send()
  socket.send("Hello!");

  // oder mit emit und eigenen Event Namen
  socket.emit("greetings", "Hello my friend!");
});

// Umgehen mit dem Event, das mit socket.send() geschickt worden ist
socket.on("message", data => {
  console.log(data);
});

// Umgehen mit dem Event, das mit socket.emit() geschickt worden ist
socket.on("greetings", data => {
  console.log(data);
});
```

Abbildung 7: Socket.IO Client Beispiel

Damit der Server die Verbindung annehmen kann, müssen folgende Kriterien erfüllt sein:

- Der Browser unterstützt WebSocket
- Die Verbindung wird nicht von Elementen wie Firewall gestört

Die API ist auf der Server-Seite dem Client sehr ähnlich, man bekommt auch wieder ein `socket` Objekt, welches von der EventEmitter Klasse von Node.js erbt.

```
const io = require("socket.io")(3000);

io.on("connection", socket => {
  // Entweder mit send()
  socket.send("Hello!");

  // oder mit emit() und eigenen Event Namen
  socket.emit("greetings", "Hello from the Server");

  // Umgehen mit dem Event, das mit socket.send() geschickt worden ist
  socket.on("message", (data) => {
    console.log(data);
  });

  // Umgehen mit dem Event, das mit socket.emit() geschickt worden ist
  socket.on("greetings", data => {
    console.log(data);
  });
});
```

Abbildung 8: Socket.IO Server Beispiel

#### 4.1.3.1 Unterschied Socket.IO zu WebSocket

Socket.IO ist keine WebSocket Implementation. Auch wenn Socket.IO WebSocket als Transportmittel benutzt, werden bei jedem Paket zusätzliche Metadaten angehängt. Das ist auch der Grund, warum ein Socket.IO Client keine Verbindung mit einem schlichten WebSocket Server herstellen kann, und umgekehrt. Man kann sich Socket.IO also als einen Wrapper rund um die WebSocket API vorstellen.

## 4.2 Deployment [R]

Das Scribble-Fight Browser-Game wurde in die Leocloud, ein Cloud-System der HTL-Leonding, unter https://student.cloud.htl-leonding.ac.at/t.rafetseder/scribble-fight/ deployed. Zuerst wurde mithilfe von der Docker-Technologie ein Docker-Image erstellt und auf die Leocloud hochgeladen. Das Deployment wurde dann mithilfe von Kubernetes umgesetzt. Was Docker und Kubernetes genau ist, folgt in den nächsten Sektionen.

### 4.2.1 Docker [R]

Die Software Docker ist eine Technologie zum Containerisieren von Prozessen, die dann unabhängig voneinander und isoliert ausgeführt werden können. Diese isolierte Prozesse

nennt man dann Container. Durch die Unabhängigkeit, die dadurch entsteht, wird die Infrastruktur besser genutzt und auch die Sicherheit bewahrt, die sich aus der Arbeit mit voneinander getrennten System ergibt. Docker arbeitet mit einem Image-basierten Bereitstellungsmodell. Dieses wird gerne bei Containertools verwendet, da Applikationen mit all deren Dependencies, egal in welcher Umgebung, genutzt werden können.

- Wenn man einmal seine containerisierte Applikation getestet hat, kann man sich sicher sein, dass die Applikation auf jeder anderen Umgebung, auf dem Docker installiert ist, auch funktioniert
- Alle Docker Container sind komplett voneinander unabhängig
- Falls Skalierung notwendig ist, kann man schnell neue Container erstellen
- Im Gegensatz zu virtuellen Maschinen beinhalten Container keine eigenen Operating Systems, deshalb kann man sie schneller erstellen und auch schneller starten

Wichtige Begriffe, die man im Zusammenhang mit Docker kennen sollte:

- Image: Speicherabbild eines Containers
- Container: aktive Instanz eines Images
- Dockerfile: eine Textdatei, die den Aufbau des Images beschreibt
- Registry: Unter Registry versteht man eine Ansammlung gleicher Images mit verschiedenen Tags, meistens Versionen

#### 4.2.1.1 Docker Architektur

Die Docker Architektur ist eine Server-Client Architektur. Der Docker Client kommuniziert mit dem Docker Daemon, der dann Docker Container z.B. baut und ausführt. Dieser Docker Daemon kann lokal installiert sein, aber der Client kann sich auch mit einem Daemon Remote verbinden.

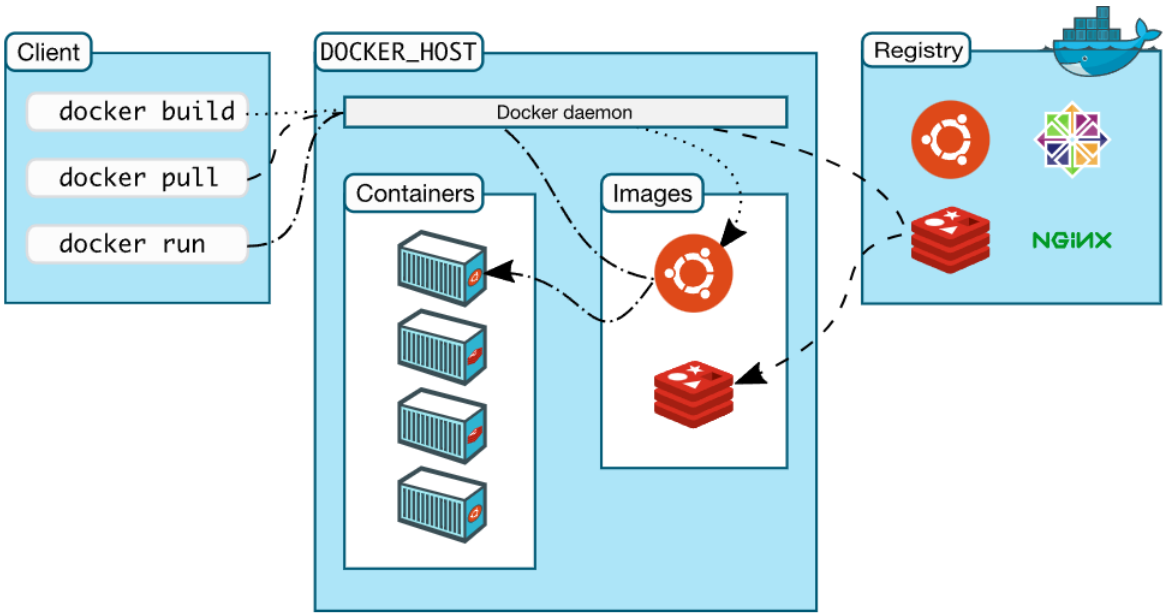


Abbildung 9: Veranschaulichung Docker Architektur

### 4.2.2 Kubernetes [R]

Kubernetes ist Open-Source-Plattform, die dafür genutzt wird, containerisierte Applikationen und Services zu verwalten. Es managed die Computer-, Netzwerk und Speicherinfrastruktur von Containern. Das Kubernetes-Projekt ist 2014 als Open-Source Projekt in die Welt gerufen worden.

Kubernetes hat mehrere Funktionen, zum Beispiel ist Kubernetes:

- eine Containerplattform
- eine Microservices-Plattform
- eine Cloud-Plattform

In dieser Diplomarbeit wird Kubernetes benutzt, um ein mit Docker gebautes Images des Scribble-Fight Browsergames auf ein Cloudsystem zu deployen. Näheres zur Umsetzung findet man hier

## 4.3 Python [H]

### 4.3.1 Flask [H]

### 4.3.2 OpenCV2 [H]

### 4.3.3 PIL [H]

### 4.3.4 TensorFlow und Keras [H]

### 4.3.5 OpenAI Gym [H]

# 5 Umsetzung

## 5.1 Web-Game [R]

### 5.1.1 Frontend [R]

#### 5.1.1.1 Architektur [R]

#### 5.1.1.2 p5.js [R]

### 5.1.2 Backend [R]

#### 5.1.2.1 Node.js [R]

#### 5.1.2.2 SocketIO [R]

### 5.1.3 Gamephysics [R]

### 5.1.4 Hitregistration [R]

### 5.1.5 Collisiondetection [R]

### 5.1.6 Regeln und Spielablauf [R]

## 5.2 Deployment [R]

### 5.2.1 Docker [R]

### 5.2.2 Leo-Cloud [R]

## 5.3 Map-Erkennung [H]

### 5.3.1 Objekterkennung [H]

#### 5.3.1.1 Beschreibung der Funktionalität [H]

### 5.3.2 Open-CV2 [H]

#### 5.3.2.1 Umwandlung der Bilder in für das Spiel brauchbare Daten [H]

### 5.3.3 Kommunikation mit der Lobby via Flask und Flask SocketIO [H]

## 5.4 KI [H]

darüber, wie das Endprodukt (KI) auszusehen hat, beziehungsweise, wie dieses aussehen kann. Limitierungen, welche vor der Forschung noch nicht bekannt und bewusst waren, begrenzten die Lernfähigkeit der KI. Auf Probleme, Lösungen und Ergebnisse wird jedoch noch näher in ... eingegangen.

### 5.4.2 Künstliche Intelligenz Definition [H]

### 5.4.3 Reinforcement Learning [H]

### 5.4.4 Die ScribbleFight-KI [H]

#### 5.4.4.1 Beschreibung der Funktionalität [H]

#### 5.4.4.2 Warum Python? [H]

### 5.4.5 Tensorflow und Keras [H]

#### 5.4.5.1 Tensorflow [H]

#### 5.4.5.2 Keras [H]

# 6 Evaluation des Projektverlaufs

## 6.1 Meilensteine

Ist es uns gut dabei gegangen diese einzuhalten?

## 6.2 Gelerntes

Haben wir bei der DA was gelernt? nein

## 6.3 Was würden wir anders machen?

JOA Was würden wir anders machen?

Aufzählungen:

- Itemize Level 1
  - Itemize Level 2
    - Itemize Level 3 (vermeiden)

1. Enumerate Level 1
   a. Enumerate Level 2
      i. Enumerate Level 3 (vermeiden)

**Desc** Level 1

**Desc** Level 2 (vermeiden)

**Desc** Level 3 (vermeiden)

# Literaturverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quellcodeverzeichnis

# Anhang