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 (SSB)

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.

Abbildung 1: Brawlhalla Spielumgebung

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 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.

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.

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

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.

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

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

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

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.

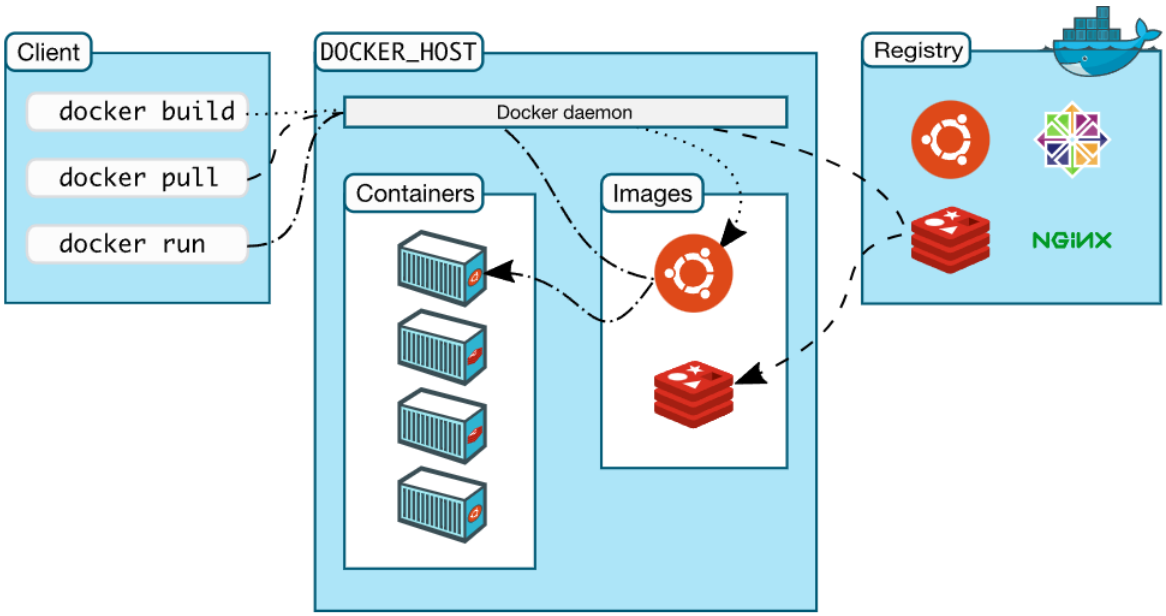

Abbildung 9: Veranschaulichung Docker Architektur

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]

Diese Programmiersprache, welche nach der Britischen comedy Serie “Monty Python” benannt ist, fand in unserer Arbeit zwei Haupteinsatzgebiete.
Erstens, zur Erkennung des Blatt Papiers und der Umwandlung der Zeichnung in eine spielbare Map und zweitens um die Künstliche Intelligenz zu erschaffen.
Die kostenlosen Bibliotheken, welche wir dabei in Verwendung haben, werden im folgenden gelistet und näher erklärt.

4.3.1 Flask [H]

Flask ist das am wohl häufigste verwendete Python Web-Framework. Somit gibt es viele Tutorials, Tools und Bibliotheken. Diese sind sehr gut bis gut dokumentiert und teilweise geprüft. Aus diesen Gründen haben wir den Teil der Bild- und Maperkennung, welche als Webanwendung funktioniert, mittels Flask umgesetzt. In Kombination mit Flask-SocketIO werden Bilder von der Webcam in Echtzeit direkt an den Server geschickt, welcher dann via OpenCV2 Informationen aus dem Bild generiert. Genau wie bei SocketIO in JavaScript, agiert Flask-SocketIO als bidirektionale Kommunikation zwischen Server und Client. Die extrem geringe Latenzzeit, welche dabei auftritt, ist wichtig um eine flüssige Verarbeitung der Bilder zu gewährleisten.

4.3.2 OpenCV2 [H]

Open “Computer Vision” (CV) wurde in unserem Kontext als Python Bibliothek verwendet. OpenCV verfügt über eine breitgefächerte Auswahl an Bildverarbeitungs-Algorithmen. Folgende wurden bei der Maperkennung eingesetzt:

- Resizing
- Farbraumkonvertierung
- Weichzeichnung
  - Median-Blur
  - Gaussian-Blur
- adaptive Schwellenwertbildung von Pixelwerten
- Konvertierung eines Zahlen Arrays in eine “.png” datei

Auf die Funktionsweise dieser Algorithmen wird im Folgenden genauer eingegangen. Um zum Beispiel ein beliebiges Bild unscharf zu zeichnen würde man so vorgehen:

Listing 1: OpenCV Demo

```
# Dieses kurze Programm soll ein Bild in Python mittels Open Computer Vision
    weichzeichnen

# Importieren der gebrauchten Bibliothek
import cv2
import numpy

# Bild einlesen
source = cv2.imread('./Pfad/zum/Bild.png', cv2.IMREAD_UNCHANGED)

# Das Quell-Bild wird nun unscharf gezeichnet
destination = cv2.GaussianBlur(source,(5,5),cv2.BORDER_DEFAULT)

# Anzeigen von dem Quellbild und dem bearbeiteten Bild
cv2.imshow('Weichzeichnung',numpy.hstack((source, destination)))

# warten, bis eine Taste gedrueckt wurde
cv2.waitKey(0)

# Alle Fenster, welche die Bilder anzeigen, werden geschlossen
cv2.destroyAllWindows()
```

4.3.3 PIL [H]

PIL (Python Image Library) ist, wie Flask und OpenCV2, eine kostenlose Zusatzbibliothek für Python. PIL wird verwendet um Bilder zu speichern und zu pixel zu manipulieren. Dabei unterstütz PIL diese Dateiformate: PPM, PNG, JPEG, GIF, TIFF, und BMP. Pixel manipulation bedeutet, dass jeder Pixel auf einem Input Bild angepasst werden kann. Beispiele hierfür sind zum Beispiel das Aufhellen oder Abdunkeln von Bildern oder das Anpassen der Sättigung. Auch Kontrast- oder Schärfeeinstellungen können mit Pixelmanipulation erzielt werden. Dafür werden meistens Matrizen und/oder Formeln pro Pixel verwendet um deren Farbwerte anzupassen

Listing 2: PIL Demo

```
# verwendete Klassen der Bibliothek einbinden
from PIL import Image, ImageFilter

source = Image.open("file.ppm") # Load an image from the file system.
destination = source.filter(ImageFilter.BLUR) # Blur the image.

# Display both images.
original_image.show()
destination.show()
```

4.3.4 TensorFlow und Keras [H]

TensorFlow ist eine Python Bibliothek, welche beim erstellen von Projekten, welche maschinelles Lernen in irgend einer Art und Weise eingebunden haben, extrem unterstütz. Wie der Name schon vermuten lässt, basiert TensorFlow auf zwei Grundlagen: Tensoren und Graphen (Flow vom Wort dataflow).
Tensoren sind besser bekannt als Skalare, Vektoren oder Matrizen. Tensoren sind also null-, ein- oder mehrdimensionale Daten-Tupel. TensorFlow bietet alles von on der effizienten Ausführung von Befehlen auf der CPU, oder GPU, über der Skalierung von Berechnungen auf viele Endgeräte bis hin zur Visualisierung der gelernten Daten mittels dem sogenannten "TensorBoard". TensorFlow ist also ein sehr mächtiges Framework, das viele Möglichkeiten bietet, künstliche Intelligenzen zu trainieren und analysieren.
Jedoch ist die Benutzerfreundlichkeit von TensorFlow sehr eingeschränkt. Viele Entwickler empfanden es als ungeeignet für schnelles Prototyping und verwendeten daher das auf TensorFlow basierende Keras.
Keras verwendet standardmäßig die GPU zum ausführen von Code und is somit um einiges schneller als TensorFlow. In dieser Diplomarbeit wurden zwei der von TensorFlow (Stable-Baselines3) bereits vorgefertigten Algorithmen, namens "A2C" und "PPO", verwendet um die KI zu trainieren. Auf die Auswertung wird in ... näher eingegangen. Die Logik hinter der Künstlichen Intelligenz wurde mittels OpenAI Gym implementiert.

4.3.5 OpenAI Gym [H]

OpenAI Gym ist ein Toolkit, mit welchem das Erlernen und Erstellen von reinforcement learning Algorithmen sehr leicht fällt. Da viele Menschen nicht wissen, was "reinforcement learning" ist, wird es im folgenden kurz erläutert.

Reinforcement Learning[H]

Reinforcement learning bedeutet auf deutsch so viel wie bestärkendes Lernen oder verstärkendes Lernen. Diese Art und Weise zu lernen ist der, wie ein Lebewesen mit einem biologischen Gehirn lernt, am ähnlichsten. Dabei basiert es auf folgendem Konzept: Ein Agent, welcher etwas erlernen soll, wird in eine ihm unbekannte Umwelt gesetzt. Dieser kennt nicht mehr als seinen eigenen Zustand und welche Aktionen er ausführen kann. Bewegt sich diese Entität nun in der Umwelt, so passieren ihm Dinge. Diese können nach einer Observation entweder zu einer negative (-) oder einer positive (+) Belohnung

führen. Das einzige Ziel des Agenten besteht nun darin sein Tun so anzupassen, dass er so viel und so effizient wie möglich an diese Belohnung gelangt. Folgende Grafik erläutert das Beispiel an einem Hund, welcher lernen soll einen Stecken zu apportieren.

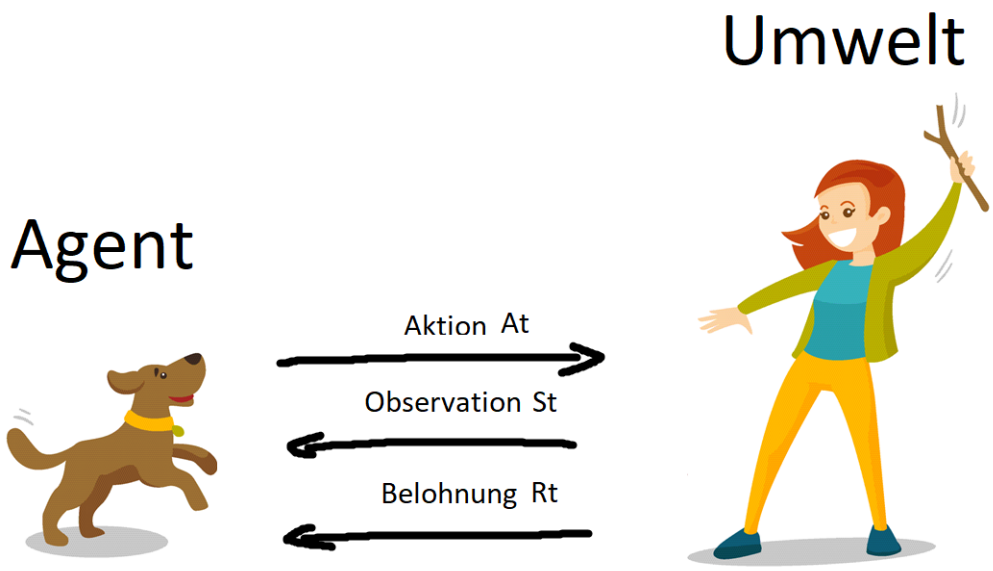

Abbildung 10: Veranschaulichung bestärkendes Lernen

OpenAI Gym hat also einen mehr oder weniger vorgegebenen Bauplan, und vorgegebene Regeln, nach welchen man so eine Künstliche Intelligenz aufbauen muss.
Weitere, weit verbreitete Formen von machine learning sind supervised und unsupervised learning.

4.3.6 Künstliche Intelligenz allgemein [H]

Künstliche Intelligenz funktioniert im Grunde genommen wie das menschliche Gehirn. Es basiert genauso auf Neuronen und deren Axonen, Dendriten und Terminale. Obwohl es verschiedene Arten von Neuronen gibt, haben sie alle gemeinsam, dass sie einen elektrischen Impuls als Reaktion auf einen Input aussenden. Abhängig von diesem Input ist durch das Neuron, welches das chemische Signal verarbeitet, der Output.

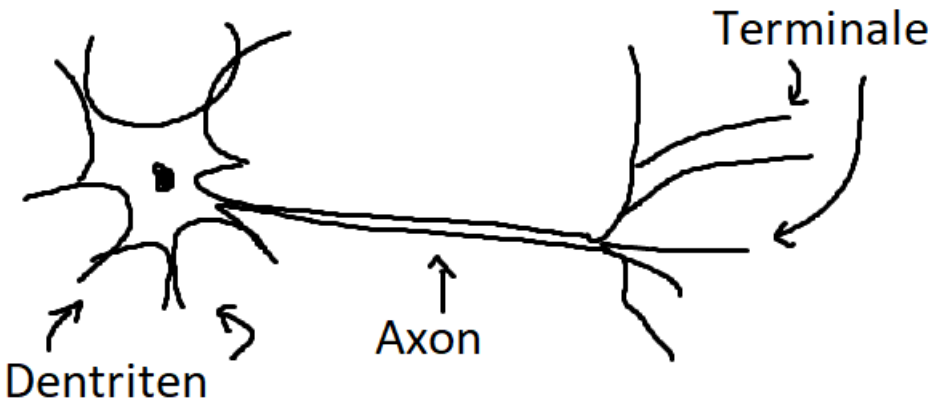

Abbildung 11: Neuron

Genauso funktioniert auch ein Neuronales Netz. Auf eine Eingabe folgt über eine Verarbeitung der Eingabe eine Ausgabe. Und genau wie bei einem Menschen, welcher etwas Neues kennenlernt, weiß auch das Neuronale Netz nicht, was dies Korrekte Antwort auf ein Problem ist. Es muss sich also herantasten an die Lösung.

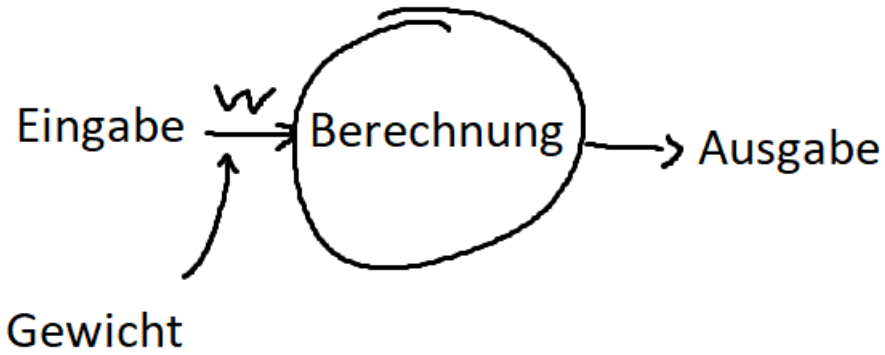

Abbildung 12: Eingabe → Berechnung → Ausgabe

Ein einfaches mathematisches Beispiel hierfür wäre, wenn man einen einfachen Faktor mit zwei Nachkommastellen herausfinden möchte. Beispiel: Euro in USD umrechnen. Hierfür ist ein Faktor kleiner 1 nötig. Bevor der ganze Prozess des lernens startet, muss

man mit einer Eingabe starten. Dies ist zum Beispiel die Zahl 1. Somit erkläre ich der künstlichen Intelligenz im übertragenen Sinne: “Bitte errechne mir wie viele USD in meiner Hand liegen.”. Es wurde also ein Input für die KI gefunden. Dieser Input wird über eine nicht ganz zufällig gewählte Gewichtung in eine Berechnung umgewandelt. Nehmen wir an, diese Gewichtung sei ein Faktor <1 also 0,99. “Nicht ganz zufällig gewählt”, weil der Anwender, welcher die KI schreibt im Vorhinein schon wusste, dass ein USD weniger Wert ist, als ein Euro; der genaue Wert war jedoch unbekannt. Nach dieser Berechnung kommt ein Wert raus – nämlich 0.99 – welcher falsch ist. Damit die künstliche Intelligenz jedoch weiß, ob sie richtig oder falsch rechnet, muss man ihr ein Feedback geben. Das heißt, dass der errechnete Wert und die Abweichung vom tatsächlichen Ergebnis (wenn dieses bekannt ist) zurückgegeben wird. In den meisten Fällen ist das Ergebnis bei präparierten Daten bereits bekannt, da diese als Trainingsdaten agieren. Wenn die künstliche Intelligenz jedoch selbst Entscheidungen vorhersagen muss, muss diese bereits mit solchen Daten trainiert worden sein, um ein akkurates Ergebnis liefern zu können. In dieser Phase lernt sie jedoch auch nicht mehr dazu. Sind die Ergebnisse unbekannt kann zum Beispiel das zuvor erklärte Reinforcement Learning als Lernstrategie herangezogen werden. Diese Abweichung wird nun mit einer weiteren Berechnung rückpropagiert und die Gewichte werden aktualisiert. Dieses Prozedere (Eingabe → Berechnung → Ausgabe → Fehler → Fehler rückpropagieren → Gewichte anpassen) wird so oft mit weiteren Trainingsdaten wiederholt, bis die KI einen minimalen Fehler (Differenz zwischen dem tatsächlichen Ergebnis und der Vorhersage) erreicht. Allerdings können hier einige Faktoren zusätzliche beeinflusst werden, bevor die KI tatsächlich zu lernen beginnt, um ein maximal genaues Ergebnis zu erzielen. Beispielsweise beträgt der Wechselkurs von Euro zu USD 0,89. Als ein USD ist nur 0,89 so viel wert wie ein Euro. Wenn man es der KI ermöglicht sich nur in zehntel-Schritte an die Lösung anzupassen, so passiert folgendes: Die KI wird das tatsächliche Ergebnis von 0,89 nie erreichen. Die von der KI errechnete Lösung beträgt entweder 0,9 oder 0,8. Wenn man diesen Anpassungswert jedoch kleiner ansetzt, auf ein Hundertstel zum Beispiel, so wird diese den Wert zwar langsamer erreichen, jedoch wird er genau dem Ziel entsprechen. Dieser Wert darf jedoch auch nie zu klein gewählt werden. Schaut man sich dies an einem etwas komplexeren Beispiel an, so ist klar zu erkennen, dass in der folgenden Kurve der tatsächliche minimale Fehler nie erreicht wird. Um dies zu vermeiden, gibt es verschiedene Methoden. Als Exempel kann eine Art Fehler-Abtastungs-Beschleunigung herangezogen werden. Hier wird der Faktor, um welchen die Abweichung korrigiert wird in einer Art Beschleunigung

angepasst. Man muss sich also den Aktuellen wert wie einen Ball vorstellen, welcher über einen Berg hinunterrollt.

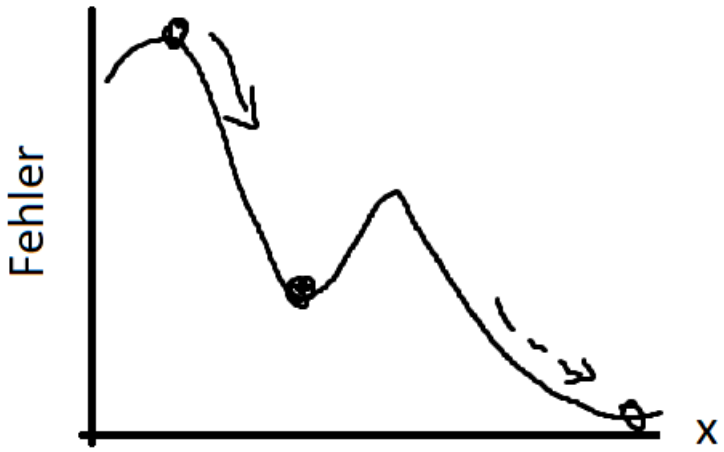

Abbildung 13: Fehlerkurve

Durch das Momentum kann der Ball nachfolgende Hindernisse überwinden. Die Gefahr dabei besteht jedoch, dass wenn der Ball nicht noch einmal angeschubst wird in einem höheren Tal zum Stehen kommt, weil das Momentum zum Zurückkommen fehlt. Es kann aber auch sein, dass der Ball nie genug Momentum hatte, da er zu wenig stark losbewegt wurde. All das und VIELES mehr fällt unter dem Begriff “Hyperparameter Tuning”. Es muss also schon zu Beginn, bevor das Neuronale Netz überhaupt zu lernen beginnt, darauf geachtet werden, dass die Parameter stimmen, damit das Ergebnis einer perfekten Lösung am ähnlichsten ist. Auch bei der Auswahl der Trainingsdaten ist enorme Vorsicht geboten. So ist es in einem amerikanischen Experiment dazu gekommen, dass eine Künstliche Intelligenz, rassistisch wurde. Diese Software sollte Richtern dabei helfen Häftlinge nach ihrer Entlassung zu beurteilen, wie hoch die Wahrscheinlichkeit sei, dass diese eine Wiederholungsstraftat begehen. Dabei kam heraus, dass dunkelhäutige Menschen weitaus gefährlicher eingestuft wurden als alle anderen. Das passierte, nicht weil diese tatsächlich eine höhere Gefahr darstellen, sondern weil in den USA dunkel-häutige Menschen öfter verhaftet werden. So stimmten auch die Probationen in den für die KI vorliegenden Trainingsdatensätzen nicht. Daraufhin meinte ein beteiligter Journalist: „Künstliche Intelligenz hat keine Meinung oder ein Bewusstsein, sondern handelt nach dem, was wir ihr vorgeben. Diese Daten und Informationen sind oft ein

Spiegel der Gesellschaft und reproduzieren so auch Vorurteile, zum Beispiel durch Über- und Unterrepräsentation“. Aaaaaa wie duat ma zitieren digga

5 Umsetzung

5.1 Web-Game [R]

5.1.1 Frontend [R]

Für die Umsetzung des Frontends wird p5.js beziehungsweise p5.play.js verwendet. Eine detaillierte Beschreibung zu diesem JavaScript-Bibliothek und wie man es in ein Projekt einbinden kann, wird in Kapitel 4.1.1 genau beschrieben. p5.play.js basiert auf einer Sprite Klasse. Diese Sprite-Klasse hat einige praktische vordefinierte Funktionen für Collisiondetection oder Animation Support. Um einen Sprite zu erstellen, wird einfach die Funktion createSprite() aufgerufen.

```
1       function setup() {
2           createCanvas(1920,1080);
3           // create a sprite
4           createSprite(50,50,30,30);
5       }
6
7       function draw() {
8           // draw all the sprites added to the sketch so far
9           // the positions will be updated automatically at every cycle
10          drawSprites();
11      }
```

Es ist zu beachten, dass die ersten 2 Parameter der Funktion jeweils die Position am Bildschirm in Pixel angeben, und die letzten 2 die Breite und die Höhe definieren. Das Ergebnis:

Abbildung 14: Einfacher Sprite

Es ist noch nicht viel zu sehen, nur ein simpler Sprite, der default-mäßig ein einfaches Rechteck mit zufälliger Farbe auf einer Position erschienen ist. Für den Player, den man in dem Web-Game sieht, wird diese Sprite-Klasse noch um ein paar Attribute erweitert:

```
class Player {
    constructor(sprite) {
        this.sprite = sprite;
        this.id = null;
        this.knockback = 1;
        this.death = 0;
        this.kills = 0;
        this.dmgDealt = 0;
        this.item = [];
        this.damagedBy = null;
        this.direction  ="";
    }
}
```

Abbildung 15: Player-Klasse

Zum Erstellen der Player-Klasse wird zwar ein Sprite benötigt, damit der Player am Bildschirm angezeigt wird, aber Attribute wie:

- id: Zur eindeutigen Identifizierung
- knockback: Wert, der erhöht wird, desto öfter man getroffen wird; desto höher der Wert, desto weiter wird man von Projektilen weggestoßen
- death: Anzahl, wie oft man gestorben ist
- kills: Anzahl an Kills, die man gemacht hat
- dmgDealt: Anzahl, wie oft man jemanden getroffen hat
- item: Array von Items, die man gerade besitzt
- direction: String, der die Richtung angibt, in die man gerade schaut (links oder rechts)

Während des Spiels werden auf der Map Items gespawned. Welcher Algorithmus dahinter steckt, wird im Kapitel TODO genauer beschrieben. Es wird zwischen 5 unterschiedlichen Items entschieden. Eine Item-Klasse hat ähnlich wie die Player-Klasse als Hauptbestandteil einen Sprite, doch auch hier werden noch weitere Attribute benötigt.

```
class Item {
    constructor(type) {
        this.type = type;
        this.dropped = false;
        switch (this.type) {
            case "bomb":
                this.ammo = 5;
                this.sprite = undefined;
            break;
            case "black_hole":
                this.ammo = 5;
                this.sprite = undefined;
            break;
            case "piano":
                this.ammo = 10;
                this.sprite = undefined;
            break;
            case "mine":
                this.ammo  =3;
                this.sprite = [];
        }
    }
}
```

Abbildung 16: Item-Klasse

- type: Gibt den Typ des Items andere
- dropped: Gibt an, ob das Item irgendwo auf der Umgebung gelandet ist
- ammo: Anzahl, wie oft man das Item benutzen kann

Je nachdem, welchen Typ das Item bekommen hat, wird auch die Anzahl wie oft man es benutzen kann, verändert. Wurde es zum Beispiel zum Type "bomb", kann man 5 Mal eine Bombe werfen.
Bei dem Typ "mine", also einer Mine, kann man mehrere Minen gleichzeitig legen, deshalb ist das Sprite-Attribut auch ein Array. Bei den anderen Items kann immer nur ein Sprite davon existieren.

Die Items

Grundsätzlich gibt es 5 unterschiedliche Items in Scribble-Fight. Je nachdem, welches Item man aufgesammelt hat, kann man verschiedene Fähigkeiten aktivieren. Für jedes Item existiert eine physics-Methode, die in der Draw-Funktion aufgerufen wird.

Keyboard-Access

Die Items werden alle durch Tasten auf der Tastatur ausgelöst. Durch p5.js kann man

sehr leicht auf die Tastatur zugreifen. Mithilfe von ASCII Code kann man jede Taste einzeln ansteuern. Mit der p5.js Methode <keyWentDown(Key)> wird überprüft, ob die Taste gerade gedrückt wurde.

Listing 3: Keyboard-Access

```
1       // E
2       if (keyWentDown(69)) {
3           bombAttack();
4       }
5       // Q
6       if (keyWentDown(81)) {
7           blackHoleAttack();
8       }
9       // R
10      if (keyWentDown(82)) {
11          pianoTime();
12      }
13      // C
14      if (keyWentDown(67)) {
15          placeMine();
16      }
17      // F
18      if (keyWentDown(70)) {
19          makeMeSmall();
20      }
```

Bomb

Das Bombem-Item wird mit der Taste <E> aktiviert.

Durch p5.play.js kann man mit <sprite.velocity.y> die vertikale Geschwindigkeit eines Sprites verändern. Dadurch kann eine Gravitation simuliert werden. Außerdem kann man mit <sprite.bounce(otherSprite)> Sprites an anderen Sprites oder Gruppen von Sprites 'abspringen' lassen. Dies wird benutzt, damit die Bombe an der Umgebung abprallt Danach wird mit <sprite.overlap(myPlayer.sprite)> überprüft, ob eine Bombe mit meinem Spieler-Sprite kollidiert. Falls ja, wird die Bombe mit <sprite.remove()> entfernt. Falls nein, wird noch überprüft, ob sich die Bombe außerhalb des Bildschirms befindet, und falls dies zutrifft, wird auch in dem Fall die Bombe entfernt.

Listing 4: Bomb Item Physics

```
1       // verringern der vertikalen Geschwindigkeit
2       bomb.velocity.y -= GRAVITY;
3       // bombe prallt an der Umgebung ab
4       bomb.bounce(environment);
5
6       // kollidert die Bombe mit meinem Player
7       if(bomb.overlapSprite(myPlayer.sprite)) {
8           // my Player gets knocked back
9           myPlayer.sprite.getsThrownAway();
10          // bomb gets deleted
11          bomb.remove();
12      } else if(bombIsOutsideMonitor()) {
13          // bomb gets deleted
14          bomb.remove();
15      }
```

Black Hole

Das Black-Hole-Item wird mit der Taste <Q> aktiviert.

Wie bei dem Bomben-Item wird auch bei dem Black-Hole-Item Gravitation simuliert. Jedoch nur für eine kurze Zeit, bis das Item in der Luft stehen bleibt und in einem Radius alle Spieler, die sich in diesem Radius befinden, anzieht, und diese auch alle Fähigkeiten nimmt. Um zu überprüfen, wie lange das Item schon existiert, stellt p5.play.js das <life>-Attribut zur Verfügung. Dieser Wert ist ein Countdown, der sich bei jedem Draw-Zyklus um 1 verringert, bis sich das Item dann bei dem Wert 0 selbst löscht. Ein weiterer wichtiger Punkt ist bei dem Item, wie es Sprites anziehen kann. Dazu wurde die attraction Funktion von p5.play.js (mit leichten Veränderungen) benutzt:

Listing 5: Attraction

```
1       // attraction
2       if (myPlayer.sprite.overlap(b)) {
3           noGravity = true;
4           var angle = atan2(myPlayer.sprite.position.y - b.position.y,
                myPlayer.sprite.position.x - b.position.x);
5           if (myPlayer.sprite.velocity.y >= -pixelWidth &&
                myPlayer.sprite.velocity.y <= pixelWidth) {
6             myPlayer.sprite.velocity.x -= cos(angle);
7           }
8           myPlayer.sprite.velocity.y -= sin(angle);
9         }
```

Die Black-Hole-Physics Funktion sieht also (vereinfacht) so aus:

Listing 6: Black Hole Item Physics

```
1       // if the item has reached a certain life, make it static and attract players
2       if (b.life <= 400) {
3           attraction(b);
4           b.velocity.y = 0;
5           b.velocity.x = 0;
6         }
7
8         // if the item has not reached a certain life, let it bounce off the
              environment
9         if (b.life > 400) {
10          b.velocity.y -= GRAVITY;
11          b.bounce(environment);
12        }
13
14        // if the item is outside of the monitor, delete it
15        if (b.position.x > windowWidth || b.position.y > windowHeight || b.life ==
              0) {
16          b.remove();
17        }
```

Piano

Das Piano-Item wird mit der Taste <R> aktiviert.

Das Piano-Item ist, wie der Namen schon vermuten lässt, ein Klavier, das am höchsten Punkt der Map erscheint und nach unten fällt. Das bedeutet, die y-Koordinate ist 0 und die x-Koordinate ist die selbe wie die, die der Sprite des Spielers hat, der das Piano aktiviert hat. Bei Kontakt zu einem Spieler oder der Umgebung wird das Klavier zerstört. Zum Überprüfen auf Kollisionen wird die p5.js Methode <sprite.collide(otherSprite)>

verwendet. Diese wird true, falls eine Kollisionen zwischen zwei Sprites oder Sprite-Gruppen stattfindet.

Listing 7: Piano-Item Physics

```
1       // check for collisions
2       if (p.collide(environment)) {
3           p.remove();
4         } else if (p.overlap(myPlayer.sprite)) {
5           myPlayer.sprite.getsThrownAway()
6           p.remove();
7         }
8         p.velocity.y -= GRAVITY;
```

Mine

Das Minen-Item wird mit der Taste <C> aktiviert. Das Minen-Item ist das einzige Item, bei dem man mehrere Instanzen auf einmal entsenden kann. Es taucht hinter dem eigenen Player-Sprite auf und fliegt so lange nach unten, bis es auf der Umgebung landet. Erst wenn es wo gelandet wird, wird es 'aktiv'. Wenn eine Mine aktiviert worden ist, und ein Spieler in Berührung mit dem Item kommt, wird dieser in die Luft gestoßen und die Mine wird gelöscht.

Listing 8: Mine-Item Physics

```
1       // check if mine has landed somewhere (if true: activate mine)
2       if (m.collide(environment) && m.touching.bottom) {
3           m.set = true;
4       }
5       if (m.overlap(myPlayer.sprite) && m.set) {
6           myPlayer.sprite.getsThrownAway();
7           m.remove();
8       }
9       m.velocity.y -= GRAVITY;
```

Size-Reduction

Das Size-Reduction-Item wird mit der Taste <F> aktiviert.

Dieses Item ist das einzige, das keinen eigenen Sprite hat. Das einzige was diese Item macht, ist, den Spieler-Sprite zu verkleinern. Dadurch wird dieser schwieriger zu treffen. Der Code, der dies umsetzt, sieht vereinfacht so aus:

Listing 9: Size-Reduction

```
1   // gets called on key press F
2   function makeMeSmall() {
3     if (doIHaveTheItem()) {
4       imSmall = true;
5       smallTimer = 10;
6     }
7   }
8
9   // gets called in draw function
10  function smallChecker() {
```

```
11    if (imSmall) {
12      // scale the sprite down at the start of countdown
13      if (smallTimer == 10) {
14        myPlayer.sprite.scale = 0.6;
15      }
16      // every second (60 frames), the countdown gets reducced
17      if (frameCount % 60 == 0 && smallTimer > 0) {
18        smallTimer--;
19      }
20      // if the countdown is over, rescale the sprite back to the original form
21      if (smallTimer == 0) {
22        myPlayer.sprite.scale = 1;
23        smallTimer = 10;
24        imSmall = false;
25      }
26    }
27
28  }
```

Die Default-Attacke

Die Default-Attacke wird mit <Left-Click> aktiviert.

p5.js bietet sehr leicht die Möglichkeit, auf User-Input zu überprüfen. Das einzige, was nötig ist um zu erkennen, ob der User gerade die linke Maustaste geklickt hat, ist die Funktion <mouseClicked()>. Mit diesen Funktionalitäten wird nun zu dem Punkt, auf dem man gerade die Maus hält und die Default-Attacke aktiviert, ein Projektil abgeschossen. Um die Information zu erhalten, auf welcher X- und Y-Position sich die Maus befindet, werden die von p5.js vordefinierten Eigenschaft <camera.mouseX> und <camera.mouseY> verwendet.

Listing 10: Size-Reduction

```
1     function mouseClicked() {
2     // Maus-Position
3     let x = camera.mouseX,
4         y = camera.mouseY;
5     // Sprite wird bei meiner Player-Sprite-Positon erstellt
6     projectile = createSprite(myPlayer.sprite.position.x,
          myPlayer.sprite.position.y, pixelWidth, pixelWidth);
7
8     // Geschwindigkeit wird auf die Position der Maus ausgerichtet
9     projectile.velocity.x = (x - myPlayer.sprite.position.x);
10    projectile.velocity.y = (y - myPlayer.sprite.position.y);
11  }
```

Movements

Es gibt 3 fundamentale Bewegungsmöglichkeiten in Scribble-Fight. Springen, links/rechts laufen und 'klettern'. Diese Bewegungen werden im Folgenden genauer erläutert.

Springen

In dem Web-Game springt man mittels Leertaste. Genau wie bei den Items, wird mittels den p5.js Methoden der User-Input ermittelt. Bei dem Springbewegung wird einfach die vertikale Geschwindigkeit des Player-Sprites so verändert, dass dieser etwas nach

oben springt.

Listing 11: Jumping

```
    // check if user pressed spacebar
    if (keyWentDown(32)) {
        jump()
    }

    function jump() {
    // user is only allowed to jump 2 times (it resets when touching the ground)
    if (!(JUMP_COUNT >= MAX_JUMP)) {
        // make the user fly up a bit (JUMP is a global variable)
        myPlayer.sprite.velocity.y = -JUMP;
        JUMP_COUNT++;
    }
}
```

Links/Rechts Laufen

Das Prinzip des Links oder Rechts Bewegens ist sehr simpel. Es wird einfach die horizontale Geschwindigkeit des Player-Sprites auf eine konstante Variable gesetzt. Wenn man sich nach rechts bewegt ist diese Konstante positiv, bei einer Linksbewegung negativ. Im Gegensatz zur Springbewegung wird diesmal aber die Methode <keyIsDown(key)> verwendet, und nicht <keyWentDown(key)>. Der Grund dafür ist, dass die Bewegung so lange anhalten soll, wie der User die Taste drückt, und nicht nur einmal pro Tastendruck.

Listing 12: Links/Rechts-Movement

```
    //A
    if (keyIsDown(65)) {
        moveLeft()
    }
    //D
    if (keyIsDown(68)) {
        moveRight()
    }

    // SPEED is a global variable
    function moveLeft() {
    myPlayer.sprite.velocity.x = -SPEED;
    }

    function moveRight() {
    myPlayer.sprite.velocity.x = SPEED;
    }
```

Bewegung auf der Spielumgebung

Die Bewegung auf der Spielumgebung wird durch die Methode <collisions()> bestimmt. Diese wird in der draw-Methode aufgerufen und wird somit 60 mal die Sekunde ausgeführt. Berührt der Sprite des Players nichts, fällt er er mit konstanter Beschleunigung nach unten. Findet jedoch eine Kollision mit der Spielumgebung statt, dann wird überprüft, welche Art von Berührung gerade stattfindet:

- Falls Berührung seitlich: Player-Sprite bekommt eine Klettergeschwindigkeit und behält diese so lange, wie die Berührung stattfindet
- Falls Berührung unten: Die vertikale Geschwindigkeit des Player-Sprites wird auf 0 gesetzt und der Sprung-Counter zurückgesetzt
- Falls Berührung oben: Mit einer Berührung die zwischen Spielumgebung und Sprite stattfindet, kann man nicht klettern noch wird der Sprung-Counter zurückgesetzt

Listing 13: Bewegung auf der Spielumgebung

```
1       // check for collisions
2       if (myPlayer.sprite.collide(environment)) {
3         // if the collision is on the side, the sprite will start "climbing" with a
              certain speed
4         if (myPlayer.sprite.touching.left || myPlayer.sprite.touching.right) {
5           myPlayer.sprite.velocity.y = CLIMBINGSPEED;
6         }
7         // jump count gets only reset when the collision is not on the top of the
              player-sprite
8         if (!myPlayer.sprite.touching.top) {
9           JUMP_COUNT = 0;
10        }
11      }
```

Richtungswechsel des Sprites

Ein weiterer wichtiger Aspekt bei der Bewegung des Player-Sprites ist, dass sich auch die Orientierung des Bildes ändern muss, wenn dieser die Richtung wechselt. Auch für diese Problemstellung stellt p5.play.js eine Lösung zu Verfügung. Mit der Methode <mirrorX()> wird der Sprite entlang seiner vertikalen Achse gespiegelt. Jedes mal, wenn der User nun also die Richtung seines Player-Sprites wechselt, wird auch der Sprites passend seiner Bewegung gespiegelt.

Listing 14: Sprite Richtungswechsel

```
1       function mirrorSprite() {
2           // A
3           if (keyWentDown(65)) {
4               mirrorSpriteLeft()
5           }
6           // D
7           if (keyWentDown(68)) {
8               mirrorSpriteRight()
9           }
10      }
11
12      function mirrorSpriteLeft() {
13          // if the mirrorX attribute is 1, then the sprite is looking to the right
14          if (myPlayer.sprite.mirrorX() === 1) {
15              myPlayer.sprite.mirrorX(myPlayer.sprite.mirrorX() * -1);
16              myPlayer.direction = "left";
17          }
18      }
19
20      function mirrorSpriteRight() {
21          // if the mirrorX attribute is 1, then the sprite is looking to the left
22          if (myPlayer.sprite.mirrorX() === -1) {
23              myPlayer.sprite.mirrorX(myPlayer.sprite.mirrorX() * -1);
24              myPlayer.direction = "right";
25          }
26      }
```

Wie gewinne ich?

Um in Scribble-Fight zu gewinnen, muss man seinen Gegner/seine Gegner drei mal erfolgreich von der Spielumgebung schießen, so, dass der Sprite des Gegners ins Nichts fällt. Dies ist durch die Attacken, die in Kapitel 5.1.1 genau beschreiben werden, möglich. Um zu überprüfen, ob der Sprite nun hinuntergefallen und somit 'gestorben' ist, wird in der draw-Methode die Funktion <deathCheck()> aufgerufen. Diese hat einige Aufgaben:

- Überprüfen ob sich der Player-Sprite außerhalb des Bildschirms befindet
- Falls ja, überprüfen ob der Player noch mindestens ein Leben hat
- Falls der Player keine Leben mehr hat, wird sein Sprite zerstört
- Hat der Player noch weitere Leben, dann wird er nach drei Sekunden an einem zufälligen Punkt, an dem er nicht direkt wieder aus dem Bildschirm fällt, gespawned
- Wird der Player respawned, werden ihm alle seine Items, die er eventuell noch hatte, wieder genommen
- Wird der Player respawned, wird sein Knockback wieder zurückgesetzt

Vereinfacht sieht die Methode also so aus:

Listing 15: Überprüfung nach Toden

```
function deathCheck() {
    // check if sprite has fallen outside of the monitor
    if (myPlayer.sprite.position.y - player_height > windowHeight && !respawnTime)
        {
        youDied();
    }
}

function youDied() {
    myPlayer.removeItem();
    myPlayer.death++;
    myPlayer.knockback = 0;
    respawnTime = true;

    // after 3 seconds the player gets respawned on a location, if he still has at
        least one live left
    setTimeout(() => {
        if (myPlayer.death < 3) {
            myPlayer.sprite.position.x = xCoordinates[Math.floor(Math.random() *
                xCoordinates.length)];
            myPlayer.sprite.position.y = 0;
            respawnTime = false;
        }
    }, 3000);
}
```

Außerdem zählt es auch als Tod, wenn der Knockback des eigenen Players über 100 ansteigt. Dies wird mit der <fatalHit()>-Methode überprüft. Diese Methode wird immer dann aufgerufen, wenn der eigene Player von irgendeinem Projekil getroffen wurde.

Listing 16: Fatal Hit

```
1       function fatalHit() {
2           if (myPlayer.knockback >= 100) {
3               youDied();
4           }
5       }
```

5.1.2 Backend [R]

Node.js [R]

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]

Nachdem der Spieler der Lobby beigetreten ist, wird es Zeit für ihn die Spielumgebung zu zeichnen und diese für die Abstimmung freizugeben. Das passiert indem dieser einen QR-Code ("Quick Response"), welcher sich in der Mitte der Lobby befindet, mit einem mobilen Endgerät, welches über eine Kamera verfügt, abscannt. Somit wird er auf eine Webseite geleitet, welche wieder auf die Kamera zugreift. Auf dieser ist bereits eine Objekterkennung vorprogrammiert. Auf die Funktionsweise und auf die technische Umsetzung wird im folgenden ... eingegangen. Wenn der Spieler merkt, dass die Objekterkennung das Blatt erkannt hat, kann dieser ein Bild aufnehmen. Im nächsten Schritt ist es nochmals möglich die erkannten Kanten zu verschieben. Wenn sich der User für das Foto als Map entscheidet kann er den ausgewählten Teil in eine top-down-2d-Perspektive umwandeln. Kurzgesagt werden dabei die vier ausgewählten Ecken, welche über die Kanten des Blatt Papiers liegen, für eine Perspektiven-Transformation mit OpenCV2 herangezogen. Wenn der Spieler dann so weit ist, kann er dieses Bild

beziehungsweise diese Map bestätigen und sendet sie somit zurück an die Lobby. Dann kann die Abstimmung beginnen.

5.3.1 Objekterkennung [H]

Das Thema “Objekterkennung” ist ein extrem schnell wachsendes und sich verbesserndes Feld in der Welt der Bildverarbeitung beziehungsweise Bildanalyse.
Die Objekterkennung ist ein großer Teil der Arbeit. Hier wird aus dem live Footage der Kamera das Blatt Papier und anschließend in einer fertigen Aufnahme die Map erkannt.

Die Objekterkennung wird zum Beispiel bei komplexen und großen Paketdiensten eingesetzt. Sollen Pakete nach Farbe gruppiert werden, so reicht bereits ein Farbsensor. Das ist in den meisten Fällen jedoch nicht genügend. Vielmals werden Pakete auf Strichcodes oder Adressen überprüft. Wenn also unterschieden werden muss, wo welches Paket landen soll, so ist eine Kamera nötig. Diese muss ein genügend gut auflösendes Bild aufnehmen, dass daraus Informationen gewonnen werden können. Diese Daten müssen zur Auswertung mathematisch beschrieben werden. Zur Übersetzung werden Verfahren, wie Kantenerkennung, Transformationen oder Größen- und Farberkennung, aber auch künstliche Intelligenzen eingesetzt. In dieser Arbeit wurden Kanten zur Objekterkennung herangezogen. Die Korrektheit des Ergebnisses ist abhängig von der Korrektheit der bereitgestellten Daten und wie genau das Objekt mathematisch beschrieben werden kann.
Ein weiteres Beispiel, wo Objekterkennung eingesetzt wird, ist in Fahrerassistenzsysteme oder autonomes Fahren. Bereits in der ersten Stufe des autonomen Fahrens (dem assistierten Fahren) werden zur Hilfe Kameras außerhalb des Fahrzeugs angebracht. In einem Auto, welches als Stufe eins autonomes Assistenzsystem eingestuft wird, werden diese Kameras für einen automatischen Spurhalteassistenten eingesetzt. Via Bildanalyse und Objekterkennung werden die Linien auf der Straße erkannt.
In höheren Stufen ist die Objekterkennung nicht mehr wegzudenken. Es müssen Verkehrsschilder und Personen auf der Straße erkannt werden um daraufhin entsprechen zu agieren.

Andere Beispiele:

- Gesichtserkennung, um das Smartphone zu entsperren.

- Qualitätskontrolle, zur Erkennung und automatischen Entfernung von kaputten oder beschädigten Teilen.
- Personenerkennung, um Menschenmassen zu analysieren.

Beschreibung der Funktionalität [H]

Wieso für Kontouren entschieden

Welche Abfolge von CV2 Bildverarbeitungsalgorithmen haben zum ergebnis geführt noch nicht auf die Algorithmen näher eingehen

5.3.2 Open-CV2 [H]

Bildverarbeitungsalgorithmen

auf die algos näher eingehen

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

Listing 17: Bild in Spielbare Map umwandeln

```
1       def getPlayableArray(img):
2           np.set_printoptions(threshold=sys.maxsize)
3
4           alpha_img = cv2.cvtColor(img, cv2.COLOR_BGR2BGRA)  # rgba
5           imgWarpGray = cv2.cvtColor(alpha_img, cv2.COLOR_BGR2GRAY)
6           blurred = cv2.GaussianBlur(imgWarpGray, (7, 7), 0)
7           imgAdaptiveThre = cv2.adaptiveThreshold(
8               blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
                    7, 2)
9           imgAdaptiveThre = cv2.bitwise_not(imgAdaptiveThre)
10          imgAdaptiveThre = cv2.medianBlur(imgAdaptiveThre, 3)
11
12          # make image square
13          imgAdaptiveThre = np.array(makeSquare(
14              cv2.cvtColor(imgAdaptiveThre, cv2.COLOR_BGR2BGRA)))
15
16          img = cv2.cvtColor(imgAdaptiveThre, cv2.COLOR_BGR2BGRA)
17
18          # pippoRGBA2 = Image.fromarray(np.array(img).astype('uint8'), mode='RGBA')
19          # pippoRGBA2.show()
20          cv2.imwrite(
21              './source/prototypes/streamFusion/output/imgAdaptiveThre.png',
                    imgAdaptiveThre)
22
23          iar = np.asarray(img).tolist()
24
25          rows = len(iar)
26          columns = len(iar[0])
27
28          meshes = 3025
29          # percent = perc(rows * columns)
30          percent = 95
31          n = math.ceil(np.sqrt(rows * columns / meshes))
32          x = 0
33          y = 0
34
35          newImg = []
36
37          while y < rows:
38              newImg.append([])
39              while x < columns:
```

```
40
41                  i = 0
42                  j = 0
43                  bg = 0
44                  while i < n:
45                      while j < n:
46                          if (y + j) < rows and (x + i) < columns:
47                              if np.all(iar[y + j][x + i][:3] == [255, 255, 255], 0):
48                                  bg += 1
49                          else:
50                              bg += 1
51                          j += 1
52                      j = 0
53                      i += 1
54
55                  bgPercent = bg / (n**2)
56                  if (bgPercent < (percent / 100)):
57                      newImg[int(y / n)].append([0, 0, 0, 255])
58                  else:
59                      newImg[int(y / n)].append([255, 255, 255, 0])
60
61                  x += n
62              x = 0
63              y += n
64
65          iar = np.asarray(newImg).tolist()
66          with open('./source/prototypes/streamFusion/output/mapArray.txt', 'w') as
                f:
67              f.writelines(repr(iar))
68
69          # pippoRGBA2 = Image.fromarray(np.array(newImg).astype('uint8'),
                mode='RGBA')
70          # pippoRGBA2.show()
71          cv2.imwrite(
72              './source/prototypes/streamFusion/output/newImg.png', np.array(newImg))
73
74          return newImg
```

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

5.4 KI [H]

Der erste Punkt ist die Künstliche Intelligenz. Im Laufe der Ausarbeitung der Diplomarbeit und der damit zusammenhängenden Forschung änderte sich oft die Vorstellung 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.

5.4.1 Lernen mit OpenAI-Gym [H]

5.4.2 Künstliche Intelligenz Definition [H]

5.4.3 Reinforcement Learning [H]

5.4.4 Die ScribbleFight-KI [H]

Beschreibung der Funktionalität [H]

Warum Python? [H]

5.4.5 Tensorflow und Keras [H]

Tensorflow [H]

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