Dokumentation

Projektname	Counter Strejk - Ikea Offensive
Projektnummer	18
Projektleiter	Tobias Weiß
Erstellt am	27.01.2022
Letzte Änderung am	02.06.2022
Status	Fertig
Aktuelle Version	1.0

Änderungsverlauf

Nr.	Datum	Version	Geänderte Kapitel	Art der Änderung	Autor
1	27.01.2022	1.0	Alle	Erstellung	Tobias Weiß
2	28.01.2022	1.1	Alle	Hinzufügung	Tobias Weiß

1. Inhalt

2. Pflichtenheft

2.1 Einleitung

Die vorlegende Dokumentation gibt über das entwickelte Produkt Auskunft. Dieses Dokument beschreibt alle Funktionalitäten die das Projekt besitzt und gibt Auskünfte über die Hintergründe wie z.B.: warum dies so abgewickelt wurde oder wie ein gewisses Problem gelöst wurde.

2.2 Allgemeines

1.1.1 Ziel und Zweck des Dokuments

Dieses Dokument beschreibt ein Software Projekt für das Unterrichtsfach «POS – Programmierung und Softwareentwicklung». Der Zweck dieses Dokumentes ist es, die Funktionalitäten des Projektes darzustellen, offene Fragen abzudecken und einen generellen Überblick über dieses «Spiel» zu verschaffen.

1.1.1 Ausgangssituation

Es gibt bei diesem Projekt keine genauen Geschäftspartner, jedoch werden unsere Lehrer, Herr Professor Gröbl und Herr Professor Winkler diese Aufgabe beurteilen.

1.1.2 Projektbezug

Das vorliegende Projekt ist ein unabhängiges Projekt.

2.3 Konzept

1.1.2 Ziel(e) des Anbieters

Das Projekt sollte dem Benutzer ein Spiel zur Verfügung stellen, in dem mit jeglichen Waffen gegen andere Spieler in Virtual Reality über das Internet gespielt werden kann. Es sollte hierbei das Augenmerk auf die Funktionalität und die Modellierung gelegt werden, um den Benutzer ein angenehmes Spielerlebnis zu ermöglichen in einer Umgebung in der er sich wohl fühlt.

1.1.3 Ziel(e) und Nutzen des Anwenders

Der Benutzer sollte am Spielstart sich Anmelden können und eine Liste von offenen «Räumen» zu Gesicht bekommen die er beitreten kann um mit den anderen Personen in diesen «Räumen» zu kämpfen. Dabei sollte ein gewisses Reservoir an Waffen zur Auswahl stehen mit der er den anderen Spieler/n Schaden zufügen kann und somit auch die Runde gewinnen kann bzw. eine virtuelle Währung erwirbt. Wenn es zeitlich möglich ist, sollte auch

eine Art «Shop» eingebaut werden, in der der Benutzer diese virtuelle Währung ausgeben kann für stärkere Waffen oder um die bereits vorhandenen Waffen zu verstärken.

1.1.4 Zielgruppe(n)

Anwender dieser Software sollten Computerspiel – Enthusiasten sein, welche über ein Virtual-Reality Headset verfügen und sich gegen andere Spieler messen möchten. Das Spiel wird mit Leichtigkeit zu Bedienen sein, sodass es auch für mehrere Altersgruppen zur Benutzung steht.

2.4 Funktionale Anforderungen

1.1.5 Multiplayer support

um Interaktionen zwischen Benutzern zu erlauben

1.1.6 Interaktionen mit Controller

Möglichkeit, mithilfe der VR-Controller mit den Objekte zu interagieren, wie z.B.: sie aufzuheben oder zu w

1.1.7 Cross-Platforming

damit Benutzer mit unterschiedlichen VR-Headsets zusammen spielen können

1.1.8 Funktionierende Waffen

2.5 Nichtfunktionale Anforderungen

1.1.9 Ansprechendes design

Von der Umgebung, Spieler und Objekte im Low-Poly style (Texturen, Materialien mit geringer Anzahl an Polygonen)

1.1.10 Gesetzliche Anforderungen

Copy-Right usw.

1.1.11 Native App

Sollte „native“ auf gewissen VR Brillen funktionieren (ohne Enterstützung eines anderen Gerätes

1.1.12 Konstante FPS

1.1.13 Animationen

Es sollte Animationen für das laufen, schießen, sterben, usw. erstellt werden

2.6 Rahmenbedingungen

1.1.14 Zeitplan

Unter normalen Bedingungen wird mit einen wöchentlicher Zeitaufwand von mindestens 2h gerechnet. Gearbeitet wird in den Programmieren und Softwareentwicklung Unterrichtsstunden und gelegentlich in meiner Freizeit, falls ich dafür Zeit habe.

1.1.15 Technische Anforderungen

Benötigt wird bei der Projektdurchführung die Game-Engine Unity, die VR-toolkit API und ein PC/Laptop auf dem es möglich ist, das Spiel zu programmieren und generell zu entwickeln. Für die Modellierung wird das Programm «Blender» benötigt. Mithilfe diesem Programm werden Beispielsweise das Aussehen der Spieler/Waffen und deren Animationen konstruiert. Außerdem wird auch noch die App «Virtual Desktop Streamer» und die «Oculus» Software benötigt um es zu erlauben, sich via der Virtual Reality App «Virtual Desktop» von der VR-Brille zum Laptop (wireless) zu verbinden. Somit kann das Spiel problemlos getestet werden, ohne es jedes mal neu zu einer .apk Datei zu Kompilieren.

1.1.16 Problemanalyse

Mögliche Probleme könnten z.B. die neue Technologie sein. Ich habe mich bereits mit der Spiele Entwicklung letztes Jahr intensiver beschäftigt aber es herrscht ein großer Unterschied zur Entwicklung einer Virtual Reality Software und kann der Zeitaufwand ebenfalls für bestimmte Funktionalitäten nur schwer eingeschätzt werden. Ebenfalls sollte Rücksicht auf die benötigte Zeit für das Modellieren in Blender genommen werden. Besonders dieser Schritt ist sehr Zeitintensiv und darf nicht vernachlässigt werden, da das Design des Spiels einen großen Eindruck beim Spieler hinterlässt.

1.1.17 Qualität

Die Qualität sollte durch andere Entwickler, Testpersonen und mir selbst festgestellt werden. Ich werde dabei mehren Bekannten/Freunde das Programm während des Entwicklungsprozesses zu Verfügung stellen, sodass diese hoffentlich mögliche Fehler erkennen. Dadurch können diese vor dem «Release» beseitigt werden.

2.7 Liefer- und Abnahmebedingungen

Die Fertigstellung des Projektes ist Anfangs Juni. Der Umfang sollte ein spielbares Virtual-Reality Computerspiel darstellen. Die Qualitätskontrolle erfolgt – wie vorhin erwähnt – über Freunde/Bekannt oder andere Entwickler

3. Umsetzung

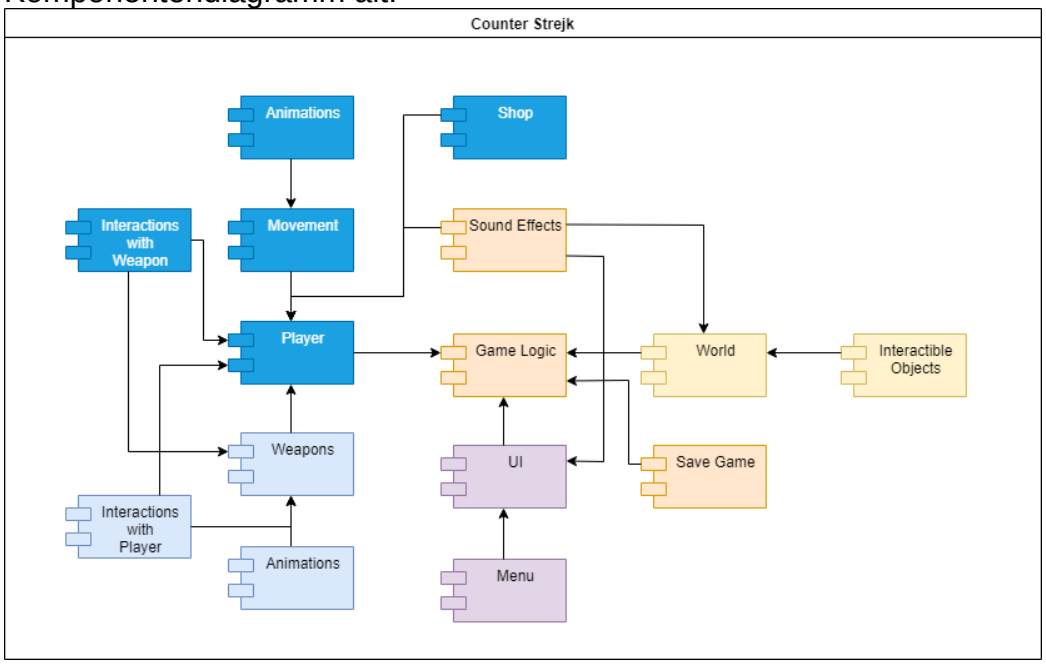
3.1 Softwaredesign

Das Design der Software im Gesamten hat sich stark verändert. Es wurde parallel zum eigentlichen Spiel ein Webserver entwickelt, welcher GET-Anfragen akzeptiert und dann dementsprechend die verbundene Funktionalität ausführt. In diesem Fall wird ein Relais geschalten, welches dann die Stromzufuhr der Schock-Elektroden zum Schock-Gerät ermöglicht.

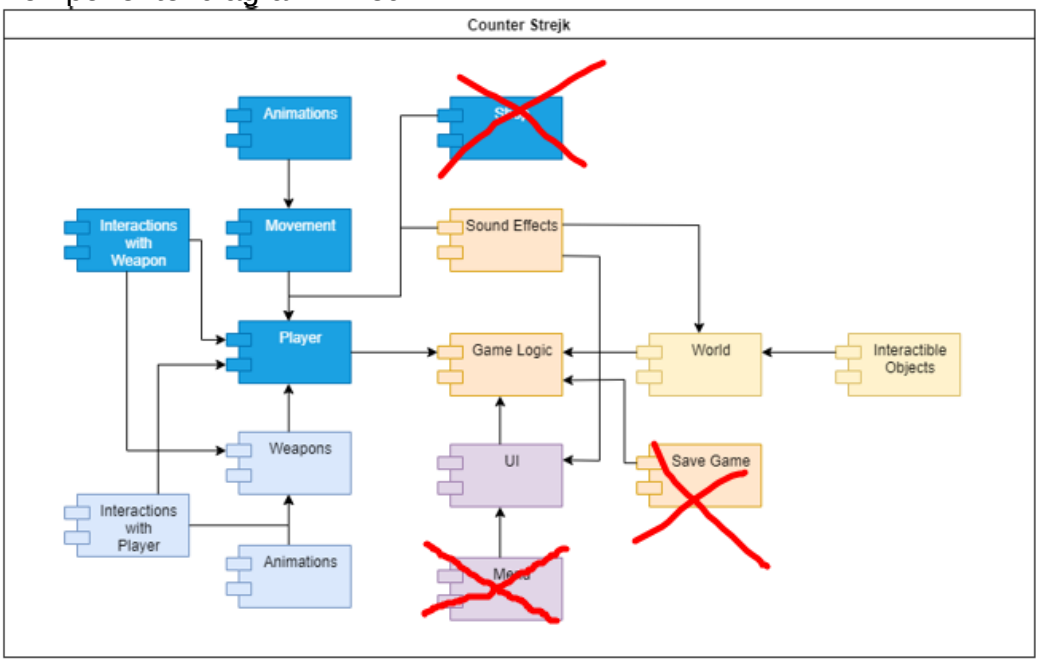
Beim Spiel selbst erfolgten nicht viele Änderungen am Design. Es konnten nur einige, im Punkt 4 und 5 angeführte, Anforderungen nicht umgesetzt werden

aufgrund der Unterschätzung des Zeitaufwandes der Programmierung und Entwicklung der einzelnen Komponenten.

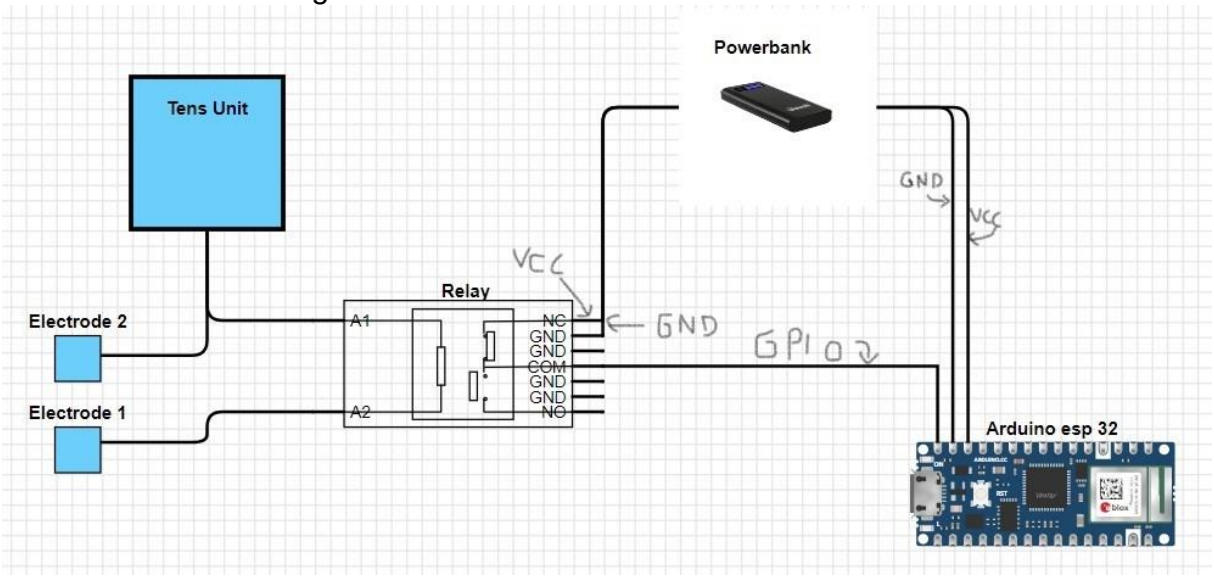
Komponentendiagramm alt:



Komponentendiagramm neu:



Circuit-board PCB diagram:

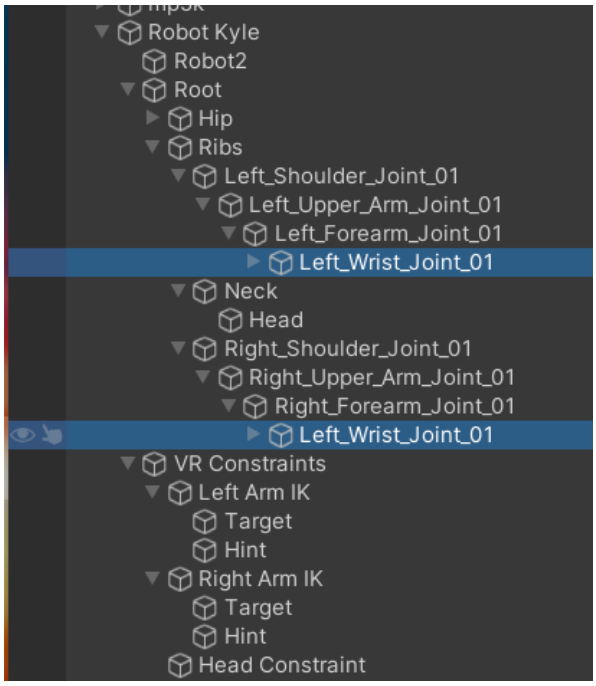


3.2 Interessante Codeausschnitte

Hands.cs

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    InputDeviceCharacteristics controllerCharacteristics = InputDeviceCharacteristics.Controller;
    InputDevices.GetDevicesWithCharacteristics(controllerCharacteristics, devices);

    if (devices[isRight].TryGetFeatureValue(CommonUsages.grip, out float gripValue))
    {
        if (!isHoldingMP5)
        {
            StartCoroutine("recordGrip", gripValue);
            StopCoroutine("recordGripMP5");
            animator.SetFloat("GripMP5", 0);
        }
        if (isHoldingMP5)
        {
            StopCoroutine("recordGrip");
            StartCoroutine("recordGripMP5", gripValue);

        }
    }
}
```

```
0 references
public void setPoseMP5()
{
    isHoldingMP5 = true;
}

0 references
public void unsetPoseMP5()
{
    isHoldingMP5 = false;
}

0 references
IEnumerator recordGripMP5(float gripValue)
{
    animator.SetFloat("GripMP5", gripValue);
    yield return null;
}

0 references
IEnumerator recordGrip(float gripValue)
{
    animator.SetFloat("Grip", gripValue);
    yield return null;
}
```

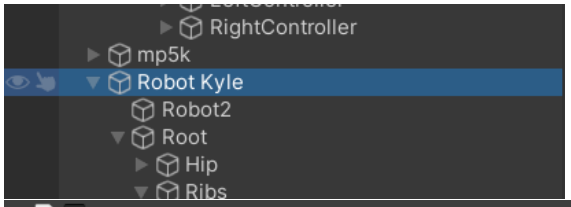


Dieses Skript ist dazu da, um die Handanimationen der VR-Hand auszuführen. Es wird hierbei im Animator eine Variable auf die momentane Grip-Value gesetzt. Hierbei wird der Grip-Button als betätigten Knopf verwendet. Dadurch wird eine «Greif» Animation simuliert.

VRRig.cs

```
public VRMap head;
public VRMap leftHand;
public VRMap rightHand;

public Transform headConstraint;
public Vector3 headBodyOffset;

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    headBodyOffset = transform.position - headConstraint.position;
}

// Update is called once per frame
Unity Message | 0 references
void LateUpdate()
{
    transform.position = headConstraint.position + headBodyOffset;
    transform.forward = Vector3.ProjectOnPlane(headConstraint.up, Vector3.up).normalized;

    head.Map();
    leftHand.Map();
    rightHand.Map();
}
```

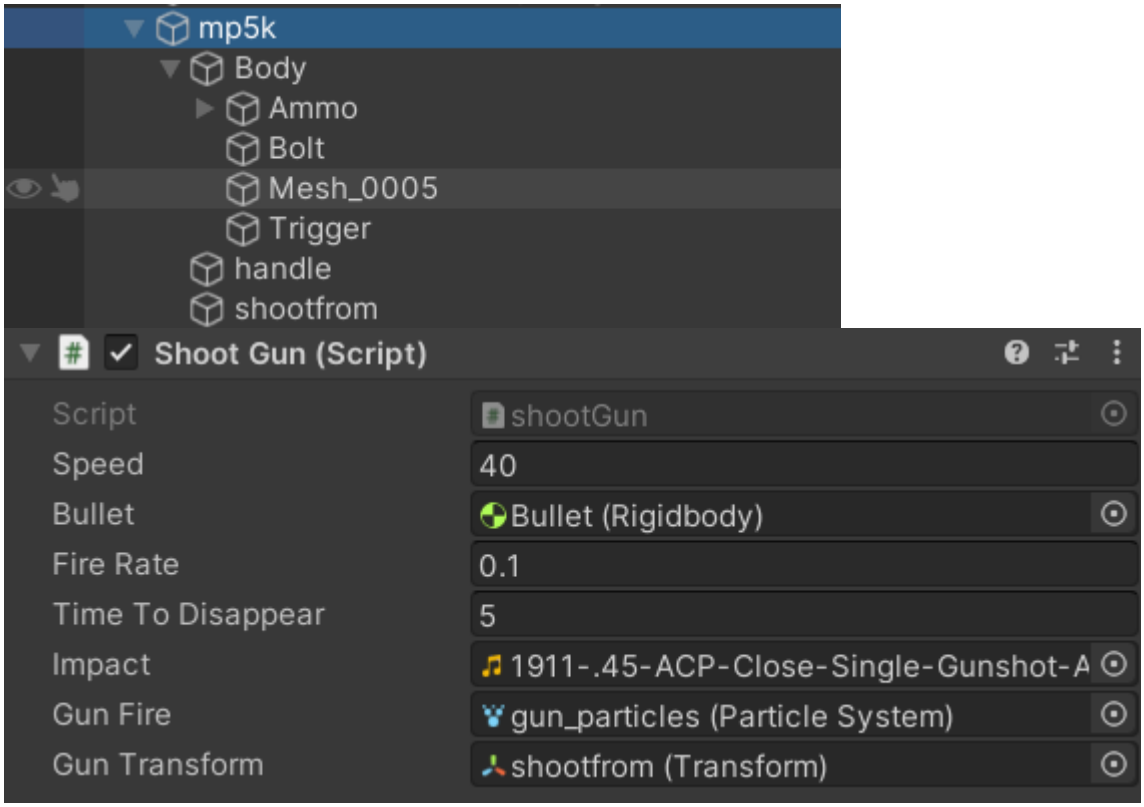
```
[System.Serializable]
3 references
public class VRMap
{
    public Transform vrTarget;
    public Transform rigTarget;
    public Vector3 trackingPositionOffset;
    public Vector3 trackingRotationOffset;

    3 references
    public void Map()
    {
        rigTarget.position = vrTarget.TransformPoint(trackingPositionOffset);
        rigTarget.rotation = vrTarget.rotation * Quaternion.Euler(trackingRotationOffset);
    }
}
```



Dieses Skript hat die Funktionalität, die Hände und den Kopf im Spiel an die eigentlichen «real-life» Positionen zu bringen. Im «VR Constraints» Object sind alle diese Positionen enthalten. «Hint» ist für die Ellbogen, «Target» für die Hände und «Head Constraint» für den Kopf zuständig.

Shootgun.cs

```
public void startShooting()
{
    while (OVRInput.Get(OVRInput.Axis1D.PrimaryIndexTrigger) > 0 || OVRInput.Get(OVRInput.Axis1D.SecondaryIndexTrigger) > 0)
    {
        Debug.Log("In shoot");
        timer += Time.deltaTime;
        if (timer > fireRate)
        {
            Debug.Log("SHOOOT");
            fire();
            timer = 0;
            audioSource.PlayOneShot(impact, 0.7f);
            gunFire.Play();
            gunFire.enableEmission = true;
        }
    }

}
public void fire()
{
    Rigidbody instantiatedProjectile = Instantiate(
        bullet, gunTransform.position, gunTransform.rotation
    ) as Rigidbody;

    instantiatedProjectile.velocity = gunTransform.TransformDirection(
        new Vector3(0, 0, speed)
    );
}
```

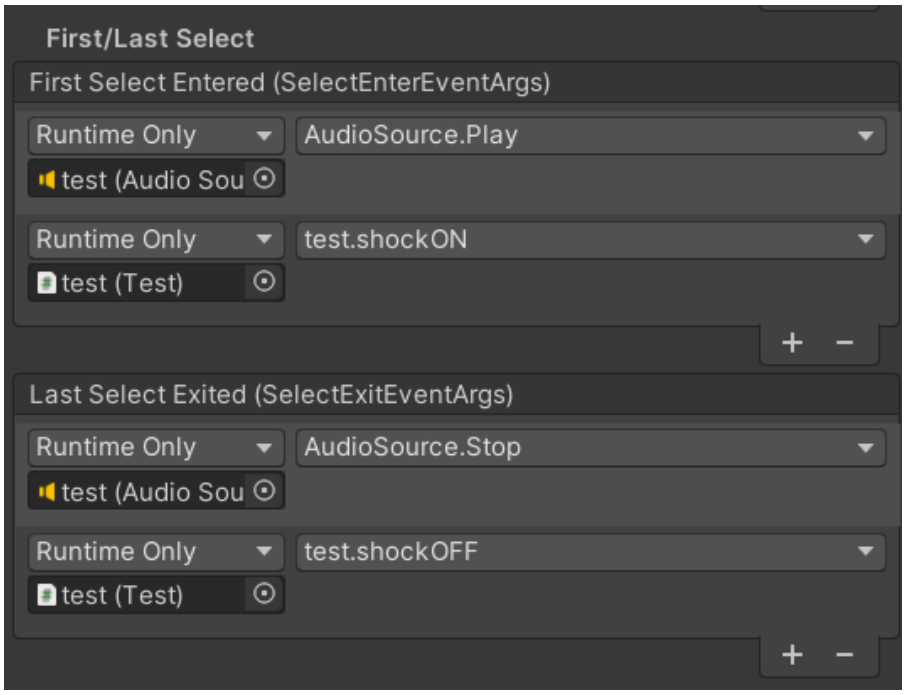


Dieses Skript ist für die Funktionalität der Waffe zuständig. Es wird als 1. Erkannt, ob ein «button-input» erfolgt. Falls dies zutrifft, wird der Timer gestartet und es wird immer geschossen sobald der «Timer» > als die

«fireRate» ist. Außerdem gibt es auch noch ein «Particle-System» und einen «Shoot-Sound».

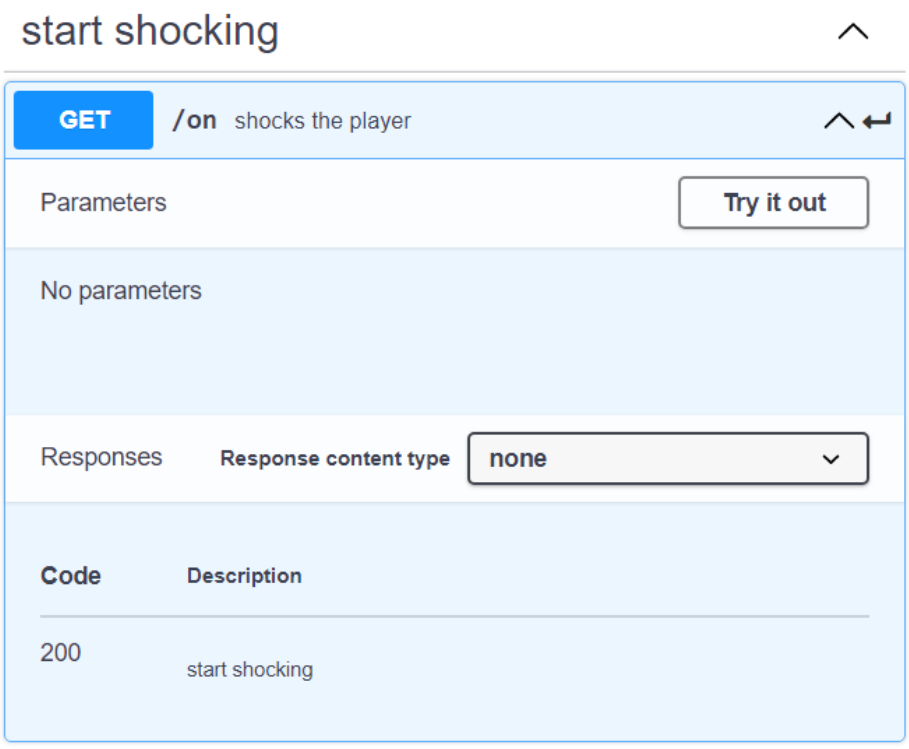
Shock.cs

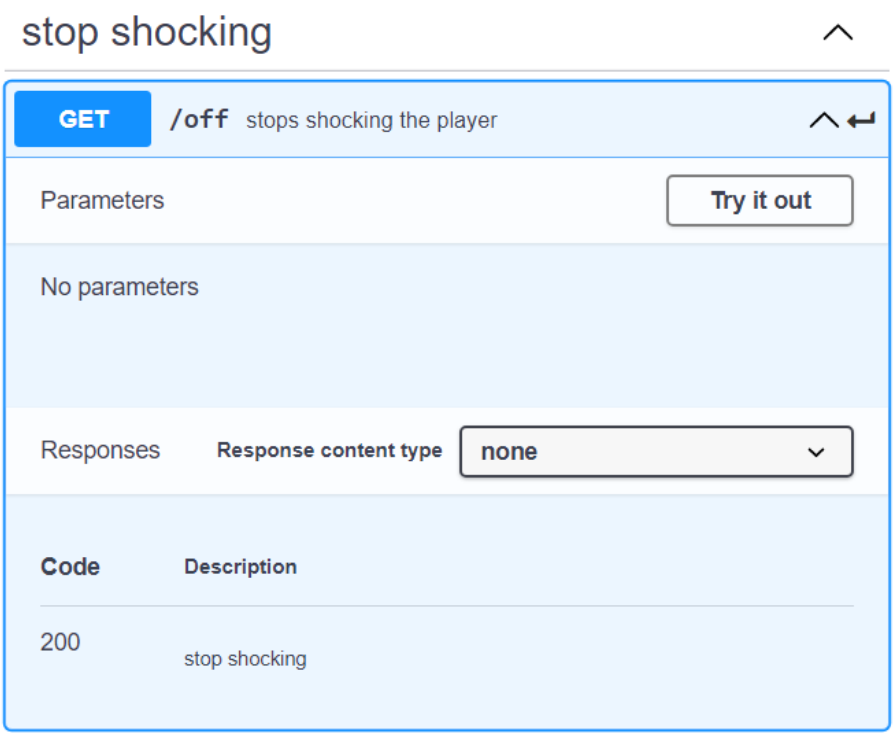
```
//private const string URL = [redacted];
private const string URL = [redacted];
public GameObject lightning;

Unity Message | 0 references
private void Start()
{
    WWW request = new WWW(URL + "/off");
}

0 references
public void shockON()
{
    Debug.Log("bzzzzz");
    WWW request = new WWW(URL + "/on");
    lightning.SetActive(true);
}

0 references
public void shockOFF()
{
    Debug.Log("no bzzzzz");
    WWW request = new WWW(URL + "/off");
    lightning.SetActive(false);
}
```



Dieses Skript verwaltet die Verbindung zur API. Hier wird eine Kugel als «Test-Object» verwendet. Sobald diese aufgehoben wird, wird eine «Request» zum Webserver gesendet welcher dann in der Bauchtasche die Verbindung vom Relais zum Schock-Gerät herstellt. Somit wird der Benutzer geschockt.

3.3 API






```
// Load Wi-Fi library
#include <WiFi.h>
```

Als Bibliothek wurde nur die „WiFi.h“ verwendet

```
// Set web server port number to 80
WiFiServer server(80);
```

Aufsetzen des Webservers

```
void setup() {
  Serial.begin(9600); // 115200
  // Initialize the output variables as outputs
  pinMode(in, OUTPUT);
  //pinMode(in, HIGH);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

Seriellen Output beginnen und pinMode initialisieren als Output. Dann wird die Verbindung zum WLAN hergestellt und danach wird gewartet bis das Programm verbunden ist. Zuletzt wird die IP Adresse ausgegeben und der Server wird gestartet.

```
void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected() {  // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
```

Hier wird überprüft ob ein Client sich verbinden möchte. Danach wird geprüft, ob Bytes vom Client kommen die der Server einliest. Dann wird der Repsonse-Code angegeben und der content-type gesetzt.

```
// turns the GPIOs on and off
if (header.indexOf("GET /on") >= 0) {
  Serial.println("GPIO on");
  digitalWrite(in, LOW);
} else if (header.indexOf("GET /off") >= 0) {
  Serial.println("GPIO off");
  digitalWrite(in, HIGH);
}
break;
```

Hier werden die Endpoints definiert und es wird der GPIO-Pin auf LOW gesetzt beim einschalten und er wird auf HIGH gesetzt beim ausschalten.

3.4 Verwendete Technologien

Unity Game Engine:

Eine Engine, die es ermöglicht mit Leichtigkeit Spiele selbst zu programmieren und zu designen. Die verwendete Programmiersprache ist C#. Alle Objekte haben Komponente wie z.B. Transform, welches die Koordinaten, Rotationen und Skalierungen des Objektes festlegt oder Rigidbody, welches das Objekt „physisch“ macht und es somit von z.B. Gravitation beeinflusst wird.

Visual Studio IDE:

Eine Programmierumgebung für diverse Programmiersprachen welche viele nützliche Funktionalitäten bietet und perfekt für Unity geeignet ist.

Arduino IDE:

Eine Programmierumgebung für das erstellen von Skripten, welche auf einem Arduino ausgeführt werden können. In dieser IDE musste außerdem noch ein gewisses Modul installiert werden, welches es ermöglicht den Code auf den „Arduino ESP32“ zu laden.

Arduino ESP32:

Ein Mikrocontroller welcher mehrere digitale Ein- und- Ausgänge besitzt. Dieser eignet sich sehr gut für kleinere Skripte oder zum Beispiel auch Webserver.

Relais:

Das SainSmart 2-CH 5V 2 Channel Relay wurde verwendet, um die Verbindung der TENS-Unit mit einer der Elektroden und auch somit den Stromfluss zu unterbrechen. Das Relais wird per GPIO am Arduino geschalten.

TENS-Unit:

Die Tens Machine Pain Unit wurde verwendet um den Schock zu ermöglichen. Dabei muss beachtet werden, dass die TENS Unit eingeschalten werden muss bevor man das Spiel ausprobieren will. Grundsätzlich kann man die Intensität, Dauer und Stärke der Stromschläge angeben.

Meta Quest 2:

Eine neuartige Entwicklung von dem damaligen Unternehmen „Oculus“ (jetzt hat Facebook/Meta sie aufgekauft). Diese Virtual Reality Brille bietet Komfort und ausgezeichneten Nutzen ohne jegliche externen Sensoren an. Verwendet wurde diese Brille bei der Entwicklung des Spiels um dieses immer wieder auszuprobieren.

Oculus Interaction SDK:

Diese neue Bibliothek ist Anfangs des Jahres 2022 erschienen und bietet voll funktionstüchtige VR-Hände (mit und ohne Controller). Diese SDK ist auch eine sehr gute Basis welche man für VR-Spiele verwenden kann.

4. Diskussion der Ergebnisse

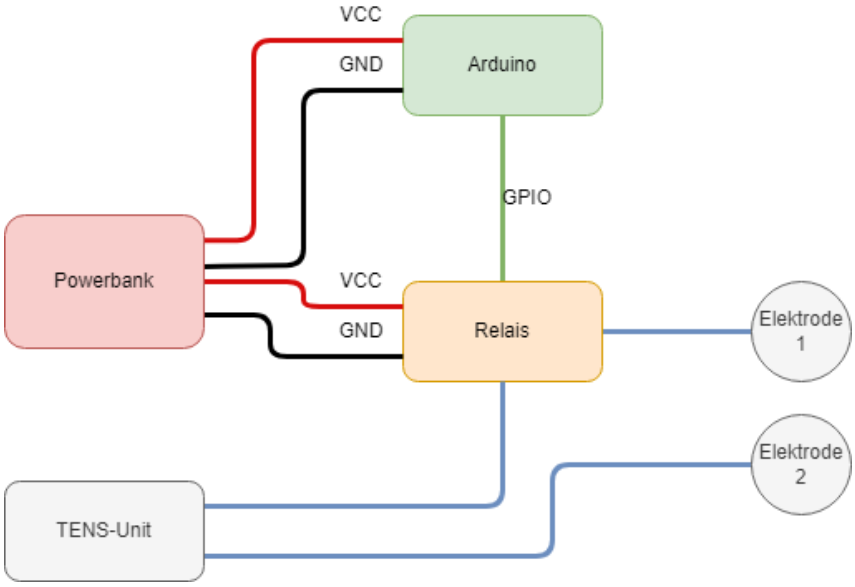
4.1 Theoretische Hintergründe

Schießen mit der Waffe:

```
public void fire()
{
    Rigidbody instantiatedProjectile = Instantiate(
        bullet, gunTransform.position, gunTransform.rotation
    ) as Rigidbody;

    instantiatedProjectile.velocity = gunTransform.TransformDirection(
        new Vector3(0, 0, speed)
    );
}
```

Hierbei wird neuer Rigidbody von dem „Bullet“ Prefab erstellt und die Geschwindigkeit von diesem wird dann modifiziert mit einem neuen Vektor wobei die „z“ Koordinate die Geschwindigkeit der Kugel ist.

Bauchtasche Elektronik:



Hier in diesem Diagramm ist der Stromfluss der „Schock-Bauchtasche“ dargestellt. Hierbei ist zu beachten dass das Relais und der Arduino eine externe Stromzufuhr benötigen. Dies wurde über das Aufteilen und Auseinanderschneiden eines USB Typ C Kabels erreicht. Vom Arduino zum Relais ist ebenfalls ein Kabel gelegt über welches dann das Kommando zum „schocken“ gegeben wird. Eines der beiden Kabel der TENS-Unit ist durchgeschnitten worden und in durch das Relais geführt. Also wenn ein Kommando vom Arduino zum Relais gelangt, wird dementsprechend der Stromfluss des Kabels der TENS-Unit geschlossen/abgebrochen.

4.2 Ausblick und Rückblick

Da die Zeit nicht ausreichte und außerdem eine neue Bibliothek zur Entwicklung von VR-Spielen in Unity erschienen ist, wurde der Fokus des

Spiels von einem Funktionierendem Online Modus auf die geforderte API versetzt. Diese war Anfangs als API zur Speicherung von Benutzerdaten gedacht. Da diese Speicherung nicht mehr nötig ist – weil der Online-Modus weggefallen ist bzw. nicht umzusetzen war in dem gegebenen Zeitraum – wurde als API ein Arduino verwendet auf dem ein Webserver läuft der die Bauchtasche «schocken» lässt.

Anfang Februar April ist die Oculus Interaction SDK erschienen. Diese hat den Grundaufbau und die Grundfunktionen des gesamten Projektes verändert bzw. hat die Softwareentwicklung an diesem Projekt vereinfacht. Aus diesem Grund wurde das alte Produkt «refactored» mit der neuen SDK.

Zukünftig sollten noch beim Programm Gegner und eine Karte mit höchster Priorität eingebaut werden. Dies wird sehr wahrscheinlich bis zum Präsentationstermin am 23.06.2022 möglich sein. Außerdem wäre noch ein «Player-Shop» angemessen für das Projekt wo der Spieler sich «Upgrades» und neue Waffen kaufen kann.