# Einleitung

Dieses Dokument beschreibt die Entwicklung eines VR-Schwertkampfspiels in Unity, inspiriert von einem Wii-Spiel. Das Ziel des Projekts ist es, eine besonders immersive VR-Erfahrung durch realistische Schwertphysik und Gegner zu schaffen.

## Projektbeschreibung

Das VR-Schwertkampfspiel soll das Gefühl eines echten Schwertkampfes vermitteln. Durch besondere Schwertphysik und realistische Gegner, die den Spieler herausfordern, wird ein immersives Erlebnis angestrebt.

## Zielsetzung

Ziel des Projekts ist es, ein funktionsfähiges und unterhaltsames VR-Schwertkampfspiel zu entwickeln, das sich durch seine Realitätsnähe und spannende Spielmechaniken auszeichnet.

# Ablauf des Projektes

## 1. Planung und Findungsphase

1.1 Erarbeitung des Konzeptes: Die Idee, ein Wii-Spiel in VR nachzugestalten, wurde entwickelt. Ziel war es, besondere Schwertphysik und möglichst realistische Gegner und Karten zu integrieren. Zudem sollte eine automatische Bewegung wie im Wii-Spiel implementiert werden.

1.2 Überlegungen zu Features: Es wurden verschiedene Features wie kleinere Erfolgserlebnisse, ein generelles Kampfsystem und Bossgegner diskutiert. Gegner können durch das Zeigen mit dem Schwert herausgefordert werden, ansonsten fordern sie den Spieler selbst heraus.

1.3 Findung von Assets: Geeignete Assets wurden gesucht und bewertet.

1.4 Besprechungen: Wöchentliche Besprechungen und Anpassungen des Projekts wurden durchgeführt.

1.5 Testphasen wurden wöchentlich durchgeführt

## 2. Start der Projektentwicklung

2.1 Generell: Aufgrund von häufigen Fehlern wurde in zwei unterschiedlichen Projekten gearbeitet.

2.2 Erstellung von zwei Karten: Wald und Stadt.

2.3 Erstellung einer Tutorial-Map: Eine spezielle Karte für das Tutorial wurde erstellt.

2.4 Erste Versuche: Die Schwertphysik wurde erstmals implementiert.

2.5 Hinzufügen von Gegnermodellen und Animationen: Erste Gegner wurden erstellt und animiert.

2.6 Hauptmenü: Ein Hauptmenü wurde hinzugefügt.

2.7 Gegnertypen: Drei verschiedene Gegnertypen wurden hinzugefügt.

2.8 Schwertphysik und Brustpanzer: Die Schwertphysik wurde verfeinert und ein Brustpanzer, der dem Spieler folgt, hinzugefügt.

2.9 Weitere Menüs und Musik: Weitere Menüs und Musik wurden hinzugefügt.

2.10 Automatische Bewegung: Das automatische Bewegen in VR wurde getestet und implementiert.

2.11 Zusammenfügen der Projekte: Die beiden Projekte wurden zusammengeführt.

2.12 Testen des Stoppen-Systems: Erste Tests des Systems, das das Stoppen bei Eintritt in den Kampfmodus ermöglicht, wurden durchgeführt.

2.13 Lebensleiste und Schadenssystem: Eine Lebensleiste und ein einfaches Schadenssystem wurden hinzugefügt.

2.14 Schwert-Grab-Mechanik: Eine Mechanik zum Greifen des Schwertes wurde hinzugefügt.

2.15 Änderungen am Hauptmenü: Das Design und der Stil der Buttons im Hauptmenü wurden geändert.

2.16 Erste Tests der APK: Die ersten Tests der APK wurden durchgeführt.

2.17 Test der Schwertphysik 2: Die Schwertphysik wurde erneut getestet und verbessert.

2.18 Hitsystem: Die erste Implementierung des Hitsystems wurde durchgeführt.

2.19 Herausforderungssystem: Ein System zum Herausfordern von Gegnern wurde hinzugefügt.

2.20 Bossgegner: Ein Bossgegner mit Nah- und Fernkampffähigkeiten wurde hinzugefügt.

2.21 Kampfverhalten von Gegnern: Das Verhalten der Gegner im Kampf wurde optimiert.

2.22 Kompletter Merge: Beide Projekte wurden vollständig zusammengeführt.

2.23 Statesystem: Ein Statesystem wurde hinzugefügt.

2.24 Pfeilmechanik: Der erste Versuch einer Pfeilmechanik wurde durchgeführt.

2.25 Hitboxen: Hitboxen für Gegner, Spieler und Schwerter wurden hinzugefügt.

2.26 Voll funktionsfähiges Combat- und Challenge-System: Das Kampf- und Herausforderungssystem wurde vollständig implementiert.

2.27 Spielerleben und Schaden: Das System für Spielerleben und Schaden wurde hinzugefügt.

2.28 Spielerhände werden rot bei Schaden: Die Hände des Spielers werden bei Schaden rot.

2.29 Gegnerverhalten optimiert: Das Verhalten der Gegner wurde weiter optimiert.

2.30 Vignette bei Schaden: Eine Vignette-Effekt bei Schaden wurde hinzugefügt.

2.31 Enemy Knockback: Ein Knockback-Effekt für Gegner wurde hinzugefügt.

2.32 Änderung des Gegner-Hit-Systems: Das Treffer-System der Gegner wurde auf drei Hitboxen umgestellt.

2.33 Anpassung der Ringrotation: Die Rotation der Ringe wurde angepasst.

2.34 Soundeffekte: Soundeffekte für Aktionen wurden hinzugefügt.

2.35 Änderung des Pfeils: Der Pfeil wurde auf einen VFX-Effekt umgestellt.

2.36 Haptisches Feedback: Haptisches Feedback wurde hinzugefügt.

2.37 Knockback verbessert: Der Knockback-Effekt wurde verbessert.

2.38 Schwarze Todesvignette: Eine schwarze Todesvignette wurde hinzugefügt.

2.39 Änderungen an der Vignette: Weitere Anpassungen an der Vignette wurden durchgeführt.

2.40 Tutorialgegner: Ein Tutorialgegner wurde hinzugefügt.

2.41 Besserer Blocking-State: Der Blocking-State wurde verbessert.

2.42 Änderung des Hit-Sounds: Der Hit-Sound wurde geändert und neue Musik hinzugefügt.

2.43 Implementierung des Tutorialgegners: Der Tutorialgegner wurde implementiert.

2.44 Layered Animationen: Layered Animationen wurden hinzugefügt.

2.45 Game Over Screen: Ein Game Over Screen wurde hinzugefügt.

2.46 Strahl mit Fade-Effekt: Ein Strahl mit Hinein- und Hinausfaden wurde hinzugefügt.

2.47 Gegner herausfordern: Die Logik, dass Gegner den Spieler selbst herausfordern, wurde implementiert.

2.48 Boss Gegner: Verhalten und Animationen sind fertig implementiert und aufeinander abgestimmt.

# Spielmechanik

Die Spielmechanik des VR-Schwertkampfspiels umfasst eine Reihe von Funktionen und Effekten, die das Spielerlebnis realistischer und immersiver gestalten.

- Schwertphysik: Die Schwertphysik wurde sorgfältig entwickelt, um realistische Bewegungen und Treffer zu simulieren.

- Automatische Bewegung: Wie im Wii-Spiel bewegt sich der Spieler automatisch, um die Immersion zu erhöhen.

- Herausforderungssystem: Gegner können durch das Zeigen mit dem Schwert herausgefordert werden oder sie fordern den Spieler selbst heraus.

- Lebensleiste und Schadenssystem: Eine Lebensleiste zeigt die Gesundheit des Spielers an, und ein Schadenssystem regelt die Auswirkungen von Treffern.

- Schwert-Grab-Mechanik: Der Spieler kann das Schwert greifen und schwingen, was zu einer realistischeren Erfahrung beiträgt.

- Kampfverhalten und Bossgegner: Gegner haben ein komplexes Kampfverhalten, und es gibt Bossgegner mit speziellen Fähigkeiten.

- Haptisches Feedback und Soundeffekte: Diese Effekte verstärken die Immersion und das Spielerlebnis.

- Vignette-Effekt und Knockback: Diese Effekte wurden hinzugefügt, um visuelles Feedback und ein besseres Gefühl für Treffer zu geben.

# Systeme und Spielmechnaniken

Um die Struktur des Projekts klar darzustellen, wird ein UML-Diagramm erstellt, das die wichtigsten Komponenten und deren Beziehungen zeigt.

Dateien und Mechaniken im Detail: Jeder hat Platz, seine Dateien und Mechaniken zu erklären und detailliert zu beschreiben.

A grey folder with white text

Description automatically generated

## Schwertphysik (David Blabl)

A white folder with a black background

Description automatically generated

Die Schwertphysik des Spiels ist von Evan Fletchers Ansatz abgeleitet. Die Idee dabei ist, dass sich das Schwert nicht aus der Hand des Spielers bewegen Soll, aber das Schwert sich an Objekten entlagen zurückbiegen lässt, wie im folgenden Bild zu sehen ist:

A black and white drawing of a sword and a circle

Description automatically generated

Die Physik soll einen möglichst Immersiven Kampf simulieren und bekannte Probleme, wie das einfach Durchschlagen von Objekten oder Gegnern, lösen.

Die Umsetzung ist über einen ConfigurableJoint in Unity gelöst. Dieser bietet die Möglichkeit auf horizontaler und vertikaler Ebene wie eine Art Feder zu agieren. Der ConfigurableJoint wird in Fletchers Ansatz als Component an ein Wrist-Objekt angeheftet und das Schwert dann als Kind Objekt an das Wrist-Objekt angefügt. Diese Implementation ist notwendig, da das normale XR Grabbable Object sämtliche Physik überschreibt und somit unbrauchbar ist.

Dies bewirkt allerdings, dass sich das Schwert auch innerhalb der Hand des Spielers bewegt und ist dadurch unrealistisch. Deshalb wurde das Konzept überarbeitet, indem das Schwert nun den ConfigurableJoint enthält und nicht mehr das Wrist-Objekt.

Das hat den Effekt, dass sich das Schwert fest in der Hand des Spielers befindet und sich lediglich Rotieren kann. In den folgenden Bildern lässt sich die Konfiguration des ConfigurableJoints erschließen:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Die X,Y und Z Motion muss beim veränderten Ansatz auf „Locked“ geändert werden um sicher zu stellen, dass sich das Schwert nicht bewegen kann.

Die Spring und Damper Werte wurden auch durch viele Tests angepasst um ein angenehmeres Ergebnis zu erzielen. Die genaue Implementierung des Ansatzes lässt sich in der Datei „SwordPos.cs“ in der Funktion „CreateConfigurableJoint()“ nachlesen.

## Brustpanzer des Spielers (David Blabl)

Da der Spieler bisher nur aus Händen und Schwert besteht, soll ein Brustpanzer unterhalb der Kamera für eine Bessere Identifizierung mit dem SpielerObjekt sorgen.

Der Brustpanzer wird über ein Skript am Spieler befestigt und übernimmt die Rotation an der Y-Achse der Kamera. Dadurch wird dem Spieler suggeriert, dass er einen Körper im Spiel hat. Dieser ist auch animiert und simuliert leichte Bewegungen des Körpers.

A screenshot of a video game

Description automatically generated

## Greifmechanik für das Schwert (David Blabl)

A white folder with a black background

Description automatically generated

Um eine realistischere Handhabung des Schwertes zu erzielen, soll es möglich sein, dass das Schwert hinter dem Spieler, auf seinem Rücken greifbar ist und danach gehalten werden muss. Lässt der Spieler das Schwert los, so kehrt es auf seinen Rücken zurück und muss es erneut greifen.

Die Implementierung dieser Mechanik ist durch einen CapsuleCollider am Griff des Schwertes gelöst und einen am Controller des Spielers. Das Schwert erhält über das Skript in der Update() Funktion die Position und Rotation des Spielers. Wenn der Spieler seinen Controller innerhalb des Colliders des Schwertes bewegt und die Greiftaste gedrückt hält, wird das Schwert als Kindobjekt an das Wrist-Objekt angeheftet.

Um Interaktionen zwischen Schwert und Wrist vor dem Greifen des Schwertes zu vermeiden, ist es nicht möglich, dass der ConfigurableJoint des Schwertes aktiv ist, während das Schwert sich auf dem Rücken befindet. Dieser kann aber nicht inaktiv oder aktiv gesetzt werden und muss somit bei jedem Greifen neu über das Skript instanziert werden.



## Herausforderungssystem (David Blabl)

A white folder with a black background

Description automatically generated

Bevor ein Gegner bekämpft werden kann, soll der Spieler die Möglichkeit haben in herauszufordern, um ihm Kontrolle über das Spiel zu geben. Sollte sich der Spieler dazu entscheiden, keinen Gegner herauszufordern sollen die Gegner ihn herausfordern.

Es hat sich durch Tests herausgestellt, dass es einen Sinnvolle Mechnik wäre, das Schwert auf den Gegner zu richten um ihn herauszufordern und diese wurde folgendermaßen implementiert: Durch betätigen der Trigger Taste auf dem Controller wird ein Raycast von der Schwertspitze aus nach vorne instanziert und über eine Line farblich gekennzeichnet.

Der Raycast und die Line sind beide auf eine bestimmte Reichweite begrenzt, sodass der Spieler keine Gegner mit zu großer Distanz zum Spieler herausfordern kann. Zudem hat die Line einen „Fade Effekt“ um sie ansprechender für den Spieler zu gestalten.   
Für den Raycast wurden die Gegner mit einem gesonderten Collider ausgestattet, der die Größe und den Durchmesser der Gegner deutlich überschreitet, um es dem Spieler zu erleichtern, den Gegner mit dem Raycast zu treffen, während er sich bewegt.

Wenn der Raycast auf einen Gegner trifft, wird ein Canvas mit einem Ring überhalb des Controllers des Spielers instanziert und der Ring füllt sich innerhalb eines Zeitintervalls. Sobald der Ring vollständig ausgefüllt ist, stoppt der Spieler und der Gegner bewegt sich auf den Spieler zu.

A screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generated

Fordert der Spieler keinen der Gegner in seiner Umgebung heraus, wird ein SphereCollider erzeugt, der an den Spieler über ein GameObject „ChallengeManager angeheftet ist. Der ChallengeManager verwaltet über eine Queue, welcher Gegner zuerst den Spieler herausfordern sollte. Der Gegner, der zuerst den Collider betreten hat, fordert den Spieler heraus, um zu vermeiden, dass Gegner verbleiben ohne bekämpft worden zu sein.

## Kampfsystem (David Blabl)

A white folder with a black background

Description automatically generated

Das Kampfsystem wird grundlegend über ein State-System realisiert. Der Wahrheitswert inCombat wird sowohl beim Spieler, als auch bei den Gegnern über einen StateController entschieden. Der StateController des Spielers beinhaltet ebenfalls eine Variable, die den derzeitigen Gegner referenziert.

Über die StateController der Gegner beziehen die Skripten „CombatSystem.cs“ und „PlayerDamage.cs“ Informationen, die notwendig für ihre Funktionsweise sind. Das CombatSystem Skript wird an das Schwert des Spielers angeheftet um dort die Logik für das hinzufügen von Schaden an Gegnern zu handhaben.

Die Wahrheitswerte „isAttacking“ und „isBlocking“ entscheiden darüber, wann ein Gegner getroffen werden kann, sowie der Abstand des Schwertes des Spielers vom Gegner seit dem letzten Treffer. Dieser Abstand und eine Abklingzeit seit dem letzten Treffer, soll verhindern, dass der Spieler dem Gegner mehr als ein Mal pro Treffer Schaden hinzufügen kann.

### Block Mechanik des Gegners

Hat ein Gegner den Status „Blocken“, so kann der Spieler ihn nur noch aus einer Richtung treffen. Die Richtung, aus der er schlagen kann wird zufällig bestimmt und drei verschiedene Collider werden entweder auf enabled oder disabled gesetzt, je nach dem welche Richtung bestimmt wurde. Sobald die Richtung bestimmt wurde, wird sie durch einen visuellen Effekt, der sich durch den Gegner hindurchbewegt, angezeigt.

A cartoon character wearing a garment

Description automatically generatedA screenshot of a video game

Description automatically generatedA cartoon of a person holding a sword

Description automatically generated

### Vignetten Effekt Schaden

### Haptisches Feedback

# Fehler im Projekt und Positive Erfahrungen

Im Verlauf des Projekts traten verschiedene Fehler auf, die jedoch zur Verbesserung der Endqualität beitrugen. Positive Erfahrungen umfassen die erfolgreiche Implementierung komplexer Spielmechaniken und das Erreichen eines hohen Grades an Immersion.

# Ausblick auf die Zukunft und Verbesserungen

In Zukunft könnten weitere Verbesserungen vorgenommen werden, wie z.B. die Optimierung der Grafik, die Hinzufügung weiterer Gegnertypen und Spielmodi sowie die Verbesserung der Benutzerfreundlichkeit und Performance.

# Fazit

Das Projekt war ein großer Erfolg und hat gezeigt, dass es möglich ist, ein immersives und realistisches VR-Schwertkampfspiel zu entwickeln. Die Erfahrungen und Erkenntnisse aus diesem Projekt werden zukünftige Entwicklungen im Bereich der VR-Spiele beeinflussen.