UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Entwicklung eines interaktiven Action-Rollenspiels mit Unity

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von

Vanessa Karolek

Erstgutachter:	Prof. Dr. Stefan Müller Institut für Computervisualistik, Leiter der Arbeitsgruppe Müller
Zweitgutachter:	Bastian Krayer, M.Sc. Institut für Computervisualistik, Arbeitsgruppe Müller

Koblenz, im Oktober 2019

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Diese Arbeit behandelt die Konzeption und Implementation eines Action-Rollenspiels mithilfe der Game Engine Unity. Im Rahmen einer Evaluation sollte das Spiel hinsichtlich der Bedienbarkeit der integrierten Steuerungsarten, der visuellen Überzeugung der Animationen und der Benutzerfreundlichkeit über zur Verfügung gestellte Hilfsmittel und Visualisierungen bewertet werden. Zusätzlich sollten Schwachstellen und Probleme im Spiel über offenes Feedback herausgefunden werden. Die Auswertung der Evaluation ergab, dass das Spiel im Hinblick auf die Bedienbarkeit und Benutzerfreundlichkeit zwar noch ausbaufähig ist, aber insgesamt einen guten Eindruck bei den Probanden hinterlassen hat.

Abstract

This thesis deals with the conception and implementation of an action role-playing game using the game engine Unity. Within the context of an evaluation, the game was supposed to be evaluated with regard to the usability of the integrated control modes, the visual conviction of the animations and the user-friendliness of the tools and visualizations provided. In addition, weaknesses and problems in the game were to be identified through open feedback. The results of the evaluation showed that the game is still expandable in terms of usability and user-friendliness, but has left a good impression on the test persons.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit sind Videospiele immer mehr Teil des menschlichen Medienalltags geworden. Der Jahresreport der deutschen Games-Branche 2019 [10] berichtet, dass im Jahr 2018 etwa 34,3 Million Deutsche mindestens gelegentlich im Kontakt mit Videospielen standen. Sicherlich hat sich der eine oder andere Spieler schon einmal die Frage gestellt, wie ein Videospiel eigentlich entwickelt wird. Zur Beantwortung dieser Frage stehen frei zugängliche *Game Engines* wie zum Beispiel Unity zur Verfügung. Sie ermöglichen die Umsetzung beliebiger Spielideen. Sowohl für Interessierte als auch für Studierende des Fachbereichs Informatik bietet das Arbeiten mit einer solchen Engine eine sehr gute Gelegenheit an, seine Programmiererfahrungen und sein Wissen in der Computergrafik im Rahmen einer Spielentwicklung zu erweitern.

1.1 Motivation

In meiner Freizeit verbringe ich auch gerne etwas Zeit mit Videospielen. Mein persönlicher Favorit im Hinblick auf die Spiel-Genres stellt das Action-Rollenspiel dar. Besonders für diese Arbeit inspiriert hat mich die von Square Enix entwickelte Spielreihe *Kingdom Hearts* [8]. Mir persönlich macht es sehr viel Spaß, in ein virtuelles Abenteuer mit herausfordernden Echtzeit-Kämpfen einzutauchen, in denen der Spieler selbst direkte Kontrolle jeglichen Kampfverhaltens der Spielfigur besitzt. Für jeden Moment muss als Spieler selbst entschieden werden, welche Aktion für das aktuelle Kampfgeschehen am sinnvollsten ist, um siegreich hervorzugehen. Eine ungünstig ausgewählte Aktion, könnte den Kampf fehlschlagen lassen. Zum Bestreiten der Kämpfe werden spontanes, strategisches Denken und schnelles Reaktionsvermögen zur Umsetzung der Strategie benötigt. Für den Spieler kann somit der eine oder andere Kampf zu einer großen Herausforderung werden. Da die Geschichte eines Action-Rollenspiels im Vergleich zu einem klassischen Rollenspiel meistens nicht sehr vertieft ist, bietet es eine sehr gute Gelegenheit an, mal selbst eine Art Action-Rollenspiel zu entwickeln und den Schwerpunkt dabei mehr auf die Spielmechaniken zu legen.

1.2 Zielsetzungen

In dieser Arbeit wird die Umsetzung eines Action-Rollenspiels mit der Game Engine Unity angestrebt. Zunächst soll über verfügbare Tutorials und dem Unity Manual [9] ein guter Einstieg in Unity gefunden werden. Anschließend soll ein Konzept für das Spiel erstellt werden, mit dessen Hilfe ein erster funktionsfähiger Prototyp implementiert werden kann. Das zu implementierende Action-Rollenspiel soll über ein attraktives Kampfsystem verfügen, ergänzt mit erzählerischem Inhalt im Sinne einer aufgabenbasierten Storyline, um einen Leitfaden im Spiel zu schaffen. Das Ziel des Spiels umfasst demnach die Erfüllung aller verfügbaren Aufgaben.

Da ich die Umsetzung eines gut optimierten Kampfsystems sehr zeitaufwändig

einschätze, soll möglichst auf zur Verfügung stehende Mittel, wie beispielsweise vorgefertigte, öffentlich zugängliche Animationen und Modelle, zurückgegriffen werden.

Nach der Implementierungsphase soll das Spiel im Rahmen einer Nutzungsstudie getestet werden. Bei bekannten Action-Rollenspielen wie *Diablo III* [2] oder *Kingdom Hearts* [8] trifft man oft auch auf verschiedene Steuerungsarten. Während Kingdom Hearts über eine eher direkte Steuerung verfügt, bei der der Spieler volle Kontrolle über die Bewegung und Ausführung der Aktionen der Spielfigur besitzt, beinhaltet Diablo III eine eher indirekte Steuerung, die sogenannte „Point To Click"- bzw. „Point 'n' Click"-Steuerung, welche primär über den Mauscursor und nur wenigen zusätzlichen Tasten der Tastatur gesteuert wird. Es ist schwierig zu entscheiden, welche der beiden Steuerungsarten sich für ein Spiel besser eignet. Aus diesem Grund sollen im Spiel sowohl eine direkte als auch eine indirekte Steuerungsart umgesetzt werden, die zum Schluss in der Nutzungsstudie gegenübergestellt werden sollen, um Auskunft darüber geben zu können, welche Steuerungsart sich für das Spiel besser eignet.

Damit die Probanden die Spielsteuerung effektiv erlernen können, möchte ich den Probanden eine Einführung in die Spielsteuerung im Spiel selbst zur Verfügung stellen. Im Rahmen der Nutzungsstudie soll unter anderem untersucht werden, ob das Spiel im Hinblick auf die zur Verfügung gestellten Hilfsmittel und Visualisierungen benutzerfreundlich genug ist.

Des Weiteren sollen für alle verfügbare Aktionen optisch passende Animationen herausgesucht werden. Diese sollen ebenfalls im Rahmen der Studie evaluiert werden.

2 Grundlagen

Dieses Kapitel soll zum besseren Verständnis nachfolgender Kapitel dienen. Zunächst ist es notwendig, in die Thematik des Genre-Konzeptes einzutauchen, um anhand dessen das Genre „Action-Rollenspiel" zu erklären und später darauf aufbauend in Kapitel 3 ein Konzept zu erstellen. Anschließend werden wichtige Features der Game Engine Unity erläutert, die bei der Umsetzung des Spiels von starker Bedeutung sein werden.

2.1 Genre als Klassifizierer von Videospielen

Heutzutage befinden sich unzählige Videospiele unterschiedlichster Art auf dem Markt. Die Auswahl reicht von meist eher simpel gehaltenen Gelegenheitsspielen bis hin zu umfangreicheren Spielen, bei denen zum Beispiel viel Action oder eine tiefgründige Geschichte im Vordergrund steht. Um ähnliche Videospiele zusammenzufassen und von Anderen abzutrennen, wurde das Genre-Konzept eingeführt. Ähnlich wie bei Filmen oder Büchern wurden auch Videospiele anhand ihrer Eigenschaften verschiedenen Genres zugeordnet. Nach Rachel I. Clarke et al. [4] sei

dieses zwar noch nicht ausgereift genug, aber Ernest Adams [1, Seite 390] ist der Meinung, dass es sich als ersten Anhaltspunkt für mögliche Unterhaltungen und Suchanfragen nach Spielen ähnlicher Art gut eigne.

Im Folgenden soll mithilfe des Genre-Konzepts und existierender Vertreter der Genres erläutert werden, wie man sich ein Action-Rollenspiel im Allgemeinen vorzustellen hat. Da es sich dabei um die Vermischung der zwei Genres Action und Rollenspiel handelt, ist es notwendig, diese zunächst zu beschreiben.

2.1.1 Action

> An action game is one in which the majority of challenges presented are tests of the player's physical skills and coordination. Puzzle-solving, tactical conflict, and exploration challenges are often present as well. [1, Seite 392]

Actionspiele stellen den Spieler vor eine Reihe von Herausforderungen, die seine physischen und koordinierenden Fähigkeiten prüfen. Typischer Gegenstand der Prüfungen ist die Schnelligkeit, Reaktionszeit, Steuerungsfähigkeit, Genauigkeit beim Zielen, zeit- oder rhythmenbasiertes Betätigen von Inputs oder das Ausführen komplizierter Tastenfolgen. Des Weiteren können auch die Erkundung der Spielwelt, das Lösen von Puzzles auf Zeitdruck oder das Erkennen von Verhaltensmustern vorkommender Spielfiguren Bestandteil der Prüfungen sein.

Videospiele des Action-Genres sind oft sehr einfach gehalten. Sie besitzen klar definierte Ziele und Wege, wie diese Ziele erreicht werden können. Diese Wege sind jedoch mit zu absolvierenden Herausforderungen versehen. Das Spiel gilt als vollendet, sobald all diese Ziele und alle damit hergehenden Herausforderungen gemeistert wurden. [1, Seite 392 f.]

Einen sehr erfolgreichen Klassiker stellt das „Jump-'n'-Run"-Spiel *Super Mario Bros.* [6] von 1985 für das Nintendo Entertainment System (NES) dar.

Abbildung 1: *Ein Screenshot eines Levels aus dem Spiel Super Mario Bros. [6]*

Das Spiel enthält eine Reihe aufbereiteter Spielabschnitte, sogenannte „Level“, die es innerhalb eines Zeitlimits zu absolvieren gilt. Abbildung 1 zeigt einen Ausschnitt eines dieser Level. Der Spieler beginnt in jedem Level bei einer Anfangsposition und muss die Spielfigur mithilfe der Spielsteuerung zu einer Zielposition führen. Währenddessen wird der Spieler jedoch mit gegnerischen Figuren und Hindernissen konfrontiert, die zum Scheitern des Levels führen können. Der Spieler muss dementsprechend seine Schnelligkeit und sein Reaktionsvermögen unter Beweis stellen, um das Level zu meistern. Zusätzlich muss der Spieler zusehen, dass er seine Anzahl zur Verfügung stehender Leben für die Spielfigur nicht auf null herabfallen lässt. Ansonsten gilt das Spiel als verloren und das Spiel beginnt wieder von Neuem. Jedes Scheitern eines Levels führt zum Abzug eines Lebens.

2.1.2 Rollenspiel

> A role-playing game is one of in which the player controls one or more characters, typically designed by the player, and guides them through a series of quests managed by the computer. Victory consists of completing these quests. Character growth in power and abilities is a key feature of the genre. Typical challenges include tactical combat, logistics, economic growth, exploration, and puzzle solving. Physical coordination challenges are rare except in RPG-action hybrids. [1, Seite 455]

Jedes Rollenspiel erzählt eine Geschichte. Das Ziel eines solchen Spiels ist es folglich, diese Geschichte zu vollenden. In Abbildung 2 wird graphisch dargestellt, wie der Ablauf eines Rollenspiels typischerweise aussieht. Bestandteil dieser Geschichte ist demzufolge das Erledigen einzelner Aufgaben, sogenannte „Quests“. Jeder Abschluss einer Quest lässt die Geschichte immer weiter voranschreiten. Die Aufträge sind meistens mit Interaktionen mit sogenannten NPCs[1], dem Bekämpfen gegnerischer Spielfiguren und dem Aufsuchen verschiedener Orte der Spielwelt verbunden. In einem finalen Gebiet findet die Geschichte meistens ihren Abschluss. Ein wichtiger Bestandteil von Rollenspielen ist die Weiterentwicklung spielbarer Charaktere. Der Anstieg verschiedener numerischer Werte regeln, wie sich ein Charakter im Laufe des Spiels weiterentwickelt. Diese werden oft „Statuswerte“ genannt und entscheiden letztlich darüber, wie stark ein Charakter ist. Das Ausmaß dieser Werte äußert sich vor allem in den Kämpfen. [1, Seite 455 ff.]

Typische Vertreter dieses Genres stellt die japanische Rollenspiel-Reihe *Final Fantasy* dar. Jedem dieser Spiele ist es gemeinsam, dass die Spielwelt von einer Bedrohung heimgesucht wird. Meistens handelt es sich dabei um einen oder mehrere Antagonisten, welche die Spielwelt zerstören wollen. Die Protagonisten setzen sich meistens aus einer Gruppe mehrerer Charaktere zusammen und nehmen es sich zur

[1] NPC ist eine Abkürzung für den englischen Begriff „Non-Player Character“. Im deutschen versteht man darunter eine nicht-spielbare Spielfigur, dessen Verhalten vom Spiel selbst festgelegt ist. Der Spieler ist meistens dazu fähig, mit NPCs interagieren zu können.

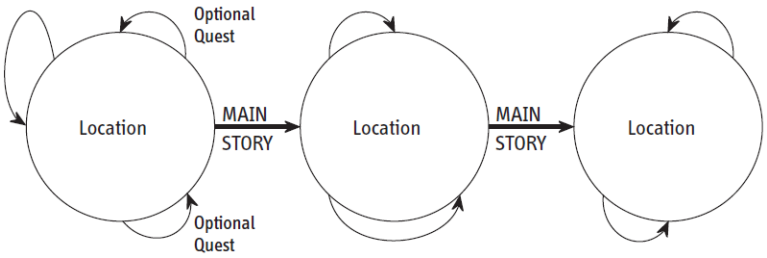


Abbildung 2: *Graphische Darstellung des Verlaufs eines typischen Rollenspiels. [1, Seite 458]*

Hauptaufgabe, die Antagonisten bei ihrem Vorhaben zu stoppen und sie endgültig in einem „finalen Kampf“ zu besiegen. Gehen die Protagonisten siegreich hervor, führt dies zur Vollendung des Spiels.

Die Geschichte wird meistens über eine Reihe von Dialogen erzählt. Dies wurde in früheren Teilen der Serie über textuelle Dialogboxen realisiert (siehe dazu Abbildung 3).

Abbildung 3: *Ein Screenshot zur Realisierung von Dialogen in Final Fantasy VII.*

Eine Dialogbox setzt sich meistens aus dem Namen der sprechenden Figur und dem Inhalt zusammen. Manch andere Spiele fügen auch Portraits der Charaktere ein, um es noch deutlicher zu machen, wer gerade spricht. In aktuelleren Titeln von Final Fantasy werden textuelle Dialoge zusätzlich mit Sprachsynchronisierungen versehen.

Bis der finale Kampf eingeleitet wird, müssen die Protagonisten eine Reihe von Aufträgen absolvieren. Damit verbunden ist das Aufsuchen und Erkunden einzelner Orte. Der Spieler wird dabei meistens in rundenbasierte Zufallskämpfe verstrickt. In Abbildung 4 sieht man einen Ausschnitt einer Kampfszene in *Final Fantasy VII* [7]. Die Kämpfe werden in Abhängigkeit der Zeit ausgeführt. Eine Spielfigur verfügt standardmäßig über die zwei Ressourcen Lebenspunkte (HP) und Magiepunkte (MP). Sie gilt als aktionsfähig, wenn die Lebenspunkte nicht null sind und der Zeitbalken (TIME) vollständig gefüllt ist. Über einen Zeiger (Zeigefinger)

wählt der Spieler eine Aktion für eine der aktionsfähigen Spielfiguren aus, die mit einem gelben Zeiger markiert ist. Können mehrere Spielfiguren eine Aktion ausführen, kann der Spieler den gelben Zeiger zwischen den Charakteren wechseln lassen. Der Spieler muss im Allgemeinen jedoch keine Spielzüge tilgen, um das Kampfgeschehen fortschreiten zu lassen. In diesem Fall würde der Gegner kontinuierlich angreifen, sobald sein Spielzug verfügbar wird. Die Magiepunkte werden für den Einsatz von Zaubern (Magic) benötigt. Zaubersprüche können je nach Vorliegen der Gegner ziemlich mächtig sein und verbrauchen beim Einsatz eine vorgegebene Menge an MP. Ein Zauberspruch ist daher nur dann einsetzbar, wenn die Spielfigur über die benötigten Magiepunkte verfügt. Normale Angriffe (Attack) können stattdessen immer eingesetzt werden und sind damit nicht von einer Ressource abhängig. Der Spieler hat außerdem die Möglichkeit, einen Gegenstand (Item) aus seinem Inventar einzusetzen, um zum Beispiel seine Ressourcen wieder aufzufüllen.

Abbildung 4: *Ein Screenshot vom Aufbau einer Kampfszene in Final Fantasy VII.*

Zum Voranschreiten der Geschichte gehört oftmals auch das erfolgreiche Bekämpfen stärkerer Gegner, sogenannte „Bossgegner“, dazu. Diese Kämpfe dauern meistens länger als gewöhnliche Zufallskämpfe und stellen den Spieler vor eine größere Herausforderung. Siegreich hervorgegangene Kämpfe werden mit Spielwährung, Erfahrungspunkten und manchmal auch Gegenständen belohnt. Erfahrungspunkte lassen die Charaktere in ihrer Charakterstufe ansteigen. Je höher die Charakterstufe ist, umso mehr steigen die Statuswerte an und umso stärker werden die Charaktere. Eine beispielhafte Übersicht zu verschiedenen Statuswerten in Final Fantasy VII [7] wird in Abbildung 5 dargestellt. Neben dem Aufstieg der Charakterstufe (LV) haben auch Ausrüstungsgegenstände (Wpn, Arm und Acc) Einfluss auf die Erhöhung oder Erniedrigung von Statuswerten.

Abbildung 5: *Ein Screenshot zur Übersicht der Statuswerte eines Charakters in Final Fantasy VII.*

2.1.3 Action-Rollenspiel

Aus Kapitel 2.1.1 und 2.1.2 konnten wesentliche Eigenschaften der beiden Genres Action und Rollenspiel dargelegt werden. Diese sollen mitsamt persönlicher Erfahrungen mit Action-Rollenspielen einen Überblick über dieses Genre geben. Das Action-Rollenspiel selbst stellt ein Hybrid der beiden Genres Action und Rollenspiel dar. In diesem Sinne werden Action- und Rollenspiel-Elemente miteinander vermischt.

Action-Rollenspiele verfügen im Vergleich zu klassischen Rollenspielen meistens über eine weniger umfangreiche und oberflächlich gehaltene Geschichte, welche über Dialoge und animierte Filmsequenzen erzählt wird. Auch in einem Action-Rollenspiel gilt es, Aufträge zu absolvieren, aber deutlich im Vordergrund und eng mit den Aufträgen verbunden steht das Bezwingen von Herausforderungen und das ständige Verbessern des spielbaren Charakters über ansteigende Statuswerte oder das Ausrüsten immer besserer Ausrüstung. Besiegte Gegner geben Erfahrungspunkte, die die Charakterstufe der Spielfigur und somit dessen Statuswerte ansteigen lassen. Was das Action-Rollenspiel primär vom klassischen Rollenspiel unterscheidet, ist das Bestreiten der Kämpfe in Echtzeit. Echtzeit-Kämpfe machen ein Spiel sehr dynamisch und stellen die physischen Fähigkeiten des Spielers auf die Probe. Der Action-Aspekt liegt daher in den Kämpfen selbst. Während man in einem klassischen Rollenspiel oft genug Zeit hatte, seinen Spielzug zu setzen, muss man bei einem Action-Rollenspiel für jede Situation sofort entscheiden, welche Aktion eingesetzt werden soll. Eine ungünstig ausgewählte Aktion kann einen Kampf schnell fehlschlagen lassen und somit zum Tod der Spielfigur führen. Der Spieler muss demnach schnell reagieren und die zur Verfügung stehenden Aktionen sinnvoll einsetzen.

Die Weiterentwicklung der Charaktere geschieht wie bei einem klassischen

Rollenspiel über das Aufsteigen in der Charakterstufe. Bei jedem Anstieg werden Punkte erworben, die man meistens auf Statuswerte oder Fähigkeiten verteilen kann. Action-Rollenspiele wie Kingdom Hearts [8] übernehmen die Verteilung oft automatisch nach einem vordefinierten Schema. Die Statuswerte lassen sich daher vom Spieler kaum beeinflussen. Der Spieler kann die Spielfiguren jedoch immer mit besseren Ausrüstungsgegenständen ausrüsten, um einige Statuswerte zu manipulieren. Bei Spielen wie Diablo III [2] dagegen werden Fähigkeiten beim Erreichen bestimmter Charakterstufen für den Einsatz freigeschaltet. Die Statuswerte errechnen sich anhand der Basiswerte der Spielfigur und den Werten der Ausrüstungsgegenstände, mit denen man die Spielfigur ausstattet. Dementsprechend muss der Spieler immer bessere Ausrüstungsgegenstände finden, um stärker zu werden.

2.2 Unity

Unity ist eine von Unity Technologies entwickelte Entwicklungsplattform, mit dessen Hilfe sich vor allem Videospiele in 2D und 3D umsetzen lassen. Diese umfasst eine Game Engine, die alle notwendigen Funktionen bereitstellt, um Spiele schnell und effizient erstellen zu können. Im Folgenden werden die wichtigsten Bereiche erläutert, die für die Umsetzung des Action-Rollenspiels in Kapitel 4 von starker Bedeutung sein werden.

2.2.1 Unity Asset Store

Ein sehr wichtiges Feature von Unity stellt der *Unity Asset Store* dar. Dabei handelt es sich um eine immer anwachsende Bibliothek für Assets. Unter Assets sind Dateien zu verstehen, die in Unity importiert und verwendet werden können. Unity Technologies aber auch Mitglieder der Unity Community stellen dort sowohl kostenlose als auch kostenpflichtige Assets zur Verfügung. Die Auswahl reicht von einzelnen Dateien wie Skripte, Modelle, Animationen oder Grafiken bis hin zu ganzen Frameworks und Projekten. Um sich manchen Entwicklungsaufwand und damit möglicherweise verbundene Kosten zu ersparen, könnte es sinnvoller sein, auf bereits verfügbare Assets zurückzugreifen. Das Importieren der Assets aus dem Store erfolgt über das in Unity integrierte Unity Asset Store Window (siehe Abbildung 6). [9]

2.2.2 GameObjects

Bei der Erstellung eines Spiels werden *GameObjects* besonders wichtig. Jeder Bestandteil eines Spiels wird als GameObject repräsentiert. Ein GameObject kann beispielsweise ein spezifisches Modell einer Spielfigur, die Kamera, eine Lichtquelle oder einfach nur ein leerer Container sein. Jedes GameObject wird in Unity über die *Scene View* in der Spielwelt platziert. Gleichzeitig werden sie auch in der *Hierarchy View* abgelegt und stellen dabei jeweils einen Kind-Knoten der Szene dar (vgl. Abbildung 7). GameObjects, die als Container eingesetzt werden, kön-

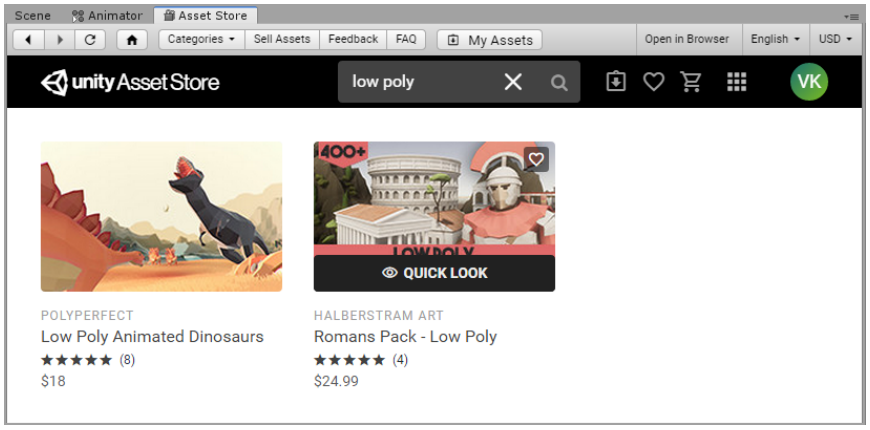


Abbildung 6: *Ein Screenshot des Unity Asset Store Windows in Unity.*

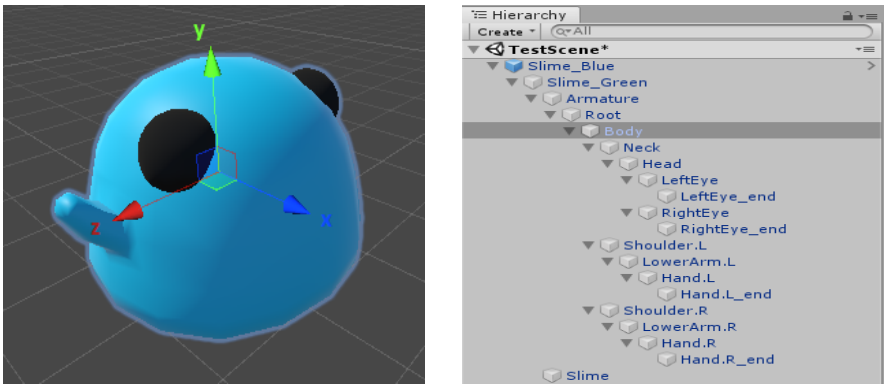


Abbildung 7: *Links: Ein spezifisches Modell als GameObject in der Scene View. Rechts: Die zum GameObject zugehörige Hierarchie.*

nen ebenfalls über Kind-Knoten verfügen. Damit werden auf Eltern-Knoten ausgeführte Transformationen wie die Translation, die Skalierung oder die Rotation auch gleichzeitig auf deren Kind-Knoten entsprechend übertragen.

Komponenten. Im Allgemeinen verfügen GameObjects über Komponenten, die ihnen jeweils eine Funktionalität zuweisen. Standardmäßig ist eine *Transform*-Komponente enthalten, die alle wichtigen Informationen zur Position und Orientierung im Raum beinhaltet. Weitere Komponenten können über den *Inspector* hinzugefügt werden. Eine große Auswahl an Komponenten stellt Unity selbst bereits zur Verfügung. Nutzer können jedoch auch eigene Komponenten in Form von Skripten hinzufügen. Diese werden in der Programmiersprache C# erstellt und verwenden die *Unity Scripting API*. Mithilfe solcher Skripte kann man beispielsweise das Verhalten von GameObjects definieren oder das Steuern dieser über Eingaben des Nutzers ermöglichen. [9]

2.2.3 Prefab System

Unity verfügt über ein sogenanntes *Prefab System*, welches es ermöglicht, GameObjects zusammen mit all ihren Komponenten und Kind-Knoten als ein wiederverwendbares Asset abzuspeichern. Mit einem solchen Asset ist es viel einfacher, künftige Änderungen an GameObjects vorzunehmen. Wird eine Änderung an einem GameObject eines spezifischen Prefabs vollzogen, kann diese auch für alle weiteren auf diesem Prefab basierenden GameObjects übernommen werden. Die Informationen einzelner GameObjects werden somit mit den Informationen des Prefabs synchronisiert. In einem Prefab können auch weitere Prefabs enthalten sein. Somit lassen sich komplexe Hierarchien aufbauen, die durch die Synchronisierung von Änderungen einfacher zu bearbeiten sind.

Die Verwendung von Prefabs wird vor allem dann sehr bedeutsam, wenn man verschiedene Objekte zur Laufzeit instanziieren möchte. So kann man zum Beispiel gegnerische Spielfiguren unter bestimmten Bedingungen und an verschiedenen Positionen erscheinen lassen. [9]

2.2.4 Mecanim

Zum Animieren von GameObjects stellt Unity ein sehr umfangreiches Animationssystem namens „Mecanim“ zur Verfügung. Es unterstützt sowohl in Unity als auch in anderen 3D-Applikationen, wie Autodesk Maya oder Blender, erstellte Animationclips. Sie enthalten Informationen darüber, wie sich bestimmte Eigenschaften eines Objekts wie die Position oder Rotation über die Zeit hinweg verändern. Animationclips eignen sich insbesondere zur Simulation menschlicher Bewegungen.

Eine Besonderheit an Mecanim ist, dass Animationclips, Transitionen und Interaktionen zwischen diesen Clips über ein visuelles Programmierwerkzeug gesetzt und bearbeitet werden können. Dadurch lässt sich dieses System ziemlich einfach bedienen.

Des Weiteren verfügt Mecanim über spezielle Features zum Behandeln menschlicher Charaktere. Über ein „Humanoid Animation Retargeting“-Verfahren werden die Korrespondenzen zwischen den Bestandteilen der Spielfigur und einer humanoiden Knochenstruktur gesetzt, um beliebige humanoide Animationen auf dem vorliegenden Modell ausführbar zu machen. Die humanoiden Animationen müssen daher nur einmalig erstellt werden. Mecanim erspart dem Nutzer damit jede Menge Animierungsaufwand.

In Abbildung 8 kann man sehen, wie Unitys Animationssystem für ein spezifisches humanoides Spielobjekt verwendet wird.

Neben der Möglichkeit, Animationen und ihre Zusammenhänge visuell festzulegen, können auch den Zuständen Verhaltensweisen im Sinne von Skripten zugewiesen werden. Mithilfe dieser lassen sich Instruktionen bereits beim Betreten, Verlassen oder während der Ausführung eines Zustandes ausführen. Zusätzlich lassen sich an bestimmten Stellen der Animationen auch AnimationEvents definieren, die zum Beispiel eine Funktion eines an dem betroffenen GameObject haftenden

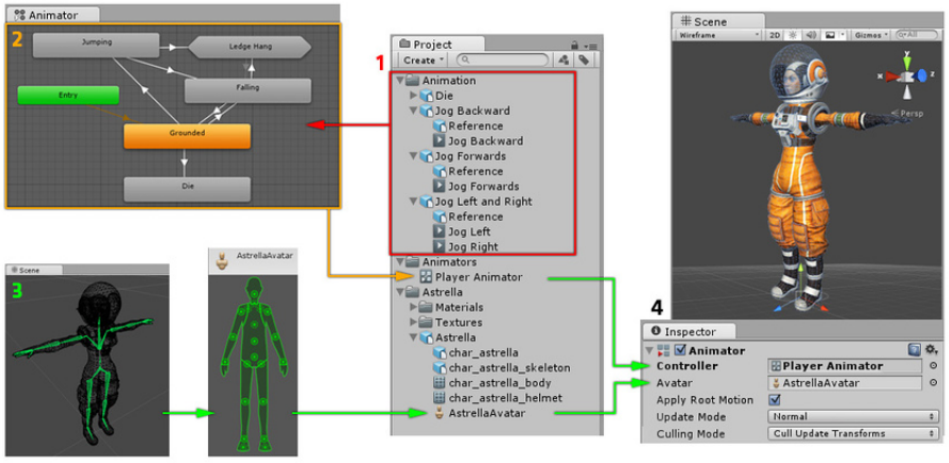


Abbildung 8: *Wie das Animationssystem Mecanim verwendet wird. [9]*

Skripts aufrufen können. [9]

2.2.5 Coroutines

Bei Situationen, in denen es notwendig ist, einen laufenden Prozess für eine bestimmte Zeit zu pausieren oder regelmäßige Aktualisierungen von Informationen vorzunehmen, kommen Coroutines ins Spiel. Im Gegensatz zu den Update-Funktionen der Unity Scripting API sind Coroutines nicht vom Frame abhängig.

In einem Skript wird eine Coroutine mit dem Aufruf der Funktion `StartCoroutine` gestartet. Als Parameter wird eine Funktion vom Typ `IEnumerator` erwartet. Eine solche Funktion muss mit einem `yield return`-Statement abschließen. Über dieses Statement können Wartezeiten eingebaut werden, die erst abklingen müssen, bis der eigentliche Prozess wieder fortgesetzt werden kann.

In Kontext von Videospielen werden Coroutinen vor allem beim Behandeln von Zeit-abhängigen Effekten wie ausgeteilter Schaden oder regenerierte Lebenspunkte pro Zeiteinheit sehr bedeutsam. [9]

2.2.6 Physics Engine

Für die Simulation physikalischen Verhaltens von GameObjects stellt Unity eine integrierte *Physics Engine* zur Verfügung. Wichtige Bestandteile dieser Engine sollen hier kurz erläutert werden.

Collider. Zur Detektion physischer Kollisionen werden Collider benötigt. Hierfür muss einem GameObject ein spezifischer Collider als Komponente hinzugefügt werden. Für 3D-Projekte stehen dafür beispielsweise Mesh Collider, Box Collider, Sphere Collider und Capsule Collider zur Verfügung, die jeweils eine eigene Form annehmen. Über Funktionen der Unity Scripting API wie `OnTriggerEnter`

oder `OnTriggerExit` kann definiert werden, was beim Betreten oder Verlassen des Collider-Bereichs über einen anderen Collider passieren soll.

Raycasting. Mithilfe der Physics Engine ist es möglich, per Raycasting-Verfahren einen Strahl beliebiger Länge in eine spezifische Richtung auszusenden. Mithilfe der `Physics.Raycast`-Methode der Unity Scripting API kann überprüft werden, ob der Strahl auf einen Collider getroffen ist. Hiermit wäre es beispielsweise möglich, spezifische Objekte in Blickrichtung des Spieler-Objektes zu detektieren. [9]

3 Konzeption des Spiels

Dieses Kapitel befasst sich mit der Formulierung eines Konzepts für die Umsetzung des Action-Rollenspiels. Mit diesem Konzept soll es ermöglicht werden, das angestrebte Spiel in der Game Engine Unity umzusetzen.

3.1 Spielziel

Ziel des Spiels ist es, eine Reihe vordefinierter Aufgaben, sogenannte Quests, zu absolvieren. Gegenstand dieser Aufgaben können Interaktionen mit bestimmten NPCs, das Aufsuchen bestimmter Orte der Spielwelt und das Ausschalten bestimmter Gegner sein. Da es sich um ein Action-Rollenspiel handelt und demnach das Kämpfen selbst sehr im Zentrum des Spiels liegt, sollen Aufgaben, die den Spieler in Kämpfe verwickeln, überwiegen. Die Anzahl und der Umfang der Aufgaben wird für diesen ersten Prototyp auf die zum Schluss durchzuführende Nutzungsstudie abgestimmt.

Zum Erreichen des Spielziels werden ein Kampf- und ein Quest-System benötigt. Außerdem muss eine Spielwelt mit all ihren vorkommenden Spielfiguren konzipiert werden, in der die Aufgaben zu absolvieren sein werden.

3.2 Spielwelt

Ein zentraler Bestandteil des Spiels ist die Spielwelt. Sie umfasst sowohl begehbare Areale als auch in ihr vorkommende Spielfiguren. Im Folgenden werden wichtige Eigenschaften und Bestandteile der Spielwelt konzeptuell dargelegt.

3.2.1 Grafische Anforderungen

Die Gestaltung der Spielwelt und aller in ihr vorkommenden Spielfiguren erfolgt in 3D. Für eine gute Umsetzung eines funktionsfähigen Kampfsystems ist eine gute Performanz des Spiels notwendig. Da diese Umsetzung sehr zeitaufwändig sein wird, sollten die grafischen Anforderungen eher niedrig gehalten werden. Hierfür soll möglichst auf Low-Poly-Modelle zurückgegriffen werden. Darunter sind 3D-Modelle zu verstehen, die geringfügig aufgelöst sind, somit nur über grobe Details

verfügen und im Vergleich zu detailreichen Modellen weniger grafischen Berechnungsaufwand erfordern.

3.2.2 Aufbau der Spielwelt

Die Spielwelt besteht aus einem quadratisch gehaltenen, hügeligen Areal, in dessen Zentrum sich ein kleines Lager befindet. In diesem Lager befinden sich alle NPCs und auch die spielbare Spielfigur wird beim Starten des Spiels von dort aus beginnen. Außerhalb des Lagers befindet sich eine begehbare Landschaft, in der gegnerische Spielfiguren ihr Unwesen treiben. Das Areal selbst wird von Bergketten umschlossen, was die Erkundung der Spielwelt auf dieses eine Areal begrenzt.

3.2.3 Vorkommende Spielfiguren

In der Spielwelt werden insgesamt drei verschiedene Gruppen von Spielfiguren auftreten, die hier kurz erläutert werden sollen.

Die Spielfigur des Spielers. Der Spieler wird beim Betreten der Spielwelt dazu befähigt, eine vorgegebene Spielfigur zu steuern. Diese startet in der Spielwelt im Zentrum des Lagers und ist mit einem Zweihandschwert ausgestattet, welches sie zum Bestreiten der Kämpfe benötigt. Mithilfe dieser Spielfigur gilt es, das Spielziel zu absolvieren. Die Spielfigur wird demzufolge fähig sein, Aufträge bearbeiten zu können.

Gegnerische Spielfiguren. Während der Bearbeitung der Aufgaben wird die Spielfigur des Spielers mit gegnerischen Spielfiguren konfrontiert. Sie sind nur außerhalb des Lagers anzutreffen und unterscheiden sich in ihrem Aussehen und in ihrem Verhalten. Es werden sowohl friedliche als auch aggressive Gegner in der Spielwelt existieren. Aggressive Gegner nutzen jederzeit die Gelegenheit dazu, den Spieler anzugreifen, wohingegen friedliche Gegner nur dann zurückschlagen, wenn der Spieler sie bereits mit einem Angriff provoziert hat.

NPCs. Um den Spieler jederzeit zu informieren und mit Aufgaben auf den Weg zu schicken, werden auch NPCs Teil der Spielwelt sein. Sie kommen nur im Lager der Spielwelt vor und verbleiben an Ort und Stelle. Es wird möglich sein, mit ihnen zu interagieren, wodurch Dialoge eröffnet und Quests ihrem Zustand entsprechend behandelt werden.

3.3 Funktionsweise des Kampfsystems

Ein Ziel dieser Arbeit ist die Umsetzung eines funktionsfähigen und attraktiven Kampfsystems. Dafür müssen mögliche Verhaltensmuster aller Spielfiguren und dessen Abhängigkeiten definiert werden.

3.3.1 Ressourcenabhängigkeiten

Am Beispiel von Final Fantasy VII in Kapitel 2.1.2 konnte man sehen, dass Aktionen meistens von vorgegebenen Ressourcen abhängig sind, um das Spiel für den

Spieler nicht zu einfach zu machen. Auch die Spielfiguren in dem von mir angestrebten Spiel sollen über Ressourcen verfügen.

Lebenspunkte (LP). Sowohl Gegner-Objekte als auch das Spieler-Objekt besitzen jeweils eine vordefinierte Menge an LP. Lebenspunkte werden um den Schadenswert eingehender Angriffe reduziert. Erreicht diese Ressource den Wert 0, stirbt die Spielfigur und ist somit nicht mehr aktionsfähig. Lebenspunkte des Spielers werden im Laufe der Zeit geringfügig wiederhergestellt, sofern er innerhalb eines bestimmten Zeitfensters keine Schadenspunkte erlitten hat.

Ausdauerpunkte (AP). Über Ausdauerpunkte verfügt lediglich nur die Spielfigur des Spielers. Diese Ressource dient dazu, mögliche Aktionen in ihrer Verfügbarkeit zu limitieren, um möglichen Exploits[2] im Spiel entgegenzuwirken und das Spiel herausfordernder zu gestalten. Es wird Aktionen geben, die einmalig oder kontinuierlich wirken können. Einige Aktionen werden auch auf die AP-Ressource zurückgreifen. Der Spieler wird somit zu strategischem Denken verleitet, um seine Aktionen für jede gegebene Situation sinnvoll einzusetzen. Ausdauerpunkte stellen sich im Laufe kurzer Zeit wieder von selbst her, sofern für eine bestimmte Zeit keine Aktion eingesetzt wurde, die diese Ressource verbraucht.

3.3.2 Aktionen des Spielers

Der Spieler wird im Spiel eine festgelegte Spielfigur selbstständig steuern können. Da die Aufgaben darauf abzielen, die Spielfigur in Kämpfe zu verwickeln, muss das Kampfsystem dem Spieler eine Reihe von Aktionen zur Verfügung stellen, die das Verhalten seiner Spielfigur unterschiedlich beeinflussen können. Die Ausführbarkeit einiger Aktionen steht dabei in Abhängigkeit der verfügbaren Ausdauerpunkte (AP). Ebenfalls wird es Aktionen geben, die nur in Kombination anderer Aktionen ausführbar sein werden. Im Gegensatz zu einem klassischen Rollenspiel wie Final Fantasy VII [7] werden Aktionen nicht in Abhängigkeit von Spielzügen sondern vollständig in Echtzeit ausgeführt. In Tabelle 1 wird eine Übersicht aller möglichen Aktionen des Spielers für einen gegebenen, hier noch nicht weiter spezifizierten Input und in Abhängigkeit des Zustands der Waffe dargestellt. Diese weitere Abhängigkeit dient dazu, einen Input mehrfach verwenden zu können und den Spieler somit nicht mit zu vielen Tasten zu überfordern. Ein Input kann daher zwei verschiedene Reaktionen auslösen. Beispielsweise führt Input 4 bei ausgerüstetem Zweihandschwert zu einem Angriff. Ist das Zweihandschwert nicht angelegt, führt Input 4 eine mögliche Interaktion mit einem NPC aus. Im Folgenden sollen die Auswirkungen aller Aktionen kurz beschrieben werden.

Fortbewegen. Die Spielfigur lässt sich in der Spielwelt in alle acht Himmelsrichtungen hinfort bewegen. Diese Aktion lässt sich völlig unabhängig von den Ausdauerpunkten kontinuierlich ausführen.

Sprinten. Während der Fortbewegung besteht die Option, die Spielfigur in Form von Sprinten schneller fortbewegen zu lassen. Diese Aktion wird bei kontinu-

[2] Unter einem Exploit wird das Ausnutzen von ungewollten, fehlerhaften Spielfeatures zum Vorteil des Spielers verstanden.

Input	Reaktion (ohne Waffe)	Reaktion (mit Waffe)
1	Fortbewegen	Fortbewegen
2	Sprinten	Sprinten
3	Zweihandschwert ausrüsten	Zweihandschwert entrüsten
4	Interagieren	Angreifen
5	-	Blocken
6	Ausweichen	Ausweichen
7	Heiltrank einsetzen	Heiltrank einsetzen

Tabelle 1: *Übersicht aller vom Spieler ausführbaren Aktionen in Abhängigkeit des Zustands der Waffe.*

ierlichem Betätigen des Inputs fortlaufend ausgeführt und führt zu einer Reduktion der Ausdauerpunkte pro Zeiteinheit.

Zweihandschwert ausrüsten und entrüsten. Da manche Aktionen nur in Abhängigkeit der Waffe auszuführen sind, muss dem Spieler die Möglichkeit geboten werden, den Zustand der Waffe zu wechseln. Über das Ausrüsten und Entrüsten des Zweihandschwertes soll dieser Zustand entsprechend verändert werden. Für den Einsatz dieser Aktion werden keine Ausdauerpunkte benötigt.

Interagieren. Der Spieler kann im waffenlosen Zustand der Spielfigur eine mögliche Interaktion mit einem NPC ausführen. Ist derzeit keine Interaktion möglich, führt das Ausführen dieser Aktion zu keiner Reaktion im Spiel. Bei der Ausführung dieser Aktion werden keine Ausdauerpunkte verbraucht.

Angreifen. Eine für das Kampfsystem wichtige Aktion ist das Angreifen. Jede Betätigung dieser Aktion führt zur Reduktion der Ausdauerpunkte um einen konstanten Wert. Wird ein Gegner von einem Angriff getroffen, werden dessen Lebenspunkte um einen konstanten Schadenswert reduziert.

Blocken. Das Blocken ist ebenfalls für das Bestreiten der Kämpfe von besonderer Bedeutung. Diese Aktion lässt sich fortlaufend ausführen. Die Spielfigur erleidet bei eingehenden Angriffen vonseiten der Gegner keinerlei Schadenspunkte. Dementsprechend werden keine Lebenspunkte reduziert, jedoch verringern sich während der Ausführung dieser Aktion die Ausdauerpunkte um einen geringfügigen Wert pro Zeiteinheit. Ferner ist der Spieler während des Blockens nicht fähig, die Spielfigur fortzubewegen. Um diese Aktion für den Einsatz attraktiv zu gestalten, verursacht der Spieler mit seinem nächsten Angriff doppelten Schaden, wenn er einen gegnerischen Angriff erfolgreich geblockt hat.

Ausweichen. Ähnlich wie beim Blocken erleidet die Spielfigur während des Ausweichens keine Schadenspunkte. Ein entscheidender Vorteil dieser Aktion ist, dass sich die Spielfigur dabei bewegt und somit Abstand zu gegnerischen Spielfiguren gewinnen kann. Diese Aktion kann jedoch nur während der Fortbewegung ausgeführt werden und führt zu einer sofortigen Reduktion der Ausdauerpunkte um einen konstanten Wert.

Heiltrank einsetzen. Da sich die Lebenspunkte der Spielfigur nur abseits der Kämpfe und nur sehr langsam wiederherstellen, soll dem Spieler jederzeit ein Heiltrank zur Verfügung stehen, mit dem er einen prozentualen Anteil seiner maximalen Lebenspunkte wieder zurückgewinnen kann. Der Einsatz dieser Aktion verbraucht keine Ausdauerpunkte, jedoch muss ein bestimmtes Zeitfenster abgewartet werden, um den Heiltrank wieder einsatzbereit zu machen.

3.3.3 Verhaltensmuster der Gegner

Da gegnerische Spielfiguren nicht über den Spieler gesteuert werden können, muss ein allgemeines Verhaltensmuster für die Gegner entwickelt werden, welches für eine gegebene Situation über das Verhalten dieser entscheidet. Zu Spielbeginn wird jedem Gegner-Objekt ein Ursprung in der Spielwelt zugewiesen, in dessen Umkreis sie per Spawn-Prinzip erscheinen werden. Es wird insgesamt fünf Zustände geben, in denen sich ein Gegner aufhalten kann. Die Entscheidung für den Aufenthalt in einem dieser Zustände geschieht in Abhängigkeit der Distanz des Spielers zum Gegner-Objekt. Wie in Kapitel 3.2.3 bereits beschrieben, können gegnerische Objekte friedlich oder feindlich gesinnt sein. Entsprechend beeinflusst diese Gesinnung ebenfalls das Erreichen dieser Zustände.

Patrouillieren. Standardmäßig bewegen sich gegnerische Spielfiguren innerhalb eines Radius um ihren Ursprung hinfort. Sie laufen dabei eine Reihe von Punkten ab, die zufällig ausgewählt werden, unterbrechen anschließend ihre Patrouille für ein kurzes, aber zufällig bestimmtes Zeitfenster und setzen diese im Anschluss wieder fort. Aggressive Gegner-Objekte können diesen Zustand in Abhängigkeit ihrer Distanz zum Spieler-Objekt verlassen, wohingegen friedliche Gegner solange in diesem Zustand verbleiben, bis sie vom Spieler über einen Angriff provoziert und somit aggressiv gestimmt wurden.

Beobachten. Nähert sich der Spieler einem aggressiven Gegner-Objekt bis zu einer vorgegebenen Distanz, unterbricht der Gegner seine Patrouille und verbleibt an seiner aktuellen Position. Um bedrohlich zu wirken, starrt er den Spieler an, indem er sich immer in Richtung des Spielers entsprechend ausrichtet. Dieses Verhalten soll dem Spieler signalisieren, dass es auf einen Kampf mit diesem Gegner-Objekt hinauslaufen wird, wenn sich der Spieler ihm weiterhin nähert. Distanziert sich der Spieler wieder, geht der Gegner seiner üblichen Patrouille wieder nach. Friedliche Gegner erreichen den Beobachten-Zustand nur dann, wenn sie über einen Angriff des Spielers schon einmal provoziert wurden und sich der Spieler anschließend von ihnen ausreichend distanziert hat.

Verfolgen. Reduziert sich die Distanz zwischen Spieler und aggressiven Gegner-Objekt noch weiter bis zu einem vorgegebenen Wert, fängt das Gegner-Objekt an, den Spieler zu verfolgen. Der Gegner verfolgt den Spieler solange, bis er ihm nah genug kommt, um ihn dann anzugreifen. Friedlich gesinnte Gegner können diesen Zustand nur dann erreichen, wenn sie über einen Angriff des Spielers schon einmal provoziert wurden und ihm ausreichend nah genug stehen, die Distanz jedoch nicht für einen Angriff ausreicht.

Angreifen. Reicht die Distanz zwischen Spieler-Objekt und Gegner-Objekt für einen möglichen Angriff aus, verbleibt der Gegner an Ort und Stelle und führt innerhalb eines vorgegebenen Zeitintervalls einzelne Angriffe aus. Durch eingehende Angriffe seitens des Spielers verschiebt sich das Zeitintervall für den nächsten Angriff um einen geringfügigen Wert, damit die Kämpfe dem Spieler gegenüber fairer werden.

Zum Ursprung zurückkehren. Distanziert sich das Gegner-Objekt durch den Verfolgungszustand zu sehr von seinem Patrouillen-Bereich, verliert es das volle Interesse an dem Kampf und läuft zügig zu einem zufällig ausgewählten Punkt seiner Patrouille zurück. Der Gegner geht anschließend in den Patrouillieren-Zustand über. Friedlich gesinnte Gegner-Objekte, die zuvor aggressiv gestimmt wurden, verhalten sich künftig wieder friedlich.

3.4 Funktionsweise des Quest-Systems

Um das in Kapitel 3.1 erklärte Spielziel zu erfüllen, wird ein Quest-System benötigt. Das Spiel muss es ermöglichen, verfügbare Aufgaben (Quests) anzunehmen, diese zu vervollständigen und schließlich wieder abzugeben. Dies soll über verschiedene Zustände der Quests realisiert werden. Quests sollen über einzelne NPCs vergeben werden. Der Ablauf der Quests erfolgt linear per Kettenreaktion. Wird eine Quest abgeschlossen, wird eine Neue zur Verfügung gestellt. Auftraggeber, Quest-Zustände und Quest-Ziel sollen dem Spieler jederzeit über visuelle Elemente einsichtig gemacht werden. Auf die einzelnen Quest-Zustände soll im Folgenden kurz eingegangen werden.

NichtVerfügbar. Verfügt ein NPC über eine Quest, die noch im nicht verfügbaren Zustand vorliegt, führt eine ausgeführte Interaktionen mit ihm seine Standard-Dialoge aus.

Verfügbar. Wird eine Quest verfügbar, stellt der NPC sie dem Spieler zur Annahme bereit. Interagiert der Spieler mit diesem NPC, schildert er sein Anliegen in Form eines Dialoges, um welches sich der Spieler kümmern soll. Gleichzeitig wird die Quest in den Zustand Angenommen versetzt.

Angenommen. Hat der Spieler eine Quest angenommen, muss er das besagte Ziel vervollständigen. Eine Interaktion mit dem Auftraggeber hat keinen Einfluss auf den Zustand der Quest und eröffnet nur einen Dialog, der den Spieler an das zu erledigende Aufgabenziel erinnert.

Vervollständigt. Wurde das Quest-Ziel erfüllt, wechselt die Quest ihren Zustand zu Vervollständigt. Die Quest kann nun beim Auftraggeber abgegeben werden.

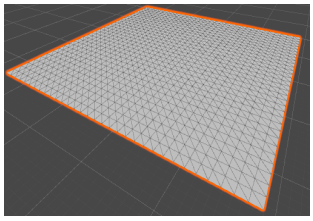
Abgeschlossen. Wird eine Quest beim Auftraggeber per Interaktion abgegeben, gilt diese als vollendet und wird in den Zustand Abgeschlossen überführt. Gleichzeitig eröffnet sich ein Dialog, in dem sich der NPC beim Spieler für seine gute Arbeit bedankt. Außerdem wird eine neue Quest bei demselben oder einem anderen NPC zur Verfügung gestellt.

4 Implementation des Spiels

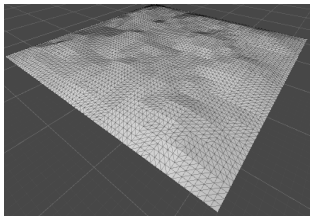
Mit der Konzeption des Spiels aus Kapitel 3 konnte ein theoretisches Grundgerüst für die Implementation erstellt werden. Im Folgenden werden die wichtigsten Bestandteile für die Umsetzung des Spiels anhand von Bildern und Programmcode dargelegt.

4.1 Erstellung der Spielwelt

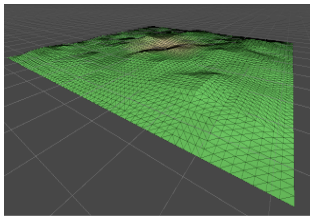
Für die Spielwelt wird ein quadratisch gehaltenes und hügeliges Low-Poly-Areal benötigt, worauf sich alle Spielfiguren fortbewegen werden. Dazu werden zwei Assets aus dem Unity Asset Store benötigt: *ProBuilder* und *PolyBrush*. ProBuilder dient hauptsächlich zur Erstellung einfacher Geometrien und eignet sich daher sehr gut zur Modellierung eines Low-Poly-Terrains. PolyBrush ermöglicht es, beliebige Meshes mit Texturen, Farben und weiteren darauf haftenden Objekten zu versehen. Außerdem lassen sich einzelne Vertices des Meshs in ihrer Höhe anpassen. Demzufolge lassen sich mit PolyBrush auch mit ProBuilder erstellte Geometrien nachbearbeiten. Nachfolgend werden die einzelnen Schritte zur Erstellung der Spielwelt dokumentiert (siehe auch Abbildung 9).



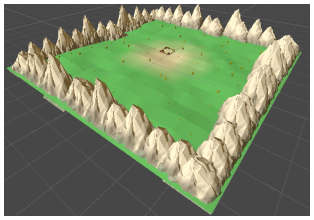
(a) *Konstruktion eines Plane-Objekts.*



(b) *Anhebung der Vertices.*



(c) *Einfärbung der Plane.*



(d) *Dekoration der Plane.*

Abbildung 9: *Darstellung des Erstellungsprozesses der Spielwelt.*

Schritt 1: Kontruktion einer geeigneten Geometrie über ProBuilder. Zur Erstellung des Areals bietet es sich zunächst an, über ProBuilder ein *Plane*-Objekt konstruieren zu lassen. Dieses Objekt setzt sich aus mehreren Vierecken zusammen. Damit der Low-Poly-Look später besser zur Geltung kommt, wird dieses Objekt über die *Subdivide*-Funktion in weitere Vierecke zerlegt.

Schritt 2: Anpassung der Geometrie mit PolyBrush. Für die hügelige Eigenschaft des Areals müssen die Vertices in ihrer Höhe angepasst werden. Hierfür wird

das PolyBrush-Tool verwendet. Ähnlich wie in einem Bildbearbeitungstool lassen sich über ein Pinsel-Werkzeug einzelne Punkte bearbeiten. Über dieses Werkzeug werden nun einzelne Vertices des Plane-Meshs angehoben, um der Plane ein hügeliges Aussehen zu verleihen.

Um dem Areal auch eine farbliche Note zu vergeben, wird es auf die gleiche Art und Weise in PolyBrush eingefärbt.

Schritt 3: Das Areal eingrenzen und dekorieren. Da das Areal bisher offen und somit noch zu verlassen ist, sollen Bergketten konstruiert werden, die das Areal hinsichtlich der Begehbarkeit begrenzen. Eine Bergkette besteht aus einer Aneinanderreihung einzelner Berge. Zur Konstruktion eines Berges ist das gleiche Verfahren wie in Schritt 1 und 2 anzuwenden.

Nun können einzelne Berg-Objekte an den Kanten des Areals platziert werden. Um die Berge in ihrem Aussehen variieren zu lassen, werden die Objekte in ihrer Skalierung unterschiedlich angepasst. Zudem werden über das PolyBrush-Tool einzelne Details wie Bäume und Blumen hinzugefügt. Für die Gestaltung des Lagers werden vorgefertigte Prefabs aus dem Unity Asset Store verwendet und auf das Areal manuell platziert.

4.2 Strukturierung des Spiels

Das Spiel verwendet insgesamt zwei verschiedene Szenen. Erstere beinhaltet einen Titelbildschirm, der über die Rotation eines Kamera-Objektes schon einmal einen ersten Einblick in die Spielwelt ermöglicht (siehe Abbildung 10). Zusätzlich wird ein Menü mit dem Titel des Spiels und den Optionen „Neues Spiel starten“ und „Spiel verlassen“ dargestellt. Die zwei Optionen können jeweils über den Mauscursor ausgewählt werden. Der Spieler hat dabei die Möglichkeit, die Spielwelt zu betreten oder die Spielanwendung wieder zu schließen.

Das Betreten der Spielwelt über die erste Option eröffnet die zweite Szene (siehe Abbildung 11). Der Spieler wird nun dazu befähigt, eine ritterlich bekleidete Spielfigur aus der Third Person Perspective zu steuern. Seine Spielfigur startet im Zentrum des Lagers. Die erste Aufgabe wird ihm bereits über ein vorhandenes UI-Element visualisiert. Vor dem Spieler-Objekt befindet sich der Auftraggeber, mit dem er interagieren muss. Interagiert er mit ihm, vervollständigt es die aktuelle Quest und der Spieler kann diese bei ihm mit der nächsten Interaktion abgeben. Anschließend erhält er seine nächste Aufgabe. Jeder Abschluss einer Quest eröffnet eine Neue. Der Plot hinter all den Aufgaben behandelt die Laufbahn des jungen Ritters, der sich dem Wachtrupp des Lagers anschließen möchte. Hierzu muss er zunächst eine Prüfung ablegen, um als Rekrut aufgenommen zu werden. Nach erfolgreichem Bestehen der Prüfung wird er zum Rekruten ernannt und wird dazu aufgefordert, sich um die Anliegen der Leute im Lager zu kümmern. Diese Anliegen wurden als Quests umgesetzt. Schließt der Spieler alle verfügbaren Quests ab, wird der junge Rekrut zum Kommandanten ernannt. Für die Durchführung der Nutzungsstudie wurde ab dieser Stelle extra eine Quest eingefügt, die den Spieler dazu auffordert, die Spielanwendung zu schließen.

Abbildung 10: *Ein Screenshot der fertigen Titelbildschirm-Szene.*

Abbildung 11: *Ein Screenshot der Spielwelt-Szene zu Spielbeginn.*

4.3 Interaktionen

Für den Spielverlauf werden Interaktionen sehr wichtig. Über diese soll es möglich sein, Informationen abgreifen oder Quests bearbeiten zu können. Für die Umsetzung der Interaktionen wurde ein Video-Tutorial vom YouTube-Kanal *Brackeys* [3] zur Hilfe genommen. Im Folgenden soll auf die einzelnen Aspekte eingegangen werden, die für die Interaktionen von entscheidender Bedeutung sind.

4.3.1 Dialoge

Interaktionen basieren auf die Ausführung von Dialogen. Ein Dialog setzt sich aus einer Reihe von Sätzen zusammen, die gesprochen werden sollen. Zur Darstellung von Dialogen bietet es sich an, eine Klasse `Dialogue` zu erstellen (siehe hierzu

Listing 1).

```
1 public class Dialogue {
2     public string[] sentences;
3 }
```

Listing 1: *Die Umsetzung der Klasse Dialogue in C#.*

Zusätzlich ist eine `DialogueManager`-Komponente zu implementieren, die regelt, wie ein Dialog auszuführen ist (siehe Listing 2). Die einzelnen Sätze eines Dialogs sollen nacheinander eingeblendet werden. Ein festgelegter Input ermöglicht die Einsicht des Folgesatzes. Um den Effekt „gesprochener“ Sätze zu erhöhen, sollen die einzelnen Buchstaben nacheinander „getippt“ werden. Diese Komponente ist nur einem GameObject hinzuzufügen, die die Rolle eines Managers übernimmt. Hierfür eignet sich ein leeres GameObject, welches nie zerstört wird.

```
1  FUNCTION HandleDialogue(Argument dialogue)
2      Open dialogue box ui
3      Enqueue all sentences of the dialogue argument
4      Execute CheckInteractionInputs()
5      Execute DisplayNextSentence()
6  END FUNCTION
7
8  FUNCTION DisplayNextSentence()
9      IF sentence queue is empty
10     or dialogue cancelling is allowed THEN
11         Stop CheckInteractionInputs
12         Close dialogue box ui
13         Stop executing this method
14     END IF
15     Store the first element of the sentence queue
16         into a variable
17     Remove the first element of the sentence queue
18     Execute TypeSentence(variable of first sentence)
19 END FUNCTION
20
21 COROUTINE CheckInteractionInputs()
22     WHILE true
23         IF player requests dialogue cancelling THEN
24             Allow dialogue cancelling
25         ELSE IF player requests to speed up the current
26         written sentence and typing this sentence has
27         not finished yet THEN
28             Allow displaying the current sentence
29                 immediately
30         ELSE IF player requests displaying the next sentence
31         and typing the sentence has finished THEN
32             Execute DisplayNextSentence()
33         END IF
34     END WHILE
```

```
END COROUTINE

COROUTINE TypeSentence(Argument sentence)
    Clear ui text
    FOR ALL letter in sentence argument DO
        IF displaying the current sentence immediately
        is allowed THEN
            Assign the sentence to the ui text
            Break loop
        END IF
        Add letter to the ui text
        Wait some time
    END FOR
END COROUTINE
```

Listing 2: *Ausführung von Dialogen über den DialogueManager.*

4.3.2 NPCs

Interaktionen finden immer zwischen mindestens zwei Akteuren statt. Kommuniziert wird über Dialoge, die über die `DialogueManager`-Komponente ausführbar gemacht worden sind. Jetzt fehlt nur noch die Möglichkeit, NPC-Objekte ansprechbar zu machen, die von dieser Komponente Gebrauch machen. Hierfür müssen die Objekte über eine `TriggerDialogue`-Methode verfügen (vgl. Listing 3). Jeder NPC besitzt einen oder mehrere Standard-Dialoge. Handelt es sich bei einem NPC-Objekt um einen Auftraggeber, verfügt er zusätzlich über Quest-Dialoge, die vom Zustand der aktuellen Quest abhängig sind. Ist die Quest des NPC-Objekts gerade aktiv, werden diese bei der Interaktion ausgeführt. Erst wenn keine Quest des NPC-Objekts aktiv ist, werden Standard-Dialoge ausgeführt.

```
FUNCTION TriggerDialogue()
    // higher priority of quest dialogues
    IF the npc object is a quest giver THEN
        Execute SetQuestDialogue()
        Start talking animation
        Stop executing this function
    END IF
    // lower priority of standard dialogues
    Let the DialogueManager execute HandleDialogue
        method with the npc's dialogues
    Start talking animation
END FUNCTION

FUNCTION SetQuestDialogue()
    IF the npc's quest is the current quest THEN
        // find dialogue corresponding to quest.state
        IF quest.state equals AVAILABLE THEN
            Message: "Quest accepted!"
```

```
19              Let QuestManager set quest.state to ACCEPTED
20              Let the DialogueManager execute HandleDialogue
21                  method with the quest accepting dialogue
22          ELSE IF quest.state equals ACCEPTED THEN
23              Let the DialogueManager execute HandleDialogue
24                  method with the quest accepted dialogue
25          ELSE IF quest.state equals COMPLETE THEN
26              Let QuestManager set quest.state to DONE
27              Message: "Quest completed!"
28              Let the DialogueManager execute HandleDialogue
29                  method with the quest completed dialogue
30          END IF
31      ELSE
32          Let the DialogueManager execute HandleDialogue
33              method with the npc's dialogues
34          Start talking animation
35      END IF
36  END FUNCTION
```

Listing 3: *Interaktionen mit NPC-Objekten.*

4.3.3 Standard vs. Point-To-Click

Sowohl das `DialogueManager`-GameObject als auch das NPC-GameObject stellen nun jeweils Funktionen bereit, mit denen es möglich ist, eine Interaktion durchzuführen. Damit das Spieler-Objekt mit NPCs interagieren kann, muss für die jeweiligen Steuerungsarten festgelegt werden, unter welchen Bedingungen die `TriggerDialogue`-Methode auszuführen ist.

Standard-Steuerung. Für die Standard-Steuerung wird eine Interaktion ausgeführt, wenn sich das Spieler-Objekt in Interaktionsreichweite befindet, seine Blickrichtung auf das NPC-Objekt gerichtet ist und der Spieler den zur Interaktion nötigen Input liefert. Zur Überprüfung, ob das Spieler-Objekt in Interaktionsreichweite ist, wurde ein Sphere-Collider verwendet, dessen Radius mit der Interaktionsreichweite übereinstimmt. Über die Funktionen `OnTriggerEnter` und `OnTriggerExit` kann abgefangen werden, ob sich das Spieler-Objekt innerhalb des Colliders befindet. Das Überprüfen der Blickrichtung der Spielfigur geschieht über das Raycasting-Verfahren der Physics Engine. Zusätzlich benötigt das NPC-Objekt einen weiteren Collider, beispielsweise einen Box-Collider, der das Modell umfasst. Es wird dabei ein Strahl ausgehend des Kopf-Objektes der Spielfigur mit einer Länge entsprechend der Interaktionsweite ausgesandt. Anschließend kann überprüft werden, ob der Collider des NPC-Objekts vom Strahl getroffen wurde. Sind diese Bedingungen erfüllt, wird eine Interaktion ermöglicht. Betätigt der Spieler nun die zur Ausführung der Interaktion benötigte Eingabe, wird die `TriggerDialogue`-Methode des NPC-Objekts ausgeführt und der Dialog eingeblendet.

Point-To-Click-Steuerung. Bei der Point-To-Click-Steuerung muss das Spieler-Objekt ebenfalls in Interaktionsreichweite sein. Zusätzlich muss sich der Mauscursor über ein NPC-Objekt befinden und der Input zur Interaktion betätigt werden. Hierfür wird aber nur der Box-Collider des NPC-Objektes benötigt. Es wird ausgehend von der Cursor-Position auf Kamera-Ebene ein Strahl ausgesandt, der überprüft, ob ein NPC-Objekt getroffen wurde. Zudem wird untersucht, ob die Distanz zwischen Spielfigur und NPC zur Interaktion ausreicht. Sind alle Bedingungen erfüllt, wird eine Interaktion ermöglicht. Der Spieler muss nur noch das NPC-Objekt mit der linken Maustaste anklicken und der Dialog wird gestartet.

In Abbildung 12 wird die Ausführung der Interaktion für beide Steuerungsarten demonstriert.

Abbildung 12: *Interaktion: Standard vs. Point-To-Click.*

4.4 Umsetzung des Quest-Systems

Mithilfe der ersten drei Videos der Playlist *Unity 5 Tutorial: Quest System* [11] konnte das konzipierte Quest-System umgesetzt werden. Die zur Umsetzung notwendigen Bestandteile werden im Folgenden erklärt.

4.4.1 Quests

Um Quests bearbeiten zu können, muss zunächst geklärt werden, was eine Quest genau ist. Hierfür bietet es sich an, eine Klasse zur Repräsentation einer Quest zu erstellen (siehe hierzu Listing 4). Eine Quest wird eindeutig über eine ID identifiziert. Zur Visualisierung der aktuellen Quest wird ein UI-Element verwendet, welches den Titel, das Ziel und den Fortschritt einer Quest festhält. Da das Quest-System so funktionieren soll, dass der Abschluss einer Quest eine neue Quest zur Annahme bereitstellt, enthält jede Quest eine weitere ID, die auf eine Folge-Quest hinweist. Die ID der ersten Quest beträgt den Wert 1. Referenziert eine Quest auf eine Quest mit ID = 0, ist keine Folge-Quest mehr zu erwarten.

```
1  public class Quest {
2      // Possible quest states
3      public enum State {
4          NOT_AVAILABLE,
5          AVAILABLE,
6          ACCEPTED,
7          COMPLETE,
8          DONE
9      }
10
11     // Some properties
12     public int id; // startet bei 1
13     public string title; // for ui
14     public State state;
15
16     // Quest goal and progress
17     public string questObjective;
18     public int currentAmount;
19     public int totalAmount;
20
21     // Quest state depended dialogues
22     public Dialogue questAcceptingDialogue;
23     public Dialogue questAcceptedDialogue;
24     public Dialogue questCompletedDialogue;
25
26     // Reference to subsequent quest
27     // - nextQuestID = 0 -> no subsequent quest
28     public int nextQuestID;
29 }
```

Listing 4: *Die Quest-Klasse in C#.*

4.4.2 Bearbeitung von Quests

Bisher wurde definiert, wie ein Quest-Objekt strukturiert ist. Zur Vollendung des Spielziels gilt es, alle Quests in den Zustand `DONE` zu überführen. Hierfür wird eine `QuestManager`-Komponente benötigt, welche darüber entscheidet, wann eine Quest zur Verfügung gestellt, angenommen, vervollständigt und schließlich abgeschlossen wird. Diese Komponente hat Einsicht auf alle existierenden Quests und auf die des Spielers. Sie ist einem GameObject hinzuzufügen, welches die Rolle eines Managers übernimmt und nie zerstört wird. Dafür bietet sich ein leeres GameObject an.

Damit dieses Quest-System funktioniert, muss die erste Quest in den Zustand `AVAILABLE` versetzt werden, während alle anderen Quests den Zustand `NOT_AVAILABLE` annehmen. Da Quests über NPC-Objekte angenommen und wieder abgeschlossen werden sollen, muss von der `QuestRequest`-Funktion des `QuestManagers` Gebrauch gemacht werden (vgl. Listing 5). Diese Funktion erlaubt es, eine Quest anzunehmen und diese somit in den Zustand `ACCEPTED` zu überführen. Das Quest-Ziel kann nun bearbeitet werden. Wurde das Quest-Ziel erfüllt, erreicht die Quest den Zustand `COMPLETE`. Bei der nächsten Interaktion wird sie in den Zustand `DONE` versetzt und somit abgeschlossen. Anschließend wird ihre referenzierte Quest verfügbar gemacht.

```
1  FUNCTION QuestRequest(Argument npc)
2      // A quest is available
3      IF npc has a quest in state AVAILABLE THEN
4          Assign this quest to the current quest variable
5          Set quest.state to ACCEPTED
6          Activate the questlog ui element
7      END IF
8
9      // A quest is active
10     IF npc has a quest in state COMPLETE THEN
11         Set quest.state to DONE
12         Assign null to the current quest variable
13         IF the quest has a chain quest THEN
14             Set chainQuest.state to AVAILABLE
15         END IF
16     END IF
17 END FUNCTION
```

Listing 5: *Regelung der Annahme und Abgabe von Quests über den QuestManager.*

Die `QuestRequest`-Funktion behandelt bisher nur die Annahme und Abgabe von Quests. Damit eine Quest vervollständigt werden kann, muss eine Funktion implementiert werden, die den Quest-Fortschritt untersucht und eine Quest in den Zustand `COMPLETE` überführen kann. Hierbei gilt es primär die `currentAmount`-Variable einer Quest anzuheben. Muss der Spieler für eine Quest beispielsweise eine bestimmte Anzahl an Gegner-Objekten besiegen, muss

bei deren Tod die `AddQuestItem`-Funktion über den QuestManager aufgerufen werden (vgl. Listing 6).

```
1  FUNCTION AddQuestItem(Argument questObjective,
2                        Argument itemAmount)
3      Find the quest with quest object equal to
4          questObjective
5      Add itemAmount to the quest.currentAmount
6      IF quest.currentAmount <= quest.totalAmount THEN
7          Message: quest.currentAmount + " / "
8              +  quest.totalAmount
9      END IF
10     IF quest.currentAmount == quest.totalAmount THEN
11         Set quest.state to COMPLETE
12         Message: "Quest completed!"
13     END IF
14 END FUNCTION
```

Listing 6: *Bearbeitung des Quest-Fortschritts über den QuestManager.*

4.5 Umsetzung des Kampfsystems

Während der Bearbeitung von Quests, soll das Spieler-Objekt in Kämpfe mit gegnerischen Objekten verwickelt werden. Zudem wird das Besiegen von Gegnern oft Gegenstand der Quests selbst sein. Damit das Kämpfen jedoch erst ermöglicht wird, müssen für das Spieler-Objekt sowie für die Gegner-Objekte jeweils ein Kampfverhalten umgesetzt werden. Es geht darum, dass sowohl Spieler-Objekt als auch Gegner-Objekt Angriffe ausführen können, die Schadenspunkte verrichten. Ausgeteilter Schaden führt zur Reduktion von Lebenspunkten. Dasjenige Objekt, das die Lebenspunkte seines Rivalen auf den Wert 0 herabfallen lässt, geht den Kampf siegreich hervor. Zur visuellen Repräsentation des Kampfverhaltens bietet es sich an, Animationen zu verwenden, die über Unitys Animationssystem Mecanim geregelt werden.

Im Folgenden soll auf die Umsetzung der Kampfverhalten beider Spielfiguren-Arten eingegangen werden.

4.5.1 Verhalten des Spieler-Objekts

Für das Verhalten des Spieler-Objekts müssen Aktionen implementiert werden, die über Nutzereingaben und mögliche zusätzliche Bedingungen ausgelöst werden können. Außerdem sollen diese visuell dargestellt werden. Zur Repräsentation des Spieler-Objekts wurde ein humanoides Modell verwendet, welches im „RPG Hero PBR HP Polyart“-Asset enthalten ist. Die eingesetzten Animationen wurden vom „RPG Character Mecanim Animation Pack FREE“-Asset aus dem Unity Asset Store entnommen. Für die Blocken-Animationen wurden passende Animationen auf der Plattform mixamo [5] herausgesucht. Bevor animiert werden konnte, mussten

ein paar Änderungen an dem Modell vorgenommen werden, um die verwendeten Animationen auf das Spieler-Objekt richtig anwenden zu können. Da das Spieler-Modell standardmäßig ein Einhandschwert sowie ein Schild in jeweils einer Hand hält, war es notwendig, das Modell an die verwendeten Animationen anzupassen. Dazu mussten das Schild-Objekt entfernt und das Schwert-Objekt größer skaliert werden. Damit die Zweihand-Animationen und das Modell miteinander harmonierten, musste das Schwert-Objekt noch entsprechend ausgerichtet werden.

Animationen. Jede Aktion des Spielers wird durch mindestens eine Animation repräsentiert. Das Spieler-Objekt verfügt insgesamt über 17 verschiedene Animationen, die in Mecanim jeweils als eigenständige Zustände dargestellt sind. Eine Übersicht über alle verfügbaren Animationen mitsamt einer kurzen Beschreibung sind für den waffenlosen Zustand in Tabelle 2 und für den Zweihand-Modus in Tabelle 3 einzusehen. Das Erreichen dieser Zustände erfolgt über das Abfangen von Inputs und Bedingungen.

Waffenloser Modus	
AnimationState	**Beschreibung**
Idle	Die Spielfigur ruht und zeigt nur leichte Bewegungen.
Moving	Die Spielfigur bewegt sich fort.
Dodging	Die Spielfigur weicht mittels einer Ausweichrolle aus. Beim Betreten des Zustands wird Ausdauer verbraucht.
EquipWeapon	Die Spielfigur rüstet ihr Zweihandschwert aus. Ein AnimationEvent an geeigneter Stelle blendet das Schwert-Objekt ein.
Damaged	Der Spielfigur wurde Schaden zugeteilt und zuckt dabei zusammen. Dem Spieler-Objekt werden die Lebenspunkte reduziert.
Dead	Die Spielfigur stirbt und fällt dabei zu Boden.

Tabelle 2: *AnimationStates im waffenlosen Modus.*

Zu Spielbeginn befindet sich der Spieler im waffenlosen Modus. Das Zweihandschwert ist dabei ausgeblendet. Um den Modus zu wechseln, muss der nötige Input zum Waffenwechsel betätigt werden. Über ein in der Animation gesetztes AnimationEvent wird eine Funktion aufgerufen, die das Schwert-Objekt aktiviert bzw. deaktiviert. Sofern das Spieler-Objekt nicht angegriffen oder eine spezifische Aktion ausgeführt wird, verbleibt die Spielfigur im Idle-Zustand des jeweiligen Modus. Der Moving-Zustand wird erreicht, wenn der Spieler den dafür notwendigen Input gedrückt hält. Lässt er den Input wieder los, wird wieder in den Idle-Zustand zurückgegangen. Befindet sich das Spieler-Objekt im Moving-Zustand und betätigt den Input zum Ausweichen, wird in den Zustand Dodging übergegangen. Der Damaged-Zustand wird erreicht, wenn das Spieler-Objekt Schaden

Zweihand-Modus	
AnimationState	**Beschreibung**
Idle	Die Spielfigur ruht und zeigt nur leichte Bewegungen.
Moving	Die Spielfigur bewegt sich fort.
Dodging	Die Spielfigur weicht mittels einer Ausweichrolle aus. Beim Betreten des Zustands wird Ausdauer verbraucht.
UnequipWeapon	Die Spielfigur entrüstet ihr Zweihandschwert. Ein AnimationEvent an geeigneter Stelle blendet das Schwert-Objekt aus.
Damaged	Der Spielfigur wurde Schaden zugeteilt und zuckt dabei zusammen. Dem Spieler-Objekt werden die Lebenspunkte reduziert.
Dead	Die Spielfigur stirbt und fällt dabei zu Boden.
Attacking: - Hit1 - Hit2 - Hit3	Die Spielfigur führt Angriffe aus. Es kann eine Angriffsfolge von bis zu drei verschiedenen Angriffen hintereinander ausgeführt werden. Ein AnimationEvent bestimmt, an welcher Stelle der Animation Schaden ausgeteilt werden kann.
Blocking: - Idle - Damaged	Die Spielfigur führt das Blocken aus. Während des Blockens zeigt die Spielfigur nur leichte Bewegungen in ihrer Pose. Wird ihr Schaden zugefügt, wird der Schaden annulliert und die Spielfigur zuckt zusammen. Ein erfolgreicher Block verdoppelt den Schaden des nächsten Angriffs.

Tabelle 3: *AnimationStates im Zweihand-Modus.*

erleidet. Stirbt das Spieler-Objekt, wird in den Zustand Dead gewechselt. Aus diesem Zustand kommt er zunächst nicht mehr raus. Lässt er sich über das daraufhin eingeblendete Menü wiederbeleben, wird das Spieler-Objekt in den Idle-Zustand des waffenlosen Modi versetzt.

Alle bisher genannten Zustandsübergänge gelten auch für den Zweihand-Modus. Im Zweihand-Modus hat der Spieler aber noch zusätzlich die Möglichkeit, anzugreifen oder zu blocken. Auch hierfür werden entsprechende Inputs betätigt werden müssen, um die Zustände Attacking und Blocking zu erreichen. Diese beiden Zustände führen beim Verlassen wieder in den Idle-Zustand.

Ausführung der Aktionen. Um einzelne Aktionen mitsamt ihrer Animationen in Gang zu setzen, ist es notwendig die Parameter per Skript zu manipulieren, welche für die Zustandsübergänge verantwortlich sind. Hierfür wird eine `HandleActions`-Funktion benötigt, die die einzelnen Inputs des Spielers abfängt und dafür sorgt, dass die entsprechende Animation ausgeführt wird (siehe

Listing 7). Zu Spielbeginn befindet sich das Spieler-Objekt im waffenlosen Zustand.

```
FUNCTION HandleActions()
    IF player requests interaction THEN
        IF a npc object is nearby that can be
        interacted with THEN
            Execute npc.TriggerDialogue()
    ELSE IF player requests using health potion THEN
        IF health potion cooldown equals 0 THEN
            Adjust player's health by 60 percent of his
                max health
            Reset cooldown
            Start a coroutine function that counts down
                the cooldown
        ELSE
            Message: "Heal potion not ready yet!"
        END IF
    ELSE IF player requests toggling weapon THEN
        IF toggling weapon is possible THEN
            IF player is unarmed THEN
                Trigger EquipWeapon animation
                // player is armed now
            ELSE
                Trigger UnequipWeapon animation
                // player is unarmed now
            END IF
        END IF
    ELSE IF player requests blocking THEN
        IF player is not unarmed THEN
            Trigger Blocking animation
        END IF
    ELSE IF player requests attacking THEN
        IF player is not unarmed THEN
            IF player has enough stamina THEN
                Trigger Attacking animation
                // AnimationEvent on animation
                // triggers a function that
                // damages all enemies in
                // front of the player
            ELSE
                Message: "Not enough stamina!"
            END IF
        END IF
    ELSE IF player requests moving THEN
        IF player can move THEN
            Trigger Moving animation
            IF player requests sprinting THEN
                IF player has enough stamina THEN
                    Set animation speed to fast
```

```
48                  ELSE
49                      Set animation speed to normal
50                      Message: "Not enough stamina!"
51                  END IF
52              ELSE
53                  Set animation speed to normal
54              END IF
55              IF player is not dodging THEN
56                  IF player requests dodging THEN
57                      IF player has enough stamina THEN
58                          Trigger Dodging animation
59                          Multiply a speed factor to the
60                              current movement direction
61                              for faster movement while
62                              dogding
63                      ELSE
64                          Message: "Not enough stamina!"
65                  ELSE
66                      Calculate new movement direction
67                  END IF
68                  Set player's forward vector to normalized
69                      movement direction multiplied by
70                      delta time
71                  Let player move into movement direction
72                      multiplied by movement speed and
73                      delta time
74              END IF
75          ELSE
76              Trigger Idle animation
77              Reset movement speed // error handling
78          END IF
79      END IF
80      // No action requested by player
81  END FUNCTION
```

Listing 7: *Ausführung der Aktionen über eingehende Inputs.*

4.5.2 Standard vs. Point To Click

Die zwei Steuerungsarten Standard und Point-To-Click lassen sich im Hinblick auf die Ausführung von Aktionen unterschiedlich bedienen. Im Folgenden soll auf die sich in der Ausführung unterscheidenden Aktionen eingegangen werden. Außerdem soll auf etwaige Probleme eingegangen werden, die sich bei der Umsetzung der Steuerungsart Point-To-Click hinsichtlich des Kampfsystems ergeben haben.

Fortbewegung. Eine sehr entscheidende Aktion stellt die Fortbewegung dar. Für die Standard-Steuerung bieten sich die Tasten W, A, S und D an. Diese Tasten finden in den meisten Tastatur-basierten Spielen Anwendung. Während A und D eine horizontale Bewegung simulieren, stellen W und S eine vertikale Bewegung

dar. Zusätzlich soll die Fortbewegung Kamera-relativ funktionieren, das heißt, die x- und z-Bewegungsvektoren errechnen sich anhand der x- und z-Vektoren des Kamera-Objeks. Da die Kamera unter Umständen schräg steht, wird zur Berechnung des z-Bewegungsvektors der z-Vektor der Kamera auf die Horizontalebene projiziert. Dies geschieht, indem die y-Komponente des Vektors auf 0 gesetzt wird. Anschließend wird der Vektor normalisiert. Der x-Bewegungsvektor ist das normalisierte Ergebnis des Kreuzprodukts zwischen Standard-Up-Vektor (`Vector3.up`) und dem bereits errechneten z-Bewegungsvektor. Die Bewegungsrichtung der Spielfigur berechnet sich über die Summe aus horizontaler und vertikaler Bewegung. Die Eingaben des Nutzers entscheiden, ob eine horizontale oder vertikale Bewegung stattfindet. Entsprechend ändert sich die Bewegungsrichtung. Zusätzlich muss eine Gravitation auf die Spielfigur einwirken, wenn sie sich über das hügelige Areal hinfort bewegen soll. Dies geschieht, indem das Spieler-Objekt zusätzlich noch um einen Down-Vektor (`-Vector3.up`) nach unten verschoben wird, sobald sich die Spielfigur nicht auf dem Boden befindet. Eine `gravity`-Variable bestimmt, wie schnell sich das Objekt zum Boden hin bewegt.

Zur Fortbewegung mit der Steuerungsart Point-To-Click muss der Spieler den Mauscursor über das Areal bewegen und dann die linke Maustaste gedrückt halten. Über die Verwendung von Collidern und des Raycasting-Verfahrens konnte so ein Punkt in der begehbaren Fläche bestimmt werden, in dessen Richtung sich die Spielfigur bewegen soll. Dieser Punkt wird dann als aktuelles Target-Objekt gesetzt, in dessen Richtung sich das Spieler-Objekt dann hinbewegt. Der Bewegungsvektor errechnet sich dementsprechend aus der Position des Spielers und der Target-Position. Außerdem kann sich der Spieler auch auf Gegner-Objekte hinzu bewegen, indem er den Cursor auf ein Gegner-Objekt bewegt und dann die linke Maustaste gedrückt hält. Der Gegner wird dann als Target gesetzt. Ein Target bleibt solange erhalten, bis der Spieler die linke Maustaste wieder loslässt.

Angreifen. Bei der Standard-Steuerung musste der Spieler nur die linke Maustaste hintereinander kurz betätigen, um kontinuierlich anzugreifen. Dies wurde beim Point-To-Click komplett verändert. Zunächst muss der Spieler das Gegner-Objekt als Target festlegen. Befindet sich das Spieler-Objekt nah genug zum Gegner, führt es innerhalb eines vorgegebenen Zeitintervalls einzelne Angriffe aus. Es wird solange angegriffen, bis das Target verlorengegangen ist, wie es beispielsweise beim Tod des Gegners oder beim Loslassen der linken Maustaste der Fall ist. Auch hierfür wurden Collider und Raycasts verwendet.

Schwert ausrüsten und entrüsten. Zum Ausrüsten bzw. Entrüsten der Waffe musste der Spieler bei der Standard-Steuerung die Taste Q betätigen. Die Point-To-Click-Steuerung bietet diese Taste zur manuellen Ausführung auch noch an. Zusätzlich wird eine automatische Abhilfe geleistet, wenn diese Aktion zur Ausführung anderer Aktionen notwendig wird. Ist zum Beispiel ein Gegner als aktuelles Target gesetzt und die Waffe nicht ausgerüstet, wird das Schwert angelegt. Soll mit einem NPC-Objekt interagiert werden, legt das Spieler-Objekt zunächst die Waffe ab.

Probleme bei der Umsetzung der Steuerungsart Point-To-Click. Es war hauptsächlich problematisch, einen geeigneten Kamera-Offset zu finden, mit dem es möglich war, nur über das Heran- bzw. Herauszoomen der Kamera alles in der Spielwelt einzusehen. Aus diesem Grund wurde der Point-To-Click-Steuerung zusätzlich die Möglichkeit hinzugefügt, die Kamera zu rotieren. Eigentlich ist dies für ein Point-To-Click-basiertes Spiel eher untypisch, aber auf diese Weise ist es für den Spieler viel mehr einsichtig, wohin er die Spielfigur bewegt. Es dient daher nur als eine Notlösung. Die Kamera-Rotation konnte ausgeführt werden, indem der Spieler die mittlere Maustaste gedrückt hält und dabei den Mauscursor mindestens horizontal fortbewegt. Die Kamera richtete sich dann entsprechend der Mausbewegung aus. So war es dem Spieler immerhin noch möglich, das Spiel mit dieser Steuerungsart gut spielen zu können.

4.5.3 Verhalten der Gegner-Objekte

Für die Gegner-Objekte gilt es ebenfalls, Aktionen zu implementieren. Doch der feine Unterschied zwischen diesen Objekten und dem Spieler-Objekt ist, dass der Spieler keinen Einfluss auf ihre ausgeführten Aktionen hat. Es wird ein Verhaltensmuster implementiert werden müssen, welches das Gegner-Objekt bei der Ausführung seiner Aktionen steuert. Demnach werden Zustände definiert, die die einzelnen Aktionen regeln. Das Abfangen von Bedingungen wird hierbei besonders wichtig. Die Animationen stammen aus dem „Level 1 Monster Pack“-Asset, wo auch ihre Charaktermodelle vorliegen. Es wurden insgesamt vier verschiedene Modelle mit entsprechenden Animationen für die Gegner verwendet. Zusätzlich wurde zur Absolvierung einer Quest ein modifiziertes Spieler-Objekt verwendet, welches auf demselben Verhaltensmuster der Gegner basiert.

Verhaltensmuster. Es gibt insgesamt fünf Zustände, die ein Gegner-Objekt erreichen kann: `PATROLLING`, `OBSERVING`, `FOLLOWING`, `ATTACKING` und `RETURNING`. Diese wurden in der Konzeption bereits präzise beschrieben. Eine `HandleState`-Funktion regelt, in welchen Zustand der Gegner bei einer gegebenen Situation versetzt werden muss (vgl. Listing 8).

```
1  FUNCTION HandleState()
2      Calculate distance to player object
3      IF enemy object is too far away from spawn
4      area THEN
5          Set enemy.state to RETURNING
6      ELSE IF player object is dead THEN
7          Set enemy.state to PATROLLING
8      ELSE IF enemy type is aggressive
9      and distance to player is less than
10     attack distance THEN
11         Set enemy.state to ATTACKING
12     ELSE IF enemy type is aggressive
13     and distance to player is less than
14     follow distance THEN
```

```
15          Set enemy.state to FOLLOWING
16      ELSE IF enemy type is aggressive
17      and distance to player is less than
18      observe distance THEN
19          Set enemy.state to OBSERVING
20      ELSE
21          Set enemy.state to PATROLLING
22      END IF
23  END FUNCTION
```

Listing 8: *Regeln der Zustände der Gegner.*

Zusätzlich wird bei der Initialisierung eine Coroutine `HandleActions` gestartet, die die Aktionen dem Zustand entsprechend ausführt (siehe Listing 9). Diese Coroutine wird unterbrochen, wenn das Gegner-Objekt stirbt.

```
1   COROUTINE HandleActions()
2       WHILE enemy is not dead
3           IF enemy.state equals RETURNING THEN
4               IF enemy type is passive THEN
5                   Set enemy to passive behavior again
6               END IF
7               Make enemy move to a random patrol point
8                   around his spawn area
9               Trigger Moving animation
10          ELSE IF enemy.state equals PATROLLING THEN
11              Execute HandlePatrol()
12          ELSE IF enemy.state equals OBSERVING THEN
13              Look at player.position
14              Trigger Idle animation
15          ELSE IF enemy.state equals FOLLOWING THEN
16              Make enemy move to player.position
17              Trigger Moving animation
18          ELSE IF enemy.state equals ATTACKING THEN
19              Look at player.position
20              Trigger Idle animation
21              IF player is not dead
22              and enemy is able to attack THEN
23                  Trigger Attack animation
24                  // AnimationEvent on animation
25                  // triggers a function that will
26                  // apply damage to the player
27              END IF
28          END IF
29      END WHILE
30  END COROUTINE
31
32  COROUTINE HandlePatrol()
33      WHILE enemy.state equals PATROLLING
34          Choose a random patrol point around the
```

```
35              spawn area
36          Move to the patrol point for a random
37              amount of seconds
38          Wait for a random amount of seconds
39      END WHILE
40  END COROUTINE
```

Listing 9: *Steuern der Aktionen der Gegner.*

4.5.4 Simulation der Kämpfe

Um einen Kampf in Gang zu setzen, müssen sich Spieler-Objekt und Gegner-Objekt nah genug stehen. Das Verhalten des Gegners kann dazu beitragen, sich dem Spieler selbstständig zu nähern, primär ist dafür jedoch der Spieler selbst verantwortlich. Das Kampfgeschehen wird über die Ausführung einzelner Aktionen beider Parteien simuliert. Greift der Spieler an, wird geprüft, ob der Gegner vom Angriff getroffen wurde oder nicht. Das Gleiche gilt auch umgekehrt. Über AnimationEvents, die an bestimmte Stellen der Angriffsanimationen gesetzt wurden, kann festgelegt werden, wann ein Angriff theoretisch treffen könnte. Anschließend ruft das jeweilige AnimationEvent eine `Hit`-Funktion auf, die überprüft, ob sich die zwei Rivalen nah genug stehen, sodass der Angriff auch tatsächlich getroffen hat. Für die Gegner gibt es nur ein rivalisierendes Objekt, nämlich das Spieler-Objekt. Umgekehrt gibt es für das Spieler-Objekt mehrere rivalisierende Objekte, wovon einige von ihnen unter Umständen gleichzeitig an einem Kampf beteiligt sein können. Stehen dem Spieler demnach mehrere Objekte so nah genug, dass ein ausgehender Angriff all diese Objekte treffen würde, muss der Schaden auch jedem Objekt ausgeteilt werden. Außerdem darf dem Spieler kein Schaden zugefügt werden, wenn er sich in der Blocken-Animation befindet. Blockt der Spieler einen gegnerischen Treffer erfolgreich, verrichtet sein nächster Angriff doppelten Schaden. In den Listings 10 und 11 werden die `Hit`-Funktionen beider Spielfiguren-Arten umgesetzt.

```
1   FUNCTION Hit()
2       FOR ALL enemy in enemies DO
3           Calculate distance to enemy
4           IF enemy is not dead
5           and enemy is in attack distance
6           and enemy is in front of the player THEN
7               IF player has blocked successfully before THEN
8                   Adjust enemy.health by -2 * damage
9                   Trigger enemy's Damaged animation
10              ELSE
11                  Adjust enemy.health by -damage
12                  Trigger enemy's Damaged animation
13              END IF
14          END IF
15      END FOR
```

```
16 END FUNCTION
```

Listing 10: *Landen von Treffern vonseiten des Spieler-Objekts.*

```
1  FUNCTION Hit()
2      Calculate distance to player
3      IF player is not dead
4      and player is not dodging
5      and player is in attack distance
6      and player is in front of the enemy THEN
7          IF player is blocking THEN
8              // No damage dealt
9              Trigger player's Blocking-Damaged animation
10             Add a delay time to enemy's next attack time
11         ELSE
12             Adjust player.health by -damage
13             Trigger player's Damaged animation
14         END IF
15     END IF
16 END FUNCTION
```

Listing 11: *Landen von Treffern vonseiten des Gegner-Objekts.*

5 Evaluation

Dieses Kapitel befasst sich mit der Evaluation des Spiels. Es wird dargelegt, welches Ziel diese Evaluation verfolgt und wie dieses Ziel über die Durchführung dieser Nutzungsstudie erreicht wurde. Das Ergebnis dieser Evaluation soll einen Ausblick für die zukünftige Weiterentwicklung des Spiels geben.

5.1 Durchführung der Nutzungsstudie

Um den derzeitigen Prototypen des Spiels bewerten zu können, ist es notwendig, die Spieler selbst einzubeziehen. Da das Spiel so konzipiert wurde, dass es jeder Spieler unabhängig von seinen Spielerfahrungen erlernen und somit auch absolvieren kann, wurden mögliche Probanden sowohl aus dem Studierendenkreis am Campus Koblenz der Universität Koblenz-Landau als auch aus meiner Verwandtschaft für diese Nutzungsstudie zur Teilnahme motiviert.

Die einzelnen Tests mit den Probanden wurden in meiner Anwesenheit durchgeführt. Zu Beginn eines Tests wurde dem Proband kurz erklärt, um wessen Spielgenre es sich handelt, was den Proband erwarten wird und was das Ziel des Spiels ist. Anschließend startete der Proband das Spiel und folgte dessen Anweisungen. Während des Spielens notierte ich mir geschilderte Anmerkungen oder Verbesserungsvorschläge, nicht genutzte Spielfeatures und ungewolltes Verhalten des

Spiels. Hatte der Proband das Spiel vollendet, wurde er zur Umfrage übergeleitet, die den Fragebogen (siehe dazu Kapitel 5.2) beinhaltete. Zur besseren Handhabung wurde auf einen schriftlichen Fragebogen verzichtet. Stattdessen wurde der Fragebogen als elektronische Umfrage über Microsoft Forms[3] betrieben. Über einen Link zur Umfrage konnte diese gestartet werden. Probanden konnten auf diese Weise sehr einfach ihre Eintragungen und Antworttexte vornehmen. Außerdem konnte garantiert werden, dass für diese Studie zwingend erforderliche Fragen auch wirklich beantwortet werden. Die Umfrage ist demnach erst dann vollendet, sobald alle erforderlichen Fragen beantwortet wurden. Während der Bearbeitung des Fragebogens hatte der Proband jederzeit die Möglichkeit, Fragen zu stellen, falls Unsicherheiten beim Verständnis der Fragen aufgetreten sind.

Insgesamt haben 14 Probanden an der Nutzungsstudie teilgenommen und diese erfolgreich bis zum Ende bearbeitet.

5.2 Fragebogen

Der Fragebogen (vgl. A.1 im Anhang) wurde den Probanden nach der Vollendung des Spiels zur Bearbeitung vorgelegt. Er gliedert sich in insgesamt vier Abschnitte. Die Beantwortung der Fragen in den ersten drei Abschnitten erfolgt über eine Likert-Skala mit den vier Antwortmöglichkeiten „Stimmt absolut“, „Stimmt eher“, „Stimmt eher nicht“ und „Stimmt gar nicht“. Die Tendenz zum „Ankreuzen der Mitte“ soll damit vermieden werden. Pro Frage müssen sich die Probanden für eine auf sie zutreffende Antwortmöglichkeit entscheiden. Der letzte Abschnitt hingegen verfügt über offen gehaltene Fragen, zu denen sich die Probanden frei äußern können. Im Folgenden soll kurz auf die einzelnen Abschnitte eingegangen werden.

Der erste Abschnitt befasst sich mit der Untersuchung der Bedienbarkeit der zwei Steuerungsarten „Standard“ und „Point To Click“. Dabei soll untersucht werden, wie gut sich diese beiden Steuerungsarten bedienen lassen und welche von beiden sich für dieses Spiel besser eignet. Des Weiteren soll vor allem herausgefunden werden, ob noch Optimierungsbedarf für die Steuerungsarten besteht.

Da ich mich während der Umsetzung des Spiels sehr lange mit der Suche nach optisch gut passenden Animationen für die einzelnen Spielfiguren beschäftigt habe, soll im zweiten Abschnitt überprüft werden, ob die Animationen visuell überzeugend waren. Die Spielfiguren wurden in drei Gruppen aufgeteilt: Spieler, Gegner und NPCs. Hierbei wurde der Proband aufgefordert, die Animationen dieser drei Gruppen seiner persönlichen Wahrnehmung entsprechend zu beurteilen.

Die Benutzerfreundlichkeit des Spiels über geeignete Hilfestellungen und Visualisierungen im Spiel selbst wird im dritten Abschnitt ermittelt. Probanden sollen für sich selbst beurteilen, ob sie mit den zu erledigenden Aufgaben gut zurechtgekommen sind und ob die visuellen Hilfestellungen bei der Aufgabenbewältigung und beim Erlernen der Spielsteuerung geholfen haben.

[3] Microsoft Forms ist eine Anwendung von Microsoft, mit dessen Hilfe sich insbesondere elektronische Umfragen erstellen lassen. Weitere Informationen zur Anwendung unter: https://support.office.com/de-de/forms

Im letzten, eher offen gehaltenen Abschnitt soll herausgefunden werden, welche Probleme sich im Spiel ergeben haben und was man an dem Spiel künftig noch verbessern kann. Die Probanden können aufzählen, was ihnen gefallen oder nicht gefallen hat. Außerdem müssen die Probanden explizit schildern, wie sie den Schwierigkeitsgrad der Kämpfe einschätzen und welche der beiden Steuerungsarten sie für das Spiel selbst bevorzugen würden. Probanden haben auch die Möglichkeit, Verbesserungsvorschläge zu nennen. Zu guter Letzt werden die Probanden darum gebeten, Auskunft darüber zu geben, ob sie Spaß beim Spielen empfunden haben und wie sie das Spiel im Allgemeinen anhand von fünf Sternen bewerten würden. Der Bewertungsmaßstab erstreckt sich dabei von „Sehr schlecht“ (kein Stern) bis „Sehr gut“ (fünf Sterne). Je mehr Sterne vergeben werden, umso besser wurde das Spiel empfunden.

5.3 Auswertung des Fragebogens

Dieses Kapitel befasst sich mit der Auswertung der Umfrage. Die Ergebnisse der Auswertung sind im Anhang A.2 einzusehen. Im Folgenden wird auf die Ergebnisse der einzelnen Abschnitte eingegangen.

5.3.1 Bedienbarkeit der Steuerungsarten

Den Beurteilungen der Probanden zufolge wurde die Standard-Steuerung im Gegensatz zur Point-To-Click-Steuerung deutlich besser empfunden. Im Folgenden soll auf die einzelnen Beurteilungen zuzüglich der Antworten des letzten Abschnitts der Umfrage eingegangen werden.

Standard-Steuerung. Insgesamt wurde die Standard-Steuerung von den Probanden größtenteils positiv bewertet (vgl. Abbildungen 13 und 14). Mit der Fortbewegung und dem Bekämpfen einzelner Gegner-Figuren hatten 85,7% der Probanden keine Probleme. Selbst das Bekämpfen mehrerer Gegner gleichzeitig schien für den Großteil der Probanden relativ gut machbar gewesen zu sein. 3 von 14 Probanden schien dies jedoch eher schwierig gefallen zu sein. Die Schwierigkeiten könnten möglicherweise mit der Reaktionsfähigkeit des Spiels auf Nutzereingaben zusammenhängen, denn 28,6% der Probanden empfanden den Übergang von Nutzereingabe zur Reaktion im Spiel eher weniger flüssig. Die Erreichbarkeit und das Ausführen einzelner Aktionen wurde größtenteils sehr positiv beurteilt. Während die Probanden spielten, konnte ich jedoch beobachten, dass für einige Probanden der Weg der Hand zur Taste H für den Einsatz des Heiltranks zu lang ist. Die Spielfigur starb oft aus dem Grund, dass der Proband die H-Taste zu spät erreicht oder sich manchmal verdrückt hatte. Einzelne Anmerkungen seitens der Probanden bestätigten mich in meiner Beobachtung. Zur besseren Bedienbarkeit der Steuerung müsste man daher diese Aktion auf eine zur Hand näherliegende Taste verlegen. Einige Probanden gaben im Bereich der Kritik noch an, dass die Hit-Boxen in der Standard-Steuerung zu klein war oder die Angriffsreichweite kürzer erschien als sie tatsächlich war. Dies machte die Gegner etwas schwieriger zu treffen. Zur Ver-

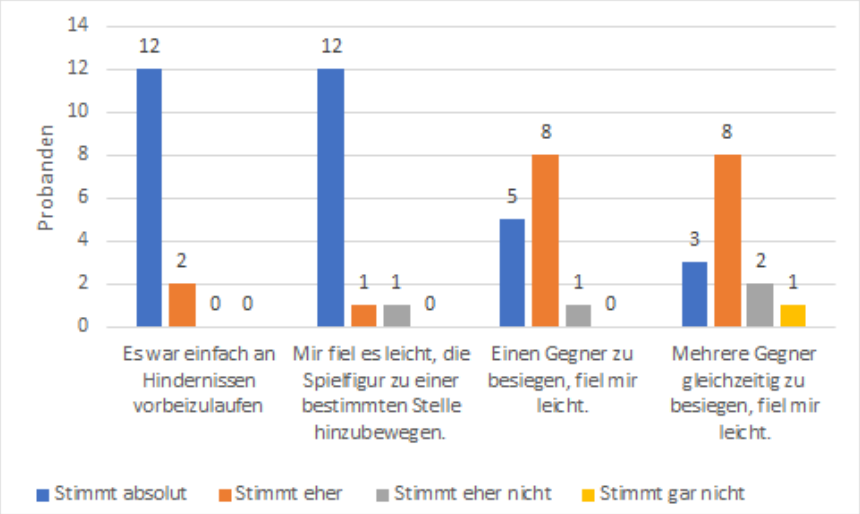


Abbildung 13: *Teil 1: Beurteilung der Standard-Steuerung anhand vorgegebener Aussagen.*

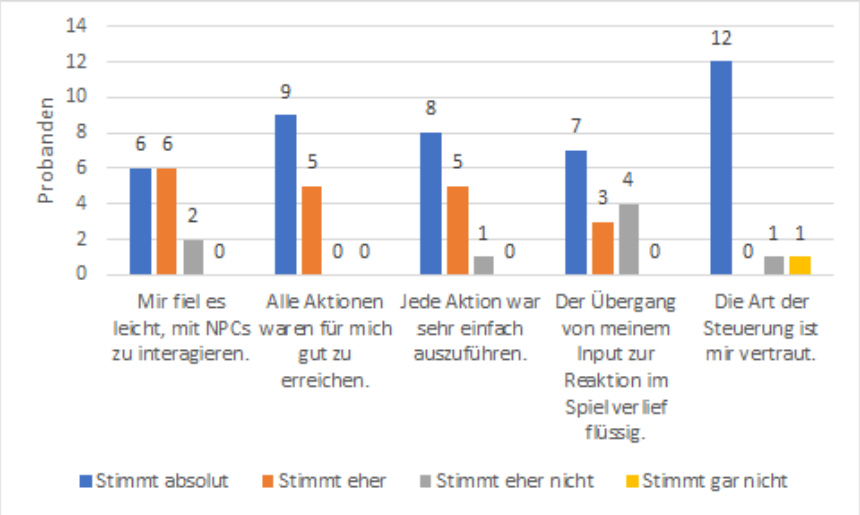


Abbildung 14: *Teil 2: Beurteilung der Standard-Steuerung anhand vorgegebener Aussagen.*

besserung der Bedienbarkeit könnte man daher auch die Angriffsreichweite und die Hit-Boxen anpassen. Sonst schien die Auslegung der Tasten im Allgemeinen den Probanden vertraut gewesen zu sein und dies ist wahrscheinlich ein Indikator dafür, dass sie mit der Steuerung überwiegend gut zurechtgekommen sind. Das Interagieren mit den NPCs könnte man auch noch besser optimieren. Zwar empfand der Großteil der Probanden, dass es für sie mindestens eher leicht gefallen ist, jedoch konnte ich bei jedem Probanden beobachten, dass sie die Spielfigur oft in ih-

rer Position und Ausrichtung korrigieren mussten, damit eine Interaktion mit dem NPC möglich wurde. Offenbar fühlten sich die Probanden nicht dadurch gestört. Trotzdem würde ich unabhängig von den Beurteilungen den Optimierungsbedarf an dieser Stelle nicht außer Acht lassen.

Point-To-Click-Steuerung. Im Vergleich zur Standard-Steuerung stieß die Point-To-Click-Steuerung eher auf Probleme (vgl. Abbildungen 15 und 16). Das

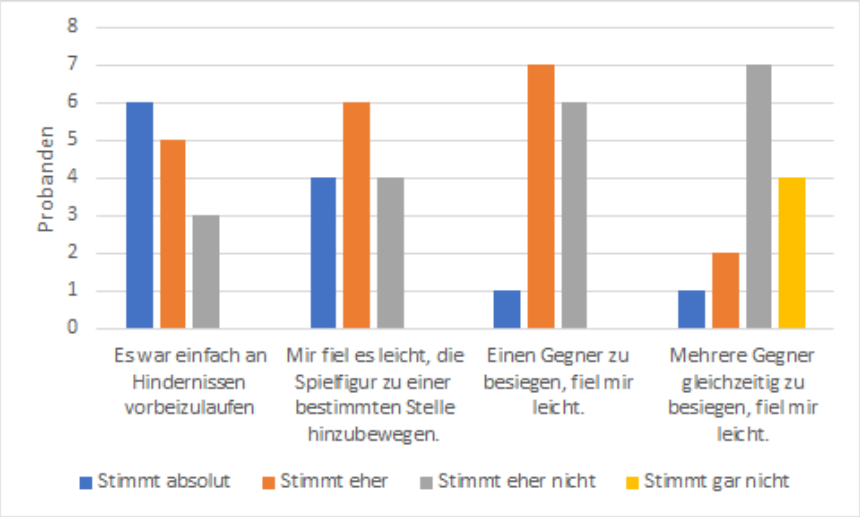


Abbildung 15: *Teil 1: Beurteilung der Point-To-Click-Steuerung anhand vorgegebener Aussagen.*

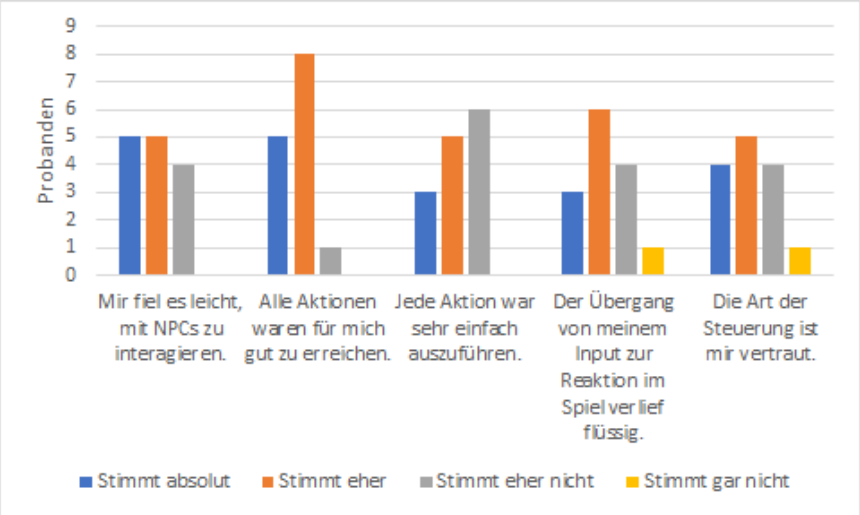


Abbildung 16: *Teil 2: Beurteilung der Point-To-Click-Steuerung anhand vorgegebener Aussagen.*

Fortbewegen der Spielfigur schien zwar noch gut machbar gewesen zu sein, aber die Probleme zeigten sich vermehrt in den Kämpfen. Während es noch eher machbar war, einen einzelnen Gegner zu besiegen, fiel es den Probanden viel schwerer, mehrere Gegner gleichzeitig erfolgreich zu bekämpfen. Hauptsächlich lag es an den zu kleinen Hit-Boxen, wie sich einige Probanden äußerten oder als Kritik niederschrieben. Gegner-Objekte waren nur schwer anzuvisieren und das Spieler-Objekt stand dabei auch viel zu oft im Weg. Leider wurde dieser Fall in der Implementation nicht sinnvoll abgefangen. Dies machte die Angreifen-Aktion viel schwieriger zu erreichen und auszuführen. Es besteht daher absolute Dringlichkeit, den Fall mit dem Cursor über dem Spieler-Objekt richtig abzudecken und die Hit-Boxen für diese Steuerungsart zu vergrößern. Mit NPCs zu interagieren, schien für die Probanden in ihrer Ausführbarkeit in Ordnung gewesen zu sein. Es wurde kritisiert, dass man als Spieler nur schwer einsehen konnte, ab welcher Entfernung interagiert werden konnte. Man empfand es sinnvoller, dass am Cursor erst dann das Interaktionssymbol erscheint, wenn das Spieler-Objekt nah genug zum NPC steht und der Cursor über dem NPC-Objekt ist. Eigentlich wollte ich den Übergang von Bewegung zur Interaktion anders gestalten, was mir aus Zeitmangel leider nicht mehr möglich gewesen war. Dennoch gilt es, auch die Interagieren-Aktion noch weiter auszubauen. Auch für diese Steuerungsart sollte die Reaktionsfähigkeit des Spiels auf Nutzereingaben noch verbessert werden.

5.3.2 Animationen

Die Beurteilungen der Animationen aller Spielfiguren hielt sich überwiegend positiv (vgl. Abbildungen 17, 18 und 19).

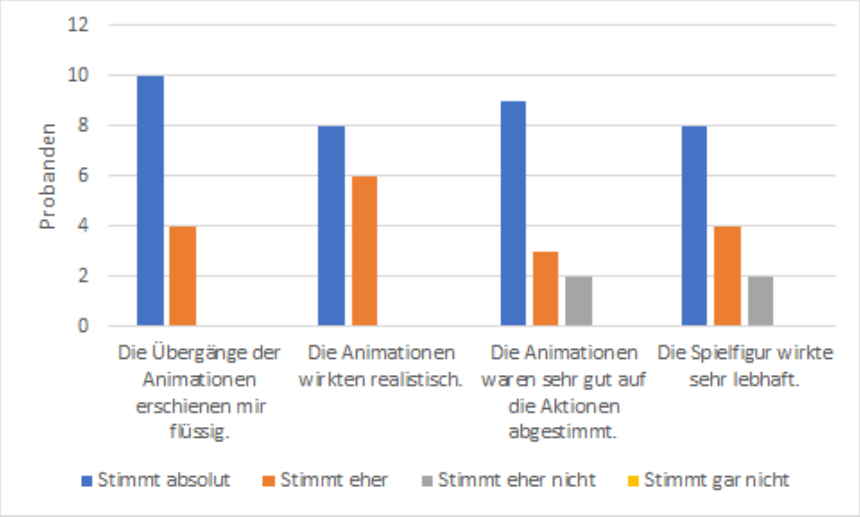


Abbildung 17: *Beurteilung der Animationen des Spieler-Objekts.*

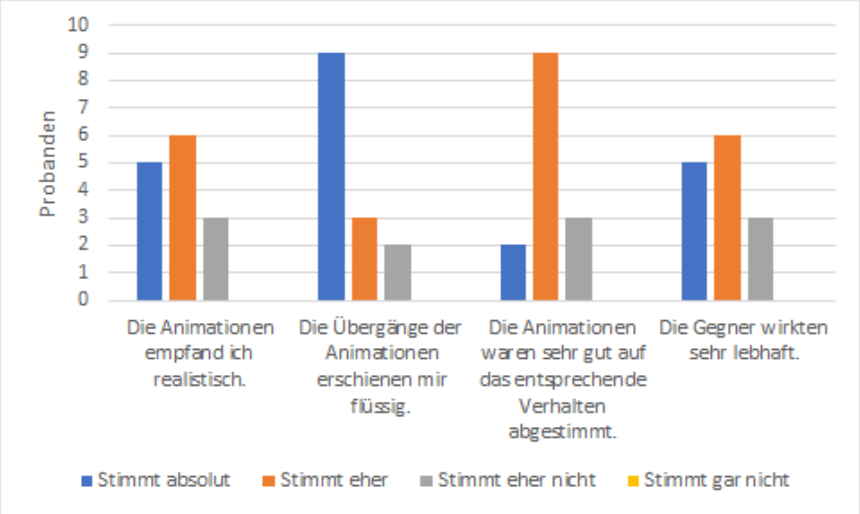


Abbildung 18: *Beurteilung der Animationen der Gegner-Objekte.*

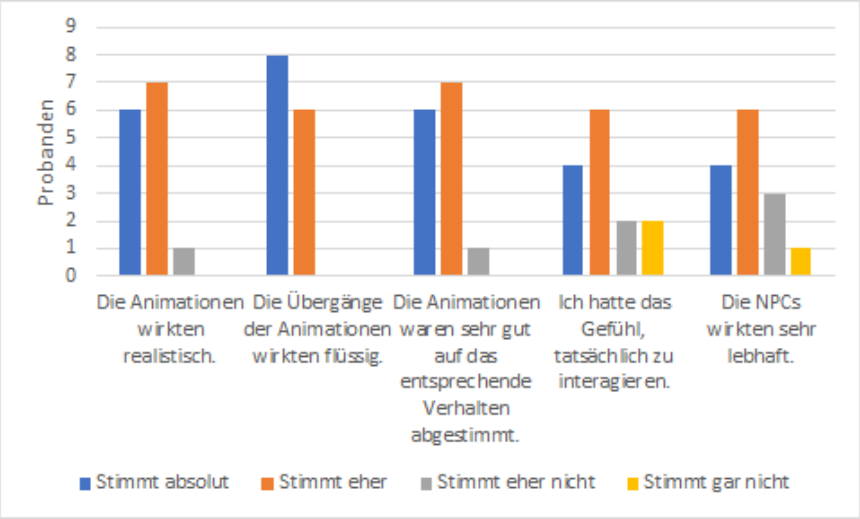


Abbildung 19: *Beurteilung der Animationen der NPC-Objekte.*

Spieler-Objekt. Das Verhalten des Spieler-Objekts wurde überwiegend als gut einsichtig und gut gewählt empfunden. Nur 2 Probanden wünschten sich für die Aktionen des Spieler-Objekt optisch passendere und lebhaftere Animationen.

Gegner-Objekte. Der Großteil der Probanden konnte eher gut einsehen, wie sich die Gegner-Objekte verhalten. Manche Animationen der Gegner schienen für einzelne Aktionen eher unpassend gewesen zu sein. Besonders kritisiert wurde die Angriffsanimation der Fledermaus-Objekte. Man konnte nur sehr schlecht deuten, wann ein Fledermaus-Objekt angreift. Für dieses Objekt müsste daher eine einsichtigere Angriffsanimation erstellt werden.

NPC-Objekte. Die Animationen der NPC-Objekte wurden von den meisten Probanden auch positiv vernommen. Das Verhalten der NPCs war sehr gut einzusehen. Um jedoch alle Probanden zufriedenzustellen, könnte man die NPCs noch mit lebhafteren Animationen versehen, die vor allem die Interaktionen realistischer erscheinen lassen.

5.3.3 Benutzerfreundlichkeit

Insgesamt lässt sich aussagen, dass das Spiel mittels der zur Verfügung gestellten Hilfsmittel und Visualisierungen schon eher als benutzerfreundlich zu betrachten ist (vgl. Abbildungen 20, 21 und 22).

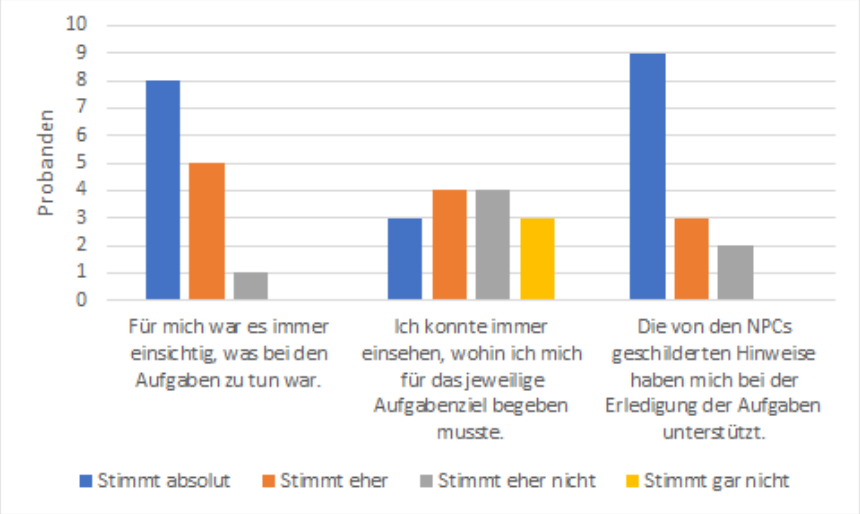


Abbildung 20: *Teil 1: Beurteilung der Benutzerfreundlichkeit hinsichtlich der Aufgabenbewältigung.*

Unterstützung bei der Aufgabenbewältigung. 92,9% der Probanden war es immer einsichtig, was bei den Quests zu tun war. Die Erklärungen der NPCs haben dem Großteil der Probanden gut erklären können, was zu tun war. Wenn es darum ging, sich zum Quest-Ziel hinzubegeben, teilten sich die Meinungen. 50% der Probanden wussten oft nicht, wo sich das Quest-Ziel befand. Eine entscheidende Ursache hierfür war die Orientierungslosigkeit, wie die Probanden schilderten und in der Kritik erwähnten. Es wurde sich vermehrt die Einbettung eines Kompasses oder einer Karte der Spielwelt gewünscht. Der Einsatz von Quest-Symbolen schien auf jeden Fall ein sehr guter Anhaltspunkt für die Probanden gewesen zu sein, um Auftraggeber schneller ausfindig zu machen. Diese Symbole wurden ihnen zufolge auch sehr verständlich erklärt. Das Einblenden des Quest-Fortschritts fanden die Probanden ebenfalls sehr hilfreich. So konnten sich die Probanden direkt nach Vervollständigung der Quest zurück zum Auftraggeber hinbegeben.

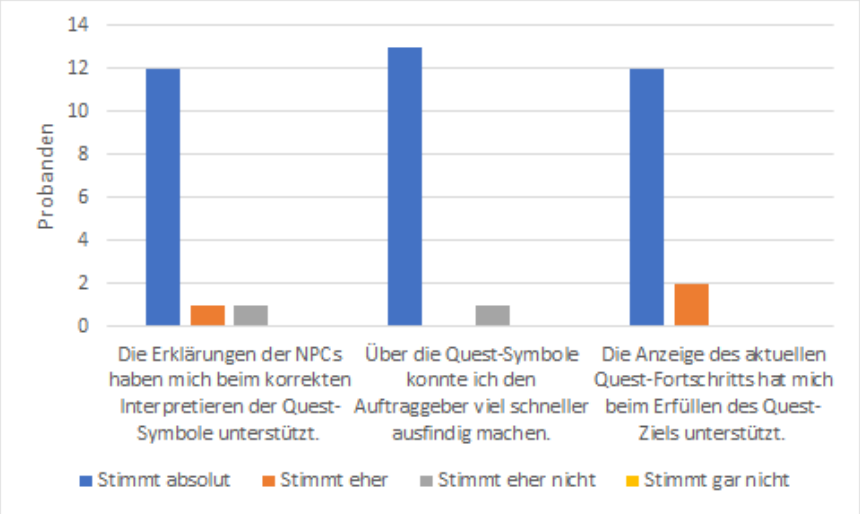


Abbildung 21: *Teil 2: Beurteilung der Benutzerfreundlichkeit hinsichtlich der Aufgabenbewältigung.*

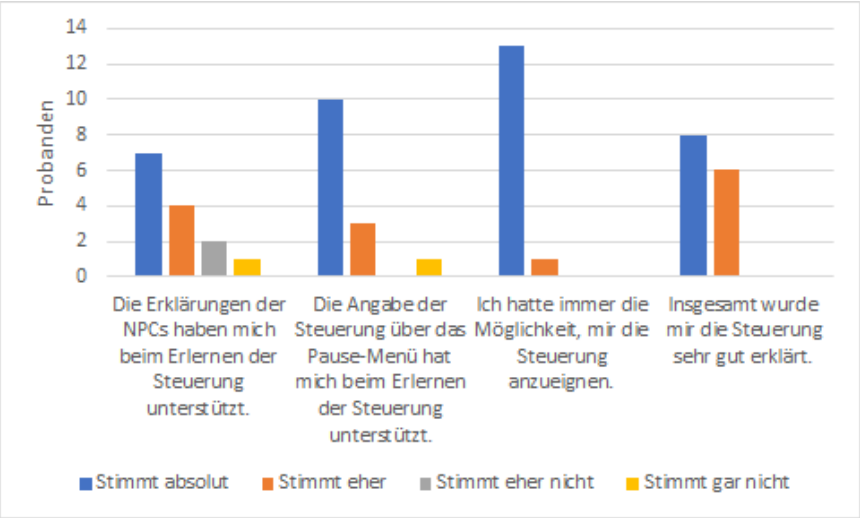


Abbildung 22: *Beurteilung der Benutzerfreundlichkeit hinsichtlich der Erlernbarkeit der Steuerung.*

Unterstützung beim Erlernen der Steuerung. Dieser Teil der Auswertung lässt nur bedingt Aussagen bestätigen oder widerlegen. Die Problematik hierbei war, dass die Probanden die Hilfestellungen zur Erklärung der Steuerung kaum bis gar nicht genutzt haben. Offenbar reichte es vielen Probanden aus, nur die nötigsten Aktionen zu kennen, um das Spiel vollenden zu können. Im Fragebogen fehlte demnach eine auswählbare Option, die besagt, dass eine Hilfestellung von den Probanden nicht in Anspruch genommen wurde. Diese Probanden mussten

demnach eine Option auswählen, die nicht auf sie zutrafen. Über Beobachtungen konnte jedoch festgestellt werden, welche Antworten ungültig sind und welche nicht. Diejenigen Probanden, die sich von den NPC-Objekten aufklären ließen, waren der Meinung, dass diese Erklärungen hilfreich gewesen sind. Ein primärer Anhaltspunkt war für alle Probanden zumindest die Einsicht der Steuerung über das Steuerungsmenü. Die Probanden bestätigten oder gaben zumindest zu, dass sie jederzeit die Möglichkeit besaßen, die Steuerung zu erlernen. Da manche Probanden die NPCs oft nur einmal angesprochen haben, könnte man daraus schlussfolgern, dass sie weitere Dialoge nicht erwarten würden. In der Kritik wurde unter anderem auch erwähnt, dass das wiederholte Initiieren der Dialoge eher negativ empfunden wurde und die Dialoge selbst oft zu lang waren. Wahrscheinlich hätten sie auch eine andere Form von Tutorial erwartet, wie es bei gängigen Spielen der Fall ist. Eine solche Form in dieser Arbeit aber noch zusätzlich umzusetzen, wäre aus mangelnder Zeit leider nicht mehr möglich gewesen. Für die zukünftige Weiterentwicklung des Spiels wäre es aber auf jeden Fall eine Überlegung wert, eine attraktivere Einführung in das Spiel zu gestalten, die nicht nur auf Interaktionen basiert.

5.3.4 Kritik und Verbesserungsvorschläge

Im letzten Abschnitt der Umfrage wurde sehr viel positives, aber auch negatives Feedback geschildert (vgl. Anhang A.2). 13 von 14 Teilnehmern haben tatsächlich Spaß an dem Spiel empfunden. Die Bewertung des Spiels insgesamt beläuft sich im Durchschnitt auf rund vier Sterne, was der Zuordnung „Gut“ entspricht. Um einen Ausblick für eine zukünftige Weiterentwicklung des Spiels geben zu können, wird jedoch in diesem Kapitel nur auf die problematischen Aspekte eingegangen. Die positiv empfundenen Aspekte können im Anhang nachgelesen werden.

Sehr problematisch war auf jeden Fall die Point-To-Click-Steuerung. Die Kamera-Rotation, die eigentlich nur als Notlösung eingebettet wurde, sei viel zu steif gewesen. Es wurde vorgeschlagen, die Kamera über einen Shortcut immer in Blickrichtung des Spieler-Objekts auszurichten. Dies könnte die Bedienbarkeit der Steuerung sicherlich etwas verbessern. Zudem waren die Hit-Boxen der Gegner-Objekte oft viel zu klein, was es erschwerte, die Gegner richtig anzuvisieren und anzugreifen. Problematisch war dabei auch, dass das Spieler-Objekt das Anvisieren nahezu unmöglich gemacht hat, wenn der Mauscursor auf ihn zeigte. Dieser Fall ist bei der Implementation leider nicht gut abgefangen worden. Man müsste dementsprechend die Hit-Boxen anpassen und den Fall „Cursor auf dem Spieler-Objekt“ noch besser umsetzen. Ein weiterer Kritikpunkt an der Point-To-Click-Steuerung war, dass der Action-Aspekt eher untergegangen ist. Sollte diese Steuerung in Zukunft weiterhin bestehen bleiben, müsste man sich auf jeden Fall überlegen, wie man diese attraktiver und actionreicher gestalten könnte.

Die Standard-Steuerung wurde im Gegensatz zur Point-To-Click-Steuerung viel actionreicher empfunden. Zudem wurde sie noch am meisten bevorzugt, da sie sich deutlich besser bedienen ließ und weniger auf Probleme stieß. Der Schwierig-

keitsgrad der Kämpfe war in dieser Steuerungsart demzufolge auch viel einfacher als in der Point-To-Click-Steuerung.

Von den Aktionen wurde hauptsächlich die Ausweichen-Aktion bemängelt, weil sie keinen besonderen Nutzen hatte. Die Blocken-Aktion hatte sie wegen ihrem aktivierbaren Vorteil völlig unbrauchbar gemacht. Zusätzlich war das Blocken für das Kampfgeschehen viel zu mächtig, weil sich jeder Angriff komplett blocken ließ und dazu noch den Schaden des nächsten Angriffs verdoppelte. Diese Aktion sollte daher in ihrer Verfügbarkeit mehr eingeschränkt werden. Außerdem sollte sich für das Ausweichen ein guter Vorteil überlegt werden, damit es sich auch lohnt, diese Aktion im Kampf einzusetzen. Einer der Probanden hätte es sinnvoller empfunden, wenn man an Gegner-Objekte vorbeirollen könnte, anstatt gegen sie zu rollen.

Die Orientierung in der Spielwelt stellte auch ein zentrales Problem dar. Diesbezüglich hat das Spiel nur sehr wenige Hilfen zur Verfügung gestellt, welche aber nicht ausreichend genug waren. Es wurde sich gewünscht, eine Karte der Spielwelt oder einen Kompass einblenden zu können, um sich anhand dessen in der Spielwelt richtig orientieren zu können. Der Aspekt der Orientierung wurde in der Implementierung ziemlich vernachlässigt und muss in Zukunft mehr berücksichtigt werden.

Manche Probanden empfanden die Dialog-Texte oft zu lang. Zudem wurde das wiederholte Initiieren der Dialoge bemängelt, was wahrscheinlich auch der Grund dafür war, weshalb sich einige Probanden nicht die Erklärungen der NPC-Objekte durchgelesen haben. Es wäre an dieser Stelle wahrscheinlich sinnvoller gewesen, eine Auswahl der Dialoge zur Verfügung zu stellen, die durch Anklicken dieser ausgelöst werden. Auch müsste die Einführung ins Spiel generell überarbeitet werden.

Um das Kampfsystem noch actionreicher und die Ausdauerpunkte noch gebräuchlicher zu machen, wurde vorgeschlagen, noch zusätzliche Spezialangriffe und erlernbare Fähigkeiten anhand eines Skill-Trees zu integrieren. Ich stimme zu, dass ein einfacher Angriff das Kämpfen selbst doch sehr eintönig und träge macht. Eine Erweiterung des Kampfsystems um zusätzliche Angriffe und Fähigkeit würde dazu beitragen, die Kämpfe attraktiver zu gestalten. Zudem wurde sich gewünscht, die Gegner-Objekte noch stärker variieren zu lassen, sodass die Kämpfe interessanter werden. Man könnte sich überlegen, ob man die Verhaltensmuster der Gegner je nach Art unterschiedlich gestaltet oder ob man nicht sogar Boss-Gegner einführt, um spannende Kämpfe zu kreieren.

Das Quest-System wurde überwiegend positiv angenommen. Ein Proband hätte es aber noch viel besser empfunden, wenn es möglich gewesen wäre, mehrere Quests gleichzeitig bearbeiten zu können. Des Weiteren wurde sich gewünscht, die Auswahl an Quest-Arten zu vergrößern, die den Schwerpunkt zum Beispiel auf die Erkundung oder auf das Sammeln von Gegenständen setzen. Manchmal hatten die Probanden auch etwas Probleme, die Quest-Marker richtig zu entdecken. Es wurde sich gewünscht, diese noch zu vergrößern, um sie besser einsehen zu können.

6 Ausblick und Fazit

Im Rahmen dieser Arbeit wurde ein funktionsfähiges Action-Rollenspiel in der Game Engine Unity umgesetzt und anschließend evaluiert. Die Evaluation zeigte, dass das Spiel insgesamt zwar gut umgesetzt war, aber noch einige Baustellen aufweist, um die man sich in Zukunft noch kümmern sollte.

Um das Spiel noch benutzerfreundlicher zu gestalten, sollten auf jeden Fall Navigationselemente eingebettet werden, damit das Orientieren innerhalb der Spielwelt deutlich verbessert wird. Des Weiteren muss an der Point-To-Click-Steuerung gearbeitet werden, da sie bisher noch sehr anfällig für Probleme ist und dem Spieler damit beim Spielen nicht viel Freude bereitet. Wenn man sich an dieser Stelle nun entscheiden sollte, auf welche Steuerungsart sich in diesem Spiel beschränkt werden müsste, fällt die Entscheidung so gut wie einstimmig auf die Standard-Steuerung. Sie ließ sich von beiden Steuerungsarten noch am einfachsten bedienen. Leider reichten die zeitlichen Ressourcen nicht mehr aus, um die Point-To-Click-Steuerung ausreichend gut zu optimieren. Anstatt die Steuerung zu vereinfachen, hat sie sie eher erschwert. Für die Einführung in das Spiel sollte auf Dialogführungen möglichst verzichtet und ein attraktiveres Tutorial-System gestaltet werden, welches besser zugänglich ist und das Erlernen der Spielmechaniken besser ermöglicht. Zudem wäre es sicherlich auch von Vorteil, wenn sich die Spieler eigene Tastenbelegungen zusammenstellen können, um das Spiel ihrem Belieben nach zu bedienen. Das bisherige Kampfsystem scheint zwar bisher ganz gut zu funktionieren, sollte aber hinsichtlich des Aspekts „Attraktivität“ noch erweitert werden. Bisher wurde überhaupt nicht bemängelt, dass das Spiel über kein auditives Feedback verfügt. Es wäre aber eine Überlegung wert, Musik und Sound-Effekte in das Spiel zu integrieren, um das Spiel attraktiver zu gestalten.

Zum Abschluss lässt sich sagen, dass durch die Arbeit mit der Unity Game Engine sehr viele neue Erfahrungen, auch im Hinblick auf die Programmierfähigkeiten, gewonnen werden konnten. Der Umfang dieser Bachelorarbeit war durch die selbstständige Themenwahl zwar sehr hoch angesetzt, ermöglichte jedoch, viele verschiedene Bereiche der Game Engine kennenzulernen und anzuwenden. Es ist erstaunlich, wie umfangreich eine Game Engine doch ist. Im Entwicklungsprozess haben sich diverse Problemstellungen ergeben, die in der Implementation oft erfolgreich umgesetzt werden konnten. Trotzdem konnten aber auch noch viele Schwachstellen im Rahmen der Evaluation festgestellt werden. Dennoch gab es aber auch viel positives Feedback, welches die erbrachten Leistungen wertschätzte. Ein Action-Rollenspiel sehr gut umzusetzen, ist schon ziemlich schwierig und komplex. Aber der Versuch, ein solches Spiel mal selbstständig von null auf zu entwickeln, bot sich als eine sehr gute Gelegenheit an, viel zu programmieren und auch andere Themengebiete wie die Modellierung oder Animation von Objekten kennenzulernen. Es ist daher für Studierende des Fachbereichs Informatik , insbesondere des Studiengangs Computervisualistik, auf jeden Fall empfehlenswert, sich mal mit einer Game Engine auseinanderzusetzen.

Literatur

[1] ADAMS, Ernest: *Fundamentals of Game Design*. 2. Edition. Thousand Oaks, CA, USA : New Riders Publishing, 2009. – ISBN 0321643372, 9780321643377

[2] BLIZZARD ENTERTAINMENT: *Diablo III*. https://www.blizzard.com/de-de/games/d3/. – zuletzt aufgerufen am 17.09.2019

[3] BRACKEYS: *How to make a Dialogue System in Unity*. https://www.youtube.com/watch?v=_nRzoTzeyxU&list=PLPV2KyIb3jR5qEyOlJImGFoHcxg9XUQci&index=3&t=0s. – zuletzt aufgerufen am 24.05.2019

[4] CLARKE, Rachel ; LEE, J. ; CLARK, N.: Why Video Game Genres Fail: A Classificatory Analysis. In: *Games and Culture* 12 (2015), 07. http://dx.doi.org/10.1177/1555412015591900. – DOI 10.1177/1555412015591900

[5] MIXAMO, ADOBE SYSTEMS INCORPORATED: *mixamo*. https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack. – zuletzt aufgerufen am 24.05.2019

[6] NINTENDO: *Super Mario Bros.* https://www.nintendo.de/Spiele/NES/Super-Mario-Bros--803853.html. – zuletzt aufgerufen am 17.09.2019

[7] SQUARE ENIX: *Final Fantasy VII*. https://square-enix-games.com/de_DE/games/final-fantasy-vii. – zuletzt aufgerufen am 17.09.2019

[8] SQUARE ENIX: *Kingdom Hearts*. https://www.kingdomhearts.com/about/de/. – zuletzt aufgerufen am 17.09.2019

[9] UNITY TECHNOLOGIES: *Unity Manual*. https://docs.unity3d.com/2018.3/Documentation/Manual/. – zuletzt aufgerufen am 17.09.2019

[10] VERBAND DER DEUTSCHEN GAMES-BRANCHE: Jahresreport der deutschen Games-Branche 2019. Version: 2019. https://www.game.de/publikationen/jahresreport-der-deutschen-games-branche-2019/. 2019. – Forschungsbericht

[11] XOCTOMANX: *Unity 5 Tutorial: Quest System*. https://www.youtube.com/playlist?list=PLj0TSSTwoqAy0abF90Ov3H7SDDj9jcpGD. – zuletzt aufgerufen am 29.07.2019

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

A Anhang

A.1 Fragebogen

Umfrage zum Action-Rollenspiel

Im Rahmen meiner Bachelorarbeit möchte ich mit dieser Umfrage mein selbst programmiertes Action-Rollenspiel evaluieren.

Nachdem Sie nun das Spiel ausprobiert haben, bitte ich Sie, die folgenden Fragen nach Ihrem persönlichen Empfinden zu beantworten.

Alle von Ihnen angegebenen Daten werden selbstverständlich vertraulich behandelt und nur im Rahmen dieser Arbeit zur Auswertung dieser Umfrage verwendet.

Bei Fragen wenden Sie sich gerne an mich.

Ich danke Ihnen für Ihre Teilnahme!
Vanessa Karolek

Bedienbarkeit

1. Wie empfanden Sie die „Standard“-Steuerung?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Es war einfach, an Hindernissen vorbeizulaufen.	○	○	○	○
Mir fiel es leicht, die Spielfigur zu einer bestimmten Stelle hinzubewegen.	○	○	○	○
Einen Gegner zu besiegen, fiel mir leicht.	○	○	○	○
Mehrere Gegner gleichzeitig zu besiegen, fiel mir leicht.	○	○	○	○
Mir fiel es leicht, mit NPCs zu interagieren.	○	○	○	○
Alle Aktionen waren für mich gut zu erreichen.	○	○	○	○
Jede Aktion war sehr einfach auszuführen.	○	○	○	○
Der Übergang von meinem Input zur Reaktion im Spiel verlief flüssig.	○	○	○	○
Die Art der Steuerung ist mir vertraut.	○	○	○	○

2. Wie empfanden Sie die „Point-To-Click“-Steuerung?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Es war einfach, an Hindernissen vorbeizulaufen.	○	○	○	○
Mir fiel es leicht, die Spielfigur zu einer bestimmten Stelle hinzubewegen.	○	○	○	○
Einen Gegner zu besiegen, fiel mir leicht.	○	○	○	○
Mehrere Gegner gleichzeitig zu besiegen, fiel mir leicht.	○	○	○	○
Mir fiel es leicht, mit NPCs zu interagieren.	○	○	○	○
Alle Aktionen waren für mich gut zu erreichen.	○	○	○	○
Jede Aktion war sehr einfach auszuführen.	○	○	○	○
Der Übergang von meinem Input zur Reaktion im Spiel verlief flüssig.	○	○	○	○
Die Art der Steuerung ist mir vertraut.	○	○	○	○

Visuelle Überzeugung

3. Wie empfanden Sie die einzelnen Animationen des Spielers?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Die Übergänge der Animationen erschienen mir flüssig.	○	○	○	○
Die Animationen wirkten realistisch.	○	○	○	○
Die Animationen waren sehr gut auf die Aktionen abgestimmt.	○	○	○	○
Die Spielfigur wirkte sehr lebhaft.	○	○	○	○

4. Wie empfanden Sie die einzelnen Animationen der Gegner?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Die Animationen empfand ich realistisch.	○	○	○	○
Die Übergänge der Animationen erschienen mir flüssig.	○	○	○	○
Die Animationen waren sehr gut auf das entsprechende Verhalten abgestimmt.	○	○	○	○
Die Gegner wirkten sehr lebhaft.	○	○	○	○

5. Wie empfanden Sie die einzelnen Animationen der NPCs?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Die Animationen wirkten realistisch.	○	○	○	○
Die Übergänge der Animationen wirkten flüssig.	○	○	○	○
Die Animationen waren sehr gut auf das entsprechende Verhalten abgestimmt.	○	○	○	○
Ich hatte das Gefühl, tatsächlich zu interagieren.	○	○	○	○
Die NPCs wirkten sehr lebhaft.	○	○	○	○

Benutzerfreundlichkeit

6. Wurden Sie durch geeignete Visualisierungen beim Erledigen der Aufgaben (Quests) unterstützt?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Für mich war es immer einsichtig, was bei den Aufgaben zu tun war.	○	○	○	○
Ich konnte immer einsehen, wohin ich mich für das jeweilige Aufgabenziel begeben musste.	○	○	○	○
Die von den NPCs geschilderten Hinweise haben mich bei der Erledigung der Aufgaben unterstützt.	○	○	○	○
Die Erklärungen der NPCs haben mich beim korrekten Interpretieren der Quest-Symbole unterstützt.	○	○	○	○
Über die Quest-Symbole konnte ich den Auftraggeber viel schneller ausfindig machen.	○	○	○	○
Die Anzeige des aktuellen Quest-Fortschritts hat mich beim Erfüllen des Quest-Ziels unterstützt.	○	○	○	○

7. Wurden Sie durch geeignete Visualisierungen beim Erlernen der Steuerung unterstützt?

	Stimmt absolut	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht
Die Erklärungen der NPCs haben mich beim Erlernen der Steuerung unterstützt.	○	○	○	○
Die Angabe der Steuerung über das Pause-Menü hat mich beim Erlernen der Steuerung unterstützt.	○	○	○	○
Ich hatte immer die Möglichkeit, mir die Steuerung anzueignen.	○	○	○	○
Insgesamt wurde mir die Steuerung sehr gut erklärt.	○	○	○	○

Kritik / Verbesserungsvorschläge

Mit den folgenden Fragen würde ich gerne in Erfahrung bringen, was ich an dem Spiel in Zukunft noch verbessern oder ergänzen könnte.

8. Was hat Ihnen an dem Spiel besonders gut gefallen?

……
……
……
……

9. Was hat Ihnen an dem Spiel nicht so gut gefallen?

……
……
……
……

10. Wie würden Sie den Schwierigkeitsgrad der Kämpfe nach Ihrem persönlichen Empfinden einschätzen und warum?

……
……
……
……

11. Welche der beiden Steuerungsarten halten Sie für das Spiel am sinnvollsten und warum?

……
……
……
……

12. Welche Verbesserungsvorschläge würden Sie dem Spiel geben?

……
……
……
……

13. Hat es Ihnen Spaß gemacht, dieses Spiel zu spielen?

- ○ Ja.
- ○ Nein.

14. Wie würden Sie das Spiel allgemein bewerten?

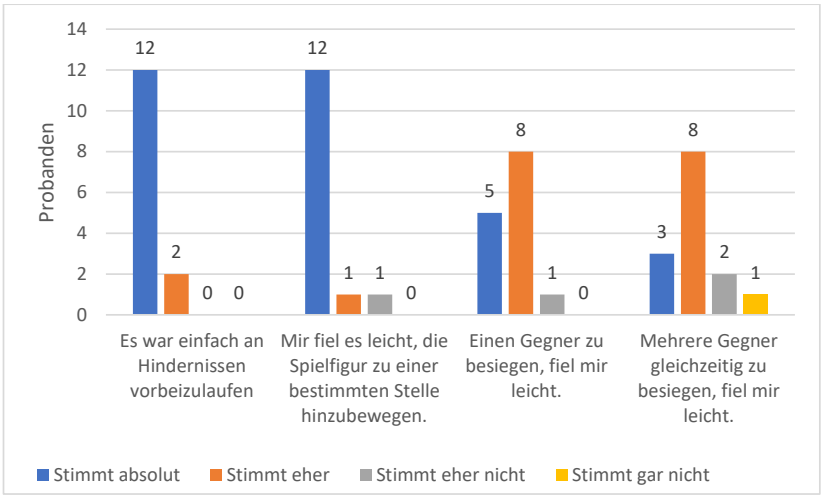
Sehr schlecht ☆ ☆ ☆ ☆ ☆ Sehr gut

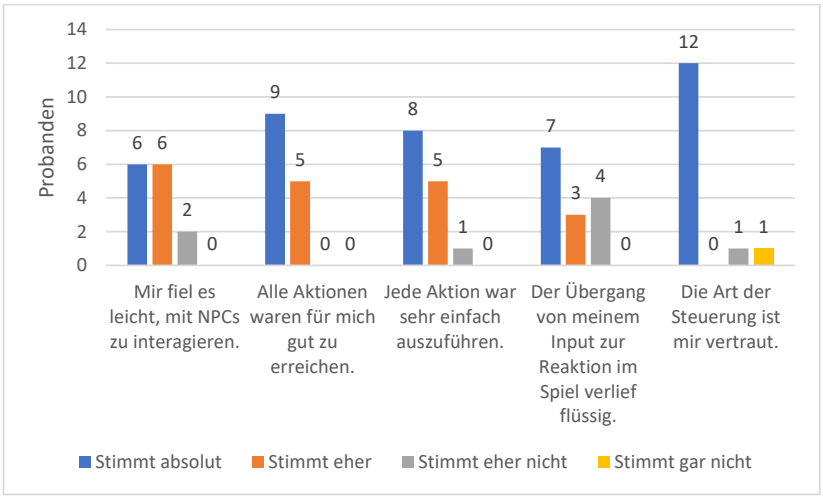
A.2 Ergebnisse des Fragebogens

Auswertung des Fragebogens

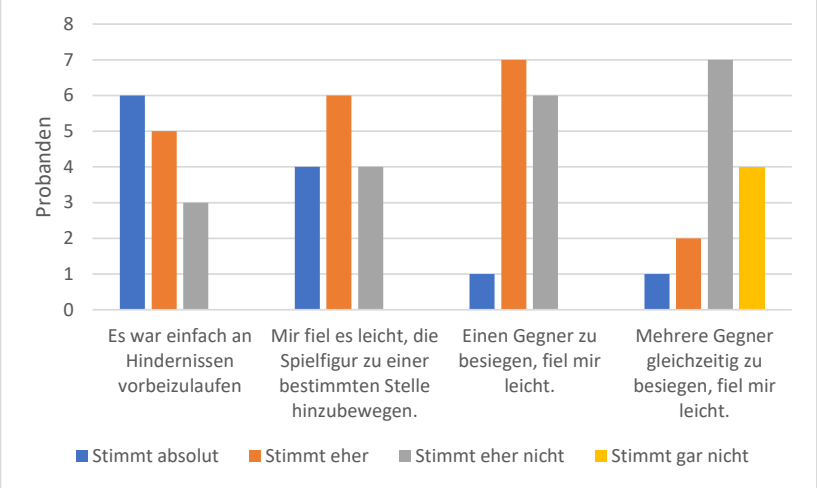
Bedienbarkeit

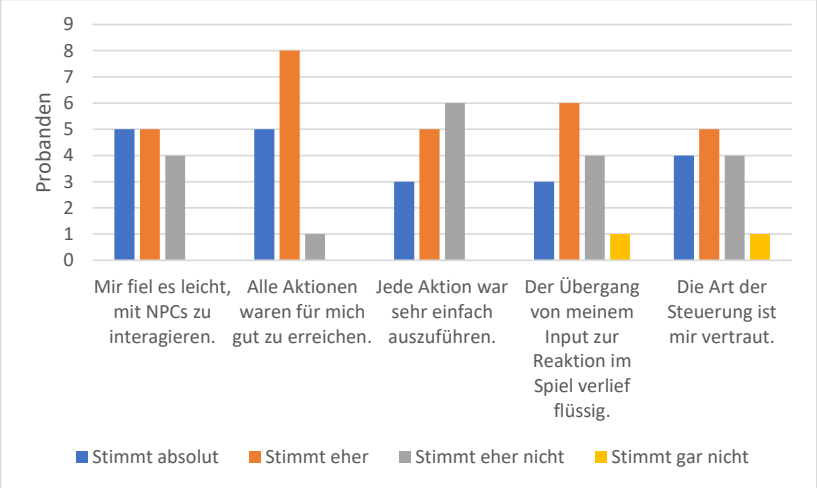
1. Wie empfanden Sie die "Standard"-Steuerung?





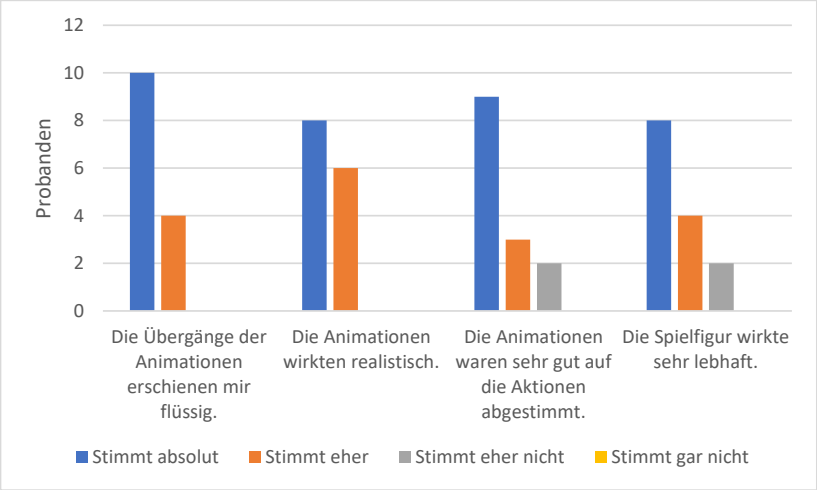
2. Wie empfanden Sie die "Point-To-Click"-Steuerung?



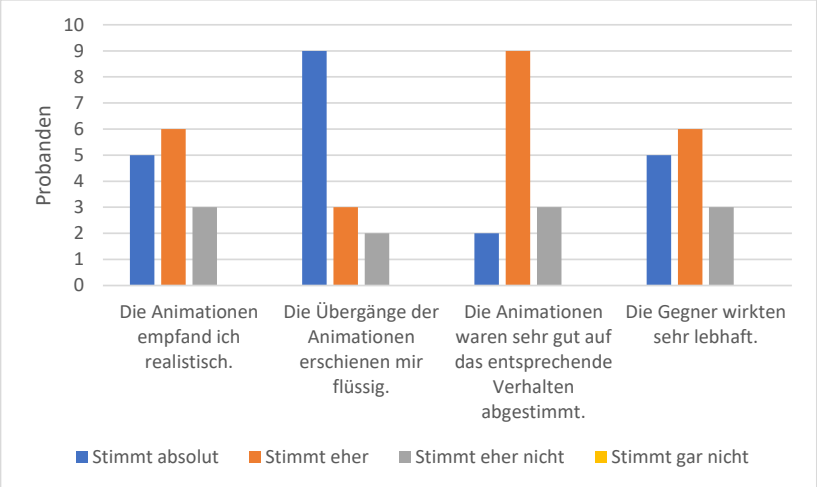


Visuelle Überzeugung

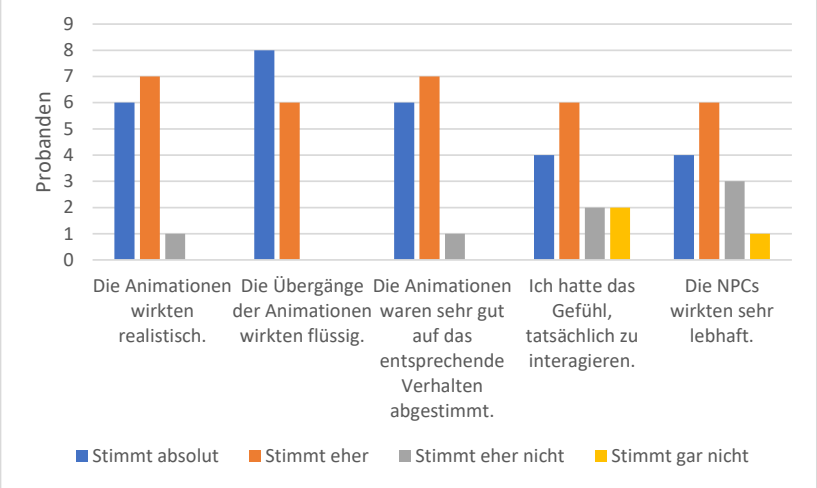
3. Wie empfanden Sie die einzelnen Animationen des Spielers?



4. Wie empfanden Sie die einzelnen Animationen der Gegner?

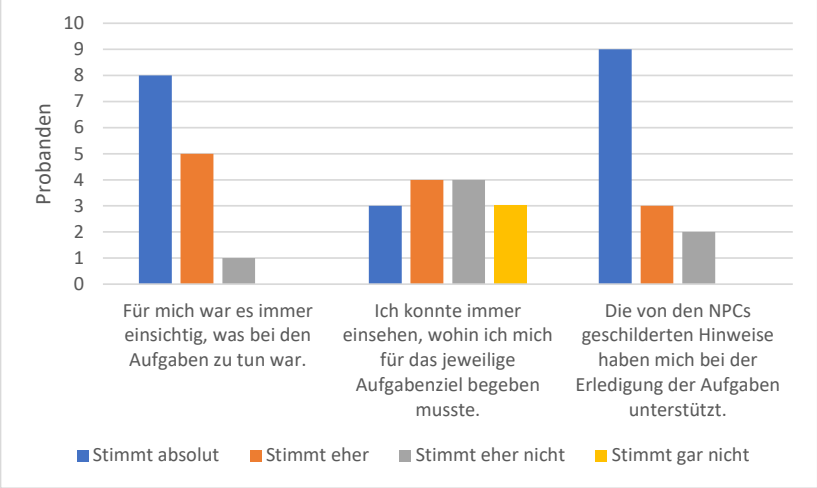


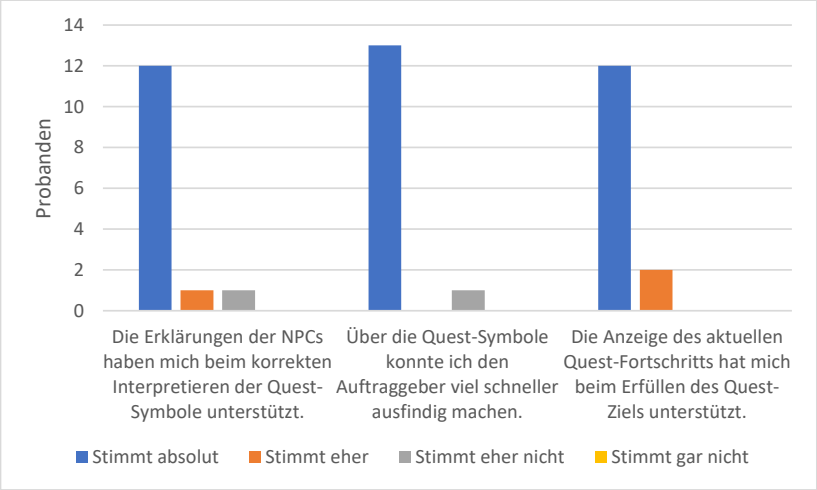
5. Wie empfanden Sie die einzelnen Animationen der NPCs?



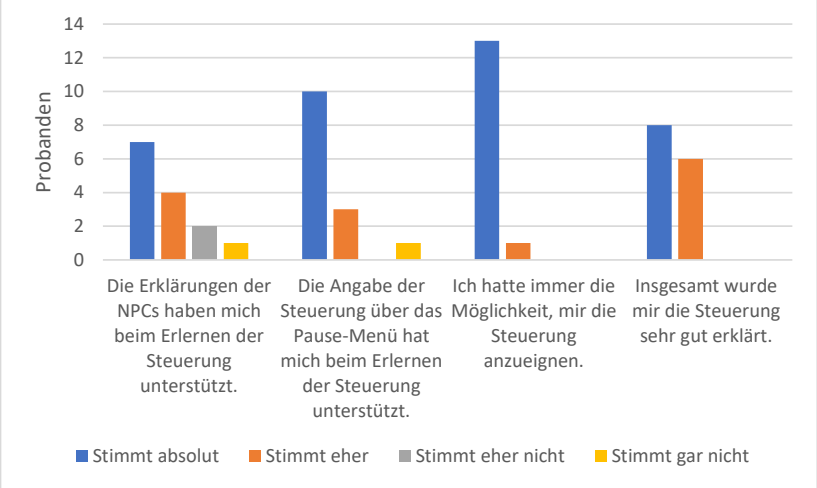
Benutzerfreundlichkeit

6. Wurden Sie durch geeignete Visualisierungen beim Erledigen der Aufgaben (Quests) unterstützt?





7. Wurden Sie durch geeignete Visualisierungen beim Erlernen der Steuerung unterstützt?



Kritik / Verbesserungsvorschläge

8. Was hat Ihnen an dem Spiel besonders gut gefallen?
    - Animationen der Gegner
    - Der Erkundungs-Aspekt
    - Ich mochte die Dialoge, die auf die 4. Wand angespielt haben, das hat mir immer ein Grinsen ins Gesicht gelockt. Die Charaktere waren sehr schön gestaltet, zwar minimalistisch, aber doch mit einem gewissen Charme.
    - Die süß-gestalteten, unterschiedlich-farbigen Gegner; ausgeglichener Cooldown der Heiltränke; Sprint <3; Quest-anzeigende Zeichen; doppelte Stärke nach Blocken
    - Der Grafik Stil
    - Das unterschiedliche Verhalten der Gegner (Hasen eher passiv, Geister und Schleim eher aggressiv) Unterschiedliches Design der NPCs Größe der Spielwelt
    - Der Artstil, die Gespräche mit den NPCs, die Quests :D
    - Die Möglichkeit, mit allen NPCs zu interagieren. Man wusste durch die ausführlichen Erklärungen der NPCs immer, was man zu tun hat. Die Möglichkeit, mit Gegnern zu kämpfen.
    - Sehr gute Erklärungen, Positionierungen, Animationen, Story-Plot
    - Das doch actionreiche gameplay der standard-steuerung, inklusive konter-system und AP-system. Auch die änderung der steuerungsart hat das gameplay doch bemerklich verändert.
    - - die Standardsteuerung war sehr intuitiv und einfach - die Kameraeinstellung bei der Standardsteuerung musste nicht zusätzlich bewegt werden, um "geradeaus" zu schauen - die Liebe zu Detail (z.B. die Animation der Fackeln) - alles war einheitlich in dem "Lowpoly"-Look gestaltet, daher wurde das sozusagen zum Stil des Spiels - Animationen waren sehr stimmtig (z.B. hat man dem Player angesehen, dass das Zweihandschwert sehr schwer sein muss, durch seine Körperhaltung beim Rennen und Laufen) - intuitives Questsystem
    - Ich fand es gut, dass der Block einen Sinn hatte und man damit gut arbeiten konnte um recht wenig Schaden zu erhalten, aber trotzdem Schaden auszuteilen. Das Verhalten der Hasen!
    - Friedliche waren von aggressiven Gegnern gut zu unterscheiden. Man wurde durch das Lob der NPC`s nach bestandener Aufgabe gut motiviert weiter zu machen. Das Spiel ist sehr übersichtlich aufgebaut.

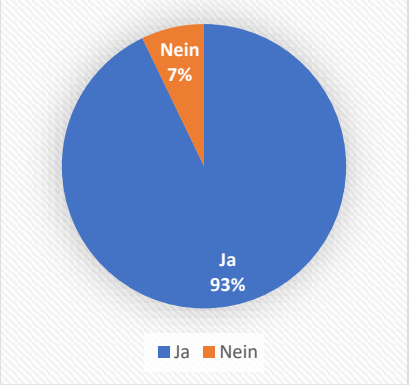
9. Was hat Ihnen an dem Spiel nicht so gefallen?
    - - Bewegung der Kamera bei PTC-Steuerung etwas steif
    - Wiederholtes Initiieren von Dialogen
    - Die Hitboxen, sowohl bei der WASD Steuerung als auch bei der Point-To-Click Steuerung, waren oft nicht gut zu treffen
    - bei Point-N-Click -> die Entfernung, ab der Dialoge angezeigt werden vs geführt werden können; Fledermaus-Angriffe nicht unbedingt ersichtlich; Angreifen manchmal schwammig, wenn Pointer zu nahe an Spieler/ mehrere Gegner
    - Das Kampfsystem war zu träge.
    - keine Karte / Kompass
    - Quest 20 Hasen zu töten, war zu langatmig, wenn man nicht viele Hasen getriggert und gleichzeitig getötet hat relativ lange langweilige Wege
    - Die zweite Steuerung. Die Reichweite des Schwerts, die man anders einschätzt, als sie ist.
    - Interaktion mit den NPCs nicht von allen Seiten optimal. Die Kamera konnte nicht auf der vertikalen Achse bewegt werden. Die Point-To-Klick Steuerung war gewöhnungsbedürftig.
    - Lange Interaktionstexte der NPC's, wenn man am Rand des Spiels war, konnte man das Hauptlager nicht mehr sehen
    - die point-to-klick steuerung hat das gameplay zwar sehr verändert, aber stand sich selbst meistens auch im weg, vor allem bei der steuerung und interkation mit npcs und gegnern. Das kämpfen wurde dadurch weniger actionreich und die konter-mechanik war nichtmerh so zu gebrauchen, das das angreifen über ein zeitintervall festgelegt worden ist
    - - Point-to-klick Steuerung war schwierig beim Kampf, da die Spielfigur oft den Bereich überdeckt hat, der zum Angreifen des Gegners geführt hätte - das Rollen zum Ausweichen habe ich nie im Kampf verwendet, da das Blocken meist ausgereicht hat und man beim Rollen eher gegen den Gegner gerollt ist, statt an ihm vorbei
    - Ich fand die "Rolle" nicht unterstützend im Kampf, weil der Block diese outclassed hat.
    - Ich konnte die Himmelsrichtungen schlecht deuten und war deshalb öfter etwas orientierungslos.

10. Wie würden Sie den Schwierigkeitsgrad der Kämpfe nach Ihrem persönlichen Empfinden einschätzen und warum?
    - Sehr angemessen (gutes Feedback bezüglich Angriffs- und Block-Aktion)
    - Einfach, da der Heiltrank beliebig oft eingesetzt werden kann und sich alle Attacken blockieren lassen.
    - Bei dem Kampf gegen mehrere Gegner war der Schaden doch etwas hoch, sodass ich immer so lange weggelaufen bin, bis ich meinen Heiltrank wieder einsetzen konnte.
    - die Kämpfe selbst waren recht ausgeglichen, nur die Steuerung war teilweise schwammig, sodass man manchmal mehr Schaden einstecken muss als nötig
    - Bei mehreren Gegner unmöglich alles zu blocken und dadurch sehr schwer, aber wenn man sich Zeit lässt und geduldig ist, sehr einfach.
    - Eher Leicht
    - War schon schwer, weil nicht jeder Angriff saß und man im Kampf gegen mehrere Gegner zu schwach war
    - Angemessen. Leider war es aber schwer die Hasen im Verbund anzugreifen, da sie zu weit auseinander standen. Und die Angriffsreichweite des Schwertes musste man sich erstmal aneignen.
    - Eher leicht, da die Gegner nur ein Angriff ausgeführt haben. So konnte man im Rhythmus immer Blocken und Angreifen.
    - waren einfach. Schleimer schwieriger zu besiegen als Geister.
    - Im standard-stil eher leicht. Die gegneranimationen waren doch sehr eindeutig und das blocken hat jeglichen schaden annuliert. in Verbindung mit dem Kontern wurden die kämpfe doch schnell beendet. Im point-to-klick gemischt. erschwert wurde das ganze da vorallem durch das zeitintervall der angriffe, wodurch es nichtmehr so gut möglich war, auf die gegner zu reagieren
    - Die Schwierigkeit war in Ordnung. Es war nicht zu schwer und doch fordernd. Gut war, dass der Heiltrank sich nach einer Zeit immer aufgefüllt hat und man sich im Zweifelsfall damit heilen konnte, oder sich aus dem Staub machen konnte.
    - Ich fand ihn ohne Block schwer, wenn man viele Gegner gepullt hatte und mit relativ machbar. Ich fand den AOE-Angriff sehr gut dabei.
    - Als "Nicht-Spieler" hatte ich am Anfang noch Probleme bei den Kämpfen, was mit der Zeit aber immer besser wurde.

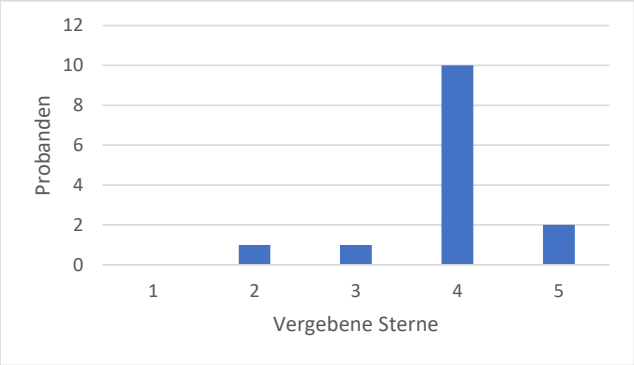
11. Welche der beiden Steuerungsarten halten Sie für das Spiel am sinnvollsten und warum?
    - PTC cool, Standard spielt sich aber leicht besser, da diese mir mehr vertraut ist
    - Definitiv WASD, weil man beide Hände gleichzeitig benutzen kann (Kamera und Bewegung sind separiert) und die Shift-Taste (Sprint) in dem Tastenbereich liegt.
    - Ich würde die WASD Steuerung bevorzugen, da man bei den Kämpfen dann eher das Gefühl hat, selbst etwas zu tun, anstatt durchgehen die Maus gedrückt zu halten Außerdem liegt die Hand bei der Point and Click Steuerung sowieso wie bei der WASD Steuerung auf der Tastatur
    - von den Kämpfen und Interaktionen mit NPCs her fand ich die Standardsteuerung besser, da die Steuerung hier nicht so schwammig war
    - Standard, weil bei Point-and-Click der Charakter Aktionen erschwert/blockiert hat, ansonsten bei den wenigen Aktionsmöglichkeiten eigentlich gleich sinnvoll.
    - WASD, weil ich es gewohnt bin
    - Ich denk eine Kombination aus beiden wäre Sinnvoll. Im Dorf und Kampf die Click and Point Steuerung und fürs Laufen die WASD Steuerung
    - Die erste , da sie mir interaktiver und reaktioneller erschien als die zweite. Das Steuern der Kamera bei der zweiten Steuerart war zu ungewohnt und wirkte nicht ganz passend, da man leider weniger sah, wenn man herauszoomt. Außerdem hatte ich ein Problem bei der zweiten Steuerung mit dem langanhaltenden Drücken der Maustaste.
    - Die Standard-Steuerung, das es für die meisten die gewohnte Steuerung ist. Außerdem fiel dabei die Steuerung der Kamera leichter.
    - Standardsteuerung
    - definitiv die standard steuerung, da ich da viel besser auf das feedback der gegner reagieren konnte.
    - Standardsteuerung, da sie intuitiver ist, was aber auch von der Spielerfahrung abhängt.
    - Ich fand die Standard-Steuerung besser, da diese intuitiver war, aber ich glaube, dass wenn ich mit der PtC-Steuerung üben könnte, ich auch mit der besser klar kommen würde.
    - Einfacher für mich als "Nicht-Spieler" war Point to click, da ich mit der Maus besser zurecht komme, als mit der Tastenkombination.

12. Welche Verbesserungsvorschläge würden Sie dem Spiel geben?
    - Einen Kompass im Spiel integrieren, damit man die Himmelsrichtungen besser erkennen kann.
    - Keine weiteren, als die Punkte, die bereits oben beschrieben sind
    - Höhere Spawn-Raten, weiterer Kamera-Sichtbereich im Standard-Modus.
    - Die Angriffskraft der Gegner etwas runterschrauben Da man AP für den Angriff verbraucht, fände ich eine Art Spezialangriff gut, vor allem beim Kampf gegen mehrere Gegner Ein Kompass hätte mir beim Finden der Quest-Orte sehr geholfen
    - Kompass, vielleicht nicht ganz so weitläufige Welt (da man viel latschen muss), Schwert im Laufen wegstecken können
    - Kampfsystem reaktiver machen.
    - Quests gleichzeitig annehmen können, Karte / Kompass
    - kürzere Wege und dramatischere Questtexte, aber sonst gefällt mir das Spiel recht gut
    - Die Reichweite des Interagierens und des Angreifens zu erhöhen. Das Laufen außerhalb des Kämpfens vielleicht ausdauerunabhängig machen, damit man die langen Laufwege ununterbrochen zurücklegen kann.
    - Eine Karte zur Navigationshilfe. Eine größere Auswahl an Quest-Arten (Erkunden, Sammeln, etc...). Weitere Angriffstechniken, evtl. ein Skill-System Die Möglichkeit, Objekte (Bäume oder so) zu Kleinholz zu verarbeiten :) Sonst siehe Punkt 9.
    - siehe nr9
    - Deutlicheres Feedback beim blocken eines angriffs, zusätzliche fähigkeiten, damit das ap system besser genutzt werden kann. Diese würden auch die in der point-to-klick steuerung das spiel wesentlich besser machen, da ich als spieler dann mehr knöpfe zu drücken hätte größere Questmarker Kompass, damit ich weiss, in welche richtung ich mich bewege
    - Die Kameraführung bei der Point-to-klick Steuerung könnte noch durch einen Shortcut erweitern, der dafür sorgt, dass die Kamera in die Blickrichtung des Spielers zeigt. So müsste man nicht immer manuell dafür sorgen, dass die Kamera sich ausrichtet.
    - Ich habe leider keine direkten spielzerstörenden Bugs gefunden und würde mir etwas Variation in den Angriffen des Spielers, in Form von Skills, und der Gegner wünschen, damit der Kampf interessanter wird. Ich habe keinen signifikanten Unterschied zwischen den Gegner gefunden, außer der größe der Hitboxen.

13. Hat es Ihnen Spaß gemacht, dieses Spiel zu spielen?



14. Wie würden Sie das Spiel allgemein bewerten?



Durchschnittliche Bewertung 3.93