



# NoRPG: Entwicklung einer spielbasierten Lernspielplattform

## STUDIENARBEIT T2\_3201

des Studiengangs Angewandte Informatik  
an der Dualen Hochschule Baden-Württemberg Karlsruhe  
von  
**Mehmet Ali Incekara & Tom Wolske**

Abgabedatum 14. Mai 2017

<b>Bearbeitungszeitraum</b>	26 Wochen
<b>Matrikelnummer - Mehmet Ali Incekara</b>	5672336
<b>Matrikelnummer - Tom Wolske</b>	1156973
<b>Kurs</b>	TINF14B2
<b>Gutachter der Studienakademie</b>	Prof. Dr. Kay Berkling

# **Erklärung**

Gemäß §5 (2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 18. Mai 2009 erkläre ich hiermit,

1. dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
2. dass die Übernahme von Zitaten und Gedankengut anderer Autoren gekennzeichnet wurde.
3. dass die eingereichte elektronische Fassung exakt mit der schriftlichen übereinstimmt.
4. dass ich die Projektarbeit keiner externen Prüfung vorgelegt habe.

---

Karlsruhe, den 14. Mai 2017

---

Ort, Datum

---

Tom Wolske

---

Karlsruhe, den 14. Mai 2017

---

Ort, Datum

---

Mehmet Ali Incekara

# **Abstract**

This thesis deals with the implementation of a game-based learning game platform. The goal of this work corresponds to the fourth goal of the Global Goals quality education. It describes how to secure an integrated education system for all and to promote equal and high-quality lifelong learning opportunities. Through the implementation as a game, the children should be stimulated to have fun at learning.

However, to ensure that the learning process is structured, this goal is supported by the high-quality academic Common Core State standards for math and English. These outline the learning objectives a student should know and be capable of at the end of each class.

In the first half of this thesis, in addition to the theoretical foundations, all functional and non-functional requirements for the planned implementation are defined in a software requirements specification.

The implementation of NoRPG is described in the second half. In addition to the architecture and its components, the implementation of the user interface is also addressed and evaluated. The evaluation is carried out using a remote usability test. In the last part special components and their implementation are described. This includes, for example, the implementation of the complete system for the interactions.

The result of this work is an app, which is a good step in the right direction, to use learning games more meaningfully and to animate children an effective learning.

# Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Implementierung einer gamifizierten Lernspielplattform. Das Ziel dieser Arbeit entspricht dabei dem vierten Ziel der Global Goals hochwertigen Bildung. Diese beschreibt die Sicherung eines integrierenden Bildungssystems für alle und die Förderung von gleichberechtigten und hochwertigen lebenslangen Lernchancen. Durch die Implementierung als ein Spiel sollen die Kinder angeregt werden, Spaß am Lernen zu haben.

Damit allerdings das Lernen strukturiert stattfindet, wird dieses Ziel dabei durch die hochqualitativen akademischen Common Core State Standards für Mathe und Englisch unterstützt. Diese skizzieren Lernziele, welche ein Schüler am Ende einer jeden Klasse wissen und fähig sein sollte.

In der ersten Hälfte dieser Arbeit werden neben den theoretischen Grundlagen alle funktionalen und nicht-funktionalen Anforderungen für die geplante Umsetzung in einem Software Requirements Specification definiert.

In der zweiten Hälfte wird die Implementierung von NoRPG beschrieben. Neben der Architektur und deren Komponenten wird zudem die Umsetzung der Benutzeroberfläche behandelt und evaluiert. Die Evaluation wird mit Hilfe eines Remote Usability Tests durchgeführt. Im letzten Teil werden besondere Komponenten und deren Umsetzungen beschrieben. Dazu zählt beispielsweise die Implementierung des vollständigen Systems für die Interaktionen.

Das Ergebnis dieser Arbeit ist eine App, die einen guten Schritt in die richtige Richtung ist, um Lernspiele sinnvoller zu nutzen und Kinder effektiv zu animieren etwas lernen zu wollen.

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Ziel der Arbeit . . . . .	2
1.3 Aufbau der Arbeit . . . . .	3
<b>2 NoRPG</b>	<b>4</b>
2.1 The Global Goals . . . . .	5
2.2 Common Core State Standards . . . . .	6
2.3 Gamifizierung . . . . .	7
2.4 Die Geschichte . . . . .	8
<b>3 Software Requirements Specification</b>	<b>11</b>
3.1 Einführung . . . . .	11
3.1.1 Zweck . . . . .	11
3.1.2 Umfang . . . . .	11
3.2 Allgemeine Beschreibung . . . . .	12
3.2.1 Produktperspektive . . . . .	12
3.2.2 Produktfunktionen . . . . .	13
3.2.3 Benutzermerkmale . . . . .	14
3.2.4 Einschränkungen . . . . .	14
3.2.5 Annahmen und Abhängigkeiten . . . . .	15
3.2.6 Aufteilung der Anforderungen . . . . .	15
3.3 Spezifische Anforderungen . . . . .	15
3.3.1 Externe Schnittstellen . . . . .	16
3.3.2 Funktionale Anforderungen . . . . .	21
3.3.3 Performanz Anforderungen . . . . .	30
3.3.4 Datenbank Anforderungen . . . . .	31
3.3.5 Entwurfsbeschränkungen . . . . .	32
3.3.6 Benutzerfreundlichkeit . . . . .	32
3.3.7 Zuverlässigkeit . . . . .	34
3.3.8 Verfügbarkeit . . . . .	34

3.3.9	Sicherheit . . . . .	35
3.3.10	Wartbarkeit . . . . .	35
3.3.11	Portabilität . . . . .	36
<b>4</b>	<b>Technische Grundlagen</b>	<b>37</b>
4.1	Unity . . . . .	37
4.1.1	Komponenten der Entwicklungsumgebung . . . . .	37
4.1.2	Grafische Oberflächen mit Unity . . . . .	39
4.2	Visual Studio . . . . .	42
4.3	C Sharp . . . . .	42
4.3.1	Allgemeiner aufbau C# . . . . .	42
4.3.2	Unity Skripte . . . . .	43
<b>5</b>	<b>Umsetzung</b>	<b>45</b>
5.1	Architektur . . . . .	45
5.2	Datenhaltung und Anbindung an eine Datenbank . . . . .	46
5.2.1	Anbindung an einen Server . . . . .	46
5.2.2	Datenhaltung auf dem Handy . . . . .	48
5.3	Benutzerschnittstelle . . . . .	50
5.3.1	Benutzeroberfläche . . . . .	50
5.3.2	Usability Evaluation . . . . .	57
5.4	C# Skripte . . . . .	62
5.4.1	Player . . . . .	62
5.4.2	Kamera . . . . .	64
5.4.3	Portale . . . . .	65
5.4.4	Interaktionsmöglichkeiten . . . . .	67
5.4.5	Pfadfindungssystem Schiff . . . . .	69
<b>6</b>	<b>Fazit und Ausblick</b>	<b>73</b>
<b>Literatur</b>		<b>IX</b>
<b>Anhang</b>		<b>XI</b>

# **Abkürzungsverzeichnis**

<b>SRS</b>	Software Requirements Specification
<b>RPG</b>	Role Player Game
<b>MMORPG</b>	Massive Multiplayer Online Role Player Game
<b>MOOC</b>	Massive Open Online Course
<b>CCSS</b>	Common Core State Standard
<b>APK</b>	Android Application Package
<b>UI</b>	User Interfaces
<b>HUD</b>	Head-Up Display
<b>DBMS</b>	Datenbank Management System
<b>NPC</b>	Non-Player Character
<b>JSON</b>	Javascript Object Notation
<b>MVC</b>	Model-View-Controller
<b>JIT</b>	Just-In-Time

# Abbildungsverzeichnis

1	High-Level-View von NoRPG . . . . .	12
2	Mockup: Login-Screen . . . . .	17
3	Mockups: Registrierungsformular und Charaktererstellung . . . . .	18
4	Mockup: Head-Up Display . . . . .	18
5	Mockup: Menü . . . . .	19
6	Mockups: Spielliste, Fortschrittsanzeige, Karte und Erfolgsübersicht . . . . .	19
7	Overall Use Case Diagramm . . . . .	21
8	Activity Diagramm: Create Character . . . . .	22
9	Activity Diagramm: Open Map . . . . .	23
10	Activity Diagramm: Show Games . . . . .	24
11	Activity Diagramm: View Progress . . . . .	25
12	Activity Diagramm: Achievements . . . . .	26
13	Activity Diagramm: Character Control . . . . .	28
14	Activity Diagramm: Game Interaction . . . . .	30
15	Darstellung von Tablemappings . . . . .	38
16	Render-Modus: Screen Space - Overlay . . . . .	40
17	Render-Modus: Screen Space - Camera . . . . .	41
18	Render-Modus: World Space . . . . .	42
19	Softwarearchitektur . . . . .	45
20	User Interface: Startbildschirm . . . . .	51
21	User Interface: Registrierung . . . . .	52
22	User Interface: Head-Up Display . . . . .	54
23	User Interfaces: Kommunikation, Menü und Einstellungen-Fenster . . . . .	55
24	Evaluation Auswertung - Der durchschnittliche Teste . . . . .	59
25	Evaluation Auswertung - Die durchschnittliche Hard- und Software . . . . .	60

# Listings

1	Hello World in C# . . . . .	43
2	Aufbau eines Unity Skriptes . . . . .	44
3	Skript SendDataToServer.cs . . . . .	47
4	Methode LoadSettings . . . . .	49
5	CharacterController.cs . . . . .	63
6	Methode LateUpdate aus ThirdPersonCamera.cs . . . . .	64
7	PortalToTargetScene . . . . .	65
8	NPCCommunication.cs . . . . .	67
9	NPC JSON . . . . .	68
10	Skript EditorPathScript.cs . . . . .	69
11	Methode selectFirstClass . . . . .	70
12	Methode playerGetOnShip . . . . .	71
13	Methode moveShip . . . . .	71
14	LoadingScreen.cs . . . . .	XX
15	IConfigReader.cs . . . . .	XX
16	MappingStandardsToCoursesBean.cs . . . . .	XX
17	MappingStandardsToCourses.cs . . . . .	XXI

# 1 Einleitung

Spiele, jeder kennt sie und spielt regelmäßig welche. Sie sind ein fester Bestandteil unserer Kultur und das schon seit tausenden Jahren. Die ersten Gesellschaftsspiele wurden noch im Sand mit Stöcken oder Steinen gespielt. Eines der frühesten Spiele wird auch heutzutage noch gespielt, dabei handelt es sich um Mühle, ein Spiel welches die Ägypter vor bereits 3000 Jahren gespielt haben.<sup>1</sup> Spiele haben sich seit dem jedoch weiterentwickelt und dienen heutzutage nicht nur zum munteren Zeitvertreib.

Ob als Brett, Karten oder Glücksspiel, Spiele sind überall zu finden und jeder kann sie spielen. Seit 1972 entwickeln sich darüber hinaus weitere Spiele, Videospiele. Sie nutzen die immer größer werdende Rechenleistung von Computern aus, um uns immer realistisch ausschauender Spiele zu liefern. Um den Überblick über die Vielzahl an Videospiele zu behalten, haben sich in den letzten Jahren verschiedene Plattformen etabliert, die versuchen dem Nutzer das zu bieten, was sie suchen. Dabei stellen diese viele verschiedene Arten von Spielen für die Gamer bereit, die einen beim Spielen die Zeit vergessen lassen. Beispiele für solche Plattformen sind unter anderem Steam, Uplay oder Origin.

Allerdings können Spiele uns nicht nur die Zeit vergessen lassen und für heitere Stunden sorgen, sie können uns auch Wissen vermitteln. Sei es beispielsweise durch eine Geschichte die sich real abgespielt hat, wie der erste Weltkrieg. Dieses Wissen wird unterbewusst an den Nutzer vermittelt, ohne dass er aktiv versucht dieses zu lernen.

Für diesen Zweig hat sich eine eigene Branche entwickelt, welche sich mit Lernspielen befasst und versucht uns, über Videospiele, neues Wissen zu vermitteln. Viele dieser Spiele nutzen bekannte Figuren, welche die Kinder aus dem Fernsehen kennen, um dieses Wissen zu vermitteln.

Diese Spiele werden hauptsächlich in den Schulen eingesetzt, um den Kindern wissen spielerisch zu vermitteln. Jedoch profitiert nicht jedes Kind von diesem Vorteil. Sei es, weil die Schule keine Computer hat oder weil das Kind nicht eine Schule besuchen kann. Für diesen Zweck wurde die Plattform Hone<sup>2</sup> entwickelt, mit der Kinder, die nicht zur Schule gehen können, die Möglichkeit haben, Wissen zu erlangen.

---

<sup>1</sup> vgl. gesellschaftsspiele.de [5] (2015)

<sup>2</sup> siehe <http://hone-kids.herokuapp.com/>

## **1.1 Motivation**

Bei Hone handelt es sich um eine Spielplattform auf der sich Kinder, bevorzugt aus Regionen in denen Bildung mangelhaft ist, anmelden können. Auf dieser Web-Plattform gibt es für Kinder die Möglichkeit neue Lernspiele für verschiedene Plattformen herunterzuladen. Zusätzlich gibt es eine Ansicht der gelernten Kompetenzen.

Dieses Konzept hat zwei wesentliche Nachteile für die Benutzer. Für die Verwendung der Web-Plattform wird ein Computer benötigt und gerade weil Spiele für verschiedene Plattformen angeboten werden können, benötigt das Kind mehrere Geräte. Neben diesen Nachteilen, ist das Aussehen und die Bedienung dieser Plattform nicht reizvoll für Kinder gestaltet.

Mehr Kinder als je zuvor in der Geschichte arbeiten täglich mit immer neueren und besseren technischen mobilen Geräten. Deshalb soll eine mobile Applikation, kurz App, für Smartphones entwickelt werden, in der die Kinder auf spielerischer Weise Fortschritte machen. Durch die Umsetzung als App wird den Kindern eine Plattform angeboten, mit welcher sie unabhängig und jederzeit auf die Lerninhalte zugreifen können. Ein weiterer Vorteil ist, dass die Kinder den technischen Umgang mit Smartphones lernen. Die App wird unabhängig von Hone funktionieren und es werden keine Inhalte und Funktionen übernommen.

## **1.2 Ziel der Arbeit**

Das Ziel dieses Projektes ist es den Kindern eine Möglichkeit zu geben, mit der sie jederzeit, überall und einfach auf die Lerninhalte zugreifen und spielerisch neues Wissen erwerben können.

Dabei ist es nicht das Ziel die Lerninhalte direkt in der App abzufragen sondern Lernspiele für Smartphones anzubieten, welche es in korrekter Reihenfolge freizuschalten und zu spielen gilt.

Das Ziel dieser Arbeit ist die Vorgehensweise, Bedingungen, Probleme zu dokumentieren. Mit dieser Dokumentation soll gewährleistet werden, dass dieses Projekt von allen Verstanden wird.

## **1.3 Aufbau der Arbeit**

Die einzelnen Kapitel dieser Arbeit repräsentieren die notwendigen Schritte, das Ziel dieses Projektes zu erreichen.

Bevor überhaupt Anforderungen spezifiziert werden können, müssen zunächst die Ideen und Gründe hinter diesem Projekt beschrieben werden. Das nächste Kapitel behandelt deshalb das Konzept der App. Zudem wird das beschriebene Konzept von anderen Spielkonzepten abgegrenzt, um die Unterschiede klarzustellen.

Im darauf folgendem dritten Kapitel wird das zum Projekt dazugehörige Software Requirements Specification (SRS) behandelt. Dies dient zur Spezifikation der App. Neben funktionalen Anforderungen werden hier auch nicht-funktionale Anforderungen festgeschrieben.

Anschließend wird auf die technischen Grundlagen für die Umsetzung eingegangen. Dabei wird die gewählte Entwicklungsumgebung und weitere notwendigen Technologien beschrieben.

Darauf aufbauend wird im fünften Kapitel die Umsetzung behandelt. Dabei wird auf Besonderheiten und Probleme in der Entwicklungsphase eingegangen. Dafür werden die einzelnen Komponenten der App beschrieben.

Abgeschlossen wird diese Arbeit mit einem Fazit und Ausblick, in dem der weitere Werdegang des Projektes geschildert wird.

## 2 NoRPG

Bei NoRPG handelt es sich um eine Gamifizierung einer Lernspielplattform. Dabei soll NoRPG an ein Rollenspiel, bzw. Role Player Game (RPG), erinnern und implementiert dessen charakteristische Eigenschaften. Eine dieser Eigenschaften von RPGs ist, dass der Spielende in die Rolle realer Menschen, fiktiver Figuren, Tiere oder auch Gegenstände einer fiktiven Welt schlüpft<sup>3</sup>. Dabei wird eine Story erzählt, die das Kind erleben kann. Damit der Spieler allerdings in der Geschichte vorankommt, muss er verschiedene Missionen bzw. Quests erledigen. Dabei kann es sich um die verschiedensten Aufgaben handeln. Darüber hinaus sammelt der Spieler Objekte in der Welt, welche er anschließend im Spiel nutzen kann. Ein Beispiel für solche Spiele ist das Spiel The Witcher 3, welches auf diesen Prinzipien aufbaut<sup>4</sup>.

Des Weiteren gibt es noch Massive Multiplayer Online Role Player Games (MMORPGs). Dabei gibt es wie bei RPGs eine Story und Quests, allerdings kann man auch auf andere Spieler treffen und mit ihnen gemeinsam spielen. Das wohl berühmteste MMORPG ist dabei World of Warcraft<sup>5</sup> vom Entwickler Blizzard. In MMORPGs gibt es die Möglichkeit verschiedene Events mit mehreren Spieler gemeinsam Quests zu erfüllen. Dieser Mehrspieler-Modus grenzt die MMORPGs von den RPGs ab.

Jedoch handelt es sich bei NoRPG letztendlich nicht um ein klassisches RPG oder MMORPG, sondern um eine Lernspielplattform und bietet Lernspiele zum Herunterladen an. NoRPG soll durch die Eigenschaften eines Rollenspiels die Spieler dazu anregen, weitere Lernspiele herunterzuladen und zu spielen. Deshalb wurde sich für den Namen NoRPG entschieden, da nicht alle charakteristischen Eigenschaften implementiert werden.

NoRPG ist allerdings auch nicht mit einem Massive Open Online Course (MOOC) zu verwechseln, sondern baut nur auf einem auf. Ein MOOC bezeichnet einen kostenlosen Onlinekurs. Ein MOOC würde selbst die Lerninhalte anbieten<sup>6</sup>, wohingegen in NoRPG nur Spiele angeboten werden, die den Lerninhalt bereitstellen.

---

<sup>3</sup> vgl. Warwitz und Rudolf [rpgSinn] Seite 78ff.

<sup>4</sup> für weitere Informationen siehe <http://thewitcher.com/en/witcher3>

<sup>5</sup> für weitere Informationen siehe <https://worldofwarcraft.com/>

<sup>6</sup> Vgl. Porter [moocBook] Seite 3f.

## 2.1 The Global Goals

NoRPG ist ein Spiel, welches versucht Bildung für jeden erreichbar zu machen. Dieses Ziel ist dabei in den Global Goals definiert. Dabei handelt es sich um 17 Ziele welche bis 2030 Umgesetzt werden sollen, um das Leben für alle Menschen auf der Welt zu verbessern<sup>7</sup>. 2015 haben 193 Weltführer diese Unterzeichnet und begonnen dieses umzusetzen. Dabei sind diese Ziele umfangreich und reichen von einem besseren Umgang mit den uns zur Verfügung stehenden Ressourcen bis hin zu qualitativ hochwertiger Bildung für jeden und kostenlos.

NoRPG unterstützt dabei das Ziel, hochwertige Bildung für jeden zugänglich zu machen. Dieses Ziel wird beschrieben als Sicherung eines integrierenden Bildungssystems für alle und die Förderung von gleichberechtigten und hochwertigen lebenslangen Lernchancen<sup>8</sup>. Dieses Ziel hat weitere Unterziele, wobei nun kurz auf die für NoRPG relevanten Unterziele eingegangen wird.

- Bis 2030 sicherstellen, dass alle Mädchen und Jungen gleichberechtigt eine kostenlose und hochwertige Grund- und Sekundarschulbildung abschließen, die zu brauchbaren und effektiven Lernergebnissen führt.
- Aufbau und Weiterentwicklung von Bildungseinrichtungen, die Kinder- und Behindertengerecht und Geschlechtsspezifisch sind und für alle eine sichere, gewaltfreie, integrative und effektive Lernumgebung bieten.

Das erste Unterziel wird in NoRPG dahingehend unterstützt, dass die Common Core State Standards (CCSSs) implementiert werden. Da NoRPG für alle kostenfrei zugänglich ist und Mädchen und Jungen gleichberechtigt sind, werden auch diese zwei Aspekte des Unterziels unterstützt.

Da NoRPG keine Bildungseinrichtung ist wird das zweite Unterziel nur bedingt erfüllt. NoRPG bietet Kindern jedoch eine sichere, gewaltfreie, integrative und effektive Lernumgebung, wodurch dieses Ziel allerdings zum Teil erfüllt wird. Darüber hinaus kann diese Anwendung in Bildungseinrichtungen angewendet werden. Geschlechtsspezifisch ist das Spiel nur dahingehend, dass die Kinder zu Beginn das Geschlecht des eigenen Charakters auswählen können.

---

<sup>7</sup> vgl. Global Goals [6] (2017)

<sup>8</sup> <http://www.globalgoals.org/de/global-goals/quality-education/>

## 2.2 Common Core State Standards

Die CCSSs sind ein Set von hochqualitativen akademischen Standards für Mathe und Englisch. Diese Lernziele skizzieren, was ein Schüler am Ende jeder Klasse wissen und fähig sein sollte. Die Standards wurden geschaffen um sicherzustellen, dass alle Schüler mit den gelernten Fähigkeiten und Kenntnissen im College, Karriere und im Leben, egal wo sie leben, erfolgreich zu sein.

Das Problem des Schulsystems ist, dass die Standards von Staat zu Staat variieren und stimmen meistens nicht mit dem überein, was die Kinder wissen sollten. In Erkennung der Notwendigkeit konsequenter Lernziele wurden die CCSSs entwickelt<sup>9</sup>.

Die Standards werden dabei von den zur Verfügung gestellten Spielen erfüllt. NoRPG sorgt dafür, dass diese Standards in der korrekten Reihenfolge und vollständig erfüllt werden. Betrachtet werden zunächst nur die Standards für die ersten fünf Klassen. Die Abarbeitung der Standards entspricht nicht dem klassischen Vorgehen von Schulen, so ist es beispielsweise nicht notwendig alle Standards der ersten Klasse zu erfüllen, um Stoff der zweiten Klasse freizuschalten. So ist es möglich, wenn ein Kind den Standard Geometrie der ersten Klasse vollständig erfüllt, schon weiter mit Geometrie für die zweite Klasse fortzufahren, ohne das alle anderen Mathestandards der ersten Klasse vollständig abgeschlossen wurden.

### Mathe

Die CCSSs konzentrieren sich auf eine klare Reihe von mathematischen Fähigkeiten und Konzepten. Die Schüler lernen Konzepte in einer organisierten Weise. Die Standards ermutigen die Schüler, reale Probleme zu lösen<sup>10</sup>.

Das Fach Mathe lässt sich für die ersten fünf Klassen in insgesamt fünf Themenbereiche eingliedern. Zu den Themen gehören beispielsweise algebraisches Denken, Operationen im Zahlenraum bis 100, Maßeinheiten und Geometrie. Jedes dieser einzelnen Themen sind nochmals in Standards unterteilt. Erst durch diese genaue Gliederung wird es ermöglicht, dass alle wichtigen Inhalte abgedeckt werden. Damit das vorhin beschriebene Konzept umgesetzt werden kann, mussten sehr früh alle Abhängigkeiten zwischen den Standards ermittelt und visualisiert werden. Der vollständige Abhängigkeitsbaum ist im Anhang 6 dieser Arbeit zu finden.

<sup>9</sup> Vgl. <http://www.corestandards.org/about-the-standards/>

<sup>10</sup> Vgl. <http://www.corestandards.org/Math/>

## **Englisch**

Die CCSSs bittet die Schüler, Geschichten und Literatur zu lesen, sowie komplexere Texte, die Fakten und Hintergrundwissen in Bereichen wie Wissenschaft und Sozialwissenschaften liefern. Die Schüler werden herausgefordert und gefragt, was sie dazu bringen, auf das zurückzugreifen, was sie gelesen haben. Dies unterstreicht kritisches Denken, Problemlösung und analytische Fähigkeiten, die für den Erfolg in College, Karriere und Leben erforderlich sind<sup>11</sup>.

Das Fach Englisch lässt sich für die ersten fünf Klassen in insgesamt sechs Themenbereiche eingliedern. Zu den Themen gehört Lesen von unterschiedlichen Texten, Schreiben, Sprechen, Zuhören sowie Grammatik. Die Gliederung dieser Themen entspricht den Standards. Auch hier ist der Abhängigkeitsbaum im Anhang 6 dieser Arbeit zu finden.

## **2.3 Gamifizierung**

Dieses Ziel der Global Goals soll dabei durch die Gamifizierung der CCSSs umgesetzt werden. Gamifizierung bzw. Gamification bezieht sich auf die Analyse von spielespezifischen Eigenschaften, welche die Spiele unterhaltsam machen und diese dann in Situationen außerhalb von Spielen anzuwenden, um das Gefühl von Spaß für neue Anwendungen, wie Lernen oder Marketing, zu übertragen<sup>12</sup>. Ein Beispiel dafür ist Pay-Back. Dabei sammeln die Nutzer bei jedem Einkauf Punkte. Diese können die Kunden dann gegen Prämien eintauschen. In Videospielen sammeln die Spieler zum Beispiel Münzen um diese anschließend gegen Gegenstände einzutauschen.

Gamifizierung verwendet darüber hinaus noch weitere erfolgreiche Prinzipien aus Videospiele, um die Nutzer zu Motivieren das Produkt zu nutzen. Eines der wichtigsten Prinzipien ist die Einbettung der Handlungen in eine Geschichte. Die Erzählung einer bindenden und spannende Geschichte hilft dem Spieler sich in die Rolle seines Charakters zu versetzen und eine emotionale Bindung herzustellen. Dadurch wird das Spielerlebnis intensiver und macht dem Spieler auch mehr Spaß. Die Wichtigkeit von Spaß ist dabei nicht zu unterschätzen. Spaß motiviert die Kinder, auf eigene Faust zu lernen und das Spiel mehr zu spielen<sup>13</sup>.

Die Möglichkeit zu wählen ist ein weiteres wichtiges Element von Spielen. Gäbe es nicht die Möglichkeit Entscheidungen zu treffen würde es tatsächlich sich um Filme

<sup>11</sup> Vgl. <http://www.corestandards.org/ELA-Literacy/>

<sup>12</sup> Umformuliert vom Oxford Dictionary

<sup>13</sup> Vgl. Michael und Chen [seriousGamesFun] Seite 40

oder Bücher handeln. Diese Medien haben keine Wahlmöglichkeiten, entweder der Betrachter ist aufmerksam oder nicht. Die Wahl, in Bezug auf die Handhabung einer bestimmten Herausforderung als aktiver Teilnehmer, Kontrolle über das eigene Lernen zu haben, macht dem Spieler die eigenen Entscheidungen und deren Konsequenzen bewusster<sup>14</sup>.

Ein weiteres Prinzip welches oft verwendet wird ist direktes Feedback. Dieses trifft in den meisten Fällen zusammen mit dem Prinzip von Belohnungen auf. Es ist schwer, sich das Leben ohne jegliche Rückmeldung vorzustellen, in der Tat ist es unmöglich. Rückmeldungen sind Informationen welche den Menschen helfen die Welt um uns herum besser zu verstehen und dies ist entscheidend für die Weiterentwicklung. Durch zusätzliche Belohnungen für besondere bzw. korrekte Leistungen wird dieser Effekt verstärkt<sup>15</sup>. Allerdings ist zu beachten, dass die Belohnungen einen Nutzen für den Spieler darstellen, damit es sich lohnt Aufwand für diese Belohnungen aufzuwenden<sup>16</sup>. Vervollständigt werden diese Prinzipien durch die Implementierung eines Spielstatus durch Level und Auszeichnungen.

Weitere Prinzipien, die Onlinespiele ausmachen, ist zunächst der ständige Wettbewerb mit anderen Spieler und Möglichkeit Aktivitäten zusammen in einem Team mit anderen Spieler zu bewältigen. Darüber hinaus sind die Faktoren wie Erfolgsergebnisse, Gruppenzugehörigkeit und soziale Akzeptanz wichtig.

Im nächsten Unterkapitel wird die Idee zur Umsetzung des Prinzips der Einbettung der Handlungen des Spielers in eine Geschichte behandelt.

## 2.4 Die Geschichte

In NoRPG spielt der Spieler einen Charakter, der in dem Dorf Rutherglen wohnt. Am Anfang der Geschichte wird dieses Dorf von dem Bösewichten heimgesucht. Dieser klaut alle Farben das komplette Dorfes und alles wirkt farblos und somit auch emotionslos. Nun hat der Spieler, das Ziel die Farben wieder zurück zu bringen. Dazu muss der Spieler in verschiedenen Welten gehen und verschiedene Gegenstände finden.

Die einzelnen Welten repräsentieren eine Klassenstufe und sind dementsprechend anspruchsvoll und unterschiedlich gestaltet. Die Standards der ersten Klasse sind in der ersten Welt. Äquivalent verhält es sich auch in den anderen Welten. Neben den Standards gibt es noch Truhen zu finden, die den Spieler in der Geschichte von NoRPG

---

<sup>14</sup> Vgl. Routledge [seriousGamesPrinciples] Seite 30f.

<sup>15</sup> Vgl. Routledge [seriousGamesPrinciples] Seite 32ff.

<sup>16</sup> Vgl. Berkling, Faller und Piertzik [gamesPaper] Seite 8

voranbringen. Jede Welt ist dabei in Unterbereiche gegliedert, die der Spieler mit der Zeit erreichen kann. Dabei wird der zu erkundende Bereich immer größer, je weiter der Spieler in der Geschichte vorankommt. Die Truhen sind dabei unabhängig vom zu lernenden Lernstoff. Der Spieler muss Lernspiele und somit Standards erfüllen, um in die verschiedenen Welten freizuschalten. In den Welten allerdings kann der Spieler frei die Gegenstände aus den Truhen suchen und sammeln. Für jeden gefundenen Edelstein kehrt ein Teil der gestohlenen Farbe zurück.

Eine sehr markante Eigenschaft von RPGs ist, dass die Spielwelt offen gestaltet ist und dem Spieler nur eine ganz grobe Vorgehensweise vorgegeben wird. Der Spieler kann sich frei in der jeder Welt bewegen, kann sich mit allen Bewohnern unterhalten und verschiedene Sachen, wie beispielsweise Ruinen, etc., entdecken. Dies wird als Open World bezeichnet. Dem Spieler werden nur Grenzen vorgegeben, in denen das Kind seine eigene Herangehensweise entwickeln kann. Eine grobe Vorgabe ist notwendig, damit das lernen strukturiert stattfindet.

## **Startwelt: Das Dorf Rutherglen**

Rutherglen ist die Heimat des Spielers und dient als Brücke zwischen allen Welten. In die verschiedenen Welten gelangt der Spieler über Portale, die in Rutherglen verteilt sind. Bei dem Heimort des Spielers handelt es sich um ein verschlagenes, unscheinbares und ruhiges Dorf. Der Spieler kann in dieser Welt mit allen Bewohnern interagieren und sprechen. Diese erzählen Geschichten über das Dorf, über ihr Leben oder geben Tipps und Hinweise.

## **Welt 1: Die dichten Wälder Talhan**

Talhan ist das Waldgebiet von Rutherglen. In dieser Welt ist der Wald das primäre Element. Dieser ist in verschiedenen Ausprägungen vorhanden. Das Kind findet sich in einem großen Wald mit verschiedenen Baumarten wieder: Von Laubbäumen, über tropische Bäume bis hin zu großen Fichtenbäume. Neben diesen Elementen gibt es noch weitere Sachen zu entdecken, die die Reise des Kindes durch die Wälder Talhan spannender gestalten soll.

## **Welt 2: Die tropischen Inseln von Galapagos**

Bei der nächsten Welt handelt es sich um die tropischen Inseln von Galapagos. Diese Welt ist im Gegensatz zum Wald sehr farbenfroh, allerdings gibt es auch viel Wasser.

Galapagos besteht dabei aus mehreren Inseln, welche mit dem Schiff erreicht werden können. Auch hier gibt es auf den verschiedenen Inseln unterschiedliche Szenen zu entdecken.

### **Welt 3: Die endlose Wüste Kalahari**

Kalahari ist eine sehr große Wüste und ist trostloser wie die vorherigen Welten. In dieser Welt wird der Begriff Open World sehr deutlich. Im Vergleich zu fast allen anderen Welten gibt es keinen Weg oder Wegweiser. Dieses Open World Konzept hat den Sinn, das Gefühl einer endlosen riesigen Wüste zu transportieren. Innerhalb dieser Wüste kann das Kind Oasen, verschiedene Ruinen und ägyptische Wahrzeichen, wie beispielsweise Pyramiden, entdecken.

### **Welt 4: Das verschneite Gebirge Lhotse**

Die vorletzte und größte Welt sind die verschneite Gebirge Lhotse. In dieser Eiswelt dreht sich alles um Eis und Schnee. Neben der Wüste ist diese Welt die zweite Welt, in der das Prinzip Open World sehr deutlich wird. Es gibt viele hohe Berge, verschiedene große Höhlen und weitere Sachen.

### **Welt 5: Der Vulkan Ätna**

Die zunächst letzte Welt ist Ätna. Hier dreht sich alles um Feuer. In dieser Welt gibt es verschiedene Inselplattformen, welche durch Lava getrennt sind. Allerdings sind einige untereinander mit Brücken verbunden. Auf den verschiedenen Inseln kann der Spieler verschiedene Dinge erkunden, darunter Drachen, Vulkane oder Ruinen. Nach dem das Kind auch diese Welt bewältigt, sind alle Farben wieder zurück im Dorf.

# **3 Software Requirements Specification**

Das Software Requirements Specification (SRS) ist ein veröffentlichter Standard zur Spezifikation einer Software. Der Inhalt eines SRS ist vom Institute of Electrical and Electronics Engineers im Standard IEEE 830-1998 festgehalten.

Der Aufbau dieses Kapitels entspricht der Struktur, die in dem Standard beschrieben wird. Einige Kapitel des SRS werden allerdings nicht behandelt, da sie keine Relevanz für NoRPG haben oder an einer anderer Stelle dokumentiert werden.

## **3.1 Einführung**

Das erste Kapitel des SRS enthält eine Beschreibung und eine Übersicht über alles, was im SRS enthalten ist.

### **3.1.1 Zweck**

Das SRS beschreibt den kompletten Projektumfang und die Anforderungen an die Software NoRPG. Es illustriert den Zweck und die vollständige Erklärung für die Entwicklung der Software. Dabei werden unter anderem Systemeinschränkungen, Schnittstellen und Interaktionen mit externen Schnittstellen thematisiert<sup>17</sup>.

Die Zielgruppe des SRS bzw. die Stakeholder des Projektes sind alle Personen und Personengruppen, die in irgendeiner Verbindung mit NoRPG stehen oder jene, die Interesse an der Umsetzung haben<sup>18</sup>. Zudem dient die Spezifikation zur Kommunikation zwischen Stakeholdern und Entwicklern.

### **3.1.2 Umfang**

Dieses SRS handelt von der in Kapitel 2 beschrieben Software NoRPG.

---

<sup>17</sup> vgl. Tripp [2](1998) Seite 3

<sup>18</sup> vgl. Rozanski [12](2011) Seite 6

## 3.2 Allgemeine Beschreibung

Im zweiten Kapitel des SRS werden alle allgemeinen Faktoren, die das Produkt und seine Anforderungen betreffen, beschrieben. Dieses Kapitel bietet einen Überblick über die Systemfunktionalitäten und stellt verschiedene Arten von Stakeholdern und deren Interaktionen mit dem System vor. Dieses Kapitel behandelt jedoch nicht die spezifischen Anforderungen, sondern stellt den Hintergrund für diese dar.

### 3.2.1 Produktperspektive

Das zu beschreibende vollständige System NoRPG besteht aus mehreren Komponenten, die auf unterschiedlichsten weisen mit den Stakeholdern kommunizieren. Daher ist es besonders wichtig, das Produkt in unterschiedlichen Perspektiven mit verwandten und geplanten Produkten zu betrachten. Aus diesem Grund werden alle System-, Benutzer-, Hardware- und Softwareschnittstellen von NoRPG definiert.

Folgende Grafik 1 stellt dabei die High-Level-View von NoRPG und seinen Komponenten dar.

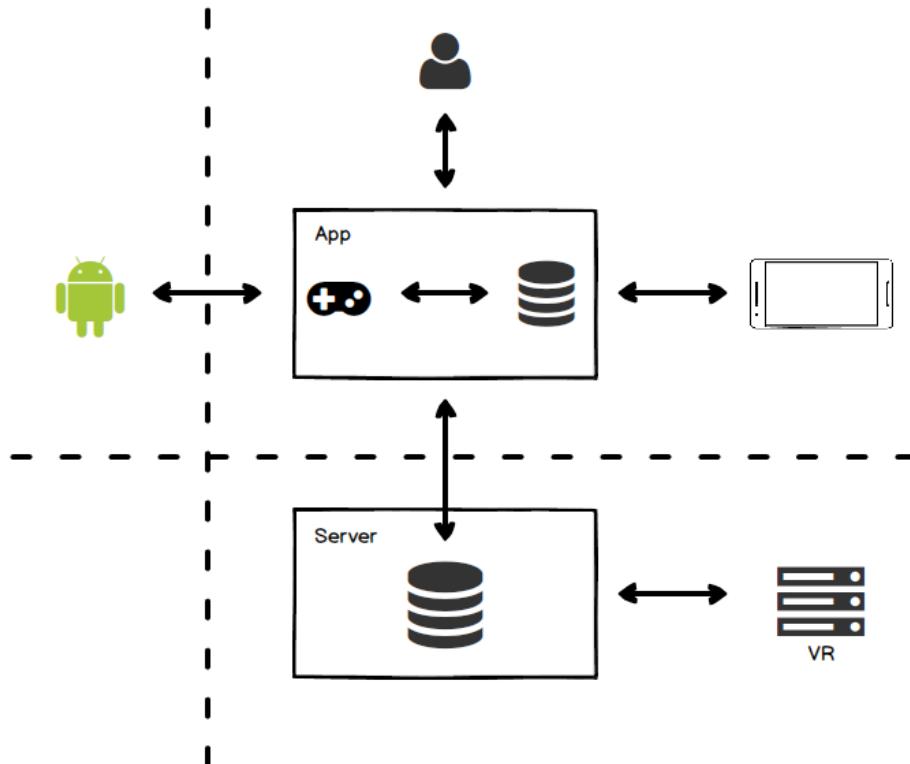


Abbildung 1: High-Level-View von NoRPG

Das vollständige System von NoRPG besteht aus zwei Kernkomponenten. Die erste Kernkomponente ist die Android App, welche sich aus dem Quellcode des Spieles und einer eingebetteten lokalen Datenbank zusammensetzt. Die zweite Kernkomponente ist eine Datenbank, die sich auf einem virtualisierten Server befindet.

Die Schnittstelle zwischen der App und dem Betriebssystem des Smartphones ist eine Systemschnittstelle. Systemschnittstellen identifizieren die Funktionalität der Software, um die Systemanforderung und Schnittstellenbeschreibung zu erfüllen, damit die Software mit dem System übereinstimmt<sup>19</sup>.

Bei der App NoRPG handelt es sich um die Benutzerschnittstelle des Systems. Benutzerschnittstellen beschreiben die Kommunikation zwischen dem System und dem User. Die Benutzeroberfläche der App ist die einzige Möglichkeit für den Anwender mit dem System zu interagieren.

Jede Schnittstelle zwischen NoRPG und Hardwarekomponenten des Systems werden als Hardwareschnittstellen bezeichnet. Das Smartphone mit all seinen Komponenten sind Hardwarekomponente, zu der eine direkte Schnittstelle existiert. Die Hardwarekomponenten eines Smartphones sind der Touchscreen, die Lautsprecher oder der WLAN-Adapter. Diese Komponenten werden in der Abbildung durch das Smartphone zusammengefasst. Eine weitere Hardwareschnittstelle gibt es zwischen dem Server und der Hardware, auf dem dieser installiert ist.

Die App kommuniziert mit der lokalen, sowie mit der serverseitigen Datenbank und verwendet dabei die Funktionen von anderen Softwareprodukten. Dabei handelt es sich um Softwareschnittstellen. Sie bilden den Übergang zwischen unterschiedlichen Programmen und ermöglichen dadurch das Nutzen derer Funktionalitäten.

### **3.2.2 Produktfunktionen**

In diesem Unterkapitel werden die wichtigsten Funktionen von NoRPG zusammengefasst. Wie in Kapitel 2 beschrieben ist das Hauptziel von NoRPG Lernspiele in einer standardisierten Reihenfolge zum Herunterladen anzubieten. Dieses Ziel macht das Downloaden von Spielen zu der Hauptfunktionalität von NoRPG.

Neben dieser Funktion gibt es allerdings weitere Produktfunktionen, um NoRPG attraktiver zu gestalten und zu personalisieren. Der Spieler wird in der Lage sein mit Elementen im Spiel zu interagieren und dabei Spielgegenstände zu sammeln. Um dem Anwender eindeutig identifizieren zu können, wird NoRPG über eine eigene

---

<sup>19</sup> vgl. Tripp [2](1998) Seite 13

Anmelde- und Registrierungsfunktion verfügen. In diesem Registrierungsprozess ist es dem Spieler möglich, seinen eigenen persönlichen Charakter zu erstellen.

Im Spiel selbst wird es neben Spieloptionen wie Qualitäts- und Audioeinstellungen noch Features geben, die das Spielerlebnis verbessern sollen. Dazu zählen Funktionen wie das öffnen einer Karte der aktuellen Spielwelt oder das betrachten des Fortschritts im jeweiligen Standard.

Die App speichert den Fortschritt und die Daten in der lokalen eingebetteten Datenbank und synchronisiert diese Informationen mit dem Server.

### **3.2.3 Benutzermerkmale**

Im Rahmen dieser Studienarbeit wird zunächst nur eine Benutzergruppe vollständig implementiert und daher nur diese hier beschrieben.

Die zu implementierende Benutzergruppe sind die User, viel mehr die Spieler. Grundsätzlich richtet sich NoRPG an Kinder, die keine Möglichkeit haben eine Schule zu besuchen. Jedoch werden keine Benutzergruppen für diese App ausgeschlossen. Egal ob jung oder alt, männlich oder weiblich, der Spieler sollte nur eine Neugier zum Lernen mitbringen.

Der Anwender benötigt Erfahrung mit der Verwendung eines Smartphones, insbesondere mit einem Android-System. Dazu zählt die Bedienung der Android-Oberfläche und die des Google Play Stores. Zudem sollten die User englische Texte lesen und verstehen können, da NoRPG zunächst nur in Englisch erscheinen wird.

### **3.2.4 Einschränkungen**

Da das SRS für die Kommunikation zwischen Entwickler und Stakeholder dient, wird zwischen Einschränkungen für Entwickler und für Spieler unterschieden.

Grundsätzlich müssen sich Entwickler an die vorgegebenen regulatorischen Richtlinien, wie beispielsweise an die Datenschutzerklärung von Google oder an das IT-Sicherheitsgesetz halten.

Da NoRPG sich an Kinder in bildungsfernen Ländern richtet, ist es besonders wichtig, dass die Texte in NoRPG einfach zu verstehen sind. Da das Spiel zunächst nur in Englisch erscheinen wird, dürfen die englischen Texte kein Fachjargon oder ähnliches beinhalten. Die App darf keine hohen Mindestanforderungen an Hardwareressourcen

haben, da der technische Standard in bildungsfernen Ländern geringer ist. Das bedeutet für die Entwickler das Spiel so gut wie möglich ressourcenschonend umzusetzen. Des Weiteren gilt es bei der Implementierung zu beachten, dass NoRPG soweit wie möglich ohne eine aktive Internetverbindung spielbar bleiben muss.

Allerdings gibt es auch Einschränkungen, welche für die Spieler gelten oder zumindest temporär. Wie schon öfter erwähnt wurde, wird das Spiel zunächst nur in Englisch erscheinen. Dementsprechend benötigt der Spieler Englischkenntnisse um die Texte im Spiel lesen und verstehen zu können. Für die Anmeldung, die Registrierung, das Herunterladen von Spielen, das Synchronisieren und installieren von Updates wird eine aktive Internetverbindung vorausgesetzt. Des Weiteren benötigt der Spieler ein Android Smartphone, welches die Mindestanforderungen von NoRPG erfüllt.

### **3.2.5 Annahmen und Abhängigkeiten**

Eine Annahme von NoRPG ist, dass es immer auf Smartphones, die genügend Leistung haben, verwendet wird. Wenn das Telefon nicht über genügend Hardwareressourcen für die Anwendung verfügt, kann es Szenarien geben, in denen die Anwendung nicht wie beabsichtigt oder überhaupt nicht funktioniert.

Eine weitere Annahme ist, dass das Smartphone und dessen Hardware sowie Software funktionieren. Das Smartphone muss sich mit dem Internet verbinden können, wenn der Benutzer sich anmelden möchte oder Lernspiele herunterladen will. Neben einer funktionierenden Internetverbindung sollten andere Hardwareelemente wie die Lautsprecher oder der Touchscreen funktionieren. Das Smartphone muss eine gültige Android Version mit einem Google Konto besitzen.

### **3.2.6 Aufteilung der Anforderungen**

In dem Fall, dass das Projekt verzögert wird, gibt es einige Anforderungen, die auf die nächste Version der Anwendung übertragen werden könnten.

## **3.3 Spezifische Anforderungen**

Das letzte Kapitel des SRS dient dazu alle Anforderungen an die Software detailliert zu beschreiben. Dies ermöglicht es Entwicklern ein System zu entwickeln, welches allen Anforderungen entspricht, und Testern, NoRPG ausreichend zu testen.

### **3.3.1 Externe Schnittstellen**

Dieser Abschnitt ist die detaillierte Beschreibung aller Ein- und Ausgänge von NoRPG. Diese Beschreibung ergänzt und vervollständigt die Schnittstellenbeschreibung von Kapitel 2.2.1.

#### **Systemschnittstellen**

NoRPG hat genau eine Schnittstelle mit einem anderen System und zwar mit Android. Android ist das Betriebssystem von Google für mobile Geräte, welches aktuell in der Version 7.1 Nougat zu erhalten ist. Viele Smartphone-Hersteller nutzen Android als Basis für ihr eigenes auf Android aufbauendes Betriebssystem.

Der Gültigkeitsbereich der Systemschnittstelle ist auf die App begrenzt und hat keine direkte Auswirkungen auf den Server.

Das Datenformat von Android ist das Android Application Package (APK) und wird für die Distribution und Installation von mobilen Apps auf Android Smartphones verwendet. Eine APK-Datei enthält den gesamten Programmcode, Ressourcen, Assets, Zertifikate und Metadaten. Verglichen kann das Datenformat von Google mit einem ZIP-Archiv<sup>20</sup>. Dieses Format muss NoRPG erfüllen, um unabhängig von den zu implementierenden Produktfunktionen auf einem Android Smartphone laufen zu können.

#### **Benutzerschnittstellen**

Die Benutzerschnittstellen bzw. User Interfaces (UI) sind der Punkt, an dem die Benutzer mit der Software interagieren. Zur Beschreibung der Benutzerschnittstellen werden logische Eigenschaften und Aspekte zur Optimierung formuliert. Für die Veranschaulichung werden Mockups verwendet. Diese stellen dar, wie die Oberfläche aussehen kann. Die am Projektende implementierte Benutzeroberfläche kann sich von den Mockups unterscheiden.

Wenn der Benutzer NoRPG das erste Mal startet oder nicht angemeldet ist, wird ihm die Login-Oberfläche (siehe Abbildung 2) präsentiert. Auf dieser Oberfläche hat der Benutzer die Möglichkeit sich mit seinem Benutzernamen und Passwort anzumelden oder sich, falls noch nicht geschehen, bei NoRPG zu registrieren. Das Smartphone muss Quer gehalten werden, da alle Elemente des Bildschirms für diese Ausrichtung

---

<sup>20</sup> vgl. Dan Morrill (Google) [11]

angeordnet sind. Diese Eigenschaft gilt ebenfalls für alle anderen Benutzerschnittstellen und wird nicht bei den Beschreibungen der anderen UIs zusätzlich erwähnt.

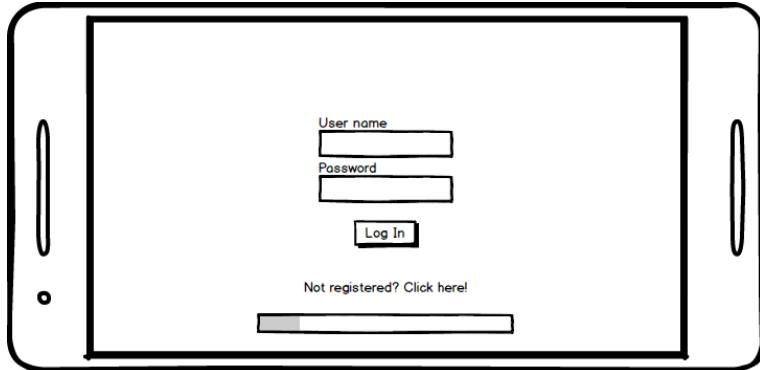


Abbildung 2: Mockup: Login-Screen

Die Hauptelemente des Login-Screens sind die Eingabefelder für Benutzername und Passwort, der Login- und Registrierungsbutton sowie ein Ladebalken, der den aktuellen Status von NoRPG zeigt. Wenn das Spiel aktualisiert wird kann hier der Status abgelesen werden. Zur Optimierung der Nutzung ist das Layout der Login-Oberfläche ein Border-Pane, in dem die Bestandteile in einer einzigen Spalte angeordnet sind. Fehler werden in einem kleinen Fenster dargestellt, wenn beispielsweise der Benutzer falsche Login-Daten eingibt oder die Internetverbindung während des Aktualisierungsprozesses abbricht.

Falls sich der Benutzer bei NoRPG registrieren möchte, hat er die Möglichkeit dies direkt in der App zu machen. Dazu klickt der Benutzer den Register-Button auf der Login-Oberfläche. Anschließend öffnet sich die Register-Oberfläche. Der vollständige Registrierungsprozess (siehe Abbildung 3) setzt sich aus zwei Schritten zusammen. Im ersten Schritt muss der Spieler das Registrierungsformular ausfüllen und bestätigen. Die Elemente des Formulars sind als Tabellen-Layout angeordnet, welches die Lesbarkeit verbessert. Felder, die nicht der erwarteten Eingabe entsprechen, werden als Falsch markiert und visuell hervorgehoben. Wenn das Anlegen des Accounts erfolgreich war, hat der Anwender die Möglichkeit im zweiten Schritt seinen persönlichen Charakter zu erstellen. Dafür bestimmt der Benutzer den Namen, das Geschlecht und Aussehen seines Charakters.

NoRPG startet erst, nachdem alles geladen wurde und der Benutzer angemeldet ist. Der Bildschirm von NoRPG besteht aus der Spielwelt (Grafik) und dem Head-Up Display (HUD). Das HUD ist eine Methode, mit der Informationen visuell als Teil der Benutzeroberfläche eines Spiels vermittelt werden. Während die Informationen, die auf dem HUD angezeigt werden, stark vom Spiel abhängen, gibt es viele Eigenschaften,

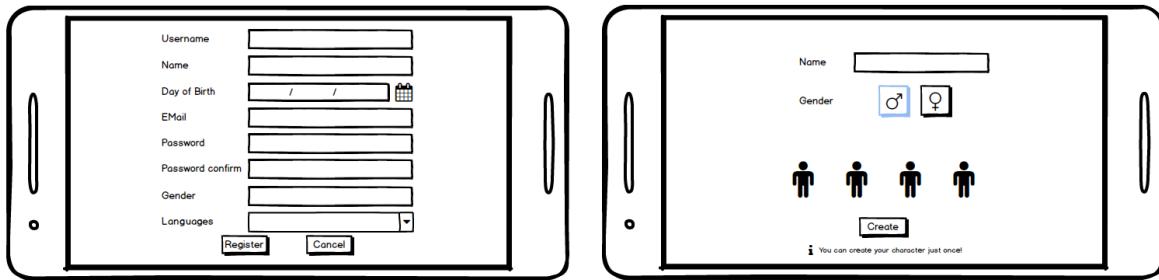


Abbildung 3: Mockups: Registrierungsformular und Charaktererstellung

die Spieler über viele Spiele erkennen. Die meisten von ihnen sind statisch auf dem Bildschirm, so dass sie während des Spiels sichtbar bleiben.



Abbildung 4: Mockup: Head-Up Display

Das Mockup in Abbildung 4 enthält alle direkt sichtbaren HUD Elemente, die während des Spieles aktiv sind. Diese Elemente sind an die Ecken des Bildschirms gebunden, so befinden sich beispielsweise die Pfeiltasten zur Bewegung des Charakters in der linken unteren Ecke des Bildschirms und die Buttons zur Interaktion mit dem Spiel in der unteren rechten Ecke.

Das Menü (siehe Abbildung 5), welches sich in der oberen linken Ecke befindet, kann geöffnet werden. Dadurch verändert sich das HUD von NoRPG und es erscheinen neue Elemente, die der Spieler sehen und benutzen kann. NoRPG ist derweil pausiert. Die anderen Elemente hingegen werden überdeckt oder reagieren nicht solange das Menü offen ist. Deshalb ergeben sich neue Optionen bzw. Möglichkeiten für den Spieler um mit NoRPG zu interagieren.

Es wird zwischen zwei Typen von Menü-Funktionen unterschieden. Eine Funktionen können direkt im Menü durchgeführt werden, wie beispielsweise das Speichern und Synchronisieren. Einige der Funktionen wiederum öffnen ein Fenster bzw. eine neue Ansicht, in dem die neuen Funktionalitäten zur Verfügung stehen. Die Fenster müs-

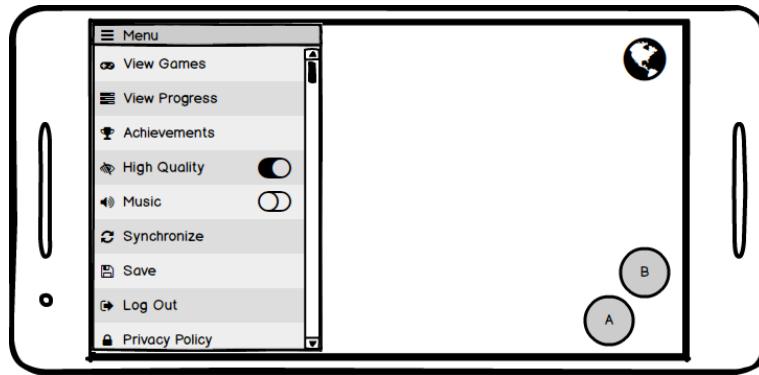


Abbildung 5: Mockup: Menü

sen erst geschlossen werden, um das Spiel fortsetzen zu können. Beispiele sind in der folgenden Abbildung 6 als Mockups zu betrachten.

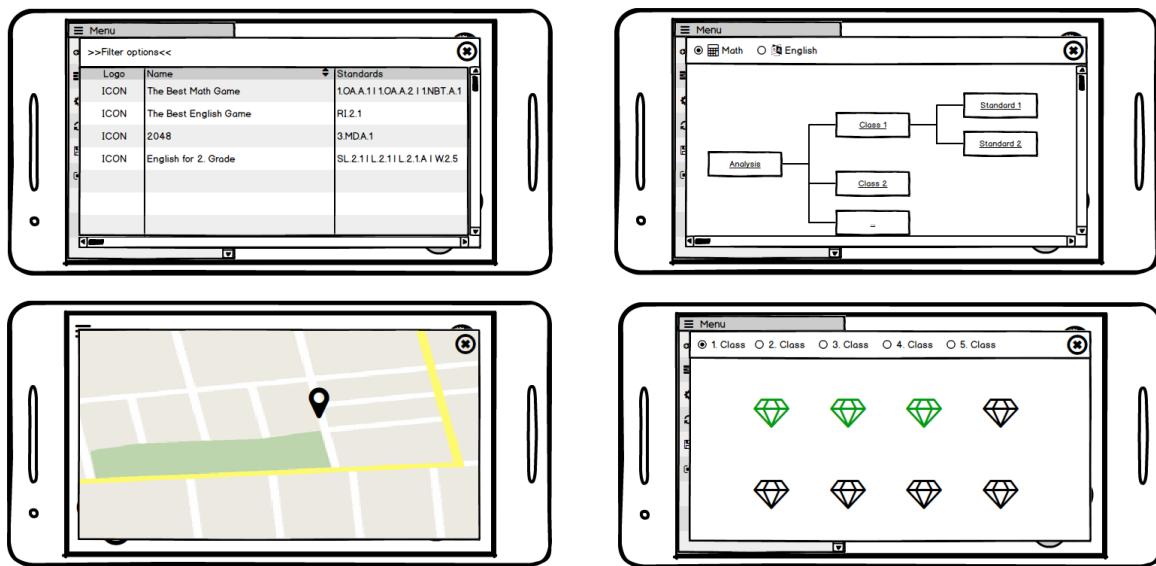


Abbildung 6: Mockups: Spielliste, Fortschrittsanzeige, Karte und Erfolgsübersicht

## Hardwareschnittstellen

Schnittstellen zwischen NoRPG und Hardwarekomponenten werden als Hardwareschnittstellen des Systems bezeichnet. Dieses Kapitel dient zur Spezifikation der logischen Eigenschaften dieser Schnittstellen.

Ein Smartphone besteht aus sehr vielen Hardwarekomponenten. Jede einzelne Komponente wird benötigt, damit das Smartphone mit seinem kompletten Funktionsumfang lauffähig ist. Jedoch spielen einige Hardwarekomponenten eine besondere Rolle für NoRPG. Neben unverzichtbaren Komponenten wie den Prozessor, Speicher oder

Akku zählen zu den Kernkomponenten der Touchscreen und der WLAN-Adapter. Der Touchscreen wird benötigt um die Eingaben des Spielers an das Spiel zu kommunizieren. Sei es die Steuerung des Charakters, das Ausfüllen des Registrierungsformulars oder die einfache Betätigung eines Buttons. Ohne den Touchscreen können keine Eingaben ohne zusätzliche Peripherie an das Spiel kommuniziert werden. Der WLAN-Adapter ist zuständig für die Verbindung mit dem Internet. Ohne eine Internetverbindung ist nicht einmal der Login funktionsfähig bzw. es wäre nicht ohne Umstände möglich NoRPG aus dem Google Play Store herunterzuladen.

Obwohl es sich bei dem Server um einen virtuellen bzw. simulierten Server handelt, weiß NoRPG nicht, dass keine physische Hardware direkt benutzt wird. Für die App scheint es, als ob sie mit einem physischen Server kommuniziert. Auch hier sind alle Komponenten des Server, auch wenn diese simuliert sind, Teil der Hardwareschnittstelle.

## **Softwareschnittstellen**

Softwareschnittstellen spezifizieren die Schnittstellen mit anderen benötigten Softwaresprodukten, welche die Nutzung derer Funktionalitäten ermöglicht.

Die vorinstallierte Software Google Play Store ist eine Plattform, die Musik, E-Books, Filme, Serien und insbesondere Apps anbietet. Der Google Play Store stellt eine Softwareschnittstelle zu NoRPG dar. NoRPG benötigt die Schnittstelle zur Spielplattform von Android, um die dort verfügbaren Lernspiel in NoRPG anzuzeigen bzw. als Download anzubieten.

Für die Verarbeitung der Toucheingaben gibt es ein C# Skript. Erst mit Hilfe dieser Software wird es möglich die Toucheingaben an das Spiel zu kommunizieren, damit die richtige Aktion in NoRPG ausgeführt wird.

Damit die lokalen Datenbanken der Clients und die Datenbank auf dem Server immer synchronisiert bleiben wird eine Datenbanksynchronisationssoftware benötigt, welcher die Synchronisation bei aktiver Internetverbindung übernimmt. Dabei handelt es sich um eine Softwareschnittstelle, da NoRPG die Funktionalität dieser Software verwendet.

Die letzte Softwareschnittstelle ist das Datenbank Management System (DBMS). Das DBMS ist die Verwaltungssoftware der Datenbank. Sie organisiert intern die strukturierte Speicherung der Daten und kontrolliert alle lesenden und schreibenden Zugriffe auf die Datenbank. Sie wird benötigt um den aktuellen Stand des Spiels zu speichern.

### 3.3.2 Funktionale Anforderungen

Use Cases dokumentieren Funktionalitäten eines Systems auf Basis von einfachen Modellen. In einem Use Case wird das nach außen sichtbare Verhalten eines Systems aus der Sicht des Nutzers beschrieben. Ein Nutzer kann hierbei eine Person, eine Rolle oder ein anderes System sein. Dieser Nutzer tritt als Akteur mit dem System in Interaktion, um ein bestimmtes Ziel zu erreichen.

Use Cases verwenden Activity Diagramme. Ein Activity Diagramm ist ein Verhaltensdiagramm der Unified Modeling Language (UML) und stellt die Vernetzung von elementaren Aktionen und deren Verbindungen mit Kontroll- und Datenflüssen grafisch dar.

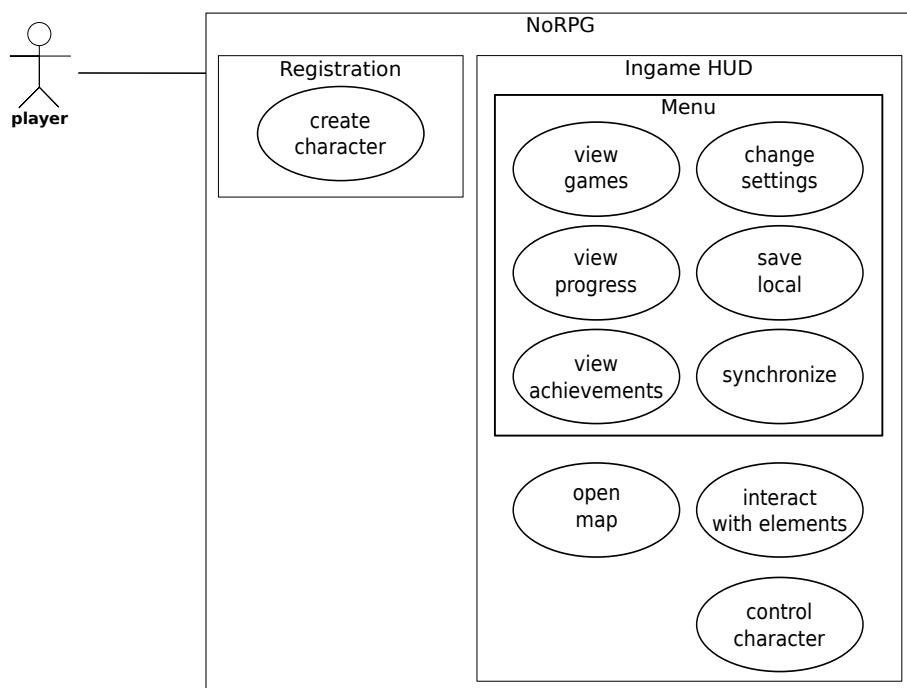


Abbildung 7: Overall Use Case Diagramm

Das abgebildete System in Abbildung 7 stellt die zu entwickelnde App für die User dar. Die App von NoRPG stellt die graphische Oberfläche und somit die beschriebenen Benutzerschnittstellen dar. Es sind nur die Funktionalitäten enthalten, die der Benutzer ausführen kann, also jene die über die Benutzerschnittstellen angesprochen werden können.

Use Cases wie Login oder Registrierung sind im Overall Use Case Diagramm nicht enthalten, da diese im Vergleich zu anderen Use Cases primitiv sind. Die abgebildeten Use Cases enthalten meistens mehrere Schritte bis die Aktion ausgeführt wird.

## Create character

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer seinen Charakter erstellen möchte. Dieser Use Case wird pro Account genau einmal im Registrierungsprozess ausgeführt.

Nach erfolgreicher Registrierung kann der Spieler seinen Charakter erstellen. Der User kann den Namen, das Geschlecht und das Aussehen des Charakters bestimmen. Ein möglicher Ablauf des Erstellungsprozesses kann aus Abbildung 8 entnommen werden.

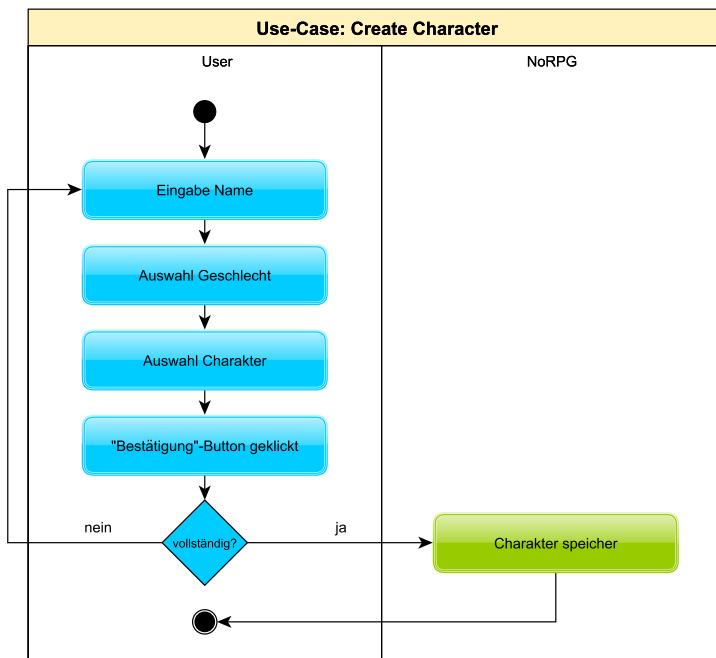


Abbildung 8: Activity Diagramm: Create Character

Der Ablauf des Prozesses kann allerdings variieren. Der Spieler kann beispielsweise erst das Geschlecht und das Aussehen bestimmen und anschließend seinem Charakter einen Namen geben. Die Variationen können allerdings nur bei den Charaktereigenschaften auftauchen.

Bevor jedoch dieser Use Case ausgeführt werden kann, muss der Spieler das Registrierungsformular vollständig ausfüllen und erfolgreich abschließen. Zudem darf der Account noch nicht existieren. Für den gesamten Registrierungsprozess ist eine aktive Internetverbindung notwendig, damit der neu angelegte Account mit den Spielereigenschaften direkt mit dem Server synchronisiert werden kann.

Nach der erfolgreichen Erstellung des Charakters, wird dieser in die Datenbank gespeichert und der User kann sich nun anmelden und das Spiel starten. Sollte die Erstel-

lung allerdings fehlschlagen oder abgebrochen, wird der User wieder auf den Hauptbildschirm von NoRPG weitergeleitet.

## Open map

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer die Karte der Spielwelt öffnet. Die Karte dient zur Orientierung und kennzeichnet den aktuellen Aufenthalt des Spielers.

Der Anwender kann die Karte der aktuellen Spielwelt durch einen Klick auf die Mini-Map öffnen. Das Activity Diagramm in Abbildung 9) zeigt den vollständigen Ablauf.

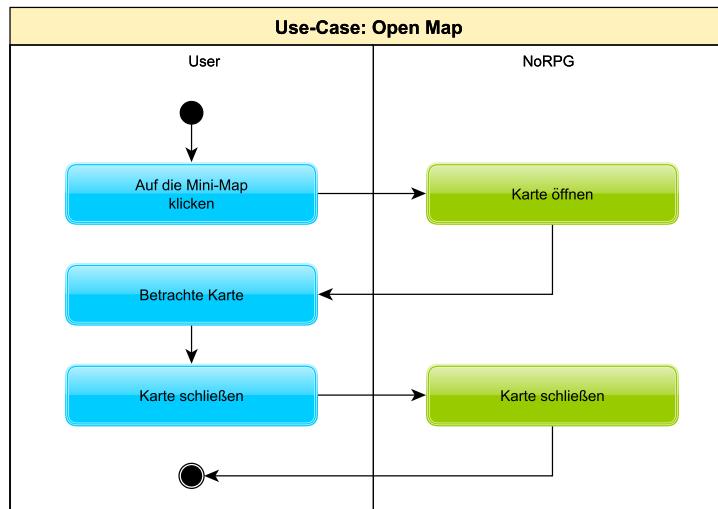


Abbildung 9: Activity Diagramm: Open Map

Die Vorbedingungen für diesen Use Case sind, dass der Spieler sich im Spiel befindet, das Menü geschlossen ist und der Spieler sich in keiner aktiven Non-Player Character (NPC) Interaktion befindet.

In einem Fehlerfall sollte sich die Karte schließen und der Benutzer wieder zum Spiel weitergeleitet werden.

## View games

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer die Liste der freigeschalteten und spielbaren Spiele öffnen will. Gefundene Spiele in NoRPG werden in diese Liste aufgenommen, so dass der Spieler an einem zentralen Ort alle spielbaren

Lernspiele anschauen kann. Die Liste enthält den Namen des Spiels, einen Download-Link sowie die Zuordnung zum entsprechenden Standard. Dadurch ist es möglich auch ohne eine aktive Internetverbindung in NoRPG voranschreiten zu können.

Das Activity Diagramm in Abbildung 10 hat im Vergleich zu den bisherigen betrachteten Activity Diagrammen eine Besonderheit. Die Aktivität "Google Play Store öffnen" findet außerhalb von NoRPG statt. Nach Beendigung dieses Schrittes wird der Spieler wieder zurück ins Spiel weitergeleitet. Allerdings wenn der Prozess von NoRPG währenddessen abgebrochen wird, startet das Spiel am letzten Speicherpunkt. Der Prozess beginnt, indem der Spieler die Funktion im Menü aufruft.

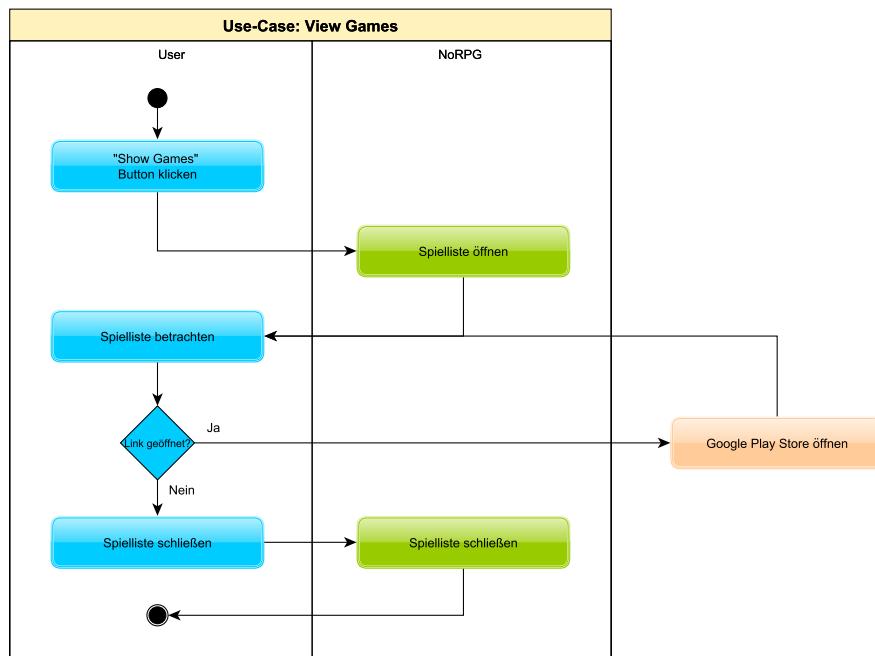


Abbildung 10: Activity Diagramm: Show Games

Der Benutzer muss sich im Spiel befinden und darf in keiner aktiven NPC Interaktion sein. Zudem muss das Menü vorher offen sein, bevor dieser Use Case ausgeführt werden kann. Auch hier gilt im Fehlerfall, dass der Spieler wieder zurück ins Spiel kommt.

### **View progress**

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer seinen Fortschritt in NoRPG betrachten möchte. Dafür wird dem Spieler pro Schulfach jeweils ein Baumdiagramm präsentiert, der die Standards und deren Reihenfolge beinhaltet. Durch eine Kennzeichnung kann der Spieler sehen welche Standards abgeschlossen sind und fehlen.

Das Activity Diagramm in Abbildung 11 ähnelt dem vom Use Case "View Games". Die einzige Ausnahme dabei ist, dass dieses Diagramm keine Aktion hat, die außerhalb von NoRPG stattfindet. Die Fortschrittsanzeige wird im Menü geöffnet. Nachdem die Anzeige offen ist kann der Spieler zwischen den Standards wechseln.

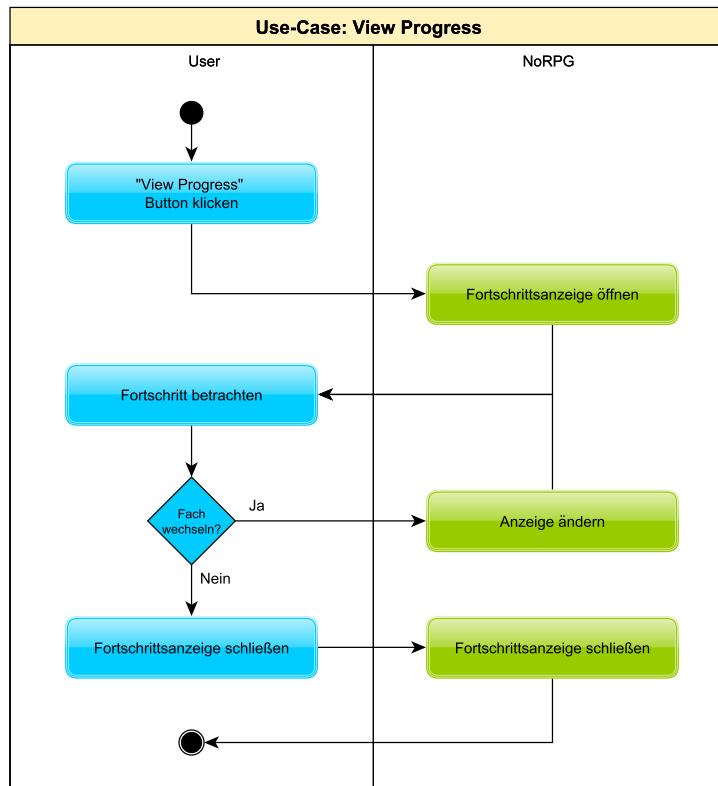


Abbildung 11: Activity Diagramm: View Progress

Dieser Use Case hat die gleichen Vorbedingungen wie der vorherige Use Case. Der Spieler muss sich im Spiel befinden, darf in keiner aktiven NPC Interaktion sein und das Menü ist vorher offen.

## View achievements

Dieser Use Case beschreibt den Anwendungsfalls, dass der Spieler seine Sammelgegenstände betrachten will. Neben den Standards kann der Anwender im Spiel Gegenstände finden, die ihn in der in Kapitel 2 beschriebenen Story weiterbringen. Diese Collectables sind auf den einzelnen Spielwelten verteilt und können dann, nachdem diese gefunden sind in der Ansicht zusammen mit den anderen betrachtet werden.

Im Prinzip ähnelt dieses Activity Diagramm dem vom Use Case "View progress". Der Spieler will seinen Fortschritt betrachten. Dieses mal allerdings seinen Fortschritt in

NoRPG. Hier kann der Spieler zwischen den Welten wechseln anstatt zwischen den Fächern und es wird eine andere Darstellungsart verwendet.

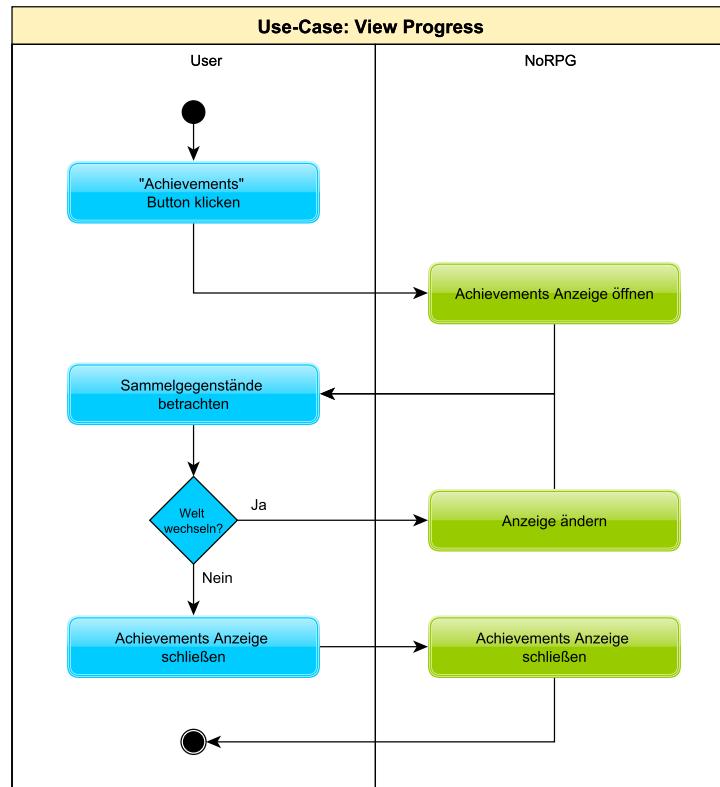


Abbildung 12: Activity Diagramm: Achievements

Dieser Use Case hat die gleichen Vorbedingungen wie der vorherige.

### Change settings

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer Einstellungen ändern möchte. In dem Rahmen dieser Studienarbeit wird es zwei Einstellungsmöglichkeiten geben. Der Spieler kann die Qualität des Spiels verändern. Ist die Option "High Quality" an, wird alles in höchster Qualität wiedergegeben. Ist jedoch diese Option aus, werden Hintergrundanimationen, die ressourcenintensiv sind, nicht wiedergegeben. Diese Option ermöglicht es Smartphones mit schlechterer Hardwareausführung das Spiel ohne Probleme spielen zu können, bietet jedoch für neuere Smartphones die Möglichkeit NoRPG in vollen zu genießen.

Die zweite Einstellungsmöglichkeit ist, dass der Spieler die Audioausgaben des Spieles verändern kann. Zu einem Spielerlebnis gehört die Musik und Soundeffekte. Diese kann der Spieler je nach Wunsch ein- oder ausschalten.

Die Vorbedingung zur Änderung der Spieleinstellungen sind, dass der Anwender sich im Spiel und sich in keiner aktiven NPC Interaktion befindet. Bei erfolgreicher Änderung sollten diese Einstellungen für diesen Account gespeichert werden, damit diese bei jedem Login, auch auf verschiedene Geräte, übernommen werden. Im Fehlerfall werden die Default-Werte der Einstellungen angewendet.

### **Save local**

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer seinen aktuellen Fortschritt manuell speichern möchte. NoRPG speichert bei bestimmten Ereignissen automatisch. Beispielsweise wenn der Spieler in eine andere Welt reist oder eine Truhe findet. Allerdings kann der Spieler seinen aktuellen Stand speichern. Es werden neben Fortschritt, Erfolge oder Charaktereigenschaften noch die Spieleinstellungen und sogar der aktuelle Standort des Spielers gespeichert. Der Spieler kann diese Funktion direkt im Menü durchführen.

Wenn das Speichern erfolgreich war, wird der Spieler durch eine kurze Nachricht informiert. Wenn keine Information erscheint kann davon ausgegangen werden, dass der Vorgang fehlgeschlagen ist. Im Fehlerfall muss die Aktion erneut durchgeführt werden.

### **Synchronize**

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer seinen lokalen Speicherstand mit dem Server manuell synchronisieren möchte. NoRPG synchronisiert automatisch, wenn eine aktive Internetverbindung vorhanden ist beim Einloggen und vor dem Ausloggen. Die Synchronisation wird nicht so oft wie das Speichern ausgeführt, da der Vorgang länger dauern kann. Ebenfalls wird die Synchronisation nicht so häufig benötigt, da der Server als eine Art Back-Up Medium fungiert. Also für den Fall, dass der Spieler sich mit einem anderen Gerät anmelden möchte oder der lokale Spielstand nicht mehr vorhanden ist. Der Spieler kann diese Funktion direkt im Menü durchführen.

Wenn das Synchronisieren erfolgreich war, wird der Spieler durch eine kurze Nachricht informiert. Wenn keine Information erscheint kann davon ausgegangen werden, dass der Vorgang fehlgeschlagen ist. Eine häufige Ursache für den Fehlschlag kann die fehlende Internetverbindung sein.

## Control character

Dieser Use Case beschreibt den Anwendungsfall, dass der Spieler seinen Charakter in NoRPG durch die Spielwelt kontrollieren möchte. Dafür verwendet der Spieler das Steuerkreuz, welches sich in der linken unteren Ecke des HUD befindet (vgl. Abbildung 4).

Das Activity Diagramm dieses Use Cases in Abbildung 13 ist im Vergleich zu den bisherigen Diagrammen besonders, denn es existiert keinen Endpunkt. Dieser ist zugleich der Startpunkt des Diagramms.

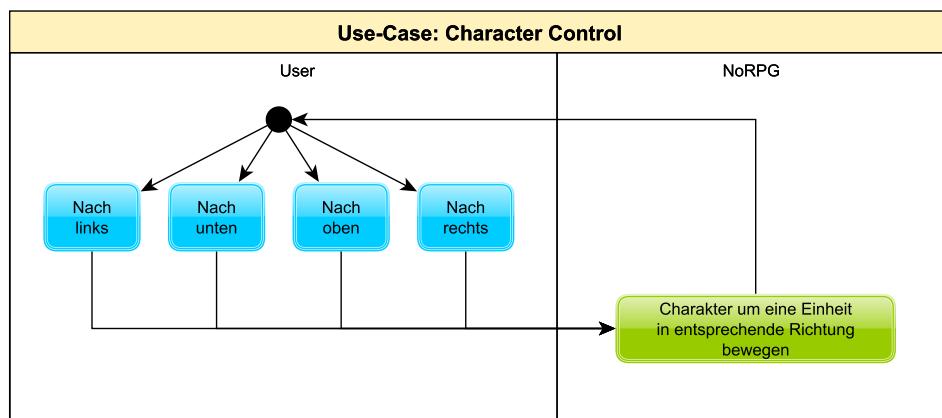


Abbildung 13: Activity Diagramm: Character Control

Wenn der Spieler das Steuerkreuz nach oben bewegt, wird der Charakter des Spielers um eine Einheit nach vorne bewegt. Nach dem Abschluss dieser Aktion kann der Spieler den Charakter direkt weiter bewegen. Wenn der Touchscreen in eine Richtung gehalten wird, dann wird dieser Ablauf ganze Zeit durchgeführt, bis die Touchscreen losgelassen wird.

Die Vorbedingung für diesen Use Case ist, dass der Spieler sich im Spiel befindet und das Menü geschlossen ist.

## Interact with elements

Dieser Use Case beschreibt den Anwendungsfall, dass der Benutzer mit einem Element im Spiel interagieren möchte. Dafür hat der Spieler zwei Tasten auf dem HUD (vgl. Abbildung 4), die dem Spieler die Möglichkeit gibt die Interaktion zu starten oder zu beenden.

In NoRPG wird es verschiedene Elemente geben, mit denen der Spieler interagieren kann.

- **NPCs:** Der Spieler kann sich mit unterschiedlichsten NPCs unterhalten, die überall in allen Spielen vorkommen. Die NPCs helfen dem Spieler, indem diese Tipps und Hinweise in den Unterhaltungen nennen.
- **Truhen:** Der Spieler kann mit Truhen interagieren. Die Truhen sind in den einzelnen Welten verteilt und enthalten die Sammelgegenstände. Durch die Interaktion öffnet sich die Truhe und der Sammelgegenstand erscheint. Nachdem die Truhe geöffnet wurde bleibt die Truhe offen und es kann nicht weiter mit ihr interagiert werden. Es handelt sich um eine einmalige Aktion.
- **Spielehändler:** Spielehändler tauchen überall auf. Diese bieten die verschiedenen Lernspiele für Standards an. Die Spielehändler können menschliche Händler oder andere Formen annehmen um die verschiedene Welten zu repräsentieren. Durch die Interaktion startet der Spieler eine Unterhaltung in dem der Spieler Informationen über den Standard erhält und das Spiel freischaltet.
- Die Interaktion mit der Umwelt (Bäume, Tiere, Gebäude) starten zunächst keine Interaktion.

Eine Interaktion kann nur dem Ä"Button gestartet werden. Wenn ein Element zur Interaktion vorhanden ist, wird erst die Interaktion gestartet. Anschließend kann der Spieler mit dem Äöder "B"Button reagieren. Der "B"Button ist zum Abbrechen der Interaktion und beendet diesen Prozess bzw. überspringt ihn. Zum Beispiel in der Interaktion mit dem Händler sorgt der "B"Button dafür, dass die Unterhaltung übersprungen wird und direkt die Spiele angezeigt werden. Mit dem Ä"Button wird die Interaktion fortgeführt bis keine Aktion mehr vorhanden ist, also in unserem Händlerbeispiel keine Unterhaltung mehr gibt. Der Ablauf ist in Abbildung 14 visualisiert dargestellt.

Die Vorbedingungen für diesen Use Case sind, dass der Spieler sich im Spiel befindet und das Menü sowie die Karte geschlossen sind. Je nach Element wird ein anderes Ergebnis erwartet. Beispielsweise wird bei einer Interaktion mit der Truhe das Ergebnis erwartet, dass die Truhe offen bleibt, der Sammelgegenstand in die Liste übernommen wurde und der Spielstand automatisch lokal gespeichert wird.



Abbildung 14: Activity Diagramm: Game Interaction

### 3.3.3 Performanz Anforderungen

Dieser Abschnitt des SRS legt sowohl die statischen als auch die dynamischen numerischen Anforderungen an die Software.

Ein sehr wichtige Anforderung betrifft die Antwortzeit des Systems. Diese Anforderung ist besonders wichtig, da der Nutzer die direkten Auswirkungen in der App mitbekommt. Eine Antwortzeit von 0,1 Sekunden gibt dem Benutzer das Gefühl, dass das System sofort reagiert. Bei einer Sekunden denkt der Benutzer immer noch, dass die Aktion ununterbrochen durchgeführt wird, jedoch bemerkt der Benutzer ein Verzögern. Im Worst Case liegt die Grenze für die Aufmerksamkeit des Benutzers bei zehn Sekunden<sup>21</sup>.

Für die Performanz Anforderungen wird zwischen der App und dem Server unterschieden. 95% der Transaktionen in der App sollten in weniger als einer Sekunde verarbeitet werden. Aufwändiger Prozesse wie die Synchronisation oder das wechseln der Spielwelten dürfen minimal länger brauchen. Für den Server gilt, dass 80% der Transaktionen in weniger als einer Sekunde verarbeitet werden. Es sind nur 80% notwendig, da der Server nur für die Synchronisation und Speicherung verwendet wird.

<sup>21</sup> vgl. Andrew Lee [8]

Der prozentuale Anteil der Synchronisation auf dem Server ist wesentlich größer als die in der App.

Der Workload ist die Anzahl an Verarbeitungen, die das System in einer gegeben Zeit durchführen kann. Die App selbst hat keine hohe Arbeitsbelastung, da diese nur einen Benutzer gleichzeitig zulässt und von der Hardware des Spielers abhängt. Vielmehr gilt es zu betrachten, mit wie viel Belastung der Server umgehen kann. Der Anzahl an Spielern ist mehr oder weniger keine Grenze gesetzt, da für jeden einzelnen Spieler zunächst eine Zeile in der Datenbank hinterlegt wird. Die Anzahl an gleichzeitigen Benutzern, die mit dem Server kommunizieren wollen, ist allerdings begrenzt. Da der Server auch seine Hardwarebegrenzungen hat kann es nur eine bestimmte Anzahl an gleichen Synchronisationsanfragen entgegennehmen. Wenn diese Anzahl übersteigt wird, arbeitet der Server nach dem First In First Out (FIFO) Prinzip. Die Clients, die als erste eine Synchronisationsanfrage senden, werden auch als erstes bearbeitet. Als eine Art der Erweiterung spezifiziert die Lastskalierbarkeit die Menge an Daten, die innerhalb bestimmter Zeitperioden sowohl bei normaler als auch bei maximaler Belastung verarbeitet werden sollen.

### 3.3.4 Datenbank Anforderungen

Die beiden eingesetzten Datenbanken unterscheiden sich von ihren Anforderungen.

Bei der Datenbank in der App handelt es sich um eine embedded, zu deutsch eingebettete, Datenbank. Sie wird zusammen mit der App ausgeliefert. Die Datenbank enthält zunächst die Spieldaten. Das ist eine Liste der Standards mit den dazugehörigen Spielen. Neben diesen Daten werden lokal noch die Daten des Spielers gespeichert. Die Spielerdaten beinhalten gespeicherte Einstellungen, die Charaktereigenschaften und den Fortschritt. Dadurch wird das Spielen ohne eine aktive Internetverbindung ermöglicht.

Die embedded Datenbank wird oft verändert, da bei jedem Fortschritt automatisch gespeichert wird. Zudem werden die Daten der Datenbank beim Einloggen und vor dem Ausloggen immer mit dem Server synchronisiert. Die Daten werden beim Ausloggen gelöscht, damit sich verschiedene Spieler auf einem Smartphone anmelden können und die Datensicherheit gewährleistet wird.

Die Integrität der Datenbank muss immer gewährleistet werden, damit die Benutzer den eigenen Fortschritt nicht fälschen können. Dies kann mit Prüfsummen und Versierung gewährleistet werden.

Die Datenbank auf dem Server enthält alle Spielerdaten und Spieldaten. Bei den Spiel-daten handelt es sich um die gleichen Daten, die auf den lokalen Datenbanken ge-speichert sind. Allerdings sind auf dem Server alle Spielerdaten gespeichert, damit die Benutzer von unterschiedlichen Geräten spielen können.

Diese Datenbank wird nicht oft verändert bzw. aktualisiert, da die Datenbank nur bei der Synchronisation verändert wird oder wenn neue Standards oder Spiele eingefügt werden. Das hat den Vorteil das nicht sehr viele Anfragen an den Server gestellt wer-den, wodurch der Workload gering gehalten wird. Zugriffen kann der Spieler nur über die App. Administratoren haben die Möglichkeit den Server über eine Virtual Machine zu erreichen und Änderungen durchzuführen.

Auch hier muss die Integrität der Datenbank immer gewährleistet werden.

### **3.3.5 Entwurfsbeschränkungen**

Bei der Entwicklung der App gilt es zu beachten, dass die Hardware-Anforderungen nicht zu hoch sind. Als Maßstab für die Hardware wird das Smartphone Samsung Galaxy S4 genommen. Grund dafür ist das Preis-Leistungsverhältnis des S4 und die Tatsache, dass Samsung eine sehr verbreitete Marke ist.

Das Samsung Galaxy S4 kostet auf Amazon (Stand 27.01.2017) ungefähr 225 Euro. Das S4 hat vorinstalliert Android 4.2 Jelly Bean und die Samsung Touchwiz 4.0 Oberfläche. Ein Update auf Android 5.0 Lollipop ist allerdings möglich. Das S4 hat 2 GByte Arbeitsspeicher und 16 GByte internen Speicher, welcher durch eine Speicherkarte erwei-tert werden kann. Bei dem Prozessor handelt es sich um einen Qualcomm Snapdragon 600 mit vier Kernen, einer Taktfrequenz von 1,9 GHz und einer 32-bit Architektur<sup>22</sup>. Für eine Internetverbindung bietet Samsungs Galaxy S4 eine drahtlose Schnittstelle, die nach IEEE 802.11a/ac/b/g/n funktioniert.

Der Speicherbedarf darf nicht größer als 200MB sein, geplant sind allerdings 150MB. Beim RAM dürfen nicht mehr als 1GB sein, geplant sind allerdings 500MB.

### **3.3.6 Benutzerfreundlichkeit**

Das Ziel der Benutzerfreundlichkeit ist eine hohe Ergonomie. Die Software-Ergonomie bezeichnet die Anpassung an die kognitiven und physischen Fähigkeiten bzw. Eigen-schaften des Benutzers, also seine Möglichkeiten zur Verarbeitung von komplexen In-

---

<sup>22</sup> vgl. Google [23]

formationen aber auch die Anpassung softwaregesteuerten Merkmale der Darstellung, wie Farben und Schriftgröße.

Damit die Benutzeroberfläche freundlich für den Benutzer ist, muss sie in das Profil des Benutzers passen. Da NoRPG sich grundsätzlich an Kinder richtet muss die Bedienung und Gestaltung der App kindgerecht sein. Damit eine App als kindgerecht bezeichnet werden kann muss es nach Wendy B. von Intel vier Prinzipien erfüllen<sup>23</sup>.

1. Freiheit: Die Fähigkeit, sich in der App innerhalb einer kontrollierten Umgebung frei bewegen zu können. Dieses Prinzip verfolgen auch Rollenspiele. Durch die offene Spielwelt kann der Spieler selbst entscheiden wohin er als nächstes hin möchte. Allerdings gilt es bei Kindern diese Umgebung zu kontrollieren, indem bestimmte Bereiche festgelegt werden müssen.
2. Komfort: Die App sollte stimulieren, jedoch darf es nicht übertrieben werden. Dies wird als Balanceakt bezeichnet. Abwechslungsreiche Stimulationen sind definitiv notwendig, beispielsweise durch die Animationen, Farben oder Musik. Dadurch wird die Wahrnehmung und Aufmerksamkeit erhöht. Allerdings ist die Linie zwischen Stimulation und Lärm bzw. zu viel Stimulation sehr dünn und kann leicht überschritten werden.
3. Vertrauen: Kinder, genau wie Erwachsene, müssen sich kompetent fühlen und wollen ihren Aktionen vertrauen.
4. Kontrolle: Kinder wollen das Gefühl, dass sie etwas vollbringen wenn sie in der App interagieren. Es werden Ziele gesetzt und Entscheidungen getroffen.

Für die kindgerechte Gestaltung gibt es allerdings noch weitere Kriterien. Es dürfen keine In-App-Käufe angeboten, keine Werbung platziert oder zu Social Media, wie beispielsweise Facebook oder Twitter, oder ähnlichen Seiten verlinkt werden. Diese Gegenstände haben in einem kindgerechten Spiel nichts zu tun. Für die Installation gilt es zu beachten, das so wenig Berechtigungen wie notwendig verlangt werden<sup>24</sup>.

Es gilt, desto einfacher die App gestaltet ist, desto mehr Kinder verstehen diese. Ein Beispiel dafür ist die Charaktersteuerung. Moderne Spiele haben keine sichtbaren Steuerkreuz, der Spieler muss nur eine Wischbewegung in die Richtung machen, in der er sich bewegen möchte. In NoRPG allerdings wird ein sichtbares Steuerkreuz gewählt, wie dies schon von vielen Geräten wie vom klassischen Gameboy oder von Playstation verwendet wird. Navigation soll schnell erkennbar und nachvollziehbar sein.

---

<sup>23</sup> vgl. Wendy B. [3]

<sup>24</sup> vgl. klick-tipps.net [7]

Schließlich gilt es die erwachsenen Nutzer nicht zu vergessen! Gemeint sind Eltern, Erziehungsberechtigte, Lehrer und jeder, der mit der App interagieren kann. Erwachsene spielen genauso eine wichtige Rolle. Egal ob diese die Kinder unterstützen, überwachen oder selbst spielen. Es ist wichtig, dass sich Eltern an die Entwickler melden können, wenn sie etwas nicht kindgerecht halten oder wenn es Probleme gibt<sup>25</sup>.

### 3.3.7 Zuverlässigkeit

Alle implementierten Produktfunktionen sollten zur Auslieferung korrekt und zuverlässig funktionieren. Dazu zählt auch, dass die Funktionen in vertretbaren Zeiten terminieren. Beispielsweise sollte die Anmeldung funktionieren, wenn der Spieler registriert ist und die richtigen Benutzerdaten eingegeben hat, oder die Benutzereingaben für die Charaktersteuerung sollen korrekt interpretiert werden.

Eine besondere Wichtigkeit hat die Implementierung der in Kapitel 2 beschriebenen CCSSs. Diese sind wichtig für die Reihenfolge der spielbaren Lernspiele, damit ein Spieler mit einem Skilllevel der ersten Klasse in Geometrie keine Lernspiele für die fünfte Klasse angezeigt kriegt. Erst dadurch wird gewährleistet und kann sichergestellt werden, dass der Spieler die Lerninhalte korrekt vermittelt kriegt.

### 3.3.8 Verfügbarkeit

Da bei jeder App eine lokale Datenbank vorhanden ist, muss der Server nicht die ganze Zeit verfügbar sein. Das gilt jedoch nur für die Benutzer, die NoRPG schon heruntergeladen und sich angemeldet haben. Wenn diese Bedingungen erfüllt sind, wird der ganze Fortschritt lokal gespeichert und kann mit dem Server manuell oder automatisch synchronisiert werden. Für den Fall, dass der Benutzer sich von einem anderen Gerät anmelden möchte, muss der Server verfügbar sein.

Daher werden nur die Verfügbarkeit des Logins bzw. Registrierung und der Synchronisation bewertet. Der Login muss eine Verfügbarkeit von 99% haben, wohingegen bei der Synchronisation eine Verfügbarkeit von 90% ausreicht.

---

<sup>25</sup> vgl. Becky White [22]

### **3.3.9 Sicherheit**

Bei Sicherheit wird zwischen zwei unterschiedlichen Typen unterschieden: Security und Safety. Security ist der Schutz vor absichtlichen Bedrohungen, wenn ein Angreifer absichtlich das Systems angreift. Im Gegensatz dazu ist Safety der Schutz vor unbeabsichtigten Bedrohungen, wenn der Benutzer durch Zufall die Sicherheitsmechanismen umgeht indem er etwas nicht beabsichtigtes ausführt.

Die Kommunikation mit dem Server und mit der lokalen eingebetteten Datenbank müssen verschlüsselt werden, damit die Credentials bei den Anmeldung oder bei der Registrierung nicht mitgelesen werden können. Des Weiteren müssen die Daten auf der lokalen Datenbank validiert werden, bevor der Server synchronisiert wird, denn es wird unter anderem auch der Spielfortschritt der Benutzer synchronisiert. Die Veränderung des Spielfortschritts wird als Schummeln bzw. Cheaten behandelt.

Die Anmeldung bzw. Registrierung ist notwendig, um NoRPG spielen zu können. Deswegen müssen die Accounts der Benutzer verschlüsselt gespeichert werden und die Passwörter dürfen bei der Anmeldung nur mittels One-Way-Functions verglichen werden. Nicht registrierte bzw. unautorisierte Benutzer erlangen keinen Zugriff auf das System.

Die gespeicherte Daten dürfen an andere Tools nur anonymisiert weitergegeben werden, für beispielsweise Analysezwecke. Diese Kommunikation mit anderen System oder Applikationen darf nur verschlüsselt geschehen.

### **3.3.10 Wartbarkeit**

Der Code von NoRPG sollte so geschrieben werden, dass der Code die Umsetzung neuer Funktionen begünstigt. Deshalb sollte die Komplexität des Quellcodes so gering wie möglich gehalten werden, indem entsprechende Methoden wie das Model-View-Controller (Model-View-Controller (MVC)) Pattern umgesetzt werden. Des Weiteren sollte das System von NoRPG Schnittstellen jeglicher Art anbieten, um das System durch weitere Komponenten wie ein Web-Tool für Administratoren zu erweitern.

Neben der Erweiterbarkeit sollte NoRPG für Fehlerfälle eine Testumgebung anbieten, um das Testen der Anwendung auf unterschiedliche Funktionen zu ermöglichen und gegebenenfalls Fehler wiederholen und simulieren zu können.

### **3.3.11 Portabilität**

Die App NoRPG ist zunächst nur für Android geplant. Andere Betriebssysteme, wie Windows Phone von Microsoft oder iOS von Apple, sind vorerst nicht vorgesehen.

Bei der vorhanden Breite an Varianten von Android-Smartphones ist sehr wichtig, dass die Portabilität innerhalb von Android Smartphones gewährleistet wird. Neben bekannten Smartphoneherstellern wie Samsung, LG oder HTC gibt es zahlreiche weitere Hersteller die auf das Android Betriebssystem setzen. Jeder Hersteller hat dabei eine große Palette an Smartphone-Modellen, wie bei Samsung die Samsung Galaxy S-Reihe, welches aktuell in der siebten Generation erhältlich ist<sup>26</sup>, oder die Samsung Galaxy Note-Reihe. Die App NoRPG muss auf allen Android-Smartphones funktionieren, solange diese die Mindestanforderungen an Software und Hardware erfüllen. Dabei muss sich die App beispielsweise an die Auflösung des Smartphones anpassen.

Die Portabilität beschreibt nicht nur die technische Sicht sondern auch in welchen Ländern und in welchen Sprachen NoRPG verfügbar sein wird. Der Release findet in allen Ländern statt, in denen der Google Play Store verfügbar ist. Zunächst wird NoRPG nur in Englisch verfügbar sein.

---

<sup>26</sup> vgl. Philippe Fischer, Michael Huch [13]

# 4 Technische Grundlagen

Im folgenden Abschnitt werden die technischen Grundlagen behandelt. Dabei wird mit den Entwicklungsumgebungen begonnen.

## 4.1 Unity

Bei Unity handelt es sich um eine Entwicklungs- und Laufzeitumgebung, mit deren Hilfe graphisch aufwändige Projekte umgesetzt werden können. Dazu gehören unter anderem Videospiele, aber auch Lernprogramme oder Apps. Dazu können in Unity 3D-Projekte, aber auch 2D beziehungsweise 2,5D Projekte umgesetzt werden. Bei einer 2,5D Anwendung handelt es sich um eine Mischung von 2D und 3D Elementen. Dank Unity sind diese Projekte nach der Entwicklung plattformübergreifend einsetzbar.<sup>27</sup> Die Entwicklungsumgebung teilt sich dabei in verschiedene Bereiche auf.

### 4.1.1 Komponenten der Entwicklungsumgebung

Das „Scene“ Fenster ist in den Standardeinstellungen in der Mitte des Bildschirms zu sehen(Siehe Abbildung 15)<sup>28</sup>. Hier wird immer die aktuelle Szene dargestellt. Darüber hinaus kann hier mit Objekten der Szene interagiert werden, um diese zu verändern, zum Beispiel an eine andere Stelle platzieren oder zu skalieren.

Wird ein Objekt in diesem Fenster ausgewählt, befinden sich im Bereich „Inspector“ zusätzliche Einstellmöglichkeiten<sup>29</sup>. Diese variieren je nach gewähltem Objekt. Auch hier können die Position oder Skalierung eines Objektes verändert werden, allerdings werden hier auch darüber hinausgehende visuellen sowie die physischen Eigenschaften der Objekte verändert.

---

<sup>27</sup> vgl. Unity3D [14] (2017)

<sup>28</sup> vgl. Unity3D [21] (2017)

<sup>29</sup> vgl. Unity3D [19] (2017)



Abbildung 15: Darstellung von Tablemappings

Objekte können des weiteren im „Hierarchy“ Fenster<sup>30</sup> ausgewählt werden. In diesem Fenster werden alle Objekte der aktuellen Szene aufgelistet. Dabei wird zwischen unterschiedlichen Ebenen unterschieden, so dass genau gesehen werden kann welche Objekte zusammengehören.

Um alle Dateien zu sehen, die zu einem Projekt gehören, gibt es das „Project“ Fenster<sup>31</sup>. Die Dateien werden dabei nach der vorliegenden Ordnerstruktur angezeigt. In diesem Fenster können Ordner, sowie andere Dateien erstellt werden.

Ein weiteres wichtiges Fenster ist das „Game“ Fenster<sup>32</sup>. Hier kann das fertige Projekt angesehen und getestet werden. Dazu gibt es oben in der Mitte der Benutzeroberfläche, ein Start, Pause und Vorlauf Button. Mit Hilfe derer kann das Programm, bevor es auf der Zielplattform abgespielt wird, in der Entwicklungsumgebung gerendert werden. Der Code wird von Unity Just In-Time (Just-In-Time (JIT)) kompiliert, und anschließend auf Mono oder dem Microsoft .NET Framework ausgeführt. Der Code steht in sogenannten Skripten, die in C#, UnitySkript (ähnlich JavaScript) oder Boo geschrieben sind.

Wenn während der Laufzeit oder im Vorfeld beim Kompilieren ein Fehler auftritt, wird dieser im „Console“ Fenster ausgegeben. Darüber hinaus werden hier Meldungen angezeigt, die explizit in den Skripten programmiert wurden. Damit es zu keinen Fehlern

<sup>30</sup> vgl. Unity3D [18] (2017)

<sup>31</sup> vgl. Unity3D [20] (2017)

<sup>32</sup> vgl. Unity3D [17] (2017)

kommt, gibt es in Unity Tests, die eine Szene auf Korrektheit prüft. Die sogenannten Integrationstest simulieren eine Szene, damit verschiedene Objekte auf ihre Eigenschaften geprüft werden können.

Nachdem nun die Benutzeroberfläche ausführlich erklärt wurde, wird nun auf die Begriffe Prefabs und Skripte eingegangen, da diese essentiell für den Umgang mit Unity sind.

## Prefabs

Bei Prefabs handelt es sich um fertige Objekte, die in Szenen verwendet werden können<sup>33</sup>. Diese können dabei als Vorlage gesehen werden, damit nicht jedes Objekt mit gleichen Eigenschaften erneut erstellt werden müssen.

## Skripte

In Skripten befindet sich die Logik von Objekten<sup>34</sup>. Ein Beispiel dafür ist das Öffnen einer Truhe. Wenn der Benutzer die Truhe anklickt und öffnen will, steht in einem Skript, was die Truhe zu tun hat. In diesem Beispiel also, dass sie sich öffnen soll.

### 4.1.2 Grafische Oberflächen mit Unity

Die grafische Oberfläche wird mit Unity's zur Verfügung gestelltem UI System implementiert. Der Canvas, zu deutsch Leinwand, stellt dabei die Hauptkomponente bzw. Elternkomponente dar. Alle UI Elemente sind Kinder des Canvas und erben somit die Eigenschaften des Canvas. Unity bietet standardmäßig eine große Anzahl an UI Elementen, wie beispielsweise Buttons, Toggle-Buttons, Dropdown-Listen und Eingabefelder<sup>35</sup>. Der Canvas verwendet das EventSystem Objekt von Unity. Das Event System ermöglicht es, alle Touchscreen-Eingaben an die korrekten UI Elemente in der App weiterzuleiten<sup>36</sup>.

Bei der Implementierung des Canvas wird zwischen drei unterschiedlichen Render-Modi unterschieden<sup>37</sup>. Je nachdem welcher Modus verwendet wird, ist die grafische Oberfläche anders in das Spiel integriert.

---

<sup>33</sup> vgl. Unity3D [15] (2017)

<sup>34</sup> vgl. Unity3D [16] (2017)

<sup>35</sup> Vgl. <https://docs.unity3d.com/Manual/UIInteractionComponents.html>

<sup>36</sup> Vgl. <https://docs.unity3d.com/Manual/EventSystem.html>

<sup>37</sup> Vgl. <https://docs.unity3d.com/Manual/UICanvas.html>

## Screen Space - Overlay

Standardmäßig ist jedes Canvas als Screen Space - Overlay implementiert. Dieser Modus platziert die UI Elemente auf den Bildschirm über die komplette Szene. Die Benutzeroberfläche wird über alle anderen Grafiken, wie die Kamera-Ansicht, gezogen. Wenn die Auflösung geändert wird, passt der Canvas die Größe aller Kindelemente automatisch an die neue Auflösung an. Deshalb ist es besonders wichtig, dass alle UI Elemente unter dem Canvas hängen.

Die folgende Grafik<sup>38</sup> 16 verdeutlicht, dass alle Elemente des Spiels hinter der Benutzeroberfläche positioniert sind.

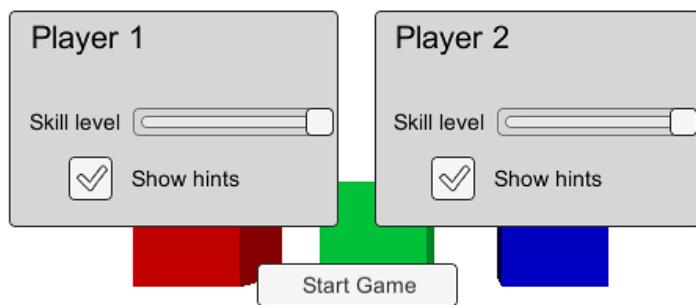


Abbildung 16: Render-Modus: Screen Space - Overlay

Die wesentliche Nachteile vom Screen Space - Overlay ist, dass der sichtbare Bereich des Spieles durch die UI Elemente verdeckt wird. Deshalb ist es besonders wichtig nur die wichtigsten Elementen immer sichtbar im Vordergrund zu lassen. Dies hat Auswirkungen auf die Positionen, Größe, Farbe und Design. Ein weiterer Nachteil ist, dass Spielelemente nicht als Teil des User Interfaces verwendet werden können.

Auf der Gegenseite können keine UI Elemente oder wichtige Informationen von den Gegenständen im Spiel verdeckt werden und ein weiterer Vorteil ist, dass die Position auf dem Bildschirm immer an die Auflösung angepasst wird und sich nicht ändert, unabhängig davon welche Aktionen der Spieler im Spiel ausführt oder auf welcher Hardware NoRPG gespielt wird.

## Screen Space - Camera

Dieser Modus ähnelt dem ersten Modus sehr, aber in diesem Render-Modus wird der Canvas eine vorgegebene Distanz vor einer bestimmten Kamera platziert. Die UI Elemente werden von der bestimmten Kamera gerendert, was bedeutet, dass die Kameraeinstellungen das Erscheinungsbild der Benutzeroberfläche beeinflussen. Wenn sich

<sup>38</sup> Bild entnommen von <https://docs.unity3d.com/Manual/class-Canvas.html>

die Auflösung des Bildschirms verkleinert, oder die Kameraposition und -winkel sich ändern, ändert sich der Canvas automatisch mit und passt die Größe der Elemente an.

Der größte Unterschied ist allerdings, wenn 3D-Objekte in der Szene näher an der Kamera sind als die UI Elemente, werden diese auch auf dem Bildschirm vor der Benutzeroberfläche platziert. Dieses Verhalten wird in der folgenden Grafik<sup>39</sup> 17 deutlich.

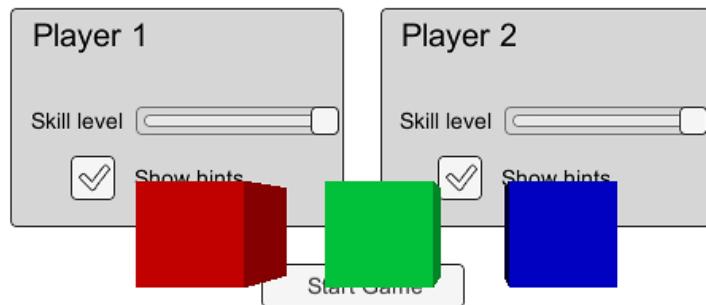


Abbildung 17: Render-Modus: Screen Space - Camera

Wenn die 3D-Objekte näher als die UI-Elemente sind können Informationen und Elemente, wie Buttons, verdeckt werden und die Interaktion mit diesen Elementen ist nicht mehr möglich. Allerdings ermöglicht dieser Modus, dass Spielemente als Teil der UI, auch wenn nur kurzfristig, funktionieren können. Trotz diesen Vorteils ist dieses Verhalten komplexer und fehleranfälliger.

## World Space

In diesem Render-Modus wird der Canvas wie jedes andere Objekt in der Szene behandelt. Die Größe und Position kann manuell gesetzt werden und die UI Elemente werden basierend auf die 3D-Position entweder vor und hinter 3D-Objekten der Szene platziert. Dies ist nützlich für UIs, die dazu bestimmt sind, ein Teil der Welt zu sein. Dies wird auch als „sterbliche“ Benutzeroberfläche bezeichnet, da diese nicht immer im Vordergrund sein muss. Im Gegensatz zum Screen Space - Camera Modus muss die UI Ebene nicht vor die Kamera platziert werden.

Die Größe des Canvas kann manuell eingestellt werden, aber seine Größe auf dem Bildschirm wird von dem Winkel und der Entfernung der Kamera abhängen. Andere Szeneobjekte können hinter, mittendrin oder vor dem Canvas sein. Dieses Verhalten wird in folgender Grafik<sup>40</sup> 18 deutlich.

<sup>39</sup> Bild entnommen von <https://docs.unity3d.com/Manual/class-Canvas.html>

<sup>40</sup> Bild entnommen von <https://docs.unity3d.com/Manual/class-Canvas.html>

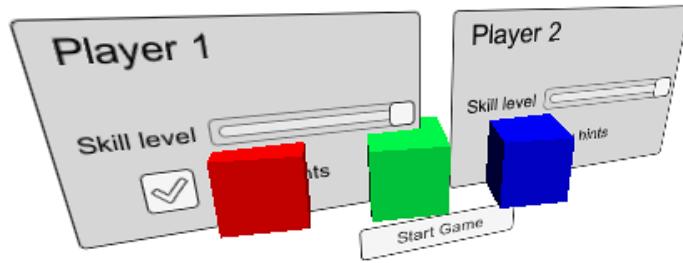


Abbildung 18: Render-Modus: World Space

## 4.2 Visual Studio

Mit Microsoft Visual Studio ist es möglich in verschiedenen Programmiersprachen zu programmieren<sup>41</sup>. Bei der Installation von Unity wird dieses dabei zusätzlich installiert. Dadurch können die Skripte aus Unity in Visual Studio geöffnet und bearbeitet werden. Zusätzlich zu Visual Studio werden auch verschiedene Plug-Ins für die IDE installiert. Dabei handelt es sich unter anderem um eine ausführliche Dokumentation von allen in Unity zur Verfügung stehenden Methoden und Klassen sowie um Test-tools, um verschiedene Tests auszuführen.

## 4.3 C Sharp

C#, gesprochen C Sharp, ist eine Programmiersprache welche von Microsoft entwickelt wurde. Sie wurde zusammen mit „.NET 1.0“ 2002 in der Version 1 veröffentlicht und ist mittlerweile in Version 6 verfügbar.<sup>42</sup>

C# orientiert sich dabei an den Programmiersprachen C, C++, Java, Delphi und Haskell und nutzt deren grundlegenden Konzepte. Aufgrund der Ähnlichkeit zu diesen Sprachen handelt es sich bei C# ebenfalls um eine Objektorientierte Sprache.

Nachfolgend wird nun auf die Grundlegenden Konventionen eingegangen, denen C# zugrunde liegen. Danach wird auf die Verwendung in Unity Skripten eingegangen.

### 4.3.1 Allgemeiner aufbau C#

Der allgemeine Aufbau von C# wird hier am Beispiel von einem Hello World Programm in Listing 1 dargestellt.

<sup>41</sup> vgl. Microsoft [10] (2015)

<sup>42</sup> vgl. Microsoft [9] (2015)

Dabei ist in Zeile eins ein Text zu sehen. Vor diesem stehen zwei Slashes. Damit wird der Text als Kommentar gekennzeichnet. Darum wird dieser Abschnitt vom Compiler beim Compilieren ignoriert. In Zeile zwei wird das Schlüsselwort „using“ gefolgt von einem Namen, in diesem Fall „System“, genutzt. Dieses dient dazu um das Package System in dem Programm zu nutzen.

```
1 // A Hello World! program in C#.
2 using System;
3 namespace HelloWorld
4 {
5     class Hello
6     {
7         static void Main()
8         {
9             Console.WriteLine("Hello World!");
10
11            // Keep the console window open in debug mode.
12            Console.WriteLine("Press any key to exit.");
13            Console.ReadKey();
14        }
15    }
16}
```

Listing 1: Hello World in C#

Als nächstes wird ein Namespace definiert. Innerhalb von diesem eine Klasse namens „Hello“. Innerhalb von dieser wiederum befindet sich der auszuführende Code.

Dieser steht in einer Methode, die in Zeile sieben definiert wird. Bei dieser Methode handelt es sich um die „Main()“ Methode. Diese wird beim Programmstart immer zuerst ausgeführt. In dieser Methode gibt es vier Zeilen Code, darunter ein Kommentar(in Zeile elf). Bei den anderen drei Zeilen handelt es sich um Konsolenausgaben bzw. Konsoleneingaben. Die Zeilen neun und zwölf geben jeweils Text auf der Konsole aus. In Zeile 13 wird dagegen ein Zeichen eingelesen, welches vom Benutzer eingegeben wird.

### 4.3.2 Unity Skripte

Die in Unity verwendeten C# Skripts erben standardmäßig von der Klasse MonoBehaviour wie in Listing 2 zu sehen. Diese Vererbung sorgt dafür, dass jede Klasse verschiedene Methoden zur Verfügung hat. Dazu zählt eine Start-Methode, die beim Laden eines Objekts mit dem Skript ausgeführt wird und eine Update-Methode, die bei jeder Frameaktualisierung ausgeführt wird. Des Weiteren können weitere Methoden genutzt werden. Außerdem sorgt die MonoBehavior Vererbung dafür, dass diese Skripte mit Objekten in Unity verknüpft werden können.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class test : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12     // Update is called once per frame
13     void Update () {
14
15    }
16 }
```

Listing 2: Aufbau eines Unity Skriptes

# 5 Umsetzung

Nachdem im vorherigen Kapitel die technischen Grundlagen erläutert wurden, wird nun auf die Umsetzung eingegangen. Dabei wird mit der umgesetzten Architektur begonnen, um einen groben Überblick über die einzelnen Komponenten zu bekommen. Anschließend wird die Datenhaltung und Anbindung an eine Datenbank und die Bestandteile der App behandelt.

## 5.1 Architektur

Die Architektur entspricht dem MVC Prinzip, bei dem der Controller für die Steuerung und Koordination zwischen Model, also Daten, und View, also Darstellung. In Abbildung 19 ist die Architektur grafisch dargestellt.

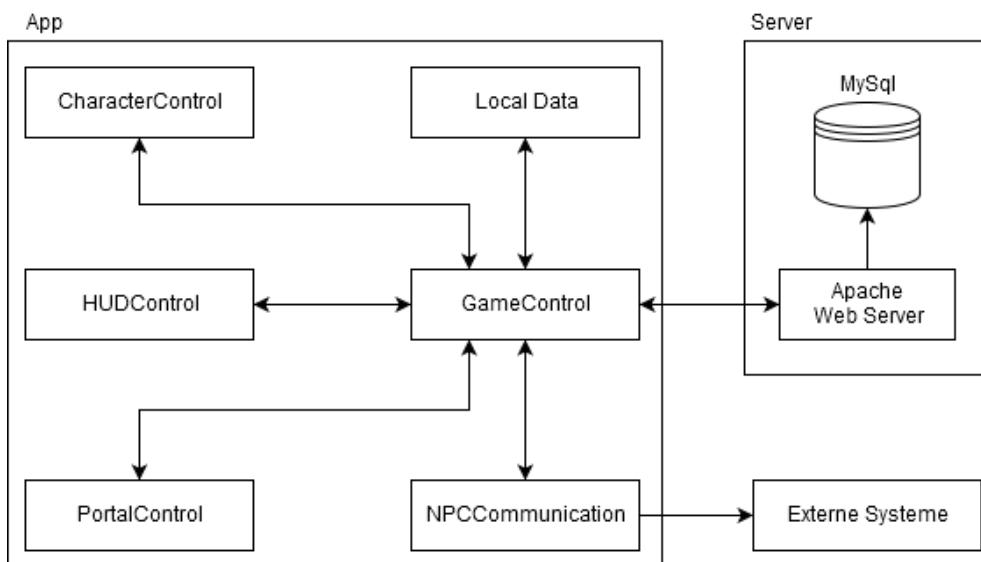


Abbildung 19: Softwarearchitektur

Das GameControl ist die zentrale Komponente in NoRPG. Diese regelt den Datenzugriff zwischen App und Server, und ist innerhalb der App mit allen weiteren Komponenten verbunden.

Innerhalb der App ist die Architektur wie eine Art Stern-Schema aufgebaut. Das GameControl fungiert als eine zentrale Steuereinheit. Dieser steuert weitere Controller, die für die Steuerung einzelner Bereiche und ihre Komponenten zuständig ist. So ist beispielsweise das HUDControl verantwortlich für die Verarbeitung der Benutzereingaben über das Touchscreen an die einzelnen UI Komponenten, so dass die richtigen Informationen angezeigt werden. Neben der Steuerung von Controllern ist das GameControl darüber hinaus auch die zentrale Stelle zum Speichern und Laden der lokalen Daten.

Außerhalb der App steht auf dem Server eine MySQL Datenbank bereit, welche sich um die persistente Datenhaltung kümmert. Diese wird mit dem GameControl über ein Apache Web Server angesprochen und gesteuert.

Die einzelnen Komponenten der Architektur werden im folgenden genauer betrachtet und erklärt.

## 5.2 Datenhaltung und Anbindung an eine Datenbank

In diesem Kapitel wird die Datenhaltung auf dem Handy und dem Server erläutert. Begonnen wird dabei mit der Beschreibung der Datenhaltung auf dem Handy. Dabei wird auf verschiedene Aspekte der Speicherung eingegangen.

### 5.2.1 Anbindung an einen Server

Zum Senden der Daten für die Registrierung wird das Skript „SendDataToServer.cs“ genutzt. In diesem werden die Daten der Registrierung zwischengespeichert und am Ende an den Server gesendet. Dazu wird die Methode „SendRegister()“ genutzt. In dieser wird die Methode „RegisterUser()“ als Coroutine gestartet. Dazu werden alle zehn Parameter übergeben. Dazu zählt beispielsweise der Username, die Email, das Passwort und der gewählte Charakter. Bei einer Coroutine handelt es sich um einen Thread, welcher beliebig gestartet, pausiert und beendet werden kann.

Innerhalb dieser Methode wird zu Beginn ein Hash erstellt, um am am Server zu überprüfen, ob die Anfrage gültig ist. Dieser besteht dabei aus Teilen der Eingabe und einem zusätzlichen geheimen Schlüssel, der nur der App und dem Server bekannt sind. Dadurch wird die Sicherheit gesteigert und es wird Angreifern erschwert unberechtigte Zugriffe auf die Datenbank zu tätigen. Bei dem Hash handelt es sich dabei um die

MD5 verschlüsselten Eingabedaten. Dadurch ist es fast nicht möglich, einen validen Hash zu bilden, ohne diese Daten zu kennen.

Nachdem der Hash erstellt wurde, werden alle Parameter in Form einer URL aneinander gehängt. Anschließend wird die URL an ein WWW Objekt übergeben und solange gewartet, bis es eine Antwort gibt. Sofern es keinen Fehler gab, wird ein Text ausgegeben, welche vom Server gesendet wird, andernfalls eine Fehlermeldung.

```
1 public class SendDataToServer : MonoBehaviour {
2     private static string secretKey = "*****";
3     public static string registerURL = "http://norpq.it.dh-karlsruhe.de/register.php?";
4     public static string loginURL = "http://norpq.it.dh-karlsruhe.de/login.php?";
5
6     [...]
7
8     private void SendRegister() {
9         StartCoroutine(RegisterUser(userText, emailText, MD5Test.Md5Sum(passwordText),
10         firstnameText, lastnameText, birthdayText, genderText,
11         countryText, native_languageText, selected_characterText));
12     }
13
14     IEnumerator RegisterUser(string user, string email, string password, string firstname,
15     string lastname, string birthday, string gender, string country,
16     string native_language, string selected_character) {
17
18         string hash = MD5Test.Md5Sum(user + email + password
19         + firstname + country
20         + selected_character + secretKey);
21
22         string post_url = registerURL
23         + "user=" + WWW.EscapeURL(user) + "&email=" + WWW.EscapeURL(email)
24         + "&password=" + WWW.EscapeURL(password) + "&firstname=" + WWW.EscapeURL(firstname)
25         + "&lastname=" + WWW.EscapeURL(lastname) + "&birthday=" + WWW.EscapeURL(birthday)
26         + "&gender=" + WWW.EscapeURL(gender) + "&country=" + WWW.EscapeURL(country)
27         + "&native_language=" + WWW.EscapeURL(native_language)
28         + "&selected_character=" + WWW.EscapeURL(selected_character)
29         + "&hash=" + hash;
30
31         WWW hs_post = new WWW(post_url);
32         yield return hs_post;
33
34         if (hs_post.error != null) {
35             print("There was an error posting the high score: " + hs_post.error);
36         } else {
37             status.text = hs_post.text;
38         }
39     }
40 }
```

Listing 3: Skript SendDataToServer.cs

Auf dem Server läuft für die Datenannahme das PHP Skript „register.php“. Dieses nimmt die Daten aus der URL entgegen und speichert diese zunächst in Variablen ab. Anschließend wird auch in diesem Skript ein Hash gebildet und mit dem gesendeten Hash abgeglichen. Sollte es hier einen Fehler geben, sendet der Server einen Fehler

zurück, aber falls die Hashes identisch sind, wird eine Verbindung zu der Datenbank aufgebaut und ein Eintrag in der accounts Tabelle erstellt. Anschließend wird eine erfolgreich Meldung an den Client gesendet.

Für den Login innerhalb der App wird identisch vorgegangen. Dabei wird jedoch kein Datensatz in die Datenbank geschrieben, sondern nur gelesen. Des Weiteren wird der Hash aus weniger Werten gebildet, da nur der Username und das verschlüsselte Passwort übermittelt werden.

Auf dem Server läuft eine MySQL Datenbank. Diese sorgt für eine permanente Datenhaltung und die Synchronisation zwischen den Clients. Die Datenbank besteht dabei aus den Tabellen „accounts“ und „player\_data“. In „accounts“ sind alle Informationen zu einem User hinterlegt, darunter auch die Daten für den Login. In der Tabelle „player\_data“ wird der Fortschritt von jedem Spieler gespeichert, damit dieser von jedem Gerät aus erreichbar ist.

## 5.2.2 Datenhaltung auf dem Handy

Die Daten auf dem Handy werden im Javascript Object Notation (JSON)-File Format gespeichert. Die JSON Datei für die Standards wird vom Server geladen.

### Standards JSON

Die JSON für die Standards ist ein wichtiger Bestandteil des Spieles. In dieser Datei befinden sich alle wichtigen Informationen zu den Standards und deren dazugehörigen Spiele. Diese Datei liegt zentral auf einem Server und kann dort gewartet und gepflegt werden. Dadurch ist es möglich, zusätzliche Standards und Spiele einzufügen, ohne dass der Nutzer die App aktualisieren muss.

Damit der Nutzer die aktuelle Datei zur Verfügung hat, versucht das Spiel zu Beginn eine Verbindung zu dem Server aufzubauen. Gelingt dies, wird die Datei von Server geladen und auf dem Handy gespeichert. Nachdem die Datei lokale gespeichert wurde, wird sie in das Spiel geladen. Dazu wird aus dem JSON eine Liste erstellt, welche überall im Spiel nutzbar ist.

Für das Laden und verarbeiten sind fünf Klassen verantwortlich. Im Folgenden wird nur detailliert auf die Klasse „WebJSONConfigReader“ eingegangen. Die anderen Klassen sind im Anhang (s. Anhang 6) abgebildet und werden nur erwähnt.

Die Klasse „MappingStandardsToCourses“ ist die Datenhaltungsklasse. In dieser ist die Struktur der einzelnen Objekte geregelt. In der Klasse „MappingStandardsToCoursesBean“ spiegelt die Struktur der JSON-Datei wieder.

Der „WebJSONConfigReader“ wird in der Klasse „LoadingScreen“ genutzt. Dort wird eine Instanz dieser Klasse erzeugt und es wird die Methode „LoadSettings“ ausgeführt und in der Variable „Settings“ gespeichert. Diese Methode ist in Listing 17 zu sehen.

```

1  public MappingStandardsToCourses LoadSettings() {
2      WWW www = new WWW(settingsURL);
3      while (!www.isDone) { }
4      string json;
5      string LocalFilePath = Application.persistentDataPath + "/v1.json";
6
7      if (string.IsNullOrEmpty(www.error)) {
8          json = www.text;
9          File.WriteAllText(LocalFilePath, json);
10     } else if (File.Exists(LocalFilePath)) {
11         json = File.ReadAllText(LocalFilePath);
12     } else {
13         json = "";
14         File.WriteAllText(LocalFilePath, json);
15     }
16
17     MappingStandardsToCoursesBean settingsBean =
18     MappingStandardsToCoursesBean.CreateFromJSON(json);
19
20     string json2 = JsonUtility.ToJson(settingsBean);
21     List<Classes> classes = new List<Classes>();
22     foreach (var classe in settingsBean.classes) {
23         List<Courses> courses = new List<Courses>();
24         foreach (var course in classe.courses) {
25             List<Standards> standards = new List<Standards>();
26             foreach (var standard in course.standards) {
27                 List<Games> games = new List<Games>();
28                 List<string> vor = new List<string>();
29                 List<string> nach = new List<string>();
30                 foreach (var game in standard.games) {
31                     games.Add(new Games(game.id, game.name, game.url));
32                 }
33                 foreach (var vorbedingung in standard.vorbedingungen) {
34                     vor.Add(vorbedingung);
35                 }
36                 foreach (var nachbedingung in standard.nachbedingungen) {
37                     nach.Add(nachbedingung);
38                 }
39                 Games[] g = games.ToArray();
40                 string[] v = vor.ToArray();
41                 string[] n = nach.ToArray();
42                 standards.Add(new Standards(standard.name, v, n, g));
43             }
44             Standards[] s = standards.ToArray();
45             courses.Add(new Courses(course.name, s));
46         }
47         Courses[] c = courses.ToArray();
48         classes.Add(new Classes(classe.name, c));
49     }

```

```
50     Classes[] cla = classes.ToArray();
51     return new MappingStandardsToCourses(cla);
52 }
```

Listing 4: Methode LoadSettings

In dieser wird zu Beginn ein neues WWW Objekt erstellt. Dieses baut eine Verbindung zum Server auf. Anschließend wird in Zeile 7 überprüft, ob es zu einem Fehler beim Verbindungsauflauf gekommen ist. Wenn es zu keinem Fehler gekommen ist, wird der Inhalt des WWW Objekts in eine Datei geschrieben und gespeichert. Für den Fall das ein Fehler aufgetreten ist und das WWW Objekt keine Verbindung zum Server aufgebaut werden kann, wird geprüft ob bereits eine ältere Datei vorhanden ist. Wenn dies der Fall ist wird der Text aus dieser Datei zum Erzeugen des Objektes genutzt. Anschließend wird in Zeile 18 aus dem Text eine JSON Struktur erzeugt. Aus dieser wird anschließend in mehreren Schritten ein mehrdimensionaler Array erzeugt und zurückgegeben.

## 5.3 Benzerschnittstelle

In diesem Abschnitt dieses Kapitels wird auf die Umsetzung der Benutzerschnittstelle eingegangen. Wie schon bereits im SRS erwähnt handelt es sich bei der Benutzerschnittstelle um die grafische Oberfläche der Anwendung. Dafür wird zunächst auf die Umsetzung der verschiedenen UI Elemente eingegangen sowie relevante Funktionalitäten beschrieben. Abschließend wird die Evaluation des Usability Tests behandelt.

### 5.3.1 Benutzeroberfläche

Die Implementierung der Benutzeroberfläche ist von Anfang der Umsetzungsphase bis zum Ende ein kontinuierlicher, iterativer und wichtiger Entwicklungsprozess. Es beginnt mit einer ganz simplen Steuerung zum Testen von Funktionen bis zum Perfektionieren aller benötigter User Interface Elemente. Als erstes werden die Startbildschirme und der Registrierungsprozess behandelt.

#### Startbildschirm und Registrierung

Das aller erste, was der Spieler von NoRPG sieht und womit er interagieren darf, ist der Startbildschirm. Dieser ist besonders wichtig, da hier die ersten Eindrücke schon

gesammelt werden. Bei dem Startbildschirm wird zwischen zwei Varianten unterschieden. In der ersten Variante startet der Spieler NoRPG das erste mal oder ist nicht mit seinem Account angemeldet. In der zweiten Variante ist der Benutzer schon angemeldet. Diese beide Varianten sind in der Grafik 20 dargestellt.

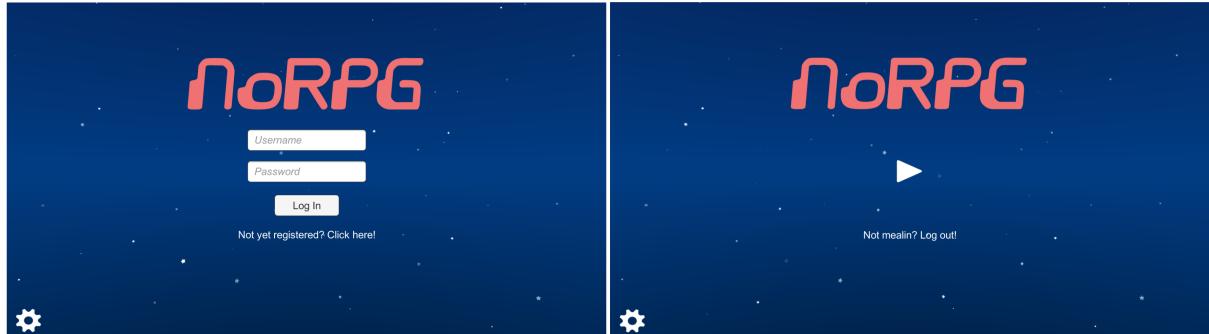


Abbildung 20: User Interface: Startbildschirm

Beide Varianten ähneln sich nicht nur im aussehen sondern auch in der Implementierung. Die Startbildschirme wurden nach dem Gestaltungsprinzip Figure/Ground entworfen und umgesetzt. Figure/Ground unterteilt die Wahrnehmung visueller Informationen in Figure (Vordergrund) und Ground (Hintergrund). Dabei liegt der Fokus auf dem Vordergrund. Das Design und Aussehen der Komponenten ist dabei Clean und Simple, zu Deutsch sauber und einfach. Dieter Rams beschreibt in seinem Buch "Weniger, aber besser / Less but better" [ramsDesign], dass einfaches Design nicht über die Subtraktion von Dingen von einem Design handelt, sondern die Gesamteffektivität des Designs zu verbessern. Elemente in der Benutzeroberfläche haben so wenig Text und Farbe wie notwendig und werden, falls möglich, nur mit Symbolen abgebildet. Dabei wird ein intuitives Design verwendet, welches die Spieler wahrscheinlich aus anderen Spielen schon kennen und nicht neu lernen müssen.

Der Vordergrund von beiden Varianten ist in zwei Abschnitte unterteilt. Der erste Abschnitt ist in der Mitte des Bildschirms positioniert und bindet die Aufmerksamkeit des Spielers. In der ersten Variante besteht dieser aus zwei Eingabefeldern für die Benutzerdaten, dem Log-In Button und einem Textfeld, um sich zu registrieren. In der zweiten Variante allerdings besteht dieser Abschnitt nur aus einem Button zum Starten des Spieles und einem Textfeld für die Abmeldung. Der zweite Abschnitt ist an der unteren Seite des Bildes positioniert und besteht aus einem Button für die Einstellungen. Dabei öffnet sich kein Fenster, sondern werden Symbole für die Einstellungsmöglichkeiten ein- und ausgeblendet.

Der Hintergrund setzt sich aus dem Logo und einem Sternenhimmel zusammen. Diese Hintergrundelemente sind 3D-Elemente, welche in der Spielwelt angebracht sind und durch eine Kamera aufgezeichnet werden. Neben diesen Hintergrundelementen

gibt es ein weiteres Element, welches erst durch die Betätigung des Log-In Buttons auftaucht. Es handelt sich um ein drehendes Icon, dass den Ladeprozess darstellt und somit dem Spieler ein Feedback gibt. Der Hintergrund ist sehr schlicht aber farbenfroh gestaltet. Zudem sind die Sterne am Himmel animiert, welches einen besonders positiven Eindruck hinterlassen soll.

Die Startbildschirme sind mit dem Standard Render-Modus Screen Space - Overlay umgesetzt worden. Dieser Modus unterstützt das Gestaltungsprinzip Figure/Ground, indem die UI Elemente vor den Hintergrund gerendert werden. Fernerhin kann der Spieler sich innerhalb des Startbildschirms nicht bewegen oder andere Interaktionen ausführen, sodass ein anderer Render-Modus notwendig wäre.

Auf Grund der Tatsache das es zwei Startbildschirme gibt, muss bevor der anzuzeigende Startbildschirm angezeigt wird eine Überprüfung stattfinden. Die aller erste Szene die in NoRPG geladen wird, ist eine nahezu leere Szene, die nur aus dem Hintergrund, einem kleinen Textfeld und einem Skript besteht. Diese Szene ist deshalb so minimalistisch gestaltet, damit das Laden und somit das Ausführen des Skriptes sehr schnell durchgeführt werden kann, so dass der Spieler im besten Fall diese Szene erst gar nicht sieht und bemerkt. Das Skript prüft, ob die in Kapitel 5.1 beschriebene lokale Datei, mit den Informationen über den Spielstand des Spielers, vorhanden und vollständig ist. Ist dies der Fall, kann davon ausgegangen werden, dass der Spieler angemeldet ist und die zweite Variante des Startbildschirms wird geladen. Der Hintergrund ist notwendig, falls das Laden länger braucht der Spieler keinen weißen beziehungsweise leeren Hintergrund sieht. Das Textfeld in dieser Szene ist angebracht, damit der Spieler im Fehlerfall informiert wird.

Die nächste Benutzeroberfläche, die behandelt wird, ist die Registrierung. Ein Teil des Registrierungsprozesses ist in Abbildung 21 abgebildet.

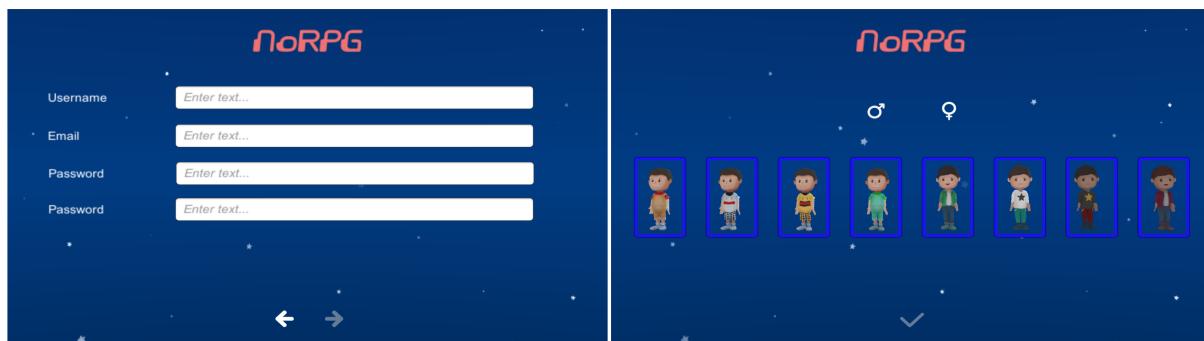


Abbildung 21: User Interface: Registrierung

Es fällt auf, dass auch hier das Gestaltungsprinzip Ground/Figure umgesetzt wurde. Dies hat die gleichen Gründe wie bei den Startbildschirmen und die Konsistenz außerhalb des eigentlichen Spieles bleibt erhalten. Neben diesen Gestaltungsprinzip wurde

versucht das Prinzip Simple und Clean weiter fortzuführen, allerdings handelt es sich bei der Registrierung um einen komplexeren Prozess. Daher wurde ein weiteres Gestaltungsprinzip verwendet.

Die UI Elemente sind tabellarisch angeordnet und deren relative Abstände gruppieren einzelne Elemente. Dieses Prinzip wird Proximity genannt. Dadurch weiß der Spieler welche Eingabefelder zu welchen Textfeldern gehören und kann somit die korrekten Informationen angeben. Auch hier ist das User Interface in zwei Abschnitte unterteilt. Der erste Abschnitt ist in der Mitte des Bildschirms positioniert und bildet das Registrierungsformular ab. Der zweite Abschnitt ist am unteren Teil des Bildes und dient zur Navigation durch den gesamten Registrierungsprozess.

Der größte Unterschied ist allerdings, dass diese Szene mit dem Render-Modus World Space implementiert wurde. Der letzte Schritt im Registrierungsprozess ist die Auswahl des Charakters. Damit die einzelnen Charaktere in die UI integriert werden konnten, musste dieser Modus umgesetzt werden. Dadurch wurde möglich, dass zunächst die 3D Objekte der Welt nicht von den UI Elementen verdeckt werden und anschließend kann der Spieler auf seinen gewünschten Charakter klicken und es wird eine Animation ausgeführt.

## **Head-Up Display im Spiel**

Das HUD bildet das User Interface innerhalb des Spieles ab. Grundsätzlich kann das HUD in vier Abschnitte unterteilt werden. Die Abschnitte sind an den Ecken des Bildschirmes positioniert. Diese Positionierung hat den Vorteil, dass die UI Elemente nicht vom Spiel ablenken und Informationen verdecken, sowie der Spieler die Elemente einfach mit den Händen erreichen kann. Abbildung 22 stellt die standardmäßige Ansicht von NoRPG dar.

Das ganze HUD ist mit dem Render Modus Screen Space - Overlay implementiert. Das HUD steht nicht im Fokus, darf allerdings nicht durch 3D Objekte verdeckt werden. Die HUD Elemente dienen zur Steuerung des Spieles und sind genauso wichtig wie das Spiel selber.

**Der Joystick** Der Joystick befindet sich in der unteren linken Ecke. Der Spieler kann mit Hilfe des Joysticks sich im Spiel fortbewegen. Von allen UI Elementen die NoRPG implementiert wurden, ist der Joystick das einzige Elemente, das nicht standardmäßig von Unity zur Verfügung stellt. Bei dem Joystick handelt es sich um ein Asset, welches



Abbildung 22: User Interface: Head-Up Display

aus dem Unity Asset Store erworben wurde. Die Besonderheit ist, dass der Joystick aus drei Elementen besteht, welche übereinander liegen.

Das oberste Element ist der sogenannte Stick. In der Abbildung 22 handelt es sich um kreisförmige hellgraue Elemente in der linken Ecke. Der Spieler interagiert mit dem Stick und kann diesen in alle Richtungen bewegen. Diese Bewegungen werden dann anschließend mit Einsatz eines Skriptes an den Charakter im Spiel übergeben.

Bei dem zweiten Element handelt es sich um die Basis des Joysticks. Bei Bewegungseingaben des Spielers ändert sich die Position nicht. Die Basis dient zum einen der Orientierung und begrenzt den Radius des Sticks. Je nachdem wie weit außerhalb der Stick sich innerhalb dieses Radius befindet, bewegt sich der Charakter mit einer unterschiedlichen Geschwindigkeit.

Das unterste Element wird als Touch Zone bezeichnet. Diese ermöglicht innerhalb eines festgelegten Bereiches, dass der Spieler seinen Daumen in eine beliebige Position setzen kann und der Joystick wird genau unter seinem Daumen platziert. Dadurch wird gewährleistet, dass junge Spieler, die eher kleinere Hände haben, und Erwachsene komfortabel den Joystick verwenden können.

**Die Interaktionbuttons** Bei den Interaktionsbuttons handelt es sich um die Buttons, die für die Steuerung der Interaktionen in NoRPG zuständig sind. Diese sind an der unteren rechten Ecke positioniert. Es wird zwischen zwei Buttons mit unterschiedlichen Zuständigkeiten unterschieden. Der rechte Button, der A-Button, ist für das Starten einer Interaktion und für das Akzeptieren bei Entscheidungsfällen verantwortlich. Der zweite Button, der B-Button, wird zum Abbrechen von Interaktionen benötigt.

Allerdings gibt es Situationen, in denen beide Buttons die gleiche Funktionalität abbilden, beispielsweise am Ende einer Konversation kann der Spieler mit beiden Buttons diese beenden. Eine Konversation ist in Abbildung 23 im oberen linken Teil abgebildet.

Damit jedoch nicht immer irgendeine Interaktion gestartet wird, wenn der Spieler die Taste verwendet, wurde ein Skript implementiert, der den Abstand vom Spieler zum nächsten Objekt im Spiel bestimmt. Erst wenn der Spieler sich in der Nähe des Objektes befindet, wird eine Interaktion gestartet. Dies wird später im Kapitel 5.4.4 genauer behandelt.

Bei den Buttons handelt es sich um standardmäßige Buttons von Unity. Allerdings sind diese mit einer animierbaren Textur überzogen. Die Textur verändert das Aussehen des Buttons und ermöglicht visuelles Feedback zurückzugeben, falls der Button geklickt wird oder für einen Moment nicht benutzbar ist.

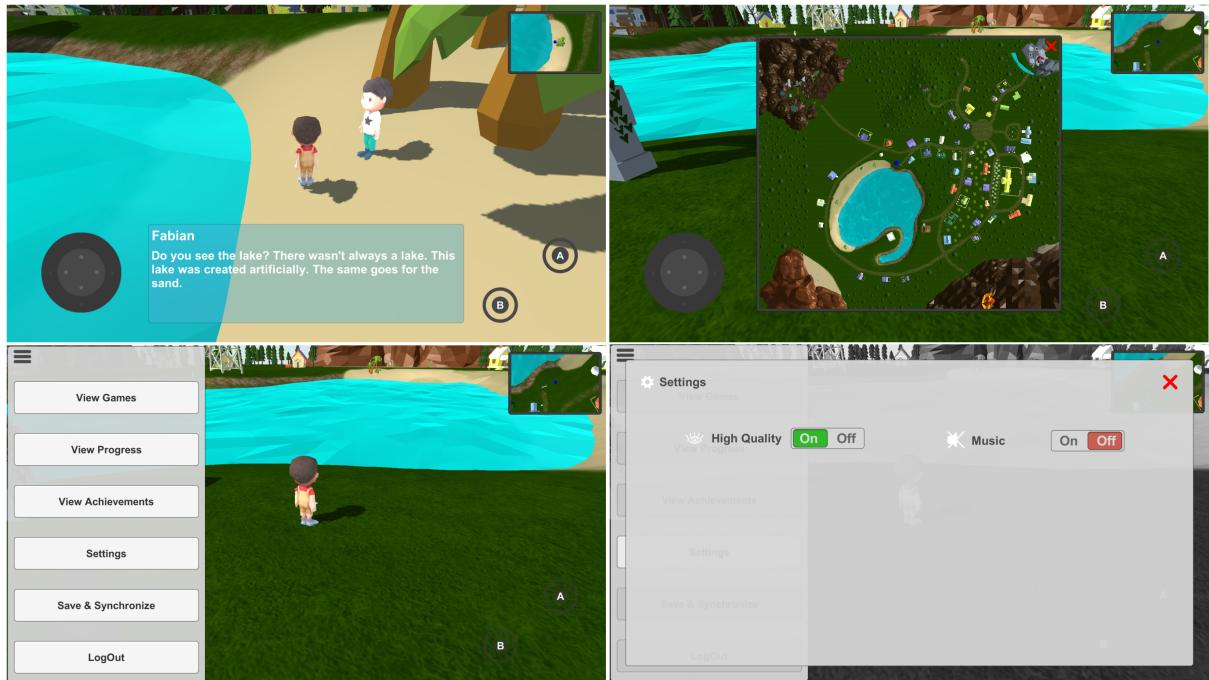


Abbildung 23: User Interfaces: Kommunikation, Menü und Einstellungen-Fenster

**Das Menü** Der Button für das Menü befindet sich in der oberen linken Ecke. Dieser ist im oberen Teil des Bildschirms positioniert, da die Häufigkeit der Benutzung wesentlich geringer ist, als bei den bisher besprochenen UI Elementen.

Bei der Verwendung des Buttons öffnet sich das Menü im linken Teil des Bildschirms. Dies ist im unteren linken Bild der Abbildung 23 abgebildet. Das geöffnete Fenster beinhaltet alle restlichen Funktionen, die nicht direkt über das HUD erreicht werden können.

nen. Beispielsweise findet der Spieler die Einstellungen oder kann seinen Fortschritt öffnen. Diese Funktionen sind in dem Menü, da die Informationen nicht immer benötigt werden und deshalb nicht die ganze Zeit sichtbar sein müssen. Während das Menü offen ist können keine anderen UI Elemente, wie die Buttons zur Interaktion verwendet werden. Der Spieler muss erst das Menü schließen um wieder mit dem Spiel fortfahren zu können.

Es wird zwischen zwei Varianten von Funktionen im Menü unterschieden. Einige der Funktionen öffnen ein weiteres Fenster, wie in der Abbildung 23 im unteren rechten Bild abgebildet, und andere werden direkt im Menü ausgeführt. So wird beispielsweise für die Einstellungen ein weiteres Fenster geöffnet, wohingegen das Speichern direkt im Menü durchgeführt wird, ohne dass sich ein Fenster öffnet.

**Die Karte** Die Karte im oberen rechten Eck des Bildschirms wird als Mini Map bezeichnet. Sie bietet dem Spieler einen Überblick über seine aktuelle Position. Realisiert wurde diese Mini Map mit einer zusätzlichen zweiten Kamera, die von oben auf den Spieler schaut. Das Objekt der Mini Map ist am Charakter angebracht, damit der Spieler und die Mini Map sich immer in der gleichen Position befinden.

Da zwei Kameras die Komplexität der Szene erhöhen, indem alle Bilder doppelt dargestellt werden, muss die zusätzliche Kamera angepasst werden. Diese rendert die Szene nicht mit der gleichen Qualität wie die Hauptkamera. Des Weiteren werden einige Elemente, wie beispielsweise Schatten oder zu kleine 3D-Objekte wie Blumen nicht in der Mini Map gerendert.

Der Spieler kann auf die Mini Map klicken und es wird die komplette Karte der aktuellen Spielwelt geöffnet. Diese Übersicht wird ebenfalls mit einer Kamera aufgenommen. Der Unterschied allerdings ist, dass diese Kamera nur eingeschaltet ist, also nur dann rendert wenn der Spieler auf die Mini Map klickt. Die Karte kann in Abbildung 23 im oberen rechten Bild betrachtet werden.

Die Kamera ist nicht wie die Mini Map am Spieler angebracht, sondern schwebt über der ganzen Szene. Die Karte zeigt ebenfalls keine Schatten oder zu kleine Objekte an, wodurch Ressourcen gespart werden.

Während die Karte offen ist kann der Spieler nicht das Menü öffnen oder eine Interaktion starten, allerdings kann sich der Spieler mit dem Joystick bewegen. Die Kamera dient zur Orientierung in der Spielwelt, deshalb darf der Spieler sich bewegen. Die Karte kann während dem Laufen geöffnet und geschlossen werden.

### 5.3.2 Usability Evaluation

Ein Usability Test wird durchgeführt, um die Gebrauchstauglichkeit einer Software oder Hardware mit den potenziellen Benutzern zu überprüfen. Usability wird durch die Attribute Erlernbarkeit, Effizienz, Einprägsamkeit, Fehlerrate und Zufriedenheit<sup>43</sup> beschrieben. Unter dem Attribut Erlernbarkeit beschreibt Nielsen in seinem Buch, dass die Anwendung leicht zu erlernen sein muss, damit der Benutzer in kurzer Zeit beginnen kann das Spiel zu spielen. Dazu zählt beispielsweise die Steuerung. Diese ist wichtig, damit der Benutzer sich im Spiel intuitiv bewegen kann. So bald der Spieler das System gelernt und verstanden hat, soll ein hohes Maß an Produktivität möglich sein. Dies wird als Effizienz bezeichnet. Das System sollte allerdings leicht zu merken und einprägsam sein, so dass auch der Gelegenheitsbenutzer in der Lage ist, nach einer gewissen Zeit ohne Verwendung der Anwendung, in das Spiel zurückzukehren, ohne alles nochmals neu lernen zu müssen. Nielsen weist zudem darauf hin, dass Fehler für den Benutzer sehr frustrierend sein können. Daher sollte das System eine geringe Fehlerquote haben, so dass Benutzer bei der Verwendung nur wenig Fehler machen und falls Probleme auftauchen, diese den Spaß am Spielen nicht einschränken<sup>44</sup>. Das letzte Attribut beschreibt den Gesamteindruck.

Es wird zwischen formativem und summativem Usability Test unterschieden. Formative Tests werden während der Entwicklung durchgeführt. Die Aufgabe ist es ein spezielles Problem bzw. Szenario zu testen. Es handelt sich dabei um eine kleine Studie für schnelles Feedback, welche wiederholt durchgeführt werden kann<sup>45</sup>. Der summative Test findet einmalig nach der Entwicklung statt, bevor das fertige Produkt ausgeliefert wird. Dabei handelt es sich um eine umfangreiche Studie, die alle Funktionen der Benutzeroberfläche bewertet<sup>46</sup>.

Ein Usability Test kann in verschiedenen Testumgebungen durchgeführt werden. Es wird zwischen Labortest, Feldtest und Remotetest unterschieden. Bei einem Labortest handelt es sich entweder um ein Usability Labor oder um einen allgemein nutzbaren Raum. Der ausschlaggebende Unterschied zwischen diesen beiden ist, dass das Usability Labor nur für Usability Tests und Evaluationen verwendet wird. Die potenziellen Tester werden eingeladen und führen die Auswertung in diesem Raum durch. Zu den Basisutensilien gehört neben Nahrung und Verpflegung die notwendige Hardware und Software, sowie ein Moderator bzw. Experte für Fragen und Probleme<sup>47</sup>. Der Feldtest oder auch Mobiler Test kann potentiell überall durchgeführt werden, sei es

<sup>43</sup> Vgl. Nielsen [NielsenUI] Seite 26

<sup>44</sup> Vgl. Nielsen [NielsenUI] Seite 27ff.

<sup>45</sup> Vgl. Rubin und Chisnell [handbookUsability] Seite 29f.

<sup>46</sup> Vgl. Rubin und Chisnell [handbookUsability] Seite 34f.

<sup>47</sup> Vgl. Rubin und Chisnell [handbookUsability] Seite 101f.

beim Kunden, in einem öffentlichen Gebäude oder in einem Café. Der Vorteil im Vergleich zu einem Labortest ist, dass der Benutzer sich einer gewohnten und reale Umgebung mit echten Lichtverhältnissen befindet. Allerdings ist es wahrscheinlicher das der Tester durch die Umwelt abgelenkt wird<sup>48</sup> und der Experte mit zum Kunden. Bei der letzten Möglichkeit ist der Benutzer und Moderator örtlich voneinander getrennt. Der Remotetest kann synchron, indem der Moderator und die Teilnehmer per Audio- oder Videokonferenz verbunden sind, oder asynchron stattfinden. Der wesentliche Vorteil ist, dass dadurch potenziell mehr Testpersonen teilnehmen und es günstiger ist. Allerdings können keine komplexen Anforderungen gestellt werden und bei Fragen ist der Tester mehr oder weniger auf sich gestellt.

## Vorbereitung und Durchführung

Bei der in diesem Kapitel beschrieben Evaluation handelt es sich um einen summariven Remotetest. Der Usability Test findet erst nach der Entwicklung von NoRPG statt und nachdem alle Funktionen implementiert wurden. Die Gründe für einen Remote-test sind, da eine es sich bei NoRPG um eine mobile Anwendung handelt und diese am besten auf dem eigenen Smartphone getestet werden kann. Dadurch werden auch verschiedenste Hardware und Android Versionen geprüft und validiert. Ein weiterer essentieller Punkt sind die Kosten. In dem Umfang dieser Studienarbeit ist es nicht möglich ein Feldtest oder Labortest durchzuführen.

In der Vorbereitung des Usability Tests gilt es zunächst die Benutzerprofile zu erstellen, die zu testenden Szenarien und Ziele, sowie den Umfang zu definieren. Anschließend die Unterlagen vorzubereiten, die Teilnehmer zu rekrutieren und den Zeitraum festzulegen. Das Profil für die Tester ist das gleiche wie das der potenziellen Spieler. Daher wird kein festes Profil für die Evaluation festgelegt, da NoRPG für jung oder alt und männlich oder weiblich ausgelegt ist. Jedoch gilt bei der Rekrutierung zu beachten, das von jeder Alters- und Geschlechtsgruppe eine gewisse Anzahl vertreten ist.

Das Ziel dieser Evaluation ist das Testen der Benutzeroberfläche, da es sich bei einer mobilen Anwendung um die einzige Interaktionsschnittstelle für die Benutzer handelt. Aus diesem Grund ist eine fehlerfreie, intuitive und gleichzeitig gut aussehende Benutzeroberfläche sehr wichtig. Zu den abzubildenden Szenarios gehört zum Beispiel die Registrierung, der Log-In oder die Liste der heruntergeladenen Spiele. Alle UI Elemente sollen in diesem summariven Test geprüft und evaluiert werden und aus diesem Grund ist der Umfang dieser Evaluation größer. Diese findet in dem Zeitraum 24.04.2017 bis 30.04.2017 statt.

---

<sup>48</sup> Vgl. Rubin und Chisnell [handbookUsability] Seite 98f.

Für die Evaluation wurde ein Dokument für die Tester erstellt. Dieses beinhaltet alle notwendigen Information und ist zugleich eine Schrittanleitung, um alle UI Elemente testen zu können. Ein ausgefülltes Formular??, dessen Publikation von dem Tester genehmigt wurde, befindet sich im Anhang dieser Arbeit. Neben den Informationen gibt es mehrere Spalten, die für die Bewertung dienen. Bewertet werden kann ein Szenario mit "++", "+", "0", "und "-".

## Bewertung

Für die Evaluation von NoRPG wurde an alle potenzielle Tester eine Prerelease Version der App und der auszufüllende Bewertungsbogen verteilt. Weitere Anweisungen wurden nicht mitgeteilt.

Zunächst werden alle zusätzlichen Informationen, die aus den Fragebögen gewonnen werden konnten, behandelt und deren Verteilung betrachtet.

Bei dem durchschnittlichen Tester handelt es sich um einen 19 Jahre alten männlichen Tester aus Deutschland. Diese Aussage beruht auf der Auswertung des Fragebogens, welche in Abbildung 24 abgebildet sind. Von allen teilnehmenden Testern sind davon 81% männlich und die übrigen 19% weiblich. Für die Auswertung wurden drei Altersgruppen definiert und festgelegt. Die meisten Teilnehmern waren über 20 Jahre, da die Evaluation von Kindern alleine schwer ausgefüllt werden kann. Die jüngeren Tester haben mit Hilfe einer Aufsichtsperson NoRPG getestet und das Ergebnis in der Evaluation festgehalten. Der Großteil der Tester kommt aus Deutschland, allerdings ergab sich die Möglichkeit, einige Tester aus der Türkei zu gewinnen. Diese Erkenntnisse haben Einfluss und Bedeutung in der Bewertung von NoRPG.

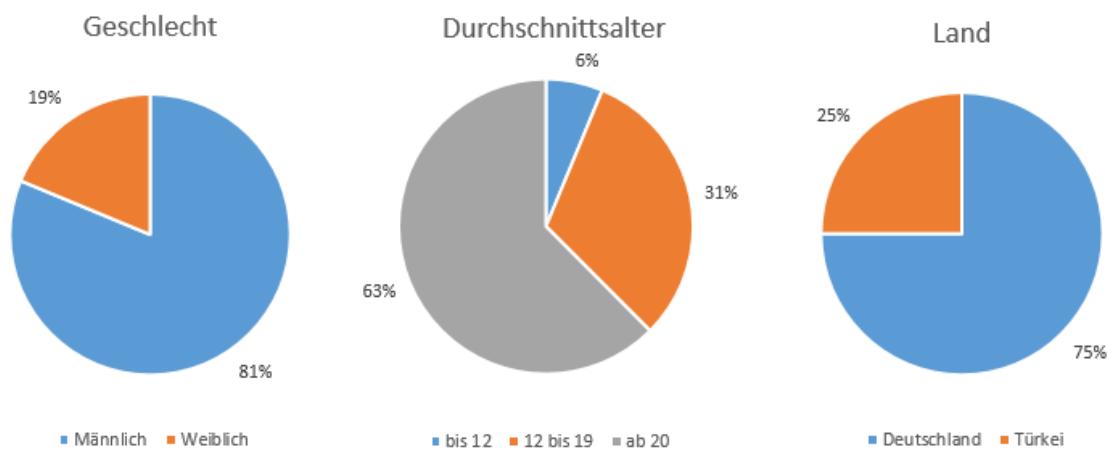


Abbildung 24: Evaluation Auswertung - Der durchschnittliche Teste

Neben dem Alter, Geschlecht und Herkunftsland konnten die Tester zudem auch die Marke des Smartphone-Herstellers und die installierte Android Version angeben. Dies hat den Zweck, ob die Einschränkungen im SRS im Vorhinein korrekt und sinnvoll waren. Das Ergebnis ist in der Abbildung 25 zu sehen. Mit einem Anteil von 50% ist die am häufigste verbreitete Smartphone Marke die des Herstellers Samsung. Auf den Smartphones war im Durchschnitt am meisten Android 7 installiert. Allerdings gibt es ein Unterschied zwischen den installierten Android Versionen. Die Testen in Deutschland hatten überwiegend die Version 7.x installiert, wohingegen in der Türkei diese Version auf keinem Testgerät installiert war.

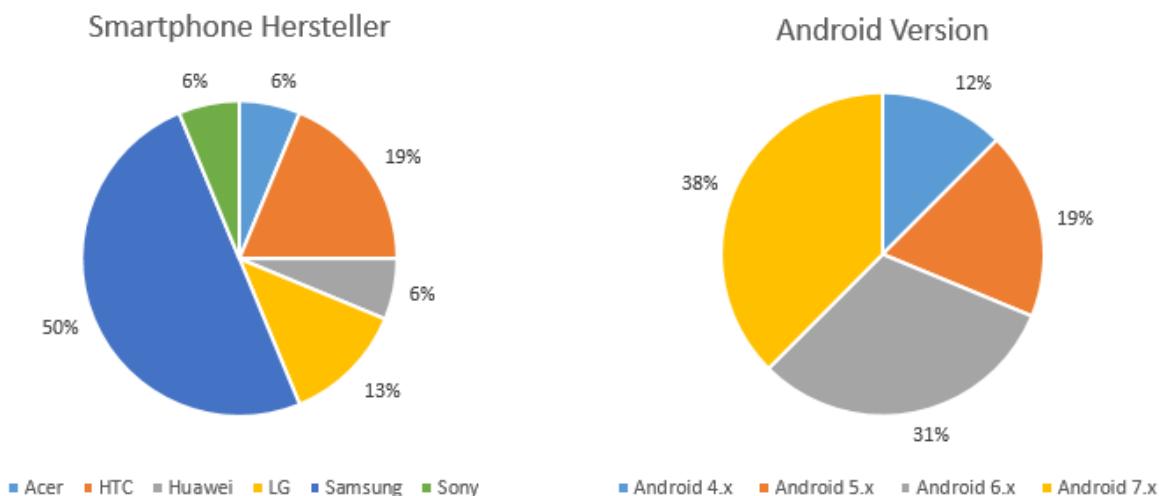


Abbildung 25: Evaluation Auswertung - Die durchschnittliche Hard- und Software

Im folgenden Abschnitt dieses Kapitels wird nun das Ergebnis der Evaluation behandelt und ausgewertet. Der einzelnen Szenarien konnten von den Testern von "–" bis "++" bewertet werden, während "++" die beste Bewertung darstellt. Damit das Ergebnis die Bewertung von allen Testern widerspiegelt, wurde "–" mit -2 und "++" mit +2 verrechnet. Dies kann in Tabelle 1 betrachtet werden. Diese entspricht dem des Bewertungsbogens, allerdings ohne nähere Beschreibung der einzelnen Szenarien. Die Punktzahl in der letzten Spalte spiegelt den genauen Durchschnitt von allen Testern wider.

Zunächst kann behauptet werden, dass die Auswertung positiv ausgefallen ist. Wenn alle Punkte von den einzelnen Szenarien zusammen betrachtet werden, ergibt es das Ergebnis von 1,1, welches in der Bewertungsmatrix unter "+" fallen würde.

Sehr positiv aufgenommen haben die Tester das Aussehen der UI Elemente. Durch die Darstellung der Funktionen mit bekannten, allerdings auch modernen, Symbolen konnten die auszuführende Aktion mit dem Element assoziiert werden. Ebenfalls damit verbunden wird das konsistente Aussehen sehr positiv beschrieben. Obwohl so

Szene	Szenario	Bewertung					Punkte
		-	-	0	+	++	
Startscreen (Logged Out User)	Log-In			x			1,4
	Change Settings			x			1,32
	Register			x			0,7
Registerszene	Create new Account			x			1,1
	Cancel Registration			x			1
Startscreen (Logged In User)	Log-In			x			1,6
	Log-Out			x			1
	Change Settings			x			1,3
Head Up Display	Overview			x			1,6
	Move Character			x			1,4
	Start Interaction			x			1,3
	Open Map				x		1,7
	Open Menu			x			1,3
	View Games				x		0,6
	View Progress	x					0,4
	View Achievements			x			0,9
	Change Settings			x			0,8
	Log-Out			x			1

Tabelle 1: Ergebnis der Evaluation - Die Bewertung

viele Funktionen im Spiel möglich sind, ist die Komplexität des Spiels sehr gering. Dies sticht in der Bewertung insbesondere mit dem Szenario "Open Map" hervor. Durch einen einfachen Klick auf die Karte vergrößert sich die Karte und der Spieler kann die ganze Spielwelt betrachten.

Allerdings gibt es je nach Benutzergruppe Auffälligkeiten und Unterschiede in den Bewertungen. So haben eher jüngere Teilnehmer meistens nur auf das Aussehen der UI Elemente geachtet und benötigten bei der Registrierung die Unterstützung eines Erwachsenen. Die schlechteste Bewertung erhielt das Szenario "View Progress". Diese beschreibt, dass der Spieler seinen Fortschritt und seine gelernten Standards betrachten will. Jedoch war diese Information für die jüngeren irrelevant und eher verwirrend.

Im Gegensatz dazu haben die älteren Tester zudem neben dem Aussehen auch beispielsweise die Eignung und eher kritischer bewertet. Es wurde häufig angemerkt, dass wichtige Informationen farblich hervorgehoben werden sollten und das zum Beispiel vor der Abmeldung der Spieler darauf hingewiesen werden.

Große Unterschiede zwischen den Bewertungen von männlichen und weiblichen Testern gab es keine. Jedoch konnten Unterschiede zwischen den Ländern festgestellt werden. So wurde das User Interface eher besser von deutschen Testern verstanden.

Viele dieser Anmerkungen, wie beispielsweise ein Dialog bei der Abmeldung oder andere eher kleinere Vorschläge wurden nach der Evaluation implementiert.

## 5.4 C# Skripte

Nachfolgend wird auf einzelne C# Skripte eingegangen, welche essentiell für die korrekte Ausführung der App benötigt werden. Neben den Skripten wird auch auf weitere Komponenten für die Skripte eingegangen.

### 5.4.1 Player

Der Player ist das Objekt, welches vom Spieler bewegt wird und ist für die Interaktion mit der kompletten Umgebung zuständig. Das Objekt setzt sich aus mehreren einzelnen Unterkomponenten zusammen, darunter fällt zum einen die Textur des Charakters, ein Teil der Minimap und ein Objekt für die Kamera.

An dem Elternobjekt Player sind Skripte und von Unity zur Verfügung gestellt Objekte angebracht. Zu diesen Objekten gehören der Rigidbody, CharacterController, Animator und ThirdPersonController. Der Rigidbody sorgt dafür, dass der Player Gravitation erfährt und nicht einfach durch die Luft schweben kann. Der CharacterController implementiert verschiedene Eigenschaften, wie beispielsweise die Breite, Höhe und den Schrittversatz. Der Schrittversatz bestimmt über welche Höhen und Objekte, wie beispielsweise Treppenstufen, der Player laufen kann. Der Animator ist zusammen mit dem ThirdPersonController dafür verantwortlich, dass sich der Player in der Szene bewegt und die Bewegungsanimationen korrekt ausgeführt werden.

Daraus resultieren folgende Möglichkeiten der Animation:

Erst durch diese Vielzahl an Animationen wird gewährleistet, dass der Player zu jeder Zeit die richtige Animation ausführt und die Bewegungen nicht unnatürlich aussehen. Damit dies passiert, werden die Werte speed und direction vom in Kapitel 5.3.1 besprochenen Joystick im Skript CharacterControll.cs an den Charakter übergeben. Diese ist für die Steuerung des Players zuständig. Hier wird die Toucheingabe über den Joystick in Weltkoordinaten umgewandelt und der Player beginnt sich zu bewegen. Das Skript ist in Listing 5 aufgeführt.

speed	direction	Ergebnis
0	0	Animation Idle
>0	0	Animation Walk
>0	0.3	Animation Walk Right Short
>0	0.5	Animation Walk Right Medium
>0	-0.3	Animation Walk Left Short
>0	-0.5	Animation Walk Left Medium
>0.5	0	Animation Run
>0.5	0.3	Animation Run Right Medium
>0.5	0.5	Animation Run Right Wide
>0.5	-0.3	Animation Run Left Medium
>0.5	-0.5	Animation Run Left Wide

Tabelle 2: Mögliche Animationen je nach Wert speed und direction

Dazu wird die Methode Update genutzt. In dieser wird, sofern ein Animator an dem Player vorhanden ist, die horizontale und vertikale Bewegung des Joysticks in die Werte für direction und speed umgewandelt. Das ganze passiert dabei in der Methode StickToWorldspace in Zeile 32.

In der Methode wird zu Beginn die rootDirection gesetzt, welche sich dabei aus der Z-Achsen Koordinate zusammensetzt. Anschließend wird die stickDirection durch den horizontalen und vertikalen Wert des Joysticks gesetzt. Abschließend wird der Wert von speed durch die Quadrierung der beiden Werte berechnet. Dieser liegt dabei zwischen Null und Eins. Danach wird das ganze in Bezug zu der Positionsrichtung der Kamera gesetzt, um die Bewegungsrichtung zu erhalten und anschließend das Kreuzprodukt aus diesen beiden Werten zu berechnen. Mit Hilfe des Wertes kann bestimmt werden, ob sich der Player nach Rechts oder nach Links bewegen soll.

Nachdem diese aufgerufen wurde und abgeschlossen ist, werden die Werte von direction und speed an den ThirdPersonController übergeben und die korrekte Animation wird startet.

```

1 public class CharacterControl : MonoBehaviour {
2     [...]
3
4     void Update() {
5         if (animator) {
6             stateInfo = animator.GetCurrentAnimatorStateInfo(0);
7             horizontal = CnInputManager.GetAxis("Horizontal");
8             vertical = CnInputManager.GetAxis("Vertical");
9
10            StickToWorldspace(this.transform, gamecam.transform, ref direction, ref speed);
11            animator.SetFloat("speed", speed);
12            animator.SetFloat("direction", direction, directionDumpTime, Time.deltaTime);
13        }
14    }
15
16    [...]

```

```

17
18     public void StickToWorldspace(Transform root, Transform camera,
19         ref float directionOut, ref float speedOut) {
20
21     Vector3 rootDirection = root.forward;
22     Vector3 stickDirection = new Vector3(horizontal, 0, vertical);
23     speedOut = stickDirection.sqrMagnitude;
24     Vector3 CameraDirection = camera.forward;
25     CameraDirection.y = 0.0f;
26
27     Quaternion referentialShift = Quaternion.FromToRotation(Vector3.forward,
28         Vector3.Normalize(CameraDirection));
29
30     Vector3 moveDirection = referentialShift * stickDirection;
31     Vector3 axisSign = Vector3.Cross(moveDirection, rootDirection);
32     float angleRootToMove = Vector3.Angle(rootDirection, moveDirection)
33         * axisSign.y >= 0 ? -1f : 1f;
34
35     angleRootToMove /= 180f;
36     directionOut = angleRootToMove * directionSpeed;
37 }
38 }
```

Listing 5: CharacterController.cs

## 5.4.2 Kamera

Bei der Kamera handelt es sich um ein Objekt, welches dem Charakter folgt. Dafür ist an dem Player ein Objekt mit dem Namen „follow“ angehängt, auf das die Kamera zeigt.

Die Kamera referenziert das Skript ThirdPersonCamera.cs. In diesem wird das Verhalten der Kamera gesteuert. Dazu werden die Methoden aus dem Listing 6 verwendet. In LateUpdate in Zeile 4 wird dabei die Position der Kamera in Bezug zum Spieler gesetzt. Dabei befindet sich die Kamera immer in einem Kreis um dem Spieler herum, wodurch die Sicht nicht eingeschränkt wird.

```

1 public class ThirdPersonCamera : MonoBehaviour {
2     [...]
3
4     void LateUpdate() {
5         Vector3 characterOffset = follow.position + new Vector3(0f, distanceUp, 0f);
6         lookDir = characterOffset - this.transform.position;
7         lookDir.y = 0;
8
9         lookDir.Normalize();
10        targetPosition = characterOffset + follow.up * distanceUp - lookDir * distanceAway;
11        CompensateForWalls(characterOffset, ref targetPosition);
12        smoothPosition(this.transform.position, targetPosition);
13        transform.LookAt(follow);
14    }
15 }
```

```

16     [...]
17
18     private void CompensateForWalls(Vector3 fromObject, ref Vector3 toTarget) {
19         RaycastHit wallHit = new RaycastHit();
20         if(Physics.Linecast(fromObject, toTarget, out wallHit)) {
21             toTarget = new Vector3(wallHit.point.x, toTarget.y, wallHit.point.z);
22         }
23     }
24 }
```

Listing 6: Methode LateUpdate aus ThirdPersonCamera.cs

Darüber hinaus wird in LateUpdate unterbunden, dass die Kamera in Wänden verschwindet und durch Objekte geschaut werden kann. Dazu wird die Methode CompensateForWalls in Zeile 21 ausgeführt. In dieser wird getestet, ob zwischen der Kamera und dem Spieler ein Objekt vorhanden ist. Wenn das der Fall ist, wird die Kamera vor dieses Objekt gesetzt.

Beim aller ersten Start des Spiels gibt es eine geskriptete Szene, in der der Spieler kurz in die Geschichte von NoRPG geführt wird. Dabei wird die Farbe der Startwelt entfernt (siehe Kapitel 2.4). Dazu wird ein Shader genutzt, welcher die Farbintensität von jedem Pixel verändert. Das Attribut Intensität des Shaders bestimmt über die Stärke der Graustufen. Dieser wird zu Beginn auf eins gesetzt und für jeden gefundenen Diamanten in den anderen Welten minimiert.

### 5.4.3 Portale

Die Portale verbinden die einzelnen Szenen, in denen die verschiedenen Welten abgebildet sind. Diese sind separiert jeweils in einer Szene implementiert, da eine riesige Szene sehr viel Performance benötigen würde.

In der Startwelt sind fünf verschiedene Portale platziert, mit denen der Spieler in die Welten kommt. Dadurch wird uns ermöglicht die einzelnen Klassen voneinander zu separieren. Jede dieser Welten hat ein Portal, welches den Spieler zurück zur Startwelt bringt. Jedes 3D Objekt in der Szene, dargestellt durch ein Steinbogen mit Partikeln, implementiert das Skript 7. Dieses Listing enthält nur die zwei wichtigsten Methoden für die Portale.

```

1 public class PortalToTargetScene : MonoBehaviour{
2     [...]
3
4     void OnTriggerEnter(){
5         string currentScene = SceneManager.GetActiveScene().name;
6         PortalControl.control.cameFrom = currentScene;
7         PortalControl.control.currentScene = targetSceneName;
8 }
```

```

9     loadingScreen.gameObject.SetActive(true);
10    StartCoroutine(LoadLevelWithRealProgress());
11 }
12 }
13 [...]
14
15 IEnumerator LoadLevelWithRealProgress() {
16     yield return new WaitForSeconds(1);
17
18     ao = SceneManager.LoadSceneAsync("Scenes/" + targetSceneName, LoadSceneMode.Single);
19     ao.allowSceneActivation = false;
20
21     while (!ao.isDone) {
22         progBar.value = ao.progress;
23
24         if (ao.progress == 0.9f) {
25             progBar.value = 1f;
26             ao.allowSceneActivation = true;
27         }
28
29         yield return null;
30     }
31 }
32 }
33 }

```

Listing 7: PortalToTargetScene

Die erste Methode `OnTriggerEnter()` wird ausgeführt, sobald der Spieler mit seinem Charakter durch das Portal geht. Dabei kollidiert der Hüllkörper des Charakters mit dem Hüllkörper des Portals und lösen die Funktion aus. Der Hüllkörper in Unity ist als Collider bezeichnet, und umschließt komplexe (nicht Polygone) Objekte, um die Kollisionserkennung zu vereinfachen. Wenn die beiden Objekte jetzt kollidieren wird zunächst der Ladebildschirm für die folgende Szene angezeigt, dies passiert in Zeile 12 von Listing 7.

Anschließend wird die asynchrone Ladefunktion `LoadLevelWithRealProgress()` ausgeführt. Das Laden einer Szene wird vom SceneManager von Unity durchgeführt und kann synchron oder asynchron ausgeführt werden. Bei der asynchronen Variante wird die Szene im Hintergrund geladen und das Spiel kann in dieser Zeit weitergespielt werden. In der anderen Variante kann der Spieler das Spiel nicht mehr steuern bis der Ladeprozess fertig ist. Neben der Synchronisationsvariante wird in Zeile 23 noch neben der zu ladenden Szene der Lademodus angegeben. Der Lademodus bestimmt, was mit der aktuellen Szene passieren soll. Mit dem Modus Single wird die alte Szene mit der neuen ersetzt, wodurch das Laden länger dauert aber weniger Performance benötigt wird.

Durch die asynchrone Ladung kann ein Ladebildschirm implementiert werden, welcher den Fortschritt des Ladeprozesses anzeigt. Dieser Ladeprozess wird in Zeile 28

berechnet. Erst wenn der asynchrone Prozess fertig ist wird die Aktivierung der Szene erlaubt. Damit die Position des Spielers in der geladenen Szene fest ist, gibt es in jeder Welt Spawnpunkte. Diese legen fest wo in der Welt der Charakter des Spielers positioniert wird.

Neben diesen eigentlichen Ladeprozess wird zudem noch der neue Standort des Spielers gespeichert, damit der Spieler nach Beendigung des Spiels automatisch wieder in der korrekten Welt startet.

Durch die häufigen Szenenwechsel dürfen keine Daten und Objekte verloren gehen. So muss der Charakter des Spielers auch in der anderen Szene noch mit seiner Auswahl stimmen oder die Einstellungen dürfen sich nicht ändern. Es gibt Objekte, wie beispielsweise das GameControl, welches alle Informationen über den Spieler hält und am Anfang erstellt wird, beim Szenenwechsel nicht zerstört sondern in diese mit übernommen wird.

#### 5.4.4 Interaktionsmöglichkeiten

Im Spiel kann der Spieler mit verschiedenen Objekten interagieren. Zu den Hauptkategorien gehören Händler, die dem Spieler Lernspiele der Standards anbieten, und Truhen, aus denen die Edelsteine gefunden werden. Darüber hinaus kann der Spieler mit verschiedenen NPC und weiteren Objekten interagieren.

Gesteuert werden alle möglichen Interaktionen mit den Objekten in der Klasse NPC-Communication und entspricht damit einem Controller. Das folgende Listing 8 bildet nur einen kleinen Abschnitt des vollständigen Skriptes ab.

```
1 public class NPCCommunication : MonoBehaviour {
2     [...]
3
4     void Start() {
5         string fileName = "npcList";
6         TextAsset textAsset = Resources.Load<TextAsset>(fileName);
7         json = textAsset.text;
8     }
9
10    public void StartCommunication(){
11        if (Vector3.Distance(player.transform.position, trader.GetClosestObject
12            ("InteractionObject", player).transform.position) < distance){
13
14            string interactableObjectName = trader.closest.name;
15            string npcType = dialogue.GetNpcType(interactableObjectName, json);
16
17            npcName.text = dialogue.GetNpcName(interactableObjectName, json);
18            npcText.text = dialogue.GetNpcText(interactableObjectName, json);
19
20            if (npcType == "Trader"){
21                acceptButton.onClick.AddListener(delegate () {OpenGameList(interactableObjectName); });
22            }
23        }
24    }
25}
```

```

22     cancelButton.onClick.AddListener(delegate() {CancelCommunication();});
23 }
24 [...]
25 else if (npcType == "Chest"){
26     acceptButton.onClick.AddListener(delegate() {OpenChest(interactableObjectName);});
27     cancelButton.onClick.AddListener(delegate() {OpenChest(interactableObjectName);});
28 }
29 else{
30     acceptButton.onClick.AddListener(delegate() {CancelCommunication();});
31     cancelButton.onClick.AddListener(delegate() {CancelCommunication();});
32 }
33 }
34 }
35 [...]
36 }
37 }
```

Listing 8: NPCCommunication.cs

In der Funktion Start wird die JSON Datei, die für die Inhalte der Interaktionen mit den Objekten zuständig ist, geladen. Durch die Implementierung als JSON wird der modulare Aufbau von NoRPG gewährleistet. Dadurch ist es immer möglich diese Datei anzupassen und beispielsweise neue Sprachen zu integrieren oder Inhalte auszuwechseln.

Das Integrieren dieser JSON Datei ist insofern anders, dass nicht alle Informationen jederzeit benötigt werden. Daher wird das JSON nicht vollständig durchlaufen, sondern es werden konkrete Anfragen gesendet. Für diesen Zweck wurde ein eigener Parser implementiert. Nach diesem Prinzip werden alle Informationen einzeln, genau und nur dann wenn sie benötigt werden, in das Spiel geladen. Im folgenden Listing 9 ist der Aufbau skizziert.

```

1 {
2     "1_Math_OA" : {
3         "npcName" : "Alfred",
4         "npcText" : "Some Text...",
5         "gamelistTitle" : "Math, First Class, Operations and Algebraic Thinking",
6         "gamelistDescription" : "Some Description...",
7         "npcType" : "Trader"
8     },
9     [...]
10    "1_Chest" : {
11        "npcText" : "You found a chest!",
12        "npcType" : "Chest"
13    },
14    [...]
15 }
```

Listing 9: NPC JSON

Der NPC Typ in Zeile 7 9 ist besonders wichtig, da anhand dieser Variable entschieden wird, welche Aktionen die beiden Interaktionsbuttons ausführen sollen. So steht

beispielsweise der Typ Trader für die Spielehändler. Neben dem Typ sind hier auch die Namen, die Texte und alle anderen Informationen gespeichert. Die Händler in den verschiedenen Welten unterscheiden sich im wesentlichen nur mit dem Objektnamen. Dabei setzt sich der Name aus drei Teilen zusammen: Der Klasse, dem Fach und dem Themengebiet. So heißt der Händler für das Themengebiet Schreiben in Englisch der ersten Klasse „1\_English\_W“. Anhand diesem Schlüssel wird die JSON durchsucht und die korrekten Texte und Aktionen geladen.

Die nächste Methode in NPCCommunication 8 ist StartCommunication, die für die korrekte Interaktion zuständig ist. Hier wird zunächst überhaupt überprüft, ob ein interaktives Objekt vor dem Spieler steht. Da es in den Welten nicht nur einen Händler gibt und diese nicht aus weiter Entfernung angesprochen werden sollen, wurde ein Skript implementiert, um den Händler zu finden, welcher am dichtesten zum Player steht. In diesem Skript wird jedes Objekt, welches den Tag "InteractionObject" hat, in einem Array gespeichert und anschließend wird für jedes Objekt geprüft wie weit es entfernt ist. Das Objekt, welches die kleinste Distanz zum Spieler hat, wird am Schluss zurück gegeben. Ist der Charakter des Spielers nah genug am Händler, wird der Dialog geöffnet und die korrekten Daten werden aus der JSON-Datei geladen.

Des Weiteren wird nachdem das korrekte Objekt ermittelt wurde, in Zeile 17 von NPCCommunication 8 zunächst nur der Typ des Objektes aus dem JSON mit Hilfe des Objektnamens ausgelesen und in die Variable geschrieben. Anhand des Typs führen die beiden Buttons für die Interaktion unterschiedliche Funktionen aus. So handelt es sich in Zeile 20 um den NPC Typ Trader und der A-Button öffnet die Liste mit den Lernspielen und der B-Button beendet die Kommunikation. Handelt es sich um eine Truhe öffnen beide Buttons die Truhe und führen die Funktion OpenChest aus.

## 5.4.5 Pfadfindungssystem Schiff

Die verschiedenen Inseln von Galapagos sind im Spiel nur mit einem Schiff erreichbar. Dabei kann der Spieler aus fünf verschiedenen Inseln auswählen und je nach Auswahl fährt das Schiff dem Spieler zur korrekten Insel. Die Pfade des Schiffes sind dabei statisch und festgelegt. Es gibt 20 Pfade, vier von jeder Insel. Diese Objekte referenzieren auf das Skript EditorPathScript (siehe Listing 10).

```

1 public class EditorPathScript : MonoBehaviour {
2
3     public Color rayColor = Color.white;
4     public List<Transform> path_objs = new List<Transform>();
5     Transform[] transforms;
6
7     void OnDrawGizmos () {
8         Gizmos.color = rayColor;

```

```

9     transforms = GetComponentsInChildren<Transform>();
10    path_objs.Clear();
11
12    foreach (Transform path_obj in transforms) {
13        if (path_obj != this.transform) {
14            path_objs.Add(path_obj);
15        }
16    }
17
18    for(int i = 0; i < path_objs.Count; i++) {
19        Vector3 position = path_objs[i].position;
20        if (i > 0) {
21            Vector3 previos = path_objs[i - 1].position;
22            Gizmos.DrawLine(previos, position);
23            Gizmos.DrawWireSphere(position, 0.3f);
24        }
25    }
26}
27

```

Listing 10: Skript EditorPathScript.cs

Dieses Skript sorgt dafür, dass die Pfade grafisch im Unity Editor angezeigt werden. Dieses Skript zeichnet dabei von einem Startpunkt zum nächsten eine Linie, solange, bis der letzte Punkt erreicht wird. Dadurch wird es möglich die Pfade grafisch zu bearbeiten und erleichtert so die Arbeit. Im folgenden Abschnitt dieses Kapitels wird beschrieben, wie das Schiff dem Pfad folgt, sobald der Player den Pfad ausgewählt hat. Dabei wird auf den Pfad von Insel eins zu zwei eingegangen. Für alle anderen Pfade ist das Vorgehen identisch.

Über eine UI wählt der Spieler mit einem Klick auf einen Button die Zielinsel aus. Es wird die Funktion im Listing 11 ausgeführt. Diese selektiert den gewünschten Pfad aus den verfügbaren Pfaden und setzt den Boolean playerOnShip in Zeile 7 auf wahr.

```

1 public void selectFirstClass () {
2     if (lastButtonClicked != "1") {
3         hud.SetActive(false);
4
5         path = GameObject.Find("Path" + lastButtonClicked + "_1").GetComponent<EditorPathScript>();
6         lastButtonClicked = "1";
7         playerOnShip = true;
8     }
9 }

```

Listing 11: Methode selectFirstClass

Da dieser Boolean nun auf wahr gesetzt ist, wird der Code der Methode playerGetOnShip ausgeführt. Dieser Code ist in Listing 12 zu sehen. Durch diese Methode bewegt sich der Player mit dem Schiff in der selben Geschwindigkeit, damit dieser nicht vom Schiff herunterfällt Darüber hinaus blockiert diese Funktion die Eingabemöglichkeiten des Players während der Fahrt.

```

1 public void playerGetOnShip () {
2     if (!playerOnShip) {
3         cc.enabled = false;
4
5         player.GetComponent<Animator>().SetFloat("speed", 0.0f);
6         player.GetComponent<Animator>().SetFloat("direction", 0.0f);
7         player.transform.position = ship.transform.position;
8
9         player_mesh.SetActive(false);
10        player_text.SetActive(false);
11
12        follow.transform.parent = ship.transform;
13        follow.transform.position = ship.transform.position + new Vector3(2.25f, 18.07f, -7.07f);
14
15        minidot.SetActive(false);
16
17        hud.SetActive(true);
18    }
19}

```

Listing 12: Methode playerGetOnShip

Des Weiteren wird die Funktion moveShip ausgeführt. Diese ist für die eigentliche Bewegung des Schiffes verantwortlich. Sobald der Boolean playerOnShip auf wahr gesetzt ist, wird zuerst die Distanz zwischen dem ersten Punkt des Pfades und des Schiffes gespeichert und die Position des Schiffes an diese Stelle verschoben. Anschließend wird der Winkel des Schiffes an die Fahrtrichtung angepasst und gedreht. Sobald die Distanz zwischen der aktuellen Position des Schiffes und dem nächsten Punkt des Pfades kleiner ist wie eine minimale Distanz, wird die currentWayPointId um eins erhöht. Solange dieser Wert nicht größer ist wie die Anzahl der Punkte des Pfades, werden diese Schritte wiederholt und das Schiff bewegt sich entlang des Pfades.

```

1 public void moveShip () {
2     if (playerOnShip) {
3         float distance = Vector3.Distance(path.path_objs[currentWayPointId].position,
4                                             transform.position);
5         transform.position = Vector3.MoveTowards(transform.position,
6                                                 path.path_objs[currentWayPointId].position, Time.deltaTime * speed);
7         player.transform.position = transform.position;
8
9         var rotation = Quaternion.LookRotation(
10            path.path_objs[currentWayPointId].position - transform.position);
11         transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
12            Time.deltaTime * rotationSpeed);
13         player.transform.rotation = ship.transform.rotation;
14
15         if (distance <= reachDistance) {
16             currentWayPointId++;
17         }
18
19         if (currentWayPointId >= path.path_objs.Count) {
20             this.transform.position = path.path_objs[path.path_objs.Count-1].position;
21             playerShipped = true;
22             cc.enabled = true;
23         }
24     }
25 }

```

```
23     playerOnShip = false;  
24     player_mesh.SetActive(true);  
25     player_text.SetActive(true);  
26     follow.transform.parent = player.transform;  
27     follow.transform.position = player.transform.position;  
28  
29     minidot.SetActive(true);  
30     player.transform.position = new Vector3(path.path_objs[path.path_objs.Count - 1]  
31         .position.x, 6.3f, path.path_objs[path.path_objs.Count - 1].position.z);  
32  
33     currentWayPointId = 0;  
34 }  
35 }  
36 }
```

Listing 13: Methode moveShip

## 6 Fazit und Ausblick

Hone ist eine Spielplattform für Kinder, auf der sie die Möglichkeit haben, Lernspiele herunterzuladen. Die Motivation für diesen Schritt ist bei Kindern jedoch gering. Darüber hinaus wollen Kinder keine Lernspiele spielen, in denen sie merken, dass sie lernen. Deshalb sollte Hone selbst zu einem Spiel werden, indem sich Kinder anmelden und anschließend spielerisch animiert werden zu lernen. Das ganze sollte als Resultat dieser Arbeit entstehen, dabei waren drei Hauptbestandteile bei der Umsetzung besonders wichtig.

Kinder sollten eine Möglichkeit bekommen, mit der sie jederzeit, überall und einfach auf die Lerninhalte zugreifen und spielerisch neues Wissen erwerben können. Diese Lerninhalte sollten dabei über externe Lernspiele eingebunden werden und in korrekter Reihenfolge freigeschaltet werden. Darüber hinaus sollte diese Arbeit als Dokumentation für dieses Projekt dienen, um Vorgehensweisen, Bedienung und Probleme zu dokumentieren.

Mithilfe der App NoRPG haben Kinder nun eine Möglichkeit überall und zu jeder Zeit spielerisch auf neue Lerninhalte zuzugreifen. Dies ist dadurch realisiert worden, dass die App immer die neusten Standards und externen Spiel zu beginn von einem Server herunterlädt, sofern Internet vorhanden ist, und somit stets aktuell ist. Darüber hinaus können die Kinder auch an verschiedenen Geräten spielen und sind somit unabhängig vom Ort.

Die externen Lernspiele werden dabei zentral auf einem Server verwaltet und können in Zukunft über ein Webinterface gepflegt und erweitert werden. Dabei erfüllt jedes Spiel mindestens einen der CCSSs. Durch eine Prüfung, ob ein Spiel bereits gespielt wurde oder nicht, wird darüber hinaus gewährleistet, dass das Kind auch nur die Spiele angezeigt bekommt, die für den Wissenstand des Kindes geeignet sind. Eine Idee für die Zukunft ist, dass Spiele die sich das Kind herunterlädt, ein Interface implementieren, welches eine Rückmeldung an die App gibt, ob das Kind das vermittelte Wissen erlangt hat. Aktuell wird davon ausgegangen, dass das Kind ein heruntergeladenes Spiel auch spielt und das benötigte Wissen lernt, da es keine Möglichkeit einer Rückmeldung von den externen Spielen gibt.

Gegen Ende der Entwicklung wurde darüber hinaus eine Evaluation des User Interfaces durchgeführt. Dabei wurde die Benutzeroberfläche mehrheitlich sehr positiv bewertet. Das konnte durch intuitive Symbole ermöglicht, die der Benutzer auch schon aus anderen Apps kennt. Durch das intuitive UI ist die App für Kinder besonders gut geeignet. Jedoch ist die Registrierung komplex und die Kinder benötigen in diesem Schritt einen Erwachsenen zur Hilfe. Darüber hinaus wurde festgestellt, dass die Fortschrittsanzeige für Kinder eher verwirrend ist und sie diese Information nicht benötigen. Als Konsequenz könnte dieser Teil in Zukunft durch ein Webinterface ersetzt werden, auf dem die Eltern den Fortschritt ihres Kindes einsehen könnten.

Ein weiterer Aspekt, der die Komplexität der App für Kinder erhöht, ist die Tatsache, dass viel Textverständnis vorausgesetzt wird. Das Kind sollte bereits flüssig lesen können. Deshalb ist diese Version der App eher für ältere Kinder geeignet. In einem nächsten Schritt kann die App durch eine Text-to-Speech Funktion erweitert werden, damit die Texte vorgelesen werden. Dadurch sinkt die Komplexität und die App ist auch für die gewünschte Zielgruppe einsetzbar.

In der Zukunft sind neben diesen Verbesserungen auch noch weitere Features denkbar. Zum einen sollen in Zukunft auch weitere Klassen, sowie Fächer angeboten werden, damit die Kinder nicht nur Englisch und Mathe lernen können. Dadurch wird die App noch attraktiver und mehr Wissen kann an die Kinder vermittelt werden.

Um das Wissen der Kinder innerhalb der App zu testen, wäre es möglich, eigene Spiele in der App anzubieten. Diese Spiele könnten dann wie eine Art Test angesehen werden, bei der gelerntes Wissen überprüft wird.

Außerdem könnten in Zukunft ein Multiplayer oder ein Ranking eingeführt werden, um eine weitere Motivation zu schaffen.

Zusammenfassend kann gesagt werden, dass die App ein guter Schritt in die richtige Richtung ist, um Lernspiele sinnvoller zu nutzen und Kinder effektiv zu animieren etwas lernen zu wollen, ohne das sie es merken.

# Literatur

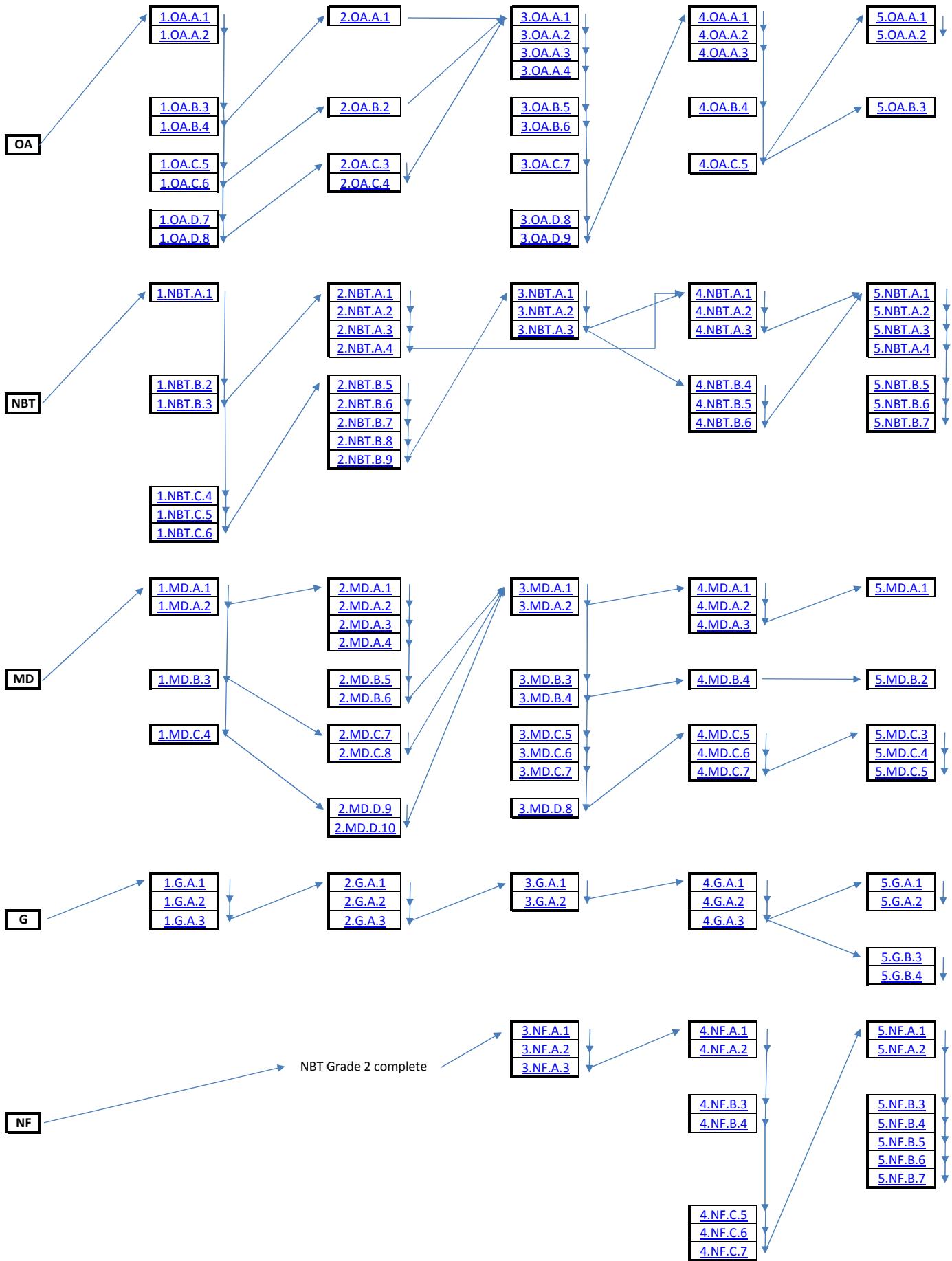
- [1] Ralf Adams. *SQL Eine Einführung mit vertiefenden Exkursen*. 1.Auflage. Carl Hanser Verlag München, 2012. ISBN: 9783446432000.
- [2] Leonard L. Tripp et al. *IEEE Recommended Practice for Software Requirements Specifications*. Techn. Ber. ISBN: 0-7381-0448-5. The Institute of Electrical und Electronics Engineers, 1998.
- [3] Wendy B. *Apps for Kids: Basic Usability Guidelines*. English. 2013. URL: <https://software.intel.com/en-us/blogs/2013/01/23/apps-for-kids-basic-usability-guidelines> (besucht am 27.01.2017).
- [4] Alan Beaulieu. *Learning SQL*. 2. Auflage. O'Reilly und Associates, 2009. ISBN: 978-0596555580.
- [5] gesellschaftsspiele.de. *Die Geschichte der Brettspiele*. German. 2015. URL: <http://www.gesellschaftsspiele.de/geschichte-brettspiele/> (besucht am 15.01.2017).
- [6] The Global Goals. *Goal 4: Hochwertige Bildung | The Global Goals*. English. 2015. URL: <http://www.globalgoals.org/de/global-goals/quality-education/> (besucht am 05.01.2017).
- [7] klick-tipps.net. *Was macht eine gute Kinder-App aus?* German. 2015. URL: <http://www.wir-machen-kinderseiten.de/blog/was-macht-eine-gute-kinder-app-aus> (besucht am 27.01.2017).
- [8] Andrew Lee. *How to write Performance Requirements with Example*. English. 2015. URL: <http://www.1202performance.com/articles/how-to-write-performance-requirements-with-example/> (besucht am 26.01.2017).
- [9] Microsoft. *Visual C#*. English. 2015. URL: <https://msdn.microsoft.com/de-de/library/kx37x362.aspx> (besucht am 21.01.2017).
- [10] Microsoft. *Visual Studio-IDE*. English. 2015. URL: <https://msdn.microsoft.com/de-de/library/dn762121.aspx> (besucht am 21.01.2017).

- [11] Dan Morrill. *Inside the Android Application Framework*. English. 2008. URL: <https://sites.google.com/site/io/inside-the-android-application-framework> (besucht am 26.01.2017).
- [12] Eoin Woods Nick Rozanski. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. 2nd revised edition. Addison Wesley, 2011. ISBN: 978-0321718334.
- [13] Michael Huch Philippe Fischer. *Die S-Klasse: Samsung Galaxy S, S2, S3, S4, S5, S6, S7 und S7 Edge*. German. 2016. URL: <http://www.computerbild.de/artikel/cb-News-Handy-Samsung-Galaxy-S-S2-S3-S4-S5-S6-S7-11332036.html> (besucht am 27.01.2017).
- [14] Unity3D. *Build One Deploy Anywhere*. English. 2017. URL: <https://unity3d.com/unity/multiplatform> (besucht am 21.01.2017).
- [15] Unity3D. *Prefabs*. English. 2017. URL: <https://docs.unity3d.com/Manual/Prefabs.html> (besucht am 21.01.2017).
- [16] Unity3D. *Skripting*. English. 2017. URL: <https://docs.unity3d.com/Manual/ScriptingConcepts.html> (besucht am 21.01.2017).
- [17] Unity3D. *The Game View*. English. 2017. URL: <http://docs.unity3d.com/Manual/GameView.html> (besucht am 21.01.2017).
- [18] Unity3D. *The HierarchyWindow*. English. 2017. URL: <http://docs.unity3d.com/Manual/Hierarchy.html> (besucht am 21.01.2017).
- [19] Unity3D. *The InspectorWindow*. English. 2017. URL: <http://docs.unity3d.com/Manual/UsingTheInspector.html> (besucht am 21.01.2017).
- [20] Unity3D. *The ProjectWindow*. English. 2017. URL: <http://docs.unity3d.com/Manual/ProjectView.html> (besucht am 21.01.2017).
- [21] Unity3D. *The Scene View Window*. English. 2017. URL: <http://docs.unity3d.com/Manual/UsingTheSceneView.html> (besucht am 21.01.2017).
- [22] Becky White. *Designing For Kids Is Not Child's Play*. English. 2016. URL: <http://www.smashingmagazine.com/2016/01/designing-apps-for-kids-is-not-childs-play/> (besucht am 27.01.2017).

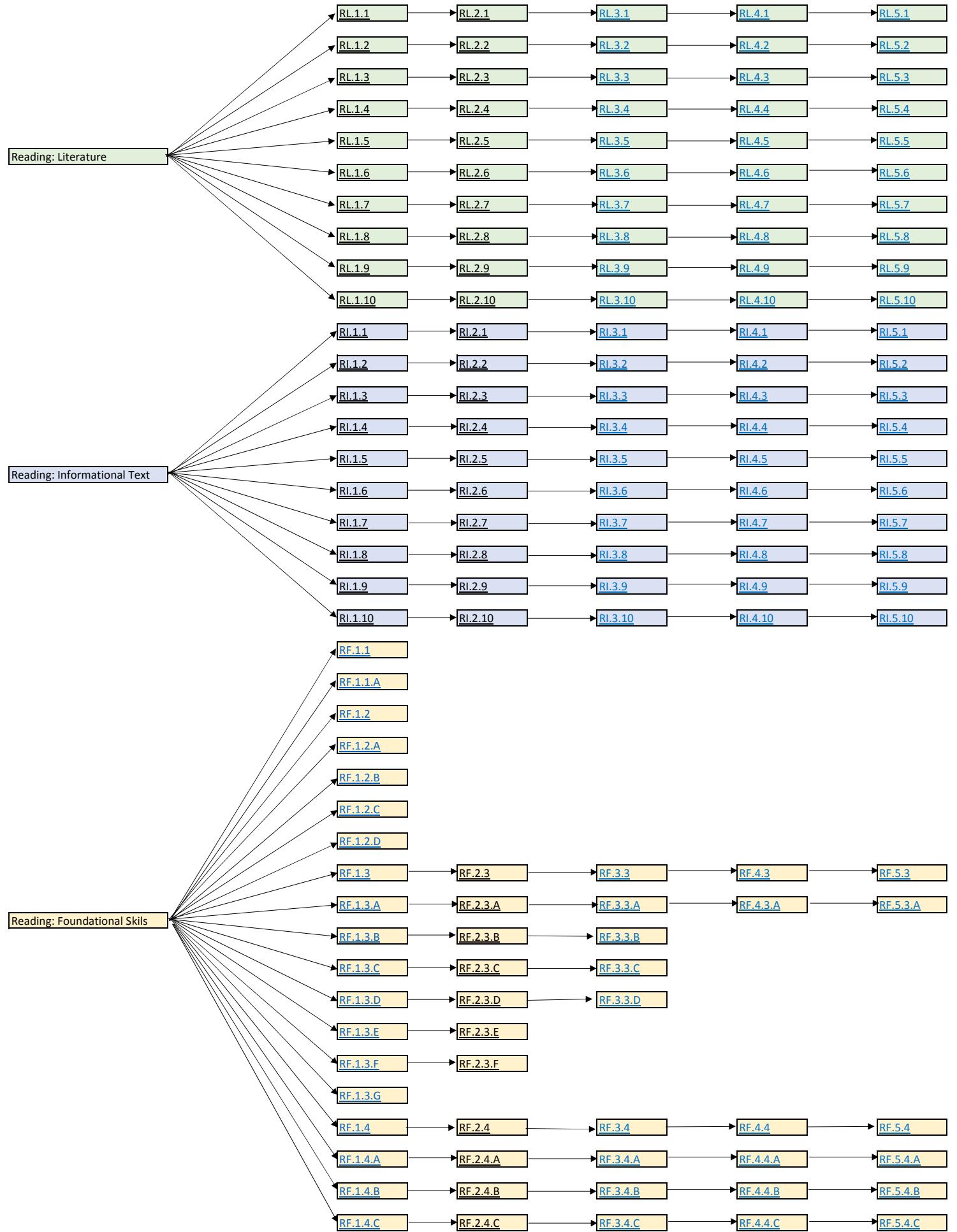
- [23] Max Wiesmüller. *Galaxy S4: Top-Model im Plastikkleid*. German. 2015. URL: [http://www\(chip.de/artikel/Samsung-Galaxy\\_S4-Handy-Test\\_61712154.html](http://www(chip.de/artikel/Samsung-Galaxy_S4-Handy-Test_61712154.html) (besucht am 27.01.2017).

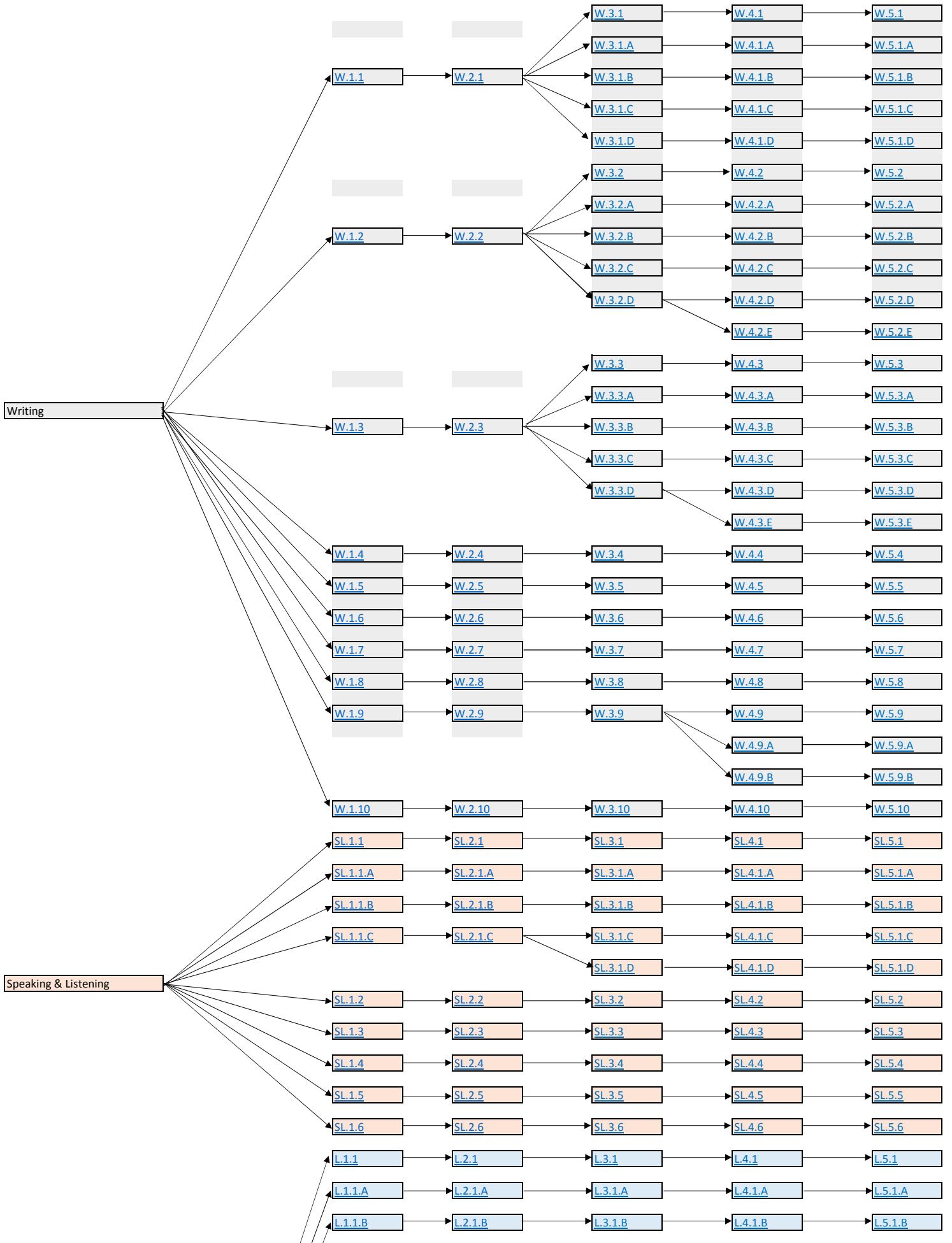
# **Anhang**

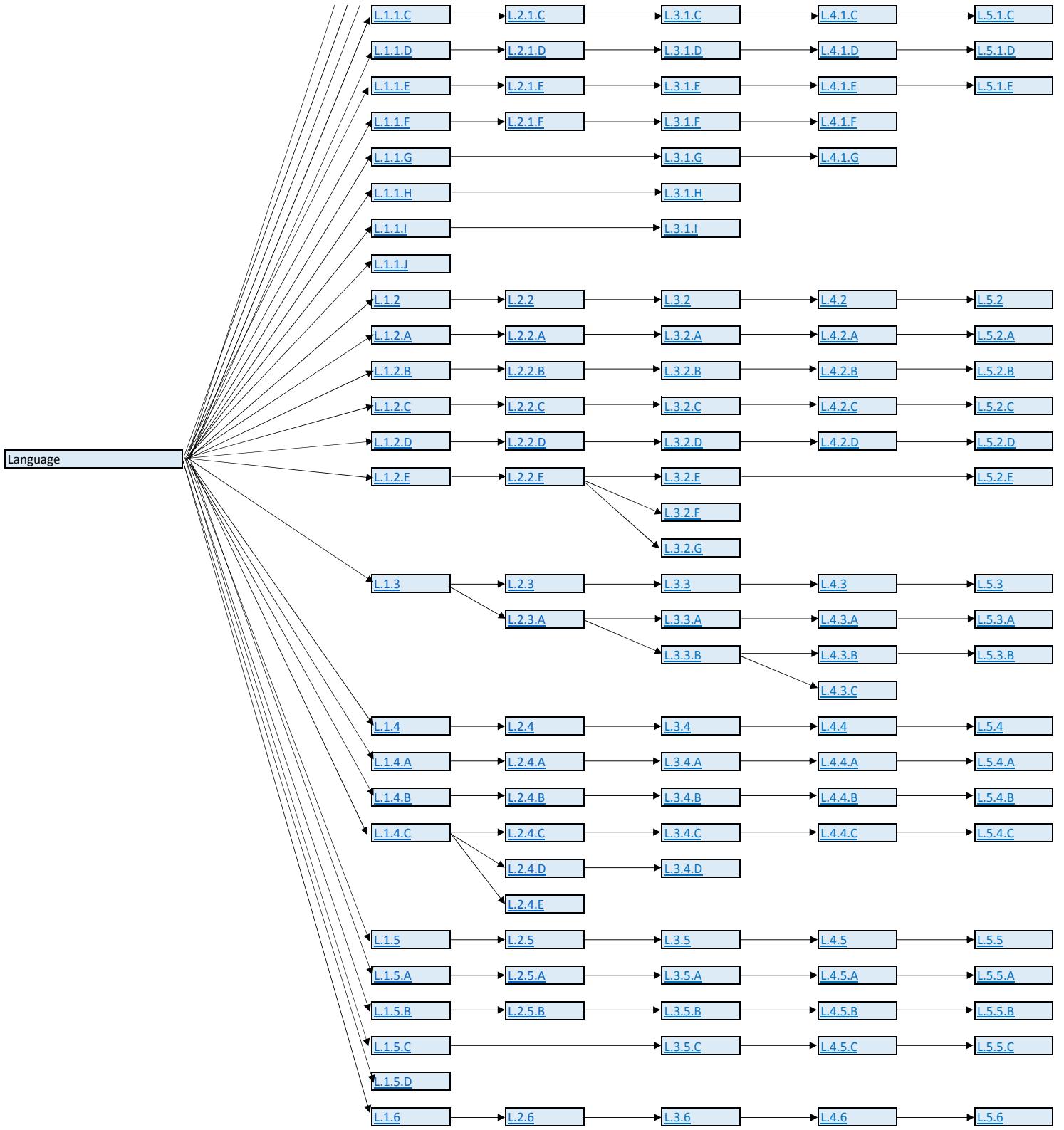
## **CCSS Mathe Abhängigkeitsbaum**



# **CCSS Englisch Abhängigkeitsbaum**







## **Usability Evaluation**

Name:	Marcel
Age:	22
Gender:	male
Country:	Germany
Smartphone and Android Version:	Samsung galaxy s2+ ; 4.4.2

NoRPG

Usability Evaluation - NoRPG					
Goal	Description	Valuation			
		--	-	0	+
<b>Startscreen (Logged Out User)</b>					
Log-In	1. Enter login data (Username: usabilityExpert, Password: norpg) 2. Click Log-In Button				x
Change Settings	1. Click Settings Button in the lower left corner Turn music off/on	2.		x	
Register	1. Click Register Button in the center of the screen				
<b>Register</b>					
Create new Account	1. Fill in the form until Account is created				x
Cancel Registration	1. Click Cancel Button in the upper right corner		x	x	
<b>Startscreen (Logged In User)</b>					
Log-In	1. Click Log-In Button in the center of the screen				
Log-Out	1. Click Log-Out Text	x			
Change Settings	1. Click Settings Button in the lower left corner Turn music off/on	2.		x	
<b>Ingame User Interface</b>					
Overview	Just have a look at the User Interface				x
Move Character	1. Use Joystick in the lower left corner to move character around the world			x	
Start Interaction	1. Go to a Non-Player Charater 2. User Interaction Button "A" in the lower right corner to start interaction 3. Navigate through interaction with "A" and "B"			x	
Open Map	1. Click on the mini map in the upper right corner				x
Open Menu	1. Click Button on the upper left corner			x	
View Games	1. Open Menu 2. Use Button "View Games" 3. Open played game		x		
View Progress	1. Open Menu 2. Use Button "View Progress" 3. View player progress		x		
View Achievements	1. Open Menu 2. Use Button "View Achievements" 3. View player achievements		x		
Change Settings	1. Open Menu 2. Use Button "Settings" 3. Change Quality Settings and/or Music Settings	x			
Log-Out	1. Open Menu 2. Use Button "Log Out"		x		
<b>Any suggestions?</b>					
Start menu: Graphic settings icon (especially on the start screen) is not intuitive. Registration, Birthday: date in the future is possible. You can click "accept" in the character creation without choosing any character. Logout: "do you really want to log out" text. Settings: Quality: Quality high/low rather than "high quality on/off". logout: "do you really want to log out?"					

# Skripte für die Datenhaltung

```
1 using UnityEngine;
2 using UnityEngine.SceneManagement;
3
4 public class LoadingScreen : MonoBehaviour {
5     private static bool isInitialized = false;
6
7     public static MappingStandardsToCourses Settings {
8         get; private set;
9     }
10
11    private static void Initialize() {
12        if (isInitialized) {
13            return;
14        }
15
16        IConfigReader configReader = new WebJSONConfigReader();
17        Settings = configReader.LoadSettings();
18
19        isInitialized = true;
20    }
21
22    public void Start() {
23        Initialize();
24
25    }
26}
```

Listing 14: LoadingScreen.cs

```
1 interface IConfigReader {
2     MappingStandardsToCourses LoadSettings();
3 }
```

Listing 15: IConfigReader.cs

```
1 using System;
2 using UnityEngine;
3
4 [Serializable]
5 public class MappingStandardsToCoursesBean {
6     public ClassBean[] classes;
7
8     public static MappingStandardsToCoursesBean CreateFromJSON(string json) {
9         return JsonUtility.FromJson<MappingStandardsToCoursesBean>(json);
10    }
11 }
12
13 [Serializable]
14 public class ClassBean {
15     public string name;
16     public CourseBean[] courses;
17 }
18
19 [Serializable]
20 public class CourseBean {
21     public string name;
```

```

22     public StandardBean[] standards;
23 }
24
25 [Serializable]
26 public class StandardBean {
27     public string name;
28     public string[] vorbedingungen;
29     public string[] nachbedingungen;
30     public GamesBean[] games;
31 }
32
33 [Serializable]
34 public class GamesBean {
35     public string id;
36     public string name;
37     public string url;
38     public string standard;
39 }
```

Listing 16: MappingStandardsToCoursesBean.cs

```

1  public class MappingStandardsToCourses {
2      public readonly Classes[] classes;
3      public MappingStandardsToCourses(Classes[] classes) {
4          this.classes = classes;
5      }
6  }
7
8  public class Classes {
9      public readonly Courses[] courses;
10     public readonly string name;
11
12     public Classes(string name, Courses[] courses) {
13         this.name = name;
14         this.courses = courses;
15     }
16 }
17
18 public class Courses {
19     public readonly Standards[] standards;
20     public readonly string name;
21
22     public Courses(string name, Standards[] standards) {
23         this.name = name;
24         this.standards = standards;
25     }
26 }
27
28 public class Standards {
29     public readonly string name;
30     public readonly string[] vorbedingungen;
31     public readonly string[] nachbedingungen;
32     public readonly Games[] games;
33
34     public Standards(string name, string[] vorbedingungen,
35                     string[] nachbedingungen, Games[] games) {
36         this.name = name;
37         this.nachbedingungen = nachbedingungen;
38         this.vorbedingungen = vorbedingungen;
```

```
39         this.games = games;
40     }
41 }
42
43 [System.Serializable]
44 public class Games {
45     public readonly string id;
46     public readonly string name;
47     public readonly string url;
48     private string standard;
49
50     public Games(string id, string name, string url) {
51         this.name = name;
52         this.id = id;
53         this.url = url;
54         this.standard = "";
55     }
56
57     public string Standard {
58         get {
59             return standard;
60         }
61
62         set {
63             standard = value;
64         }
65     }
66 }
```

Listing 17: MappingStandardsToCourses.cs