UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Entwicklung einer interaktiven Spiele-Applikation unter Android

Bachelorarbeit

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

vorgelegt von

René Döres

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Bastian Krayer (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Oktober 2019

Erklärung

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

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

Odenheim, 22.10.2019 .....

(Ort, Datum) (Unterschrift)

Institut für Computervisualistik
AG Computergraphik
Prof. Dr. Stefan Müller
Postfach 20 16 02
56 016 Koblenz
Tel.: 0261-287-2727
Fax: 0261-287-2735
E-Mail: stefanm@uni-koblenz.de

UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Aufgabenstellung für die Bachelorarbeit
René Döres (Matr.-Nr. 215 101 123)

Thema: **Entwicklung einer interaktiven Spiele-Applikation unter Android**

In der heutigen Zeit ist der Gebrauch von Smartphones ein Teil unserer Gesellschaft. Die Unterhaltungsbranche ist bei Apps stark frequentiert und bietet eine Vielfalt an Genres. Die Möglichkeiten bei der Entwicklung von Applikationen unter Android scheinen somit endlos. Die Herausforderung besteht darin mit aktuell endlichen Tools, eine logisch bedienbare Anwendung zu entwickeln, die optisch und Nutzeransprechend ist.

Ziel dieser Arbeit ist es, sich in die Softwareentwicklung mit der Android Plattform mit Hilfe von Tools einzuarbeiten. Darauf aufbauend soll eine interaktive Spiele-Applikation konzeptioniert und umgesetzt werden, um die Grenzen und Möglichkeiten zu erforschen und zu analysieren.

Schwerpunkte dieser Arbeit sind:

1. Einarbeitung in die Technischen Grundlagen
2. Konzeption einer interaktiven Spiele-Applikation
3. Erstellung der Modelle, Implementierung
4. Evaluation und Bewertung der Ergebnisse
5. Dokumentation der Ergebnisse

Koblenz, den 16.04.2019

- René Döres -

- Prof. Dr. Stefan Müller-

Abstract

The goal of this work is the induction, conception, implementation and evaluation of an interactive game application among Android. The game genre of the app is a 2D-Jump ‘n’ Run Side-Scroller, whose graphical implementation is based on the four elements earth, fire, water and wind. The application should have classic functions of a Jump ‘n’ Run game and allow the player to overcome the four game worlds to find the finish. The implementation is based on Unity Engine and Adobe Photoshop. A user test asks basic questions about the application and specific questions about the research question, which are then evaluated. The research question examines the connection between fun factor and color perception while playing the app. Represented by the natural color combinations of the four elements. At the end possibilities for expansion and future prospects will be discussed.

Zusammenfassung

Das Ziel dieser Arbeit ist die Einarbeitung, Konzeption, Implementierung und Evaluation einer interaktiven Spiele-Applikation unter Android. Das Gamegenre der App ist ein *2D-Jump ’n’ Run Side-Scroller*, dessen grafische Umsetzung auf den vier Elementen Erde, Feuer, Wasser und Wind basiert. Die Anwendung soll über klassische Funktionen eines *Jump ’n’ Run* Spiels verfügen und es dem Spieler ermöglichen die vier Spielwelten zu überwinden um ins Ziel zu finden. Die Basis der Umsetzung bilden Unity Engine und Adobe Photoshop. Ein Nutzertest stellt grundlegende Fragen zur Anwendung sowie spezifische Fragen zur Forschungsfrage, die anschließend bewertet werden. Die Forschungsfrage prüft den Zusammenhang zwischen Spaßfaktor und Farbwahrnehmung bei dem Spielen der App. Dargestellt durch die natürlichen Farbkombinationen der vier Elementen. Abschließend werden Erweiterungsmöglichkeiten und Zukunftsaussichten diskutiert.

Inhaltsangabe

1 Grundlagen 1

1.1 Android 1
1.1.1 Komponenten einer Android Anwendung 2
1.1.2 Aktivitäten Lebenszyklus 3
1.2 Entwicklungsumgebungen 4
1.2.1 Android Studio 4
1.2.2 LibGDX 5
1.2.2.1 Applikations Lebenszyklus 6
1.2.3 Unity 7
1.2.3.1 Lebenstyklus Unity 8
2.3 C Sharp 8
1.4 Android SDK und JDK 9
1.5 Sony Vegas Pro15 9
1.6 Adobe Photoshop CS6 9
1.7 Jump ‘n‘ Run 10
1.8 Zusammenfassung 11

2 Inspirierende Applikationen 12
2.1 Super Mario Bros 12
2.2 Limbo 12
2.3 The Whispered World 13
2.4 Rayman Jungle Run 14
2.5 Zusammenfassung 14

3 Konzeption 15
3.1 Spielwelt 15
3.2 Hintergrund und Grafik 16
3.3 Musikauswahl und Menü 16
3.4 Steuerung und Kamera 16
3.5 Elemente und Effekte 16
3.6 Hindernisse und Items 16
3.7 Schwierigkeitsgrad und Anzeige 16
3.8 Forschungsfrage 16

4 Umsetzung 17
4.1 Skripte 17
4.2 Szenenhierarchie Objekte 17
4.2.1 OverviewManager 17
4.2.2 CanvasManager 17
4.2.3 LevelManager 17
4.3 Nicht sichtbare Objekte 18
4.3.1 Kontrollpunkte 18
4.3.2 Todeszone 18

4.3.3 Nicht sichtbare Wand 18
4.4 Canvas in den Spielszenen 19
4.4.1 Informationsarbeitsfläche 19
4.4.2 Pausenarbeitsfläche 19
4.4.3 Steuerungsarbeitsfläche 20
4.5 Hauptmenü 21
4.5.1 Level Auswahl 23
4.5.2 Levelende und Regeln 24
4.6 Hauptkamera der Szene 25
4.7 LevelManager 25
4.8 Überbegriff Objekte 26

5 Analyse 31
5.1 Testsystem 31
5.2 Aufbau des Nutzertest 31
5.3 Durchführung des Nutzertest 31
5.4 Darstellung der Ergebnisse 31
5.4.1 Benutzerfreundlichkeit 31
5.4.2 Mechanik 32
5.4.3 Optik 34
5.4.4 Farbwarhnehmung I 35
5.4.5 Farbwarhnehmung II 39
5.4.6 Farbwarhnehmung III 39
5.5 Bewertung des Nutzertest 40
5.6 Kritische Reflexion der eigenen Untersuchung 42

6 Fazit und Ausblick 43
6.1 Ausgangssituation 43
6.2 Fazit 43
6.3 Ausblick 44

7 Anhang 44

Literatur

1 Grundlagen

In den folgenden Abschnitten werden wichtige Grundkenntnisse erläutert und relevante Themen für die Bachelor Arbeit aufgegriffen. Es soll ein Überblick über das Themengebiet erschaffen und nützliches Wissen vermittelt werden.

1.1 Android

Android (im altgriechischen Mann und Gestalt) ist eine Software-Plattform, als auch ein Betriebssystem für mobile Geräte. Gegründet von Google und dessen Unternehmen Open Handset Alliance, veröffentlicht am 23.September 2008. Es ist kompatibel mit Notebooks, Mediaplayer, Smartphones, Fernsehern, Mobiletelefonen, Tablett-Computern und Desktop-Computern. Android verwendet einen monolithischen Linux Kernel. Dadurch können auch Treiber für die Hardwarekomponenten und mögliche Funktionen direkt eingebaut sein. Linux lagert zum Beispiel Funktionalitäten in Kern-Module aus [dap].

Jeder Prozess wird in einer virtuellen Maschine erzeugt und die Android Anwendungen werden in einer JVM ausgeführt. Es handelt sich um eine freie Software, geschrieben in Java, C, C++ und anderen Programmiersprachen. Die Android Software unterliegt der Lizenz von Apache Lizenz 2.0 und der Linux Kernel der von GNU GPL v2. Dies ermöglicht es dem Anwender Android zu studieren, zu modifizieren und zu verteilen [eae].

Android ist eins der beliebtesten und am häufigsten gekauften Betriebssysteme für Smartphones. Der Marktanteil im Jahr 2017 betrug etwa 86 Prozent. Im folgenden Diagramm (Abb. 1) sind die weltweit verkauften Smartphones nach Betriebssystemen aufgeführt und werden miteinander verglichen.

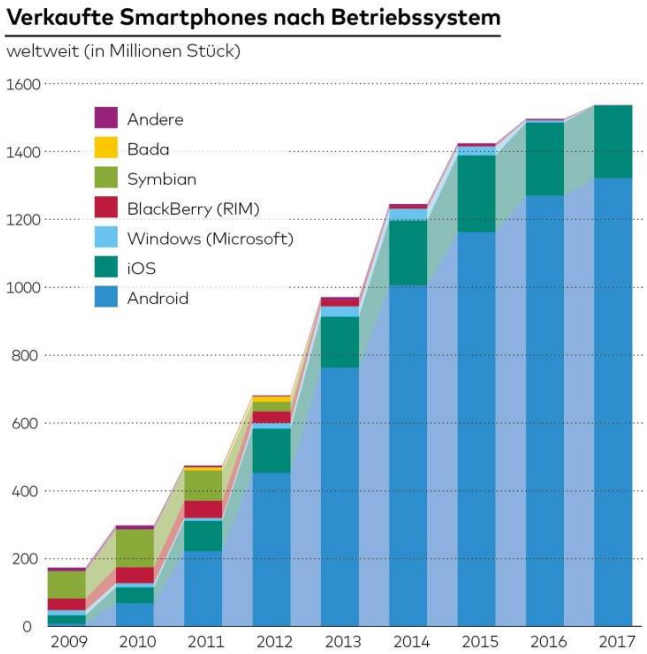

Abb. 1: Verkaufte Smartphones nach Betriebssystem 2017, [bet]

Unteranderem beinhaltet Android proprietäre Software wie Google Play, auch bekannt im Englischen als *closed source software*. Google schränkt das Recht für Nutzer und Dritte stark ein, in der Weiter- und Wiederverwendung, sowie Anpassung und Änderung. Aufgrund des geistigen Eigentumsrechts, dem Urheberrecht und zahlreicher Patente, behält Google von externen Apps einen gewissen Prozentsatz des erwirtschafteten Umsatzes. Entwickler können kostenpflichtige, wie kostenfreie Android Applikationen in dem Google Play Store veröffentlichen, unter der oben aufgeführten Prämisse. Aufgrund der Globalisierung, der wachsenden Weltbevölkerung, wirtschaftlichen Aspekten und kulturellen Aspekten, besteht eine hohe Nachfrage für Applikationen jeder Art. Heutzutage lohnt sich die Entwicklung von Anwendungssoftware wie nie zu vor [pug]. Das folgende Diagramm (Abb. 2) zeigt die Prognose der steigenden Umsatzentwicklung von Google Play für die Jahre 2014 bis 2019 in US-Dollar.

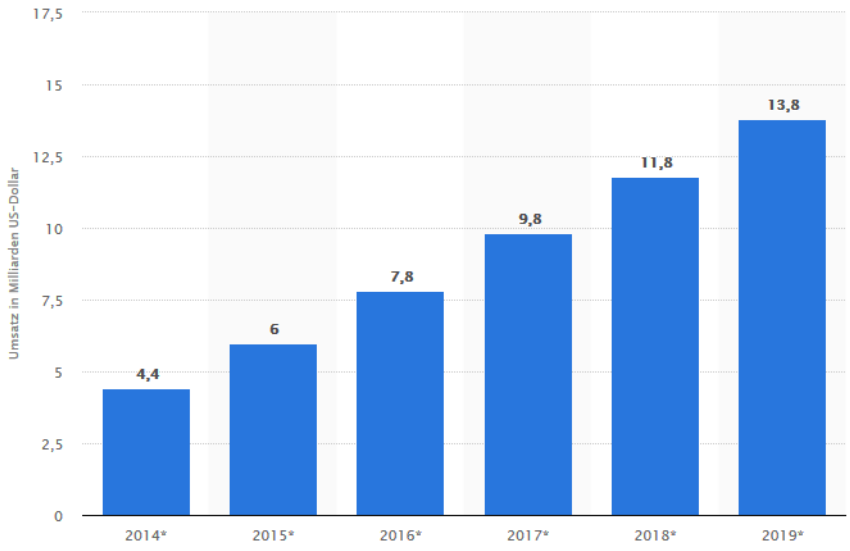

Abb. 2: Prognose der Umsatzentwicklung von Google Play, [pug]

1.1.1 Komponenten einer Android Anwendung

Eine Android Anwendung besteht aus mehreren Komponenten, dem *Android Manifest*, aus *Activities*, *Application Ressources* und *Layouts*. Im Folgenden werden diese genauer beschrieben.

Das *Manifest-File* ist der Drehpunkt jeder Android-Anwendung. Die Metadaten einer Anwendung werden in Form einer XML-Syntax hinterlegt und in einer Textdatei abgespeichert. Darunter fallen Metadaten, wie die Zugriffsrechte einer Anwendung, oder das Registrieren aller *Activities* und deren Verhalten einer Applikation. Ein einfaches Beispiel für ein Zugriffsrechte ist der Internetzugang.

Eine Applikation besteht meistens aus mehreren *Activities* mit der Fähigkeit zwischen diesen zu navigieren. Diese sind zuständig für einen Großteil der Logik einer Anwendung, ähnlich der Nutzeroberfläche, da sie auf Eingaben des Nutzers reagieren und eventuelle Ausgaben erzeugen. Ein

View besitzt Informationen über die Bildinhalte und eine *Activity* bekommt einen *View* zugeordnet, um auf Eingaben reagieren zu können. Eine Anwendung muss mindestens aus einer *Activity* bestehen, zugleich kann nur eine aktiv im Vordergrund sein. *Intents* werden hauptsächlich für das Starten von *Activties* verwendet, da diese sich dadurch selbständig ausführen können. Ähnlich eines Bindeglieds zwischen der Aufrufenden und der Aufgerufene. *Activities* werden als Java-Datei abgespeichert.

Layouts beschreiben den Bildschirmaufbau und bestimmen die Anordnung der *Views*. Sie werden einer *Acitivity* zur Laufzeit zugeordnet. *Views* sind also Elemente an der Oberfläche die dem Nutzer präsentiert werden. Bei Android ist die grafische Darstellung ein Baum von *Views*. Grafische Oberflächen werden hauptsächlich in XML-*Layoutfiles* beschrieben.

In den *Application Resources* sind alle Audio- und Bilderressourcen hinterlegt. Welche an jeder Stelle in der Anwendung erreichbar sind. Sowie die oben aufgeführten XML-*Layoutfile*, das Manifest und Java-Klassendateien [taa].

1.1.2 Aktivitäten Lebenszyklus

Während der Laufzeit einer Android Anwendung durchlaufen *Activities* mehrere Zustände. Mithilfe von Rückruffunktionen werden *Activities* über Zustandsänderungen informiert. In der nachfolgenden Grafik (Abb. 3) werden die Rückruffunktionen, auch *Callback*, in Rechtecken, die Aktivitäten farblich und die Richtung mit schwarzen Pfeilen dargestellt.

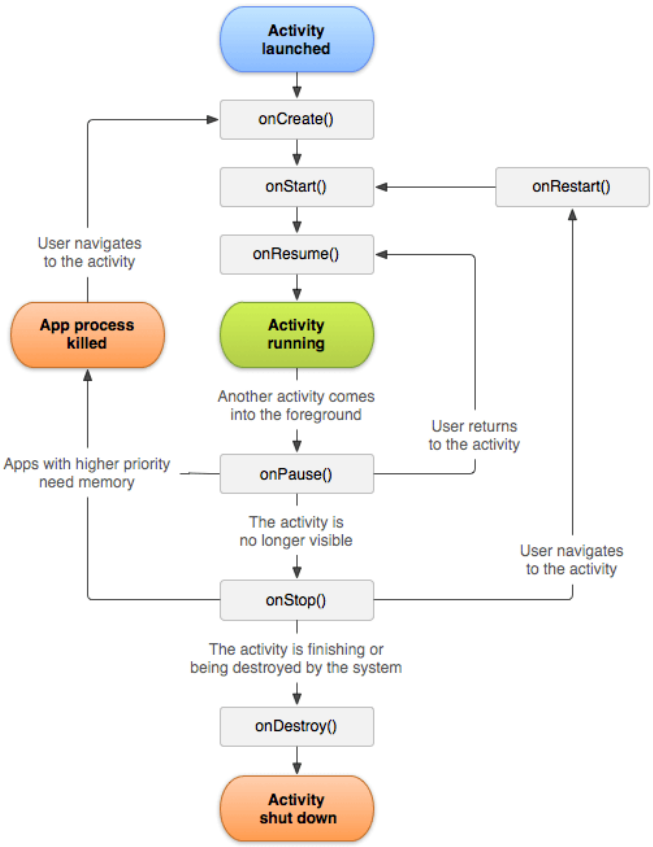

Abb. 3: Activity lifecycle, [alcy]

Die Aktivitätsklasse eines Kernsatzes stellt sieben Rückruffunktionen zur Verfügung. Damit können Übergänge zwischen den Phasen des Aktivitäts-Lebenszyklus navigieren werden. Sobald eine Aktivität einen neuen Zustand erreicht, ruft das System jeden der sieben Rückruffunktionen auf. Es gibt vier Hauptzustände: *Activity running*, *Activity launched*, *Activity shut down* und *App process killed*. Im folgenden Schema werden diese genauer erläutert.

Eine *Activity* befindet sich im *Activity running* Zustand. Es wird die Rückruffunktion *onCreate()* aufgerufen, dadurch wird die *Activity* erzeugt und bekommt einen *View* zugewiesen. Mit dem Wechseln in die Funktion *onStart()* bereitet sich die *Activity* vor in den Vordergrund zu treten und interaktiv zu werden. Um diese letztendlich auf dem Bildschirm sichtbar zu bekommen, wird die Funktion *onResume()* aufgerufen.

Eine *Activity* befindet sich im *Activity launched* Zustand. Im Gegensatz zu dem obigen Zustand, kann dieser unterbrochen werden, um eine andere *Activity* in den Vordergrund zu rufen. Dies geschieht mit der Funktion *onPause(),* dadurch wird die aktuelle Aktivität pausiert. Um diese wieder in den Vordergrund zu rufen, also sichtbar zu machen, wird die Rückruffunktion *onResume()* aufgerufen. Sobald eine andere *Activity* gestartet wurde, ist der Zustand *Activity launched* nicht mehr sichtbar und wird durch die Funktion *onStop()* gestoppt. Dieser bleibt aber im Hintergrund aktiv. Sollte nun eine andere Applikation eine höhere Priorität bekommen und Speicher benötigen, wechselt die *Activity* in den Zustand *App process killed.* Damit wird der Applikationsprozess beendet und stellt neuen Speicherplatz zu Verfügung. Es besteht weiterhin die Möglichkeit für den Nutzer die *Activity* zu reaktivieren, durch das Aufrufen der Funktion *onCreate().*Mit dem Wechseln in die Funktion *onRestart()* kann diese wieder in den Vordergrund gerufen werden.

Ist die *Activity* wieder im Zustand *Activity launched,* kann auf die Funktion *onDestroy()* zugegriffen werden. Der Funktion folgt, dass die *Activity* abgeschlossen ist oder das System diese zerstört hat. Danach wechselt die *Activity* in den Zustand *Activity shut down* und beendet den Lebenszyklus.

1.2 Entwicklungsumgebungen

Im Folgenden Abschnitt werden Plattformen vorgestellt, mit dem die Softwareentwicklung für eine Android-Anwendung realisiert werden kann.

1.2.1 Android Studio

Android Studio ist eine freie integrierte Entwicklungsumgebung von Google und wurde nach einer zweijährigen Entwicklungsdauer im Dezember 2014 veröffentlicht. Es ist eine quelloffene Umgebung und kompatibel für die Betriebssysteme Windows, macOS, Linux, Ubuntu und Chrome OS. Es ist mit Java, Kotlin und C++ programmiert und ist kostenfrei zu erwerben. Seit Ende 2015 wird das Android Developer Tool (ADT) für Eclipse nicht mehr unterstützt und basiert seit dem her auf der Entwicklungsumgebung IntelliJ IDEA 14 (IDE). Welche die Programmiersprachen Java, Kotlin, Groovy und Scala kennt. Zu dessen Besonderheiten gehören unteranderem die Unterstützung von Java EE und JUnit. Sowie Werkzeuge zum *Refactoring* von Codes, Git und einem GUI-Editor [lli].

Die aktuelle Version von Android Studie beläuft sich auf der *3.6 Canary,* veröffentlicht am 23. August 2019. Zusätzliche Funktionen zu dem bereits Beschriebenen IntelliJ IDEA komplettieren die

Entwicklungsumgebung. Im Folgenden werden weitere Funktionen gezeigt. Dazu gehören zum Beispiel ein Themen-Editor und das Android Lint, für die statische Codeanalyse. Weiterhin verwendet Android Studio das *Build-Managment-Automatisierungs-Tool*, dieses System ermöglicht dem Entwickler eine geräteunabhängige Entwicklung von Applikationen. Sowie die Unterstützung der Software ProGuard, welche kompilierte Java-Dateien komprimieren, optimieren und dessen Decodierung erschweren kann. Auch ist dem Entwickler die Verwendung von Google Diensten möglich, diese können innerhalb der IDE konfiguriert und direkt auf die Applikation angewandt werden. Als Letztes sind noch die Unterstützung von verschiedenen Programmiersprachen und die Kompatibilität mit anderen Entwicklungsumgebungen zu erwähnen. Ein Beispiel dafür ist Xamarin Studio und der angewandten Sprache C#. Der Lebenszyklus von Android Studio orientiert sich dabei an den Lebenszyklus von Android (Abb. 3) [ast].

1.2.2 LibGDX

LibGDX ist ein quelloffenes Java - *Framework* für die plattformunabhängige Spieleentwicklung. Veröffentlicht im Jahr 2009, entwickelt von Mario Zechner in den Programmiersprachen Java, C und C++. Die Entwicklungsumgebung unterstützt die Betriebssysteme Windows, Linux, MacOS, Android, BlackBerry, IOS und Webbrowser. LibGDX ist auf GitHub als Open-Source-Software freigegeben und kann kostenlos heruntergeladen werden. Die Umgebung unterliegt der Apache Lizenz 2.0 einer anerkannten Freien-Software-Lizenz. Somit kann der Entwickler sein komplettes Projekt als sein Eigenes patentieren lassen, auch unter Verwendung dieses *Frameworks*.

Nach zwei jähriger Aktualisierungspause ist am 19.07.2019 eine neue Version *1.9.10* erschienen. Die Umgebung ist mit Bibliotheken, wie zum Beispiel Box2D oder Bullet Physics erweiterbar. Es können 2D, 2.5D und 3D Spiele entwickelt und umgesetzt werden. LibGDX arbeitet unteranderem mit der OpenGL ES 2.0 Bibliothek und einer 3.0 Wrapper-Schnittstelle, welche zur 3D-Entwicklung gebraucht werden. [libd]

Im Vergleich zu Android Studio fehlen nützliche Werkzeuge wie ein vollwertiger Level-Editor. Im großen Ganzen ist es weniger als Entwicklungsumgebung zu verstehen wie z.B. ein Android Studio und wird aufgrund dessen seltener für die Spieleentwicklung genutzt. Trotzdem ist es ein sehr kompetentes *Framework*, welches auch in Android Studio eingebunden werden kann.

LibGDX besitzt wichtige Kernschnittstellen für die Interaktion mit dem zu unterstützenden Betriebssystem. Im Folgenden Text werden einige wichtige vorgestellt.

ApllicationListener.java orientiert sich an den Lebenszyklus von Android, bei dem alle Methoden in einem Thread aufgerufen werden.

Graphics.java generalisiert die Kommunikation mit der GPU. Es werden Methoden abhängig der aktuellen Version bereitgestellt für den Umgang mit Bibliotheken. Zum Beispiel zur Entwicklung in 3D .

Files.java ermöglicht ein plattformunabhängiges Lesen und Schreiben der Ressourcen.

Application.java ist der Einstiegspunkt zum Laden des Spiels. Die Schnittstelle verfügt über Methoden zum Abfragen bestimmter Informationen über das Betreibsystem und ermöglicht das Benutzen verschiedener Module, als Beispiel Grafik und Sound. Weiterhin ist sie für das Rendern der Inhalte auf der Oberfläche zuständig.

Input.java kann sensorbasierende Ereignisse verarbeiten und informiert über interaktive Benutzereingaben.

Des Weitern gibt es erwähnenswert Module die nützliche Klassen enthalten und diese bereitstellen. Sei es für die mathematische Berechnung, oder eine generische Level-Map Implementierung, oder das Laden und Speichern unterschiedlicher Ressourcen. Um einige Module erwähnt zu haben *Math, Maps, Assets, Scenes, Utils.* [liba]

1.2.2.1 Applikationen Lebenszyklus

Beim Erstellen einer neuen Aktivität muss parallel ein neuer openGL-Kontext erstellt werden. Es müssen alle grafischen Ressourcen neu geladen werden, dies hat zur Folge, dass bei mehreren Aktivitäten ein hoher Berechnungsaufwand hin zukommt. Es ist sinnvoll die Spielwelt ein eine einzelnen Szene aufzuteilen. Im Gegensatz zu Android Applikationen die aus mehreren Aktivitäten bestehen können, besteht eine libGDX Applikation nur aus einer Aktivität. Bei genauerem hinschauen besteht eine gewisse Ähnlichkeit zu Android. Die Antwort liegt in der Vergangenheit, als libGDX eine Bibliothek für Android basierende Geräte war.

Das folgende Diagramm (Abb. 4) beschreibt den Lebenszyklus. Zustände einer Applikation werden farblich repräsentiert und die Rechtecke beschreiben die Funktionen der *ApllicationListener* Schnittstelle.

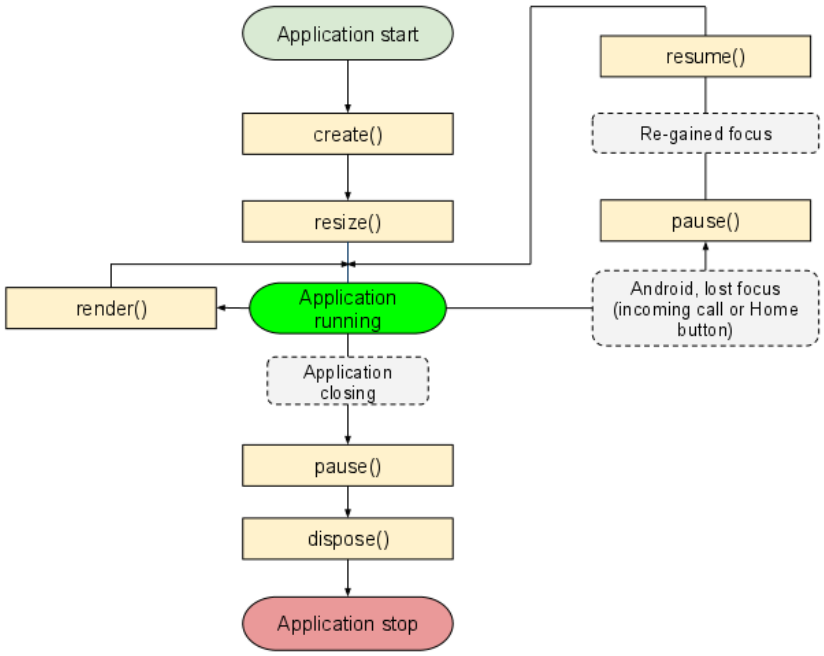

Abb. 4: Lifecycle libGDX, [lib]

Der Lebenszyklus wird von der *AplicationListener* Schnittstelle modelliert und bildet ein Teil der Logik. Die Implementation obliegt dem Entwickler und dient zu dessen Eingreifen in den Lebenszyklus.

Zusttand Application start, die Anwendung wird gestartet, dabei wird die Funktion *create()* der *ApplicationListener* Schnittstelle aufgerufen. Beim Verändern der Bildschirmgröße wird die *resize()* Funktion ausgeführt.

Die Anwendung befindet sich im *Application running* Zustand. Sobald die Oberfläche neu gerendert werden muss, wird die die Funktion *render()* ausgeführt. Wird währenddessen eine andere App geöffnet oder diese pausiert, wird die Funktion *pause()* aufgerufen. Danach kann die Anwendung mit der Funktion *resume()* wieder in den Vordergrund gebracht werden. Wird die Anwendung vom Nutzer geschlossen wird zuerst die Funktion *pause()* ausgeführt, zum stoppen um anschließen mit der *dispose()* Funktion diese zu beenden. Damit ist der Zustand *Application stop* erreicht und der Lebenszyklus wird beendet [lib].

1.2.3 Unity

Unity ist eine Entwicklungs- und Laufzeitumgebung für Spiele des amerikanischen Unternehmens Unity Technologie, veröffentlicht 2005. Die Software ist für die Betriebssysteme Windows, macOS, Linux verfügbar und besitzt eine *closed-source* Software Lizenz für ihren *code*. Der Code ist geschützt aber das entwickelte Endprodukt mit Hilfe der Plattform ist mein Eigentum. Die aktuellste Version ist die *2019.2.9* am hochgeladen am 12 Oktober 2019. Zielplattformen von Unity sind PS4, Xbox On, Wii U, Nintendo, Switch, tvOS, HoloLens, IOS, Android, PSV. Die Entwicklung von interaktiven 2D, 2.5D, 3D Anwendungen sind möglich und der Entwickler wird bestmöglich von Unity geleitet.

1.2.3.1 Lebenstyklus Unity

Der Lebenszyklus ist zu groß um darauf genauer einzugehen. Das Folgende Diagramm (Abb. 5) zeigt wie Unity die Ereignisfunktionen während der gesamten Lebensdauer eines Skriptes anordnet und wiederholt.

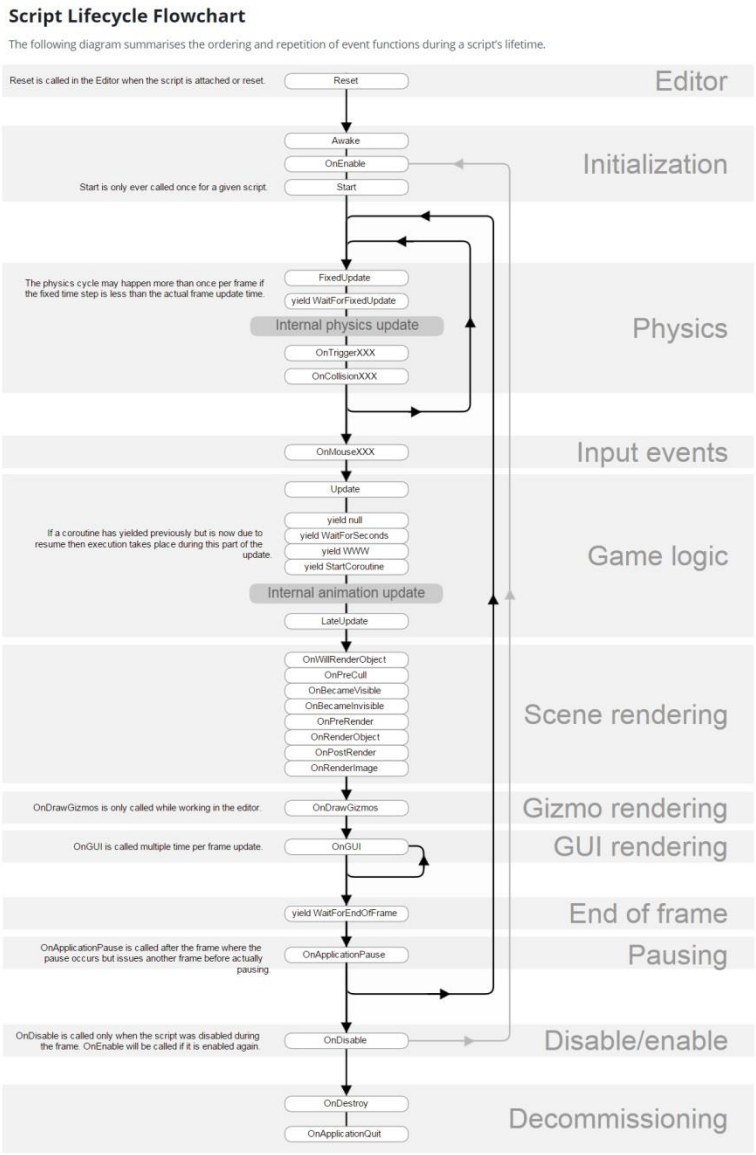

Abb. 5: Lebenszyklus Unity; [LUU]

2.3 C Sharp

C# (im englischem c sharp) ist eine objektorientierte Allzweck-Programmiersprache. Entwickelt im Jahr 2000 von Microsoft. Die aktuelle Version *8.0* ist die neunte Generation und wurde am 23. September 2019 veröffentlicht. Die Programmiersprache ist plattformunabhängig und wurde im Rahmen der .NET-Strategie entwickelt, mit dem Focus auf der webbasierenden Komponente. Sie ist auf diese optimiert und meist in diesem Kontext zu finden. Ursprünglich wurde sie für Windows entwickelt aber mit Xamarin, einer Tochtergesellschaft, kann diese auch zur Entwicklung für macOS, IOS und Android verwendet wenden. Weiterhin besteht die Möglichkeit mit .Net Native einer Werkzeugkette, plattformunabhängigen Maschinencode anzubieten, für alle C++ unterstützenden Plattformen. Passende beschreibende Paradigmen für C# sind multiparadigmatisch, strukturiert, objektorientiert, ereignisorientiert, funktional und generisch.

Ein Unterschied zur Programmiersprache C++, sind zum Beispiel Zeiger. In C++ können Zeiger mit allen Rechten inkludiert werden. In C# wird dieser nur für unsichere Codes erlaubt, welcher mit eingeschränkten Rechten agiert. Ein *Delegat*, eine Weiterentwicklung von Funktionszeigern aus C,

kann auf Methoden einer Klasse verweisen. Somit wird implizit ein Objektzeiger als Parameter an die Methoden übergeben. Zusätzlich erlauben es Attribute, Informationen über ein Objekt, eine Klasse oder eine Methode zu speichern. Diese Metadaten können via Laufzeit ausgewertet werde. Der Vorteil, Inkompatibilitäten der Methodensignatur, eine formale Schnittstelle einer Funktion oder Prozedur, zwischen aufrufenden *Delegat* und aufzurufenden Methode, werden während der Kompilierung bemerkt [cs].

2.4 Android SDK und JDK

Die Android SDK und JDK Werkzeuge sind Voraussetzung zur späteren Umsetzung der Applikation für das Betriebssystem Android.

Das Android Software Development Kit SDK enthält eine Sammlung von Entwicklungswerkzeugen, die die Entwicklung einer Android Applikation unterstützt. Tools wie Beispielcode, eine Dokumentation, Tutorials und die Lizenz der Android Open Source Project sind sehr attraktiv für den Entwickler. Das SDK ist auf Android seit März 2015 nicht mehr selbst verfügbar, jedoch ist die Softwareentwicklung weiterhin mit speziellen Android-Anwendungen möglich. Die aktuellste Version *26.1.1* wurde 2017 veröffentlicht. Das SDK Werkzeug kann bei der Installation von Android Studio mit installiert werden und unterstützt verschiedene Betriebssysteme.

Das Java Development Kit JDK verfügt über die Laufzeitumgebung Java Runtime Environment (JRE), dies ermöglicht das ausführen von Programmen weitgehend unabhängig von darunterliegenden Betriebssystemen. Die aktuellste Version *11* wurde 2018 veröffentlicht und ist für verschiedene Betriebssysteme zu erhalten. Außerdem können mit der Umgebungen Java-Anwendungen getestet werden [sjdk].

2.5 Sony Vegas Pro

Die Bearbeitungssoftware eignet sich für das schneiden von Audio, Video und dem Zusammenfügen mehrerer Elemente. Sony Vegas Pro wurde 1999 von dem Unternehmen Magix Software GmbH entwickelt und eignet sich für *non-linear editing* (NLE), dabei wird die Originaldatei nicht bearbeitet. Stattdessen werden die Änderungen von einer speziellen Software festgelegt und geändert. Die aktuelle Version *17.0 Build 384* ist seit dem 5 August 2019 verfügbar. Die Lizenz ist proprietär und das Produkt kostenpflichtig zu erwerben.

Zahlreiche Werkzeuge unterstützen den Entwickler bei der Umsetzung seiner Projekte. Nützlich für die vorliegende Arbeit sind unteranderem *Drag'n'Drop*, unterstützt von Audiostandards und chronologischer Materialorganisation. Zusätzlich unterstützen zahlreiche Effekte die Kreativität des Entwicklers und helfen bei deren Umsetzung [sony].

2.6 Adobe Photoshop CS6

Adobe Photoshop ist ein Raster-Grafik-Editor des Softwarehauses Adobe Inc und wurde 1990 entwickelt. Es eignet sich sehr gut für Erstellung und Bearbeitung von Grafiken und Animationen jeglichen Stils. Die aktuellste Version *CC 2019* ist seit dem 15. Oktober 2018 veröffentlicht und die fünfundzwanzigste Generation. Kostenpflichtig erhältlich für die Betriebssysteme macOS und

Windows. Die Lizenz ist proprietär. Es unterstützt die Gestaltung von Grafiken in 2D und 3D. Die Photoshop Dateien bestehen oft aus mehreren Ebenen und werden als Projekt im Format *.PSD* gespeichert. Zusätzlich ist es möglich die komplette Szene in ein bekanntes Format zu konvertieren und als dieses abzuspeichern. Als Beispiel PNG oder GIF Format. Es gibt die Möglichkeit die *.PSD* Datei als *.PSB* zu speichern, dabei wird die maximale Höhe auf 300.00 Pixel und die Länge auf ca. 4 Exabyte (10^{18}) ermöglicht. Zahlreiche Erweiterungen komplettieren die Software, wie zum Beispiel mit zusätzlichen Spezialeffekt Modulen oder 3D Effekt Modulen. Photoshop bringt sehr viel Werkzeuge von sich aus mit. Zu nennen sind Bildverarbeitungsfunktionen, Messfunktionen, Zeichnungsfunktionen und Farbersetzung um nur einige Begriffe zu nennen. [apc]

2.7 Jump ’n’ Run

Die Bezeichnung *Jump ‘n’ Run* hat sich in der deutschen Sprache als Genre etabliert. Aus dem englischen *jump and run*, „spring und lauf“, werden Spiele in der Computer- und Smartphoneszene bezeichnet. Im Englischen wird dieser Spielstiel als *platform game* kurz *platformer* bezeichnet. Die wesentliche Handlung eines *Jump ’n‘ Run* Spiels ist das präzise Springen und Laufen mit dem Hauptcharakter durch verschiedene Level. Die Level werden durch zwei- oder dreidimensionale Koordinatensysteme dargestellt, in denen die meisten Objekte eine Position haben. Dieser Oberbegriff teilt sich in der heutigen Zeit in mehreren Untergenres wie zum Beispiel:

- Plattform/*Platforms ‘n‘ Ladders*, überbrücken von Höhenunterschieden mittels Leitern.

- Plattform/*Shooter*, überwiegend schießende Figuren.

- Plattform/*Fighter*, überwiegend im Nahkampf agierende Figuren.

- Plattform/*Jump ’n‘ Run*, überbrücken von Hindernissen mittels Geschicklichkeit.

Die Übergänge sind heute sehr Komplex, da oft mehrere Genres in einem Spiel vereint werden. Als Beispiel *Rayman Legends* in dem der Nahkampf und Fernkampf von Gegnern, als auch überwinden und einsammeln von Objekten einen Hauptbestandteil des *Jump ’n‘ Spiels* ausmacht.

Ein wesentlicher Unterschied besteht in der visuellen Darstellung der Applikationen in 2D, 2.5D, 3D oder als Hybrid. Es wird unterschieden in *2D-Jump ’n’ Run*, dort wird die Spielwelt von der Seite betrachtet, wie z.B. bei *Super Mario Bros*. Hingegen sind *Jump ’n’ Run* mit isometrischer Perspektive, auch als Dreiviertel-Ansicht bekannt, mit dreidimensionaler Beweglichkeit der Spielfigur. Dies erzeugt einen dreidimensionalen Eindruck der Welt durch die Überblicksperspektive von schräg oben (diagonal), ein Anwendungsbeispiel ist *Simutrans*. Bei *3D-Jump ’n‘ Runs*, ist eine Zentralprojektion der Umgebung darstellt, in der sich die dreidimensionale bewegliche Spielfigur bewegen kann. Durch die Zentralprojektion in Kombination mit darstellender Geometrie können anschauliche Bilder von räumlichen Objekten dargestellt werden, konform der Bilder die beim Sehen mit dem Auge im Augpunkt entstehen [jnr].

2D-Jump ’n‘ Runs treten in verschiedenen Bewegungsparadigmen auf. Wie horizontales *Scrolling* bei *Jump ’n‘ Run* Spielen, ein Beispiel dafür ist Super Mario Bros oder vertikales *Scrolling* bei *Shootern*, wie bei dem Spiel Commando. Weiterhin gibt es die Ansicht der Vogelperspektive z.B. beim Aktion Genre, umgesetzt in Rennspielen, ein Beispiel dafür ist Jet Ski Race: Water Scoot. Als Letztes ist die isometrische Ansicht zu erwähnen, gerne bei Taktikspielen verwendet, z.B. wie in Ant Attack.

Am häufigsten vertreten ist horizontales *Scrolling*, ob mit statischem Hintergrund oder mittels Bewegungsparallaxe. Bei *Jump 'n' Run* Spielen wird oft die seitliche Ansicht auf das Spielgeschehen gewählt, auch bekannt als *Side-Scoller*. Die Laufrichtung ist sehr häufig mit: „von rechts nach links festgelegt". Ein konträres Beispiel dafür ist die Laufrichtung von rechts nach links in dem Spiel *Jungle Hunt* [ssc].

2.8 Zusammenfassung

Im Folgenden Abschnitt werden die Elemente genannt, die zur Umsetzung dieser Bachelorarbeit wichtiger Bestandteil sind. Es wird eine Applikation entwickelt für das Android Betriebssystem. Dafür wird die Entwicklungsumgebung Unity in der Version 2017.4.29f1 verwendet und interne Skripte in der Programmiersprache C Sharp geschrieben. Das Genre der App ist ein *2D-Jump'n'Run Side-Scroller*. Zur Erstellung und Bearbeitung von Grafiken wird Adobe Photshop CS6 verwendet. Sony Vegas 15 dient zum schneiden der Audiodateien. Das Android Studio und dessen SDK und JDK Werkzeug ermöglicht das kompilieren der App von dem zu unterstützenden Betriebssystem von Windows nach Android.

2 Inspirierende Applikationen

Der folgende Abschnitt zeigt Spiele die oben aufgeführte Konzepte verwenden, angelehnt an das Konzept der Thematik dieser Bachelorarbeit.

2.1 Super Mario Bros

Eine der bekanntesten *Jump 'n' Run* Reihe ist Super Mario für die Nintendo-Heimkonsolen. Es gehört zum Genre *platformer* und ist von Nintendo entwickelt und veröffentlicht. Super Mario Bros erschien das erste Mal 1985 in Japan für den Familien-Computer Famicom. Der Spieler muss mit der Spielfigur Gegner sowie Hindernisse bezwingen und überwinden. Das Ziel des Spiels ist es, die Prinzessin aus den Fängen eines Schurken zu retten. Mario rennt und springt durch die zweidimensionale, in Seitenansicht dargestellte, Spielwelt. Mario Bros greift auf das Konzept des *Side-Scrolling* zurück mit einem statischen Hintergrund. Die Spielwelt besteht aus mehreren Level, Boss Gegnern, Gegnern mit unterschiedlichsten Paradigmen, verschiedene Power Items die dem Spieler Fähigkeiten verleiten und Münzen, wie Checkpoints (Abb.5). Die Steuerung erfolgt über die konsolenabhängigen Kontroller. Es existiert eine *Canvas* mit einer Zeitdauer und einer Münzenanzeige. Der Grafikstil ist auch als Pixel Art bekannt. *Nice to know* die Entwickler verinnerlichten sich die Philosophie des Miyamoto-Mentors Yokoi, bei dem erst die Funktion entsteht und später die Form des Spielelements festlegt wird [smb]. Die Folgende Grafik (Abb. 5) zeigt den Grafikstil von Super Mario Bros.

Abb. 6: Elemente und Spielwelt aus Super Mario Bros, [smb1]

2.2 Limbo

Ist ein *2D-Side-Scoller-Computerspiel*. Es zählt zum *Genre Xbox-Live-Arkade-Spiel*. Das Indie Entwicklungsstudio Playdead hat das Spiel am 21. Juli 2010 erstmals für die Xbox360 veröffentlicht. Fünf Jahre später ist es für jede bekannte Konsole und jedes Betriebssystem zu erhalten. Das Konzept liegt bei einer unbunten Spielumgebung. Die verwendeten Farben weisen weder einen bestimmten Farbton noch eine Sättigung auf. Somit besteht eine Welt aus weiß, schwarz, Licht, Schatten und erschafft eine alptraumhafte Welt (Abb. 7). Inspiration fanden die Entwickler im Film Noir aus dem Jahr 1950. Die Spielfigur durchläuft diese Welt auf der Suche nach seiner Schwester. Gegner können flüchten, schon tot sein, sowie angreifen. Der Spieler muss Rätsel lösen, Gefahren abwenden und Fallen überwinden. Die Rätsel und Ereignisse unterliegen physikalischen Kräften wie Elektrizität, Gravitation oder Magnetismus. Dies wurde mit Hilfe einer entwickelten Physik-Engine umgesetzt. Die Hauptspielfigur besitzt ausschließlich zwei Fähigkeiten, das Springen und Greifen. Limbo greift nicht auf detaillierte dreidimensionale Modelle zurück, sondern auf einen minimalsten Stil. Dadurch konnte die Entwicklung des Gameplays in den Vordergrund gelenkt werden [lim]. Das nächste Bild zeigt ein Ausschnitt aus der Spielwelt von Limbo.

Abb. 7: Darstellung des Hauptcharakters in der Spielwelt, [lim1]

2.3 The Whispered World

The Whispered World ist ein *Punkt 'n' Click - Abenteuer* Spiel für Microsoft, Windows und Mac OS. Veröffentlich am 28. August 2009 und entwickelt von dem deutschen Unternehmen Deadalic Entertainment. Entwickelt in der Visionaire Studio Engine. Das Spiel verfügt über *2D-Scrolling* Ansichtsebenen mit Parallaxe Effekt. Üblich bei diesem Genre werden durch überziehen der Maus oder das klicken auf interaktive Bereiche der Welt Aktionen ausgeführt. Diese sind vorher interaktiv und werden durch eine bestimme Aktion in die Welt hervorgehoben. Die Hauptfigur reist mit dem Wanderzirkus seiner Familie durch die Welt. Der Spieler muss dabei Rätsel lösen, Items sammeln und seine eigene *Storyline* erschaffen. Unterschiedliche Antwortmöglichkeiten erzeugen ein individuelles Spielgefühl für den Nutzer [tww]. Das Folgende Bild (Abb. 8) zeigt verschiedene Grafiken die jeweils eine eigene Ebene bilden, später ergibt die frontale Draufsicht auf die Welt ein komplettes Bild.

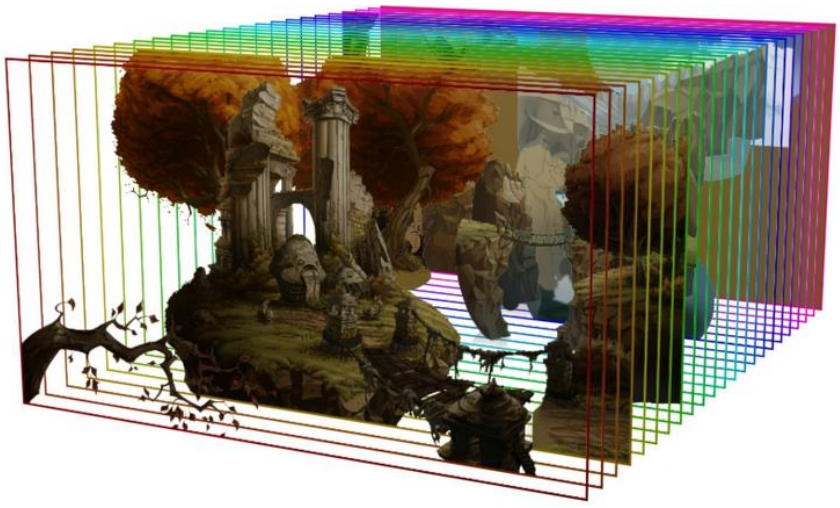

Abb. 8: Verschieden sortierte Grafik Ebenen der Welt, [tww1]

2.4 Rayman Jungle Run

Rayman Jungle Run ist eines der bekanntesten und beliebtesten *Jump 'n' Spiele* für Android. Auf dem Markt hat es Ubisoft im September 2012 gebracht und Studio Pastagames hat es entwickelt. Es ist ein *2D-Sidescroller*, im *Single-Player-Modus.* Der Grafikstil dieser Reihe ist im *Comicstyle* gehalten. Die Hauptfigur mit dem Namen Rayman muss durch neuen Level laufen und springen. Wenn ein Level mit einhundert Prozent eingesammelten *Lums* abgeschlossen ist, *Lums* sind eine Art Glühwürmchen, wird ein Bonuslevel freigeschaltet. Die Level Gestaltung ist an verschiedene Kategorien angelegt, wie z.B. einem Piratenschiff. Fähigkeiten der Hauptfigur sind unter anderem Springen, Fliegen und Klettern. Rayman rennt selbständig durch die Welt und erweitert seine Fähigkeiten im Laufe des Spiels. Er sammelt Münzen, überwindet Hindernisse und stellt sich Gegnern [ray]. Folgende Abbildung (Abb. 9) zeigt eine Spielszene aus Rayman Jungle Run.

Abb. 9: Zeigt Rayman und eine Münze einer Welt, [ray1]

2.5 Zusammenfassung

Die Gamebranche ist unendlich in Ihren Konzepten aber endlich in Ihren Möglichkeiten. In der heutigen Zeit lebt das Geschäft von der Konkurrenz, dadurch gibt es in der Entwicklerbranche einen florierenden Markt [ggm]. Darunter fallen z.B. große bekannte Firmen, Entwicklerteams und *Indie Game* Entwickler, sei es eine kleine Firma oder eine Einzelperson. Dies wird ermöglicht durch kostenlose Entwicklungsumgebungen wie bereits zuvor vorgestellt und anderen weiteren kostenpflichtigen Versionen. Eine Vielzahl an Programmiersprachen unterstützt die Entwickler bei der Umsetzung, sowie viele öffentliche Dokumentationen und eine Fülle an *Open Source* Code. Android ist das am häufigsten verkaufte Betriebssystem für Smartphones und der Play Store einer der umsatzstärksten. Die Spielszene von Smartphones ist eine der größten weltweit, besonders auf die Entwicklungsländer zurückzuführen. Im Jahr 2019 wird der globale Umsatz von Computerspielen mit 35.7 Billionen Dollar angegeben. Im Vergleich dazu liegt der Umsatz von mobilen Spielen bei 68.5 Billionen Dollar, davon 54.9 Billionen Dollar allein von Smartphones (Abb. 10). Eine der lukrativsten Gamebranchen in der Zukunft. Smartphone Applikationen erreichen nicht die grafische Leistung wie Konsolen oder Computer, jedoch ist so gut wie jedes bekannte Spielgenre auch hier vertreten, ob in 2D, 2.5D oder 3D. Zuvor wurden unterschiedliche *Side-Scoller* Genre für Android Applikationen vorgestellt und dessen Umsetzung. Es gibt zahlreiche Steuerungsmöglichkeiten der Hauptfiguren, diese sind immer individuell an das Spielkonzept angepasst. Das *Jump'n' Run* Genre gab es bereits vor den Smartphones. Dessen klassisches Wesen von dem Überwinden von Hindernissen, dem

Einsammeln von Objekten und dem Laufen und Springens durch eine Welt, ist in vielen heutigen Umsetzungen wiederzufinden. Im Folgenden Kreisdiagramm (Abb. 10) zeigt der blaue Anteil den Umsatz der mobilen Geräte.

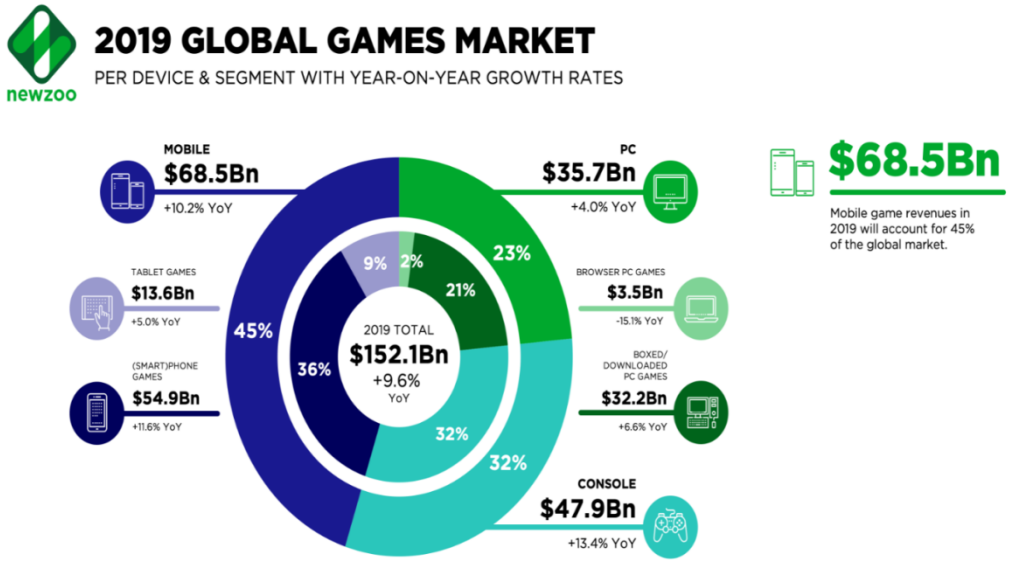

Abb. 10: Globaler Umsatz des Spielmarktes, [ggm]

3 Konzeption

Eine konkrete Aufgabenstellung wurde nicht festgelegt, da sich dieses im Laufe der Nachforschungen finden sollte. Zuerst gab es einige Ideen, wie das Abspeichern der vorigen gespielten Zeit und dem Modus seinen letzten Lauf vorauslaufen zu lassen. Die vier Elemente Erde, Feuer, Wasser und Wind waren von Anfang an ein bestimmender Bestandteil zur Umsetzung der Anwendung Die erste Idee Fähigkeiten dem Charakter zu geben die die vier Elemente widerspiegeln, habe ich verworfen, da es kein sinnvolles Konzept für eine Forschungsfrage darstellte. Zusätzlich musste ich ein Konzept finden welches diese nicht verliert und einbindet. Da ich besonders beim Erarbeiten und Bearbeiten von Grafiken mir immer wieder Gedanken zu Farben gemacht habe, hat sich mir die Frage gestellt, ob die Farbwahrnehmung die Spielwahrnehmung beeinflusst. Inspirationen zur Umsetzung habe ich mir aus aktueller *Gameart* und in Abschnitt 3 vorgestellter Spielanwendungen geholt.

3.1 Spielwelt

Die vier Elemente sollen jeweils durch eine eigene Szene dargestellt werden. Der Fokus der Arbeit liegt hierbei auf dem Darstellen der farblichen Kombinationen und den passenden Bildern zu den Elementen. Es sollen klassische *Jump ‘n‘ Run* Elemente in der Szene vorkommen, wie beispielsweise Plattformen unterschiedlichster Art in Ihrer Funktionalität, sowie abgestimmt auf die Elemente und dessen Farbe. Weiterhin soll es Münzen, Kisten, Gegner und Abgründe geben. Die Level sollen über gleiche Grundplattform Elemente verfügen, jedoch über individuelle Gestaltung der damit verbundenen Spielobjekte. Damit soll möglichst eine gleiche Umgebung geschaffen werden die einen ähnlichen Schwierigkeitsgrad hat, aber keinen beeinflussenden Lerneffekt erzeugt.

3.2 Hintergrund und Grafik

Die Anwendung soll über keinen statischen, sondern über einen dynamischen Hintergrund verfügen. Ursprünglich sollte ein animiertes Fundament gestaltet werden, jedoch war es im Hinblick auf die vier Elemente schwierig ein passendes Konzept umzusetzen. Inspiriert durch das Spiel „Alto's Adventure" sollen die Level über einen fortlaufenden Hintergrund verfügen. Dabei soll auf das *Parallax Scrolling* zurückgegriffen werden, um eine 3D Illusion zu erschaffen. Dies soll durch geschicktes Anordnen der Ebenen gewährleistet und ermöglicht werden. Der Spieler soll sich flexibel in alle Richtungen bewegen können ohne das der Spieler aus der Kamera läft. Die Grafiken des Hintergrundes sollen die Welten eines Elementes in der Natur darstellen. Des Weiteren soll gezielte Farbkombinationen, die Szenen und Objekt-Gestaltung komplementieren.

3.3 Musikauswahl und Menü

Die App soll über eine intuitive Menüführung verfügen und eine animierte Komponente beinhalten. Für die Spielszenen soll eine schnelle und mitreißende Musik und für das Menü eine ruhige Melodie gewählt werden. Es soll eine gleichbleibende Musik für die Spielszenen bestehen, damit die Musik nicht zusätzlich die Wahrnehmung der Applikation beeinflusst.

3.4 Steuerung und Kamera

Die Steuerung und Kamera sollen sehr ähnlich des Konzepts eines *Jump , 'n' Run* Spiels sein. Mit einer dynamischen Kamera die dem Spieler folgt und einem Joystick der für das Bewegen des Hauptcharakters zuständig ist. Der Spieler soll nach links wie rechts durch die Welt laufen und springen können.

3.5 Elemente und Effekte

Die Elemente sollen mit zahlreichen Objekten und dem Hintergrund in jedem Level individuell dargestellt werden. Der Fokus liegt dabei auf der Darstellung der Farbkombinationen. Zusätzlich soll eine Art Bonus für den Spieler zur Verfügung stehen, in Form eines einmaligen einzusammelnden Elementes. Der Hauptcharakter soll mit einem besonderen Effekt ausgestattet werden.

3.6 Hindernisse und Items

Die Szenen sollen über zahlreiche Hindernisse verfügen die den Spielspaß erhöhen. Items wie Münzen sollen durch das Einsammeln einen zusätzlichen Spaßfaktor generieren.

3.7 Schwierigkeitsgrad und Anzeige

Ziel ist es ein fortlaufendes Spielgefühl für den Spieler zu generieren, in dem er nahe seinem Tod generiert wird. Es soll verschiedene Arten der Folgen beim sterben mit dem Hauptcharakter geben und der Nutzer soll auf seine Geschicklichkeit angewiesen sein um die Level zu bstehen. Zusätzlich soll ein Lebenssystem für den Hauptcharakter implementiert werden. Auch eine Anzeige für die Anzahl der gesammelten Münzen ist gewünscht.

3.8 Forschungsfrage

Im Laufe der Arbeit hat sich folgende Forschungsfrage heraus entwickelt:- „Inwiefern beeinflussen Farbkombinationen im Spiel eines Jump & Run Games, in Form der vier auftretenden Elemente, die menschliche Wahrnehmung und deren Auswirkungen auf das individuelle Spielgefühl.“ Diese soll später mit Hilfe eines Nutzertests bewertet werde.

4 Umsetzungen

Im Folgenden wird die Umsetzung in Unity vorgestellt.

4.1 Skripte

Es wurde ein Vielzahl an Skripten verwendet die später erwähnt werden.

4.2 Szenenhierarchie Objekte

Im Folgenden werden *Gameobjects* vorgestellt, die als organisatorischer Vaterknoten in der Szenenhierarchie dienen. Die Szene Erde, Feuer, Wasser und Wind besitzen jeweils diese Objekte in ihrer Struktur.

4.2.1 OverviewManager

Der *OverviewManager* ist ein leeres *GameObject* und hat als Kindsknoten wichtige für den Spieler unsichtbare organisatorische Objekte. Darunter fallen mehrere *Checkpoints, HiddenWalls* und *Killzones*. Darauf wird genauer eingegangen in Abschnitt 5.3.

4.2.2 CanvasManager

Der *CanvasManager* ist ebenfalls ein leeres GameObject und enthält als Unterpunkte eine *SymbolCanvas,* eine *PauseCanvas* und eine *ControlCanvas*; Genauer erläutert in Abschnitt 5.4.

4.2.3 LevelManager

Analog zu obigen ein leeres *Gameobject* mit Kindsknoten *LevelManger* und *LevelManger 2*. Diese sind für die *Canvas* - Verwaltung zuständig und werden in Kapitel 5.5 behandelt.

4.3 Nicht sichtbare Objekte

4.3.1 Kontrollpunkt

Der *Checkpoint* ist ein *GameObject* mit einem *Box Collider 2D*, dessen Variable *Is Trigger* auf *true* gesetzt ist, damit der Charakter ohne Kollision der *Collider,* hindurchlaufen kann. Weiterhin hat es ein *Checkpoint* Skript, in dem der *LevelManager* der Szene der Variable zugeordnet ist. Dem *LevelManager* ist der erste Checkpoint der Szene zugeordnet, durch diese Kombination kann der Spieler immer zu dem aktuellen Checkpoint wiederbelebt werden. Insgesamt existieren in der Szene Erde zwanzig Checkpoints, bei Feuer zweiundzwanzig, in Wasser zwanzig und bei Luft sechsten Stück. Diese sind in den jeweiligen Szenen so platziert, dass der Spieler möglichst wenige Frustgefühle entwickelt und ein kontinuierliches Spielgefühl entwickeln kann. Weiterhin ist die Platzierung auf die Schwierigkeit der Level angepasst, wie dessen unterschiedlichen Spielanlage. Die Checkpoints werden nur beim herausfallen aus der Welt und beim Kollidieren mit Stacheln aktiviert. Die Folgende Abbildung X zeigt dessen Platzierung in Form der grünen Balken in der Szene Luft.

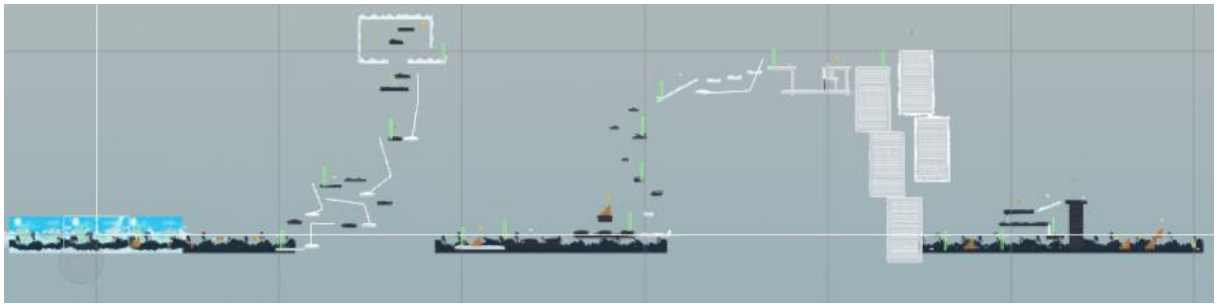

Abb. 11: Grüne Checkpoints in der Welt Luft.

4.3.2 Todeszone

Die *Killzone* ist eine *GameObject* mit einem *Box Collider 2D* und einem *Neustart* Skript. Sobald der Hauptcharakter mit Dieser kollidiert, wird über das *Neustart* Skript auf das *LevelManager* Skript zugegriffen und dessen Methode *RespwanPlayer()* ausgeführt. Wir oben beschrieben ist dem *LevelManger* ein *Checkpoint* aus der Szene hinzugefügt. Sobald der Hauptcharakter einen anderen Checkpoint erreicht wird dieser als neuer gespeichert. Die Todeszonen dienen als Abfang-Mechanismus für den Hauptcharakter in der Welt. Ohne die platzierten Todeszonen würde der Spieler unendlich nach unten durch die Szene fallen. Es sind insgesamt drei pro Spielszene eingefügt.

4.3.3 Nicht sichtbare Wand

Die *HiddenWall* ist ein *GameObject* mit einem *Box Collider 2D*. Diese dient zum Begrenzen des Spielbereichs. Dadurch existieren keine ungewollten Wege oder mögliche *Bugs* in der Spielwelt und der Spieler kann sicher dem vorgegeben Weg folgen. Das Folgende Bild (Abb.) zeigt die Organisation der drei obigen Kapitel aus der Sicht der Szene Wasser.

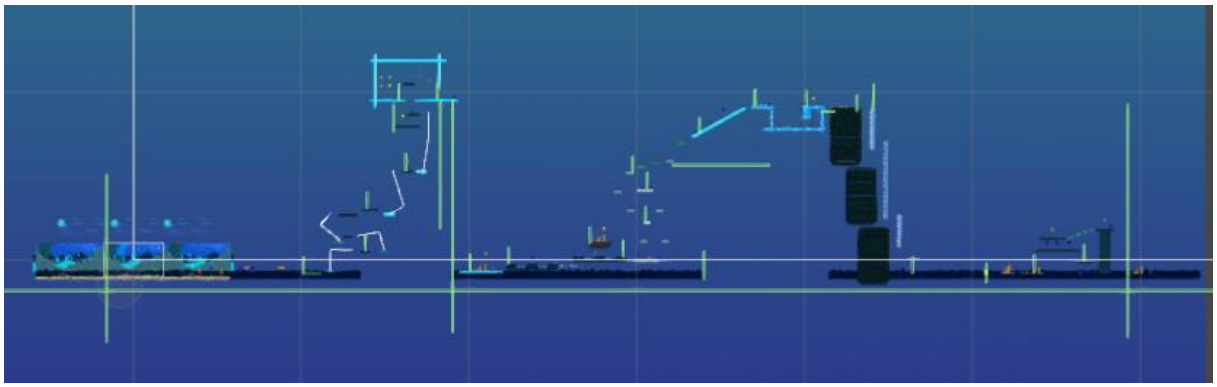

Abb. 12: Organisationsstruktur der Szene Wasser von „Nicht sichtbare Objekte“

4.4 Canvas in der Szene

Es gibt in den vier Spielszenen jeweils den *CanvasManager* der als Übersicht für die drei wichtigen Arbeitsflächen dient (vgl. Abschnitt 5.2.2).

4.4.1 Informationsarbeitsfläche:

Dem Nutzer wird eine aufgeräumte und auf das Spiel abgestimmte Informationsarbeitsfläche angezeigt. Dazu wird eine Userinterface *Canvas* aus Unity verwendet. Diese ist Bildschirmauflösung unabhängig und besteht aus sieben weiteren Kindsknoten. Die *SymbolCanvas* dient dem Spieler zur Informationenerkennung der gesammelten Münzen, der Anzahl seiner Leben, seines erspielten *Highscores* und der Anzeige des zu sammelnden Spezialitems. Zu jeder *Canvas* wird automatisch ein dazugehöriges *EventSystem Object* generiert, welcher zur Erkennung von Berührungen des Spielers auf dem Bildschirm zuständig ist.

Der Informationsarbeitsfläche gehören insgesamt vier UI Text und drei UI Image Objekte als Kindknoten an. Eine Grafische Lösung ist für das Anzeigen des Lebens, der Münze und des speziellen Element-Items gewählt. Die drei Image Objekte repräsentieren die Lebensanzeige, die Münzenanzeige und die Element - Item - Anzeige, jeweils durch ein passendes hinzugefügtes Quellbild. Die vier Text Objekte sind für das Anzeigen der jeweiligen Punktestände und den *Highscore* zuständig. Des Weiteren sind diese mit dem *LevelManger* und dem *LevelManger2* in der jeweiligen Szene verknüpft. Dadurch kann mittels Skript eine Aktualisierung bei bestimmten Events ausgelöst werden. Die *Canvas* ist so angeordnet, dass der Spieler eine gute Übersicht hat. Die Schriftart *Adventure* ist der *Font* Variable des *Text (Scripts)* zugeordnet und repräsentiert eine spielerische Darstellung.

4.4.2 Pausenarbeitsfläche

Um den Spielfluss nicht durch einen Szenenwechsel zu beeinflussen, ist die *PauseCanvas* keine eigene Szene. Zusätzlich müssten die Szenen immer neu generiert und geladen werden, dass als nicht sinnvoll erscheint. Das Pausenmenü wird während des Spielens interaktiv vom Nutzer ausgeführt und vermittelt den Eindruck nicht die Spielewelt zu verlassen. Dies wurde ebenfalls mit einer *UI Canvas* umgesetzt. Die *Canvas* besitzt ein *UI Button* als Kindknoten, welcher in der jeweiligen Spielszene zu sehen ist. Dieser ist unten Rechts in der Arbeitsfläche angeordnet ist. Dargestellt mit dem Quellbild für Buttons und dem internationalen Pausenzeichen von zwei Strichen. Dem Button ist das Pausenmanager Skript angehangen und auf die *panelPause* aus der aktuellen Szene gezeigt, dem anzuzeigenden Pausenmenü. Mit der Methode *On Click* wird auf die Methode *PauseControle* des

Pausemanager Skript zugegriffen. Wenn der Spieler den Pausenknopf drückt, wird die Zeit der Spielszene angehalten und das Pausenmenü angezeigt. Der Shader des anzuzeigendes Menüs ist am Anfang der Spielszene ausgeschaltet und wird beim aktivieren eingeschaltet.

PanelPause ist ebenfalls ein Kindknoten der *PauseCanvas* und beinhaltet die komplette Struktur des anzuzeigenden Menüs. Ein *UI Image* ist mit dem Hauptmenübild ausgestattet, diesem UI sind vier weitere Kindknoten untergeordnet, drei *UI Button* und einem *UI Toggle Element*. Das Hauptmenübild ist leicht transparent eingestellt, damit die aktuelle Spielszene und die darin liegenden Objekte noch leicht zu erkennen sind. Damit soll der Eindruck im Level zu verweilen erzeugt werden. Mit dem *UI Toggel* kann der Spieler die Spielszenenmusik an und ausschalten, dargestellt durch zwei Lautsprecherbilder. Der *Button „*Weiter“ greift auf die Methode *PauseControle* des *Pausenmanager* Skript zu, deaktiviert das *panelPause* und aktiviert die Zeit der Spielszene wieder. Der *Button* „Neustart“ greift auf die Methode *Restart* zu und lädt die aktuelle Spielszene neu. Der *Button* „Verlassen“ ist mit dem *Levelauswahl* Skript ausgestattet und führt beim Aktivieren die Methode *LoadLevel* aus. Damit kommt der Spieler wider zurück zum Hauptmenü.

4.4.3 Steuerungsarbeitsfläche

Für den *Joystick* wird ebenfalls auf eine *UI Canvas* zurückgegriffen. Die *ControlCanvas* hat ein *UI Image* als Kindknoten mit dem äußeren Ring als Bild des Joysticks. Dieser Knoten hat einen Weiteren Kindknoten ein *UI Button* der als Bild den inneren Kreis des Joysticks besitzt. Diesem ist zusätzlich das Skript Joystick Move hinzugefügt und darin ein Radius von 90 eingestellt, in dem der Spieler den Kresi bewegen kann. Der *Joystick* ermöglicht es dem Spieler durch bewegen nach links und rechts, das Laufen des Hauptcharakters. Durch das Bewegen nach oben, wird der Sprung ausgeführt ab einem eingestellten Y Wert. Wird diese Höhe überschritten löst es den Sprung aus, damit wurde ein angenehmes Maß gefunden um das Springen nicht zu früh oder zu spät auszulösen und es dem Spieler zugleich ermöglicht auch beim Laufen, wie Stehen zu springen. Erstere der Folgenden Grafiken zeigt die Arbeitsflächen in der Szene Wind wie Sie der Spieler sieht und die Zweite mit einem aktivierten Pausenmenü aus dem Level Wasser.

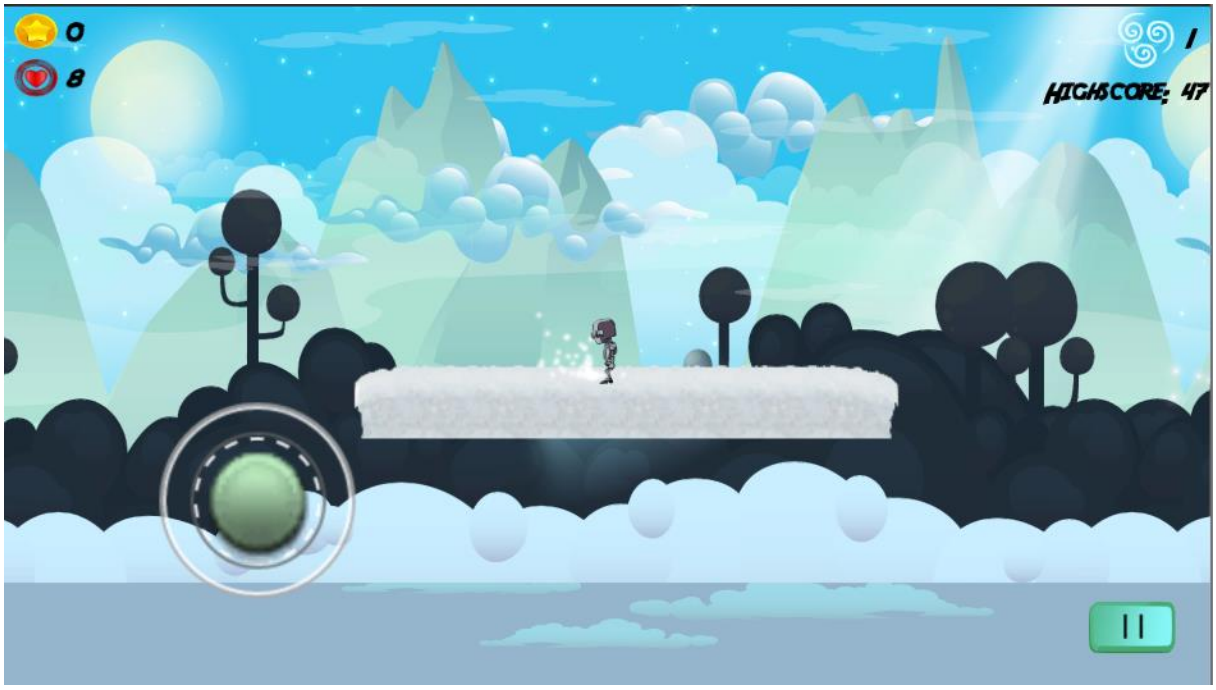

Abb. 13: Alle Arbeitsflächen sichtbar in de Szene Wind.

Abb. 14: Aktiviertes Pausenmenü in der Welt Wasser

4.5 Hauptmenü

Für das Hauptmenü wurde eine eigne Szene in der Assets Ordner Struktur von Unity erstellt. Die Szenen Hierarchie besteht aus mehreren Elementen. Zusätzlich zur gegeben *Main Camera*, sind zwei weiter GameObject hinzugefügt (Abb. 15). Darunter fällt der *AudioManager* zum verwalten der Hauptmenü Musik, dafür wird dem Objekt eine Audio Source Komponente hinzugefügt. Die Musikauswahl fiel hierfür in eine ruhige und entspannte Richtung. Die Variablen *Play on Awake* und *Loop* sind auf *true* gesetzt. Dadurch aktiviert sich die Musik direkt beim starten der Szene und wiederholt sich.

Zwei Sprung-Animationen des Hauptcharakters erzeugen eine angenehm passive Interaktion der Szene mit dem Spieler. Die interaktiven Elemente der Szene sind mit dem Benutzerinterface von Unity erstellt, aufgrund der schon vorher genannten Touchscreen Funktion und der Unabhängigkeit der Skalierung der Oberflächenansicht. Die Erstellung erfolgt ähnlich dem des vorgestellten Shemas aus Abschnitt 5.4.2.Das Menü ist mit zwei Skripten ausgestattet, dem *Quitdurchklick* und *Startdurchklicken*. Die *Button* „Levels“ und „Regeln“ besitzen ein *Levelauswahl* Script mit jeweils einer Variable, in der die zu laden Szene geschrieben ist. Level führt zur *Levelsuswahl* Szene und Regeln zur Regelszene. Der „Start“ Button greift auf die Methode *LoadLevel()* zu und startet das erste Level Erde. Der „Ende“ Button greift auf die Methode *Quit()* zu und beendet die Applikation. Dem Menü Objekt ist *ein Vertical Layout Group (Script)* hinzugefügt dies koordiniert die Anordnung der Kindknoten und dessen Ausrichtung. Es wurde ein Abstand von 20 in jede Richtung gewählt und die Ausrichtung *Middle Center* (Abb. 15).

Ebenfalls ist der *Canvas* ein *UI Toggel* hinzugefügt, dies besitz ein Kindknoten *Background* und dieses ein weiteren Kindknotent *Checkmark*. Im *Inspector* ist dem Background Objekt das Off Symbol hinzugefügt und dem *Checkmark* Objekt das On Symbol. Das *Toggel* Objekt besitzt ein *Toggel (Script)*, die Transition ist auf *Sprite Swap* gestellt. Der *Pressed Sprite* Variable ist das Off Symbol

zugewiesen, die *Toggel* Varaible *Is On* ist auf True gesetzt. Die Methode *On Value Changed* greift auf den *AudiManager* zu und setzt den *bool* Wert auf *enable.* Abbildung 17 zeigt das komplette Menü.

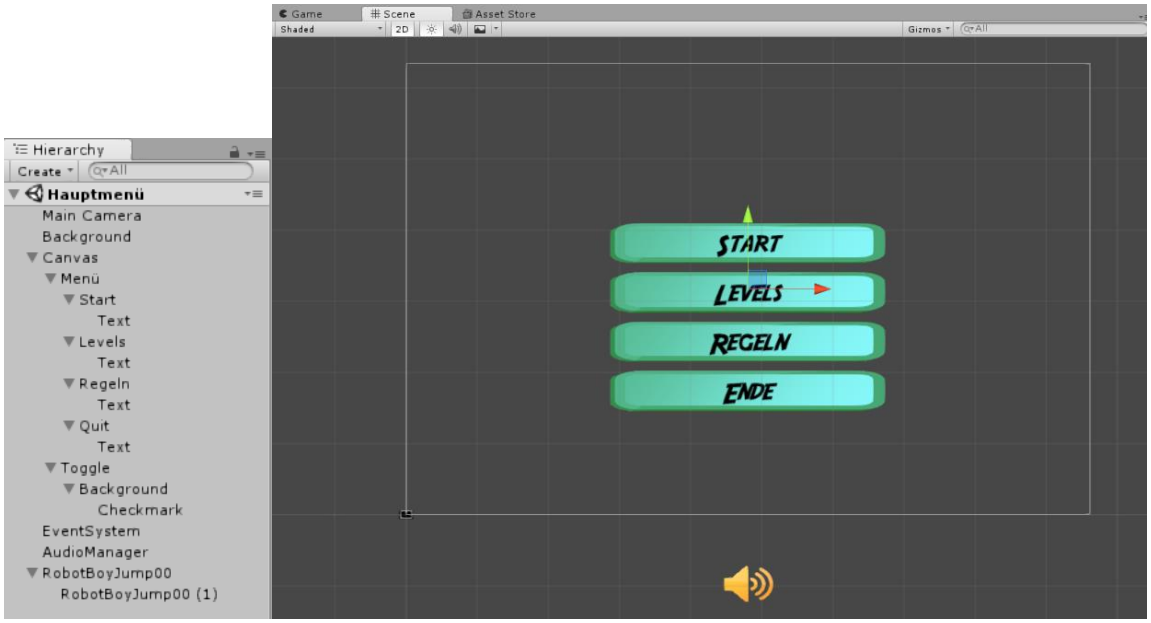

Abb. 15: Struktur

Abb.16: Canvas

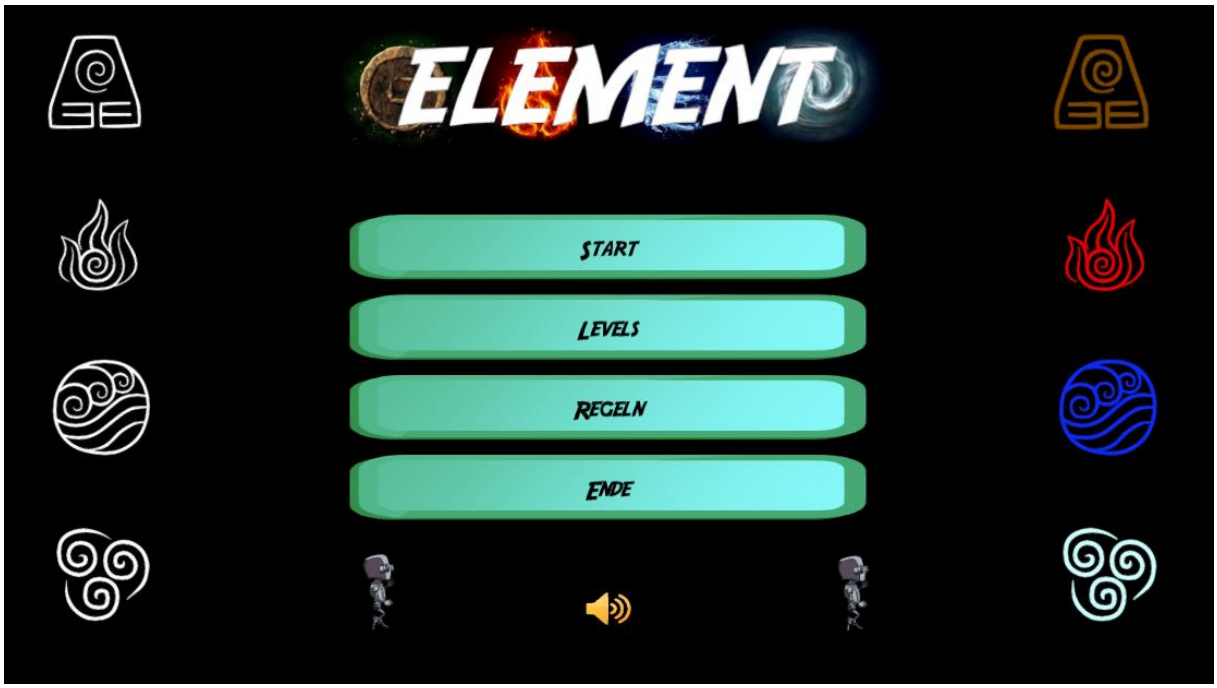

Abb. 17: Hauptmenu Ansicht des Nutzers

4.5.1 Levelauswahl

Ich verweise auf obige Abschnitte. Auch hier wird eine *UI Canvas* eingesetzt mit schon beschrieben Gründen, Skripten und Methoden. Diese Szene ist eine eigne und dient dem Spieler dazu, bei unerwartetem beendet der Applikation, das aktuelle Level wieder laden zu können. Auch hier die im Panel verwende Ankerpunkete sind passend skaliert und bieten ausreichend Abstand zum Bildschirm. Hier bekommt das Menü *GameObject* ein Layout, diesmal ein *Grid Layout Group (Script)* hinzugefügt. Damit ist auch hier ein spezifische Anordnung möglich und die automatische Skalierung das ganzen Menüs und dessen Kindknoten, bei Änderung der Bildschirmauflösung ist kein Problem. Das Folgende Bild (Abb.18) zeigt, die vier Level Auswahlmöglichkeiten mit ausgeschaltetem Sound.

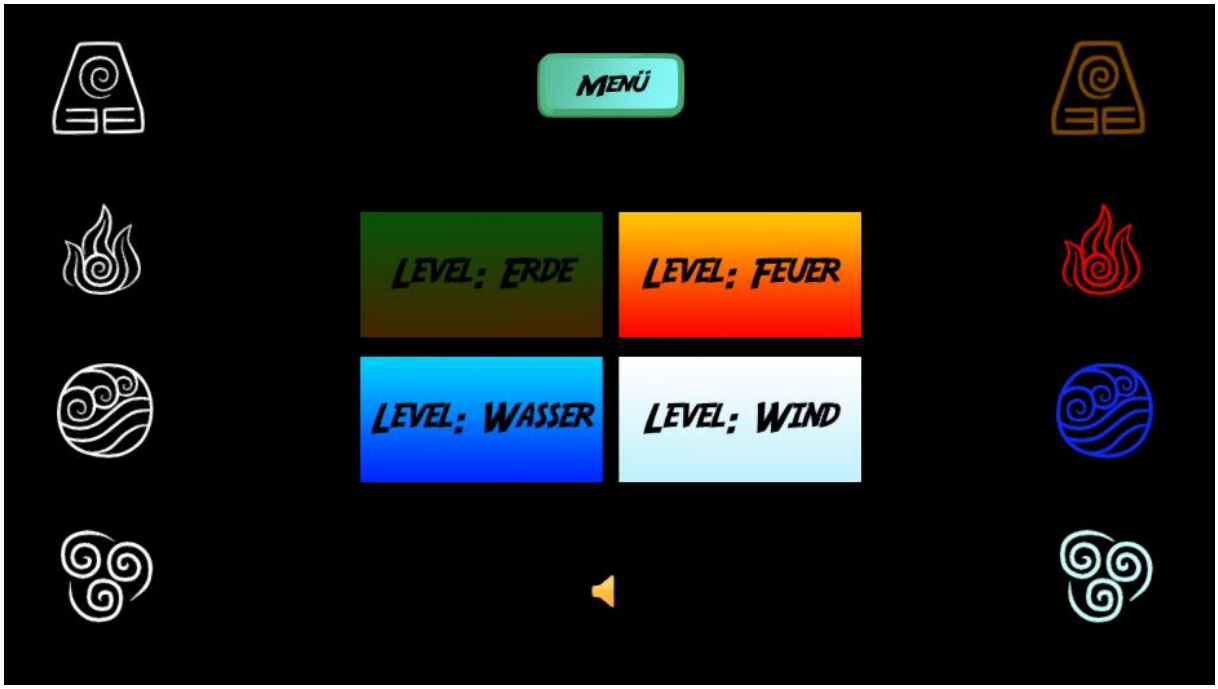

Abb. 18: Levelauswahlmenü

4.5.2 Levelende und Regel Szene

Für beide Szenen ist das obige aufgeführte Schema angewandt. Bei der *Levelende* Szene sind diesmal für die passive Interaktion die Animation Sterben und Rollen des Hauptcharakters zuständig. Es gibt nur einen Button welcher zurück zum Hauptmenü führt, analog bei der Szene Regeln. Die nächsten zwei Abbildungen zeigen dessen Aufbau.

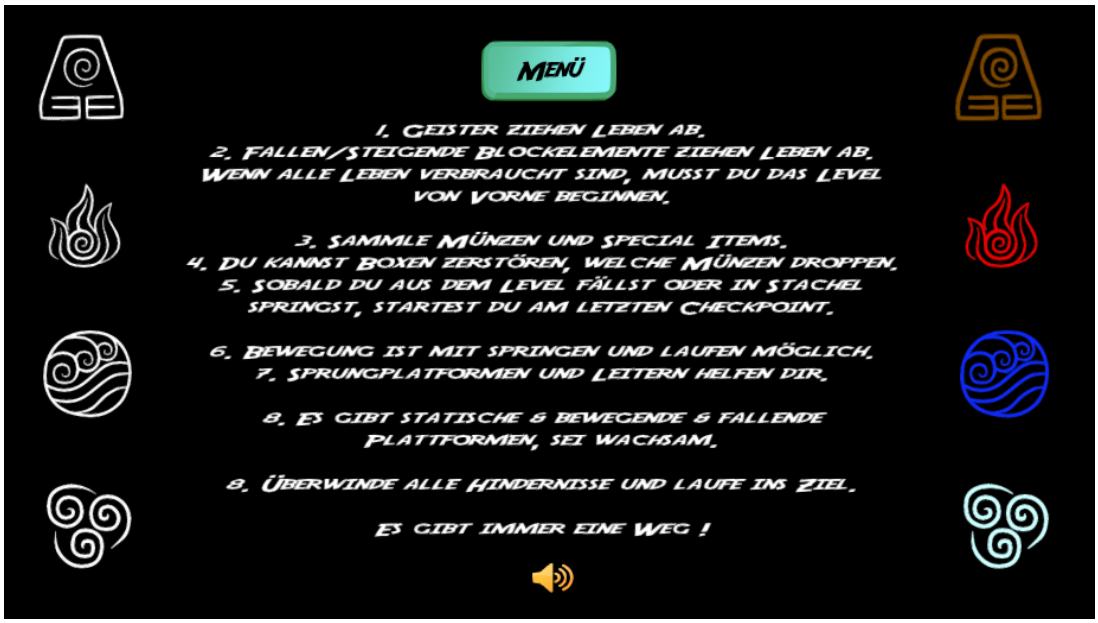

Abb. 19: Regeln

Abb. 20: Levelende

4.6 Hauptkamera der Szenen

Beim Erstellen einer Szene ist die Kamera unabhängig von Objekten. Damit der Spieler nicht verschwindet beim verlassen des Kamerabereichs, agiert der Hauptcharakter als Mittelpunkt der Hauptkamera. Der *Main Camera* ist ein *Camera2DFollow* Skript hinzugefügt, dieses enthält eine Variable Target, der der *CharacterRobotBoy* zugeordnet ist. Es wurde der Weitsichtfaktor angepasst um ein besseres Spielgefühl zu kreieren. Der Kamerabereich bewegt etwas verstärkt Voraus in die Richtung in der der Spieler läuft. Dafür ist die Variable *Look Ahead Factor* auf 15 gesetzt und der *Return Speed* auf 10. Beim Nichtbewegen zentriert sich die Kamera auf den Charakter. Die Darstellung der *Main Camera* wurde Anfangs auf Orthographic gestellt, denn dadurch werden auch 3D *GameObject* in 2D dargestellt. Die Folgenden zwei Grafiken zeigen zwei der vier Welten und dessen Größe und Aufbau (Abb. 21, Abb. 22). Analog existiert dies für Feuer und Wind.

Abb. 21 Wasserwelt

Abb. 22: Erdwelt

4.7 LevelManager

Der erste *LeveMmanager* ist für das Aktualisieren der Münzen, des *Highscores* und der des Speziellen Items verantwortlich. Dafür wird das *LevelManager* Skript verwendet. Der LevelManager2 ist für das Lebenssystem des Spielersverantwortlich und aktualisiert dessen Lebensanzeige. Beide dienen zur besseren Organisation des Spiels.

4.8 Überbegriff Objekte:

Münze Luft

Eine Münze besteht aus zwei Animationen. Die erste Animation ist eine Rotation der Münze um die Y Achse danach wechselt sie in die Zweite Animation in der die Münze glänzt. Für die erste Animation wurde mit sechs Quellbildern eine Animation erstellt, analog für die zweite Animation. Im *Inspector* wurde der Münze ein *Coin* Tag hinzugefügt. Der die Münze spezifischer beschreibt. Des Weiteren wurde ein *Circel Collider 2D* hinzugefügt und die Variable *Is Trigger* auf *true* gesetzt, damit der Spieler hindurch springen kann und die Münze trotzdem erkannt wird. Eine Münze besitzt eine *Audio Source,* die beim einsammeln des Spielers erzeugt wird, zusätzlich wird das GameObject der Münze zerstört.

Münze Physik

Ich verweise auf den obigen Abschnitt diese Münze hat zusätzlich einen *Box Collider 2D* der innerhalb des anderen *Collider* liegt. Da der Hauptcharakter an diesem hängen bleiben kann.Zusätzlich ist ein *Rigidbody 2D* Komponente hinzugefügt, um der Münze eine Schwerkraft zu verleihen. Mit dem Skript *Box* wird ein Prefab dieser Münze entsprechend der gewählten Anzahl erzeugt und noch oben in unterschiedlichste Richtungen beschleunigt. Dann verliert Sie ihre Kraft und fällt auf den Boden.

Kiste

Eine Kiste ist ein *GameObject* mit einer Bildquelle und einem *Box Collider 2D.* Sie dient als Hindernis und ist in vielen Konstruktionen vorhanden.

Holzkiste mit Münzenfunktion

Andere Bildquelle und Script Box hinzugefügt. Anzahl der erzeugten Münzen eins. Effekt Rauch Erdexplosion hinzugefügt. Beim kollidieren mit dem Gamer das ist der Tag der Hauptspielfigur wird die Animation ausgelöst, später zerstört und das Prefab einer Münze in die Szene erzeugt.

Goldkiste mit Münzenfunktion

Analog zu oben mit einer anderen Bildquelle und der Anzahl zu erzeugenden Münzen von drei.

Special Items

Das ist in jedem Level eine Animation, aus vier Bildern in jeweils einer anderen 90° Position, die das Element darstellt. Dafür wurde jeweils eine eigene Sounddatei zugeschnitten die den Sound des jeweiligen Elements beim aufsammeln abspielt.

Plattform

Diese ist mit einem Box Collider 2D Ausgestattet und der Tag ist auf *Gras* gesetzt. Das Material Slippery ist bei jeder Platform hinzugefügt, damit der Spieler nicht darin hängen bleibt.

Fallende Plattform

Analoger Aufbau zu oben zusätzlich einen *Rigidbody 2D.* Damit bekommt die Plattform eine Kraft die diese nach unten zieht die Bewgung nach X und Z ist deaktiviert.

Bewegende Plattform

- Es werden Game Objekte in der Szene erstellt die später als Punkte für den zu zeichnenden Pfad dienen. Das *Path Definition* Skript dort werden die Objekte verwendet und die Anzahl kann eingestellt werden.

- Eine zuständige Plattform bekommt das *Follow Path* Skript und der Pfad wird diesem Skript hinzugefügt. Zusätzlich kann die Geschwindigkeit eingestellt werden.

- Zum lösen der Physik und dem nicht halten der Hauptfigur auf der Plattform beim bewegen, wird der Plattform ein Kindknoten geben mit dem Plattform Holder Skript. Welches den Spieler mit der Plattform bindet, dafür muss noch der *Box Collider 2D* des *Holders* also des Kindknoten überden *Collider* der Plattform gebracht werden.

Leiter

Game Objekt mit einem *Ladder Script* und einem *Box Collider 2D* welcher auf *Is Trigger* gestellt ist. Der Spieler kann über die Leiter zu einer höheren Plattform gelangen, die Klettergeschwindigkeit ist auf vier gestellt.

Springboard

Das Springboard ist eine Animation von zwei Bildern. Es besitzt einen *Box Collider 2D* und dessen Material ist der *bounce* Effekt hinzugefügt. Dieser ist auf eine angepasste maximale Höhe eingestellt, dass der Spieler nicht unendlich nach oben Springen kann.

Geist

Hat einen *Box Collider 2D* und das Gegner Skript. Die erstellen Gegner *Prefabs* bewegen sich alle unterschiedlich, da ihre Bewegung mittels einem Intervall umgesetzt ist. Das Gesicht des Gegners wird entsprechend der Laufrichtung gespiegelt. Jeder Kontakt zieht der Hauptfigur ein Leben von insgesamt acht ab, weiterhin kann der Gegner erst wieder nach zwei Sekunden Schaden veruraschen.

Spikes

Grafik mit einem *Box Collider 2D* und dem *Neustart* Skript. Kollidiert der Spieler mit den *Spikes* wird er am letzten Checkpoint wiederbelebt.

Schilder

Helfen den Weg genau zu erkennen in der Szene.

Levelende

Das Levelende ist durch ein goldenes Tor dargestellt und hat einen *Box Collider 2D* mit dem Skript *Neues Level*. Welches beim erreichen den nächsten Index aus den Build Settings lädt.

Build Setting

Dort müssen alle Szenen geordnet eingefügt werden, damit Funktionen auf die Szenen zugreifen können und später das Spiel exportiert werden kann. Die Abbildung 24 zeigt alle Szenen.

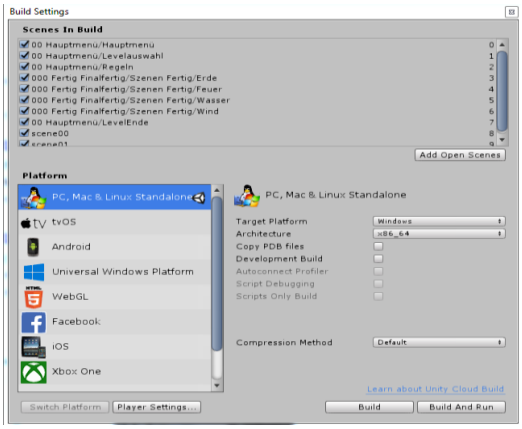

Abb. 23: Build Index von Unity

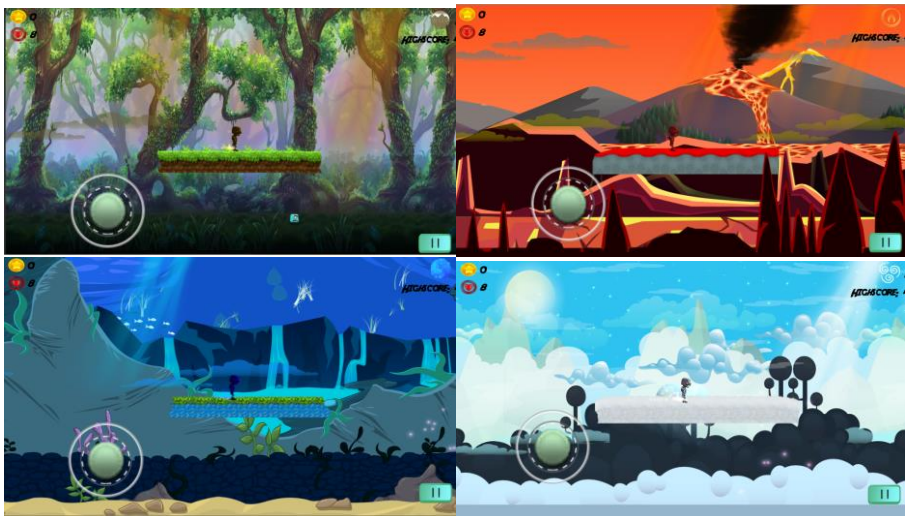

Abb. 24: Elemente als Szenen

Hauptcharakter

Der Hauptcharakter Robotboy verfügt über einen Animator bei dem alle Bewegungen und dessen Animationen verknüpft sind, Siehe Bild.

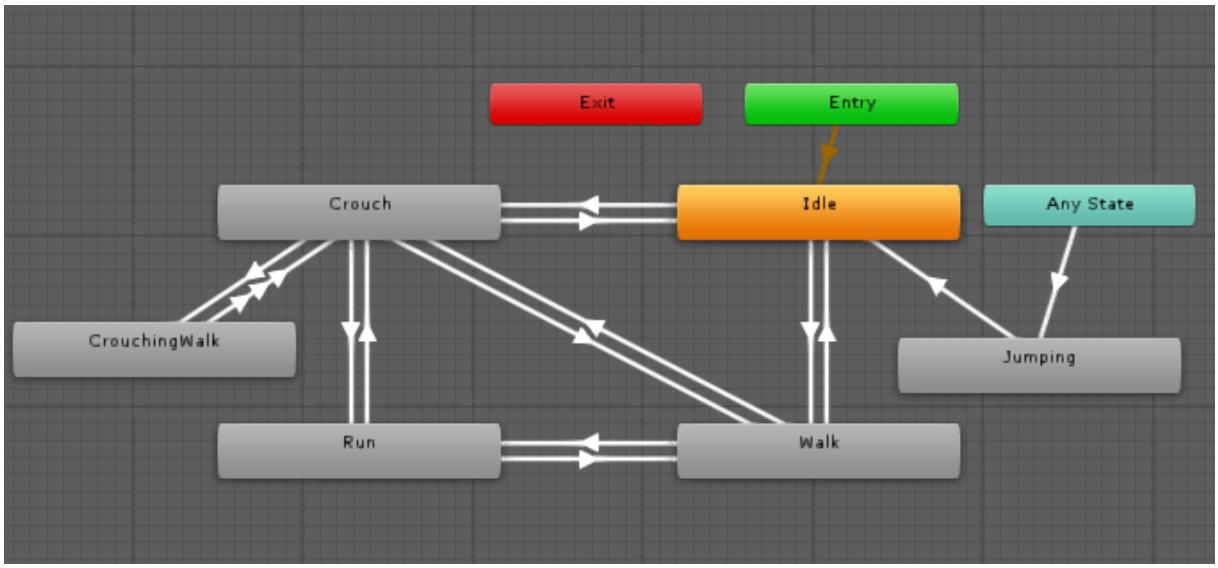

Abb. 25 Animator Charakter Robotboy

Der Charakter verfügt über zahlreiche Skripte, dem *Platformer Charakter2D*, darin ist die Laufgeschwindigkeit und die maximal Sprunghöhe festgelegt. Dem *Platform 2D User Controle* hauptsächlich für die Bewegung zuständig. Dem Kollision Effekt Script dort ist in jeder Welt eine farblich abgeglichene Partikel System, welches Mithilfe von Kraft eine Art glühende Bällchen beim Laufen auf Plattformen Hinterlässt. (*Tag Gras*)

Da mit Licht gespielt wird und manche Plattformen und Objekte ein Dunkles Material bekommen haben. Besitzt der Hauptcharakter eine Punkt Lichtquelle die die verdunkelten Elemente erscheinen lässt.

PartikelEffekte

Sind umgesetzt als fliegende Schmetterlinge und fallender Goldstaub in multipler An zahl in der Szene.

Parallax Hintergrund

Die Szenen Erde, Feuer, Wasser und Wind verfügen über eine individuelle Grafikensammlung, mit Farbkombinationen aus dem jeweiligen Element. Die Grafiken zeigen Elemente aus der Welt um diese darzustellen. Parallaxe und feststehende Grafiken sind an die *Main Camera* gebunden. Der komplette Hintergrund mit allen Grafiken wird in unterschiedliche Layer geordnet und dem Vaterknoten das *Parallax* Skript gegeben. Darin kann die Variable *ou* eingestellt werden und er Hintergrund dupliziert sich auch nach oben. Andernfalls dupliziert er sich nur der Y Achse. Der Vaterknoten einer Grafik wird nach links und rechts verdoppelt und diesem als Kindknoten angeordnet. Die Hauptkamera ist an das Skript gehaftet und der *Parallax* Effekt dort eingestellt mit geschwindigkeit. Siehe Abbildung 26 und 27 der Elemente Feuer und Wind.

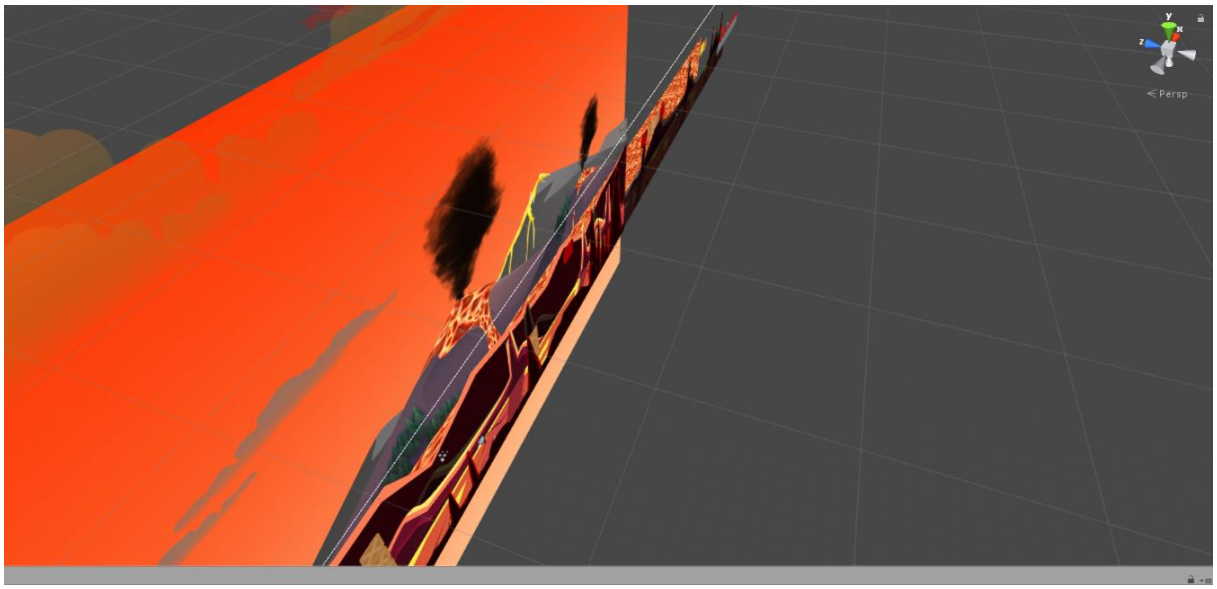

Abb. 26: Parallaxe Feuer

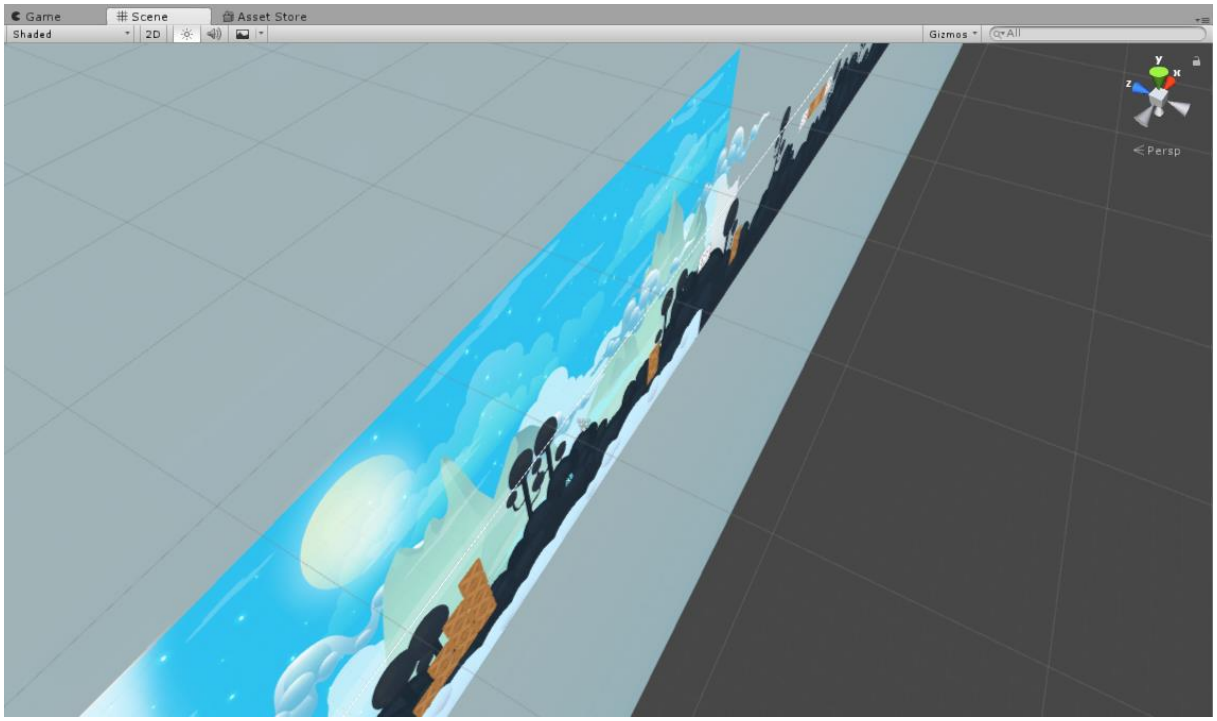

Abb. 27: Parallaxe Wind

5 Analyse

Zur Evaluation der Applikation und der Analyse der Forschungsfrage dient ein Nutzertest in Form eines Fragebogens (siehe Anhang). Einerseits beziehen sich die Fragen, auf die Benutzerfreundlichkeit, die Mechanik und die Optik des *Jump 'n' Run* Spiels. Der andere Teil bezieht sich auf die Farbwahrnehmung und Spielwahrnehmung des Nutzers.

5.1 Testsystem

Die Anwendung wurde auf einem Smartphone Samsung Galaxy S7 getestet. Die Displaygröße des Testhandys beträgt 5,1 Zoll mit 2.560 x 1.440 Pixel. Einem Octa-Core Prozessor (4 x 2,3 GHz; 4x 1,6 GHz der taktet) und 4GB Arbeitsspeicher. Weiterhin ist das Betriebssystem Android installiert in der Version *8.0 Oreo*. Die Testdatei .apk von Unity ist dem Arbeitsspeicher des Handys hinzugefügt, die Applikation „APK Installer" ist installiert und darin das Spiel.

5.2 Aufbau des Nutzertest

Der Fragebogen besteht aus einer Information in dem die Testperson alle wichtigen Informationen nachlesen kann. Er ist in sechs Sinnesabschnitte aufgeteilt. Insgesamt besteht er aus sieben Offenen Fragetypen und einundzwanzig geschlossenen Fragetypen. Es sind vorgegeben Antwortmöglichkeiten gegeben und Bewertungsskalensysteme mit unterschiedlichen Gewichtungen.

5.3 Durchführung des Nutzertest

Die Tester bekommen zuerst eine mündlich Einführung und werden im Anschluss daran gebeten, sich die Informationen des Nutzertestes durchzulesen. Im Anschluss wird das Handy der Testperson übergeben. Nach dem starten der App sollen zuerst die Regeln gelesen werden, bevor das erste Level gespielt wird. Bei auftretenden Fragen stehe ich den Testern am Anfang zu Verfügung.

5.4 Darstellung der Ergebnisse

5.4.1 Benutzerfreundlichkeit

Die Probanden haben die Applikation mit 3.6 Sternen im Durschnitt bewertet. Der Schwierigkeitsgrad ist mit fünfmal als *angemessen* und fünfmal als *völlig angemessen* beurteilt. Die Spieldauer eines Level ist von „eins nicht angemessen" bis „fünf völlig angemessen" mit 4.5 im Durschnitt abgeschlossen.

1. Wie bewerten Sie den Spaßfaktor des Spiels?

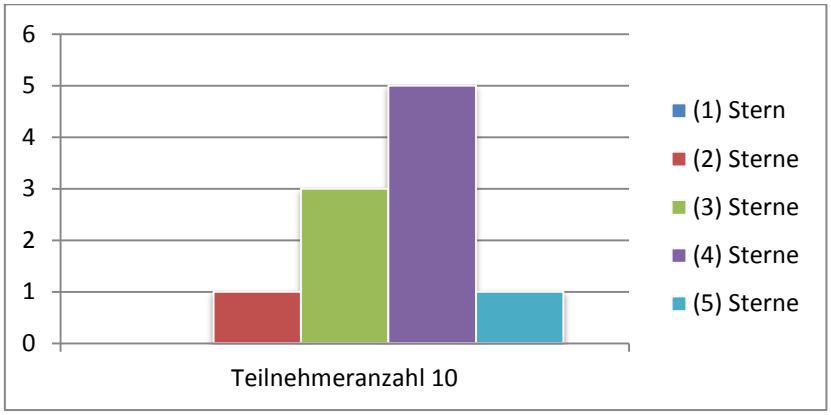

Abb. 28: Auswertung zu Frage 1

2. Wie bewerten Sie den Schwierigkeitsgrad des Spiels?

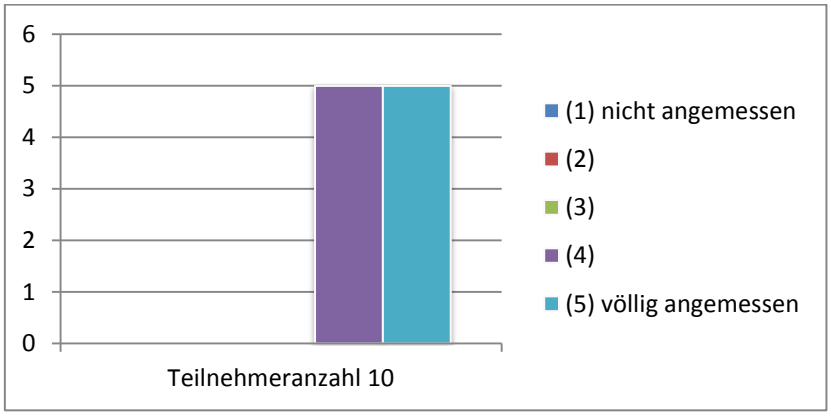

Abb. 29: Auswertung zu Frage 2

3. Wie bewerten Sie die Dauer eines Level?

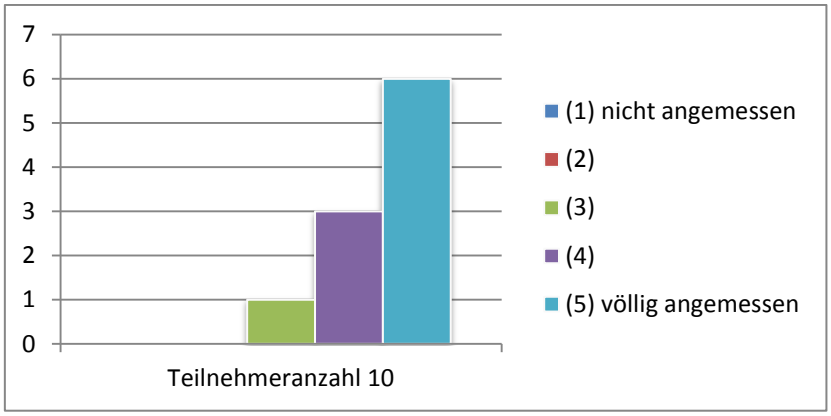

Abb. 30: Auswertung zu Frage 3

5.4.2 Mechanik

Die Menüführung ist von 60 Prozent der Teilnehmer als *intuitive* und mit 40 Prozent als *völlig intuitive* bewertet. Weiterhin hat ein Tester die Steuerbarkeit des Spiels als *weniger intuitiv* und neun Testpersonen als *neutral* wahrgenommen. Der unterschiedliche Aufbau der Level und dessen Einfluss auf das Spielgefühl ist von eins „trifft nicht zu“ bis fünf „trifft völlig zu“ von zehn Prozent der Tester als *nicht zu treffend*, von fünfzig Prozent als *neutral* und von vierzig Prozent als *zutreffend eingeordnet*.

4. Wie bewerten Sie die Menüführung des Spiels?

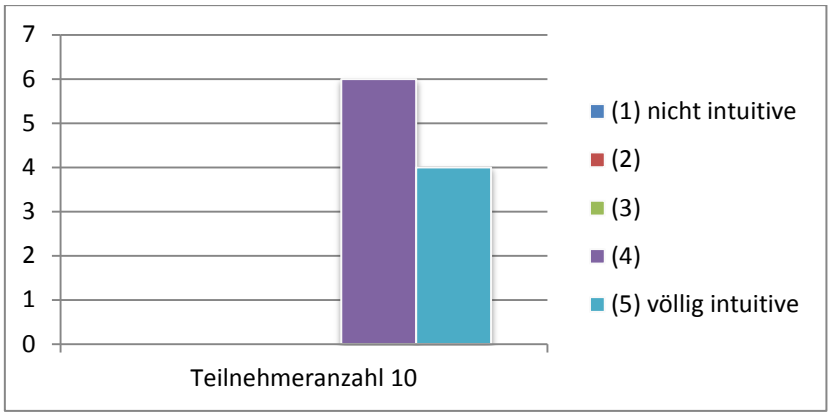

Abb. 31: Auswertung zu Frage 4

5. Wie bewerten Sie die Steuerbarkeit des Spiels?

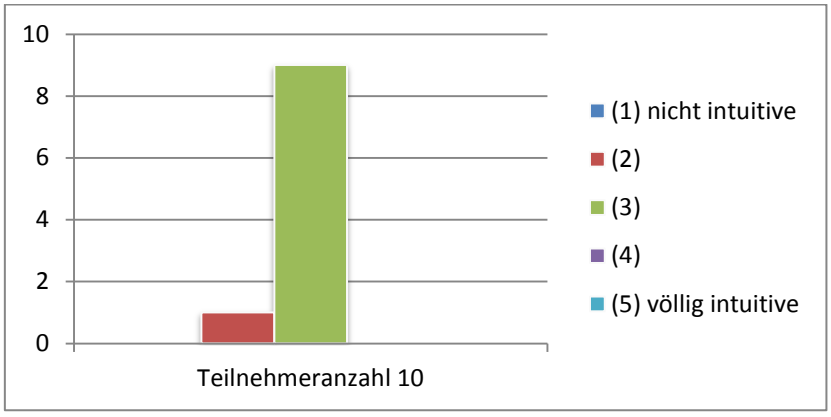

Abb. 32: Auswertung zu Frage 5

6. Hat der unterschiedliche Aufbau der Level Ihren Spaßfaktor beeinflusst?

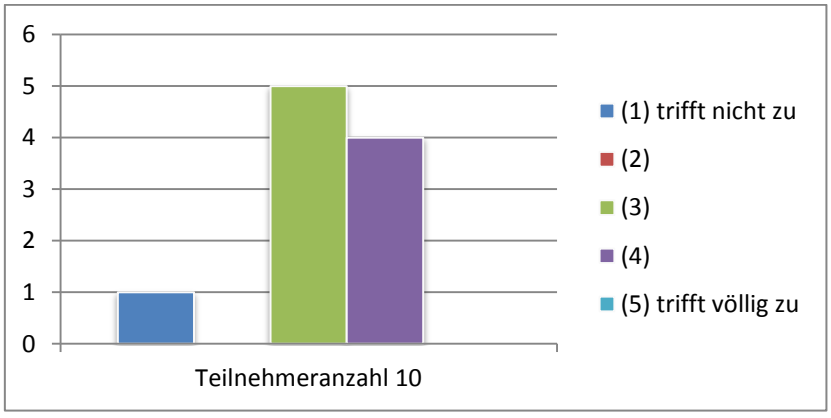

Abb. 33: Auswertung zu Frage 6

5.4.3 Optik

Die visuelle Darstellung der einzelnen Level wurde mit der Tendenz zu *ansprechend* eingestuft, mit einem Durschnittswert von 4.3. In einer Skala von eins „nicht ansprechend“ bis fünf „sehr ansprechend“ ist die visuelle Darstellung des Menüs von 20 Prozent mit *zwei*, von 20 Prozent mit *drei*, von 40 Prozent als *vier* und von 20 Prozent mit *fünf* bewertet. Die visuelle Darstellung des gesamten Spiels hat in der selbigen Bewertungsskala einen Durschnittswert von 4.2 erhalten. Die Parallaxe der Level hat ein Teilnehmer mit *weniger ansprechend*, drei *neutral*, drei a*nsprechend* und drei mit *völlig ansprechend* beurteilt.

7. Wie bewerten Sie die visuelle Darstellung der einzelnen Level?

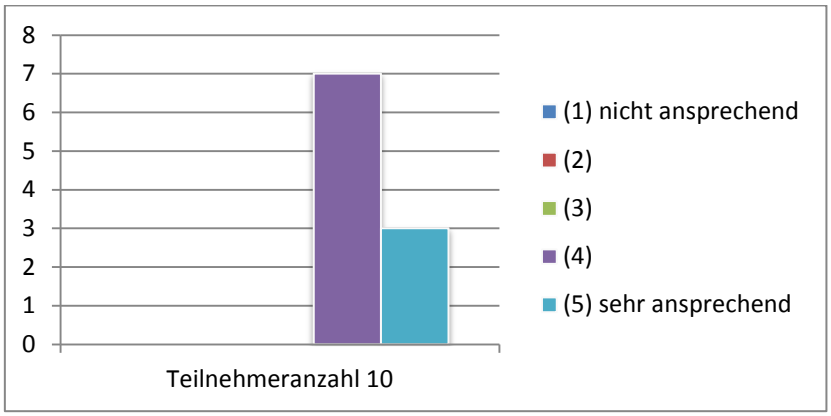

Abb. 34: Auswertung zu Frage 7

8. Wie bewerten Sie die visuelle Darstellung des Menüs?

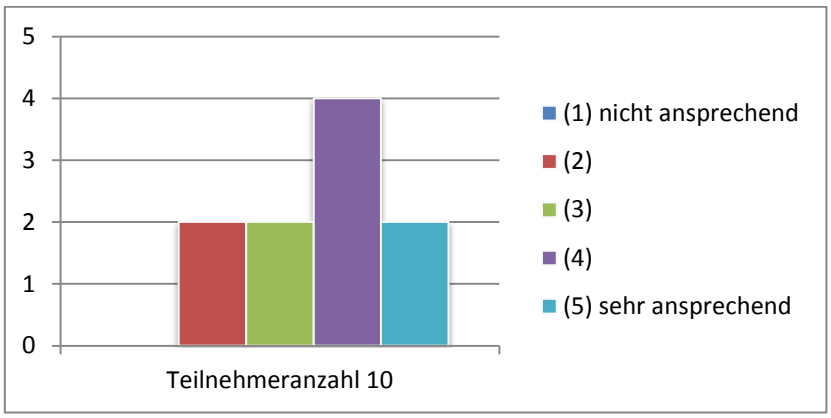

Abb. 35: Auswertung zu Frage 8

9. Wie bewerten Sie die visuelle Darstellung des gesamten Spiels?

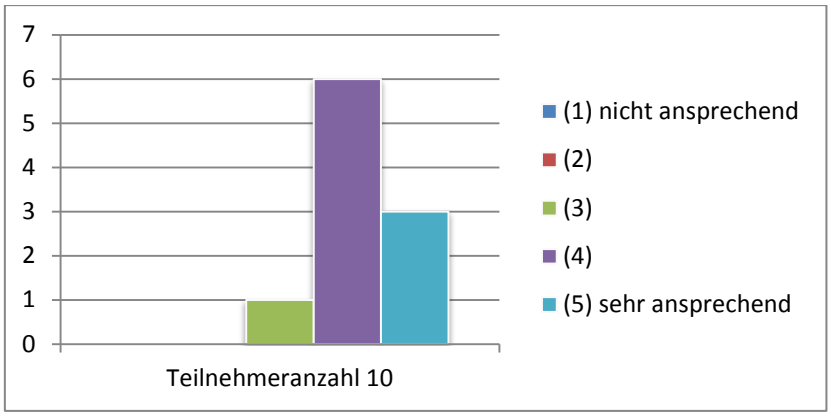

Abb. 36: Auswertung zu Frage 9

10. Wie bewerten Sie den fortlaufenden Hintergrund in den Level?

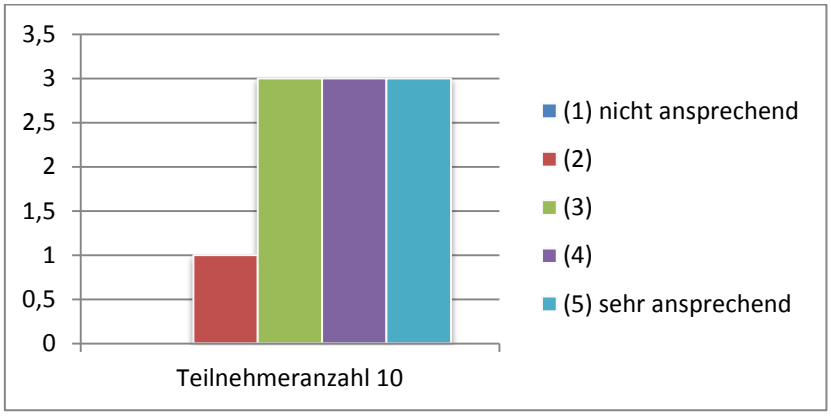

Abb. 37: Auswertung zu Frage 10

5.4.4 Farbwahrnehmung I

Den Testern ist im Level Erde die Farbe *Grün* viermal und *Braun* sechsmal in der Erinnerung geblieben. Für das Level Feuer ist zehnmal die Farbe *Rot* notiert und für das Level Wasser zehnmal die Farbe *Blau*. Im Level Wind haben sieben Teilnehmer die Farbe *Weiß* und drei Teilnehmer die Farbe *Hellblau* in der Erinnerung behalten.

Bei den vier Folgenden Durschnittswerten haben die Tester in einer Skalar von eins „nicht ansprechend“ bis fünf „sehr ansprechen“ ihre Bewertung abgegeben. Die Darstellung des Elements Feuer hat einen Durschnittswert von 4.5, Wasser 4.2, Wind 4.3 und Erde 4.2. Neunzig Prozent der Teilnehmer haben die natürliche Farbkombination in den Level beim Spielen *völlig* wahrgenommen, zehn Prozent beurteilen dies mit *neutral*.

11. Welche Farbe ist Ihnen im Level Erde am Stärksten in der Erinnerung geblieben?

Abb. 38: Auswertung zu Frage 11

12. Welche Farbe ist Ihnen im Level Feuer am Stärksten in der Erinnerung geblieben?

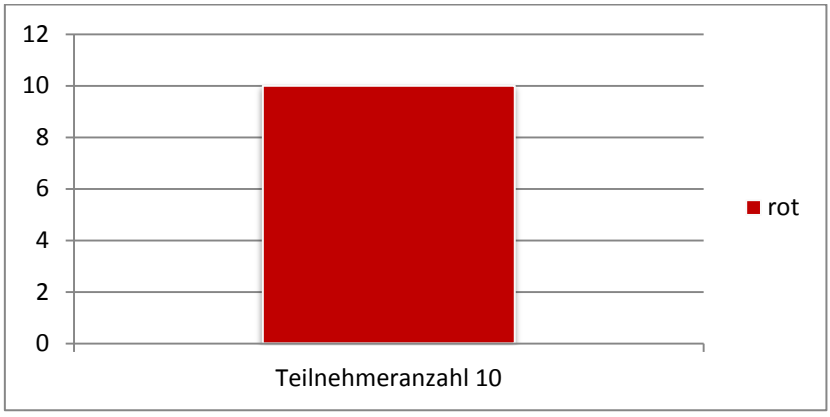

Abb. 39: Auswertung zu Frage 12

13. Welche Farbe ist Ihnen im Level Wasser am Stärksten in der Erinnerung geblieben?

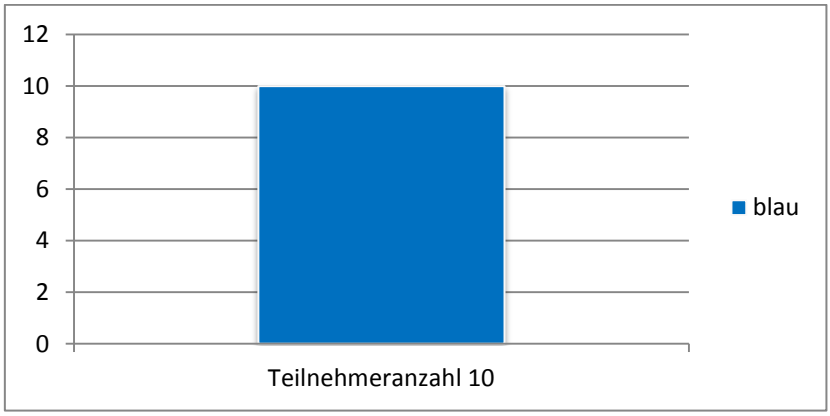

Abb. 40: Auswertung zu Frage 13

14. Welche Farbe ist Ihnen im Level Wind am Stärksten in der Erinnerung geblieben?

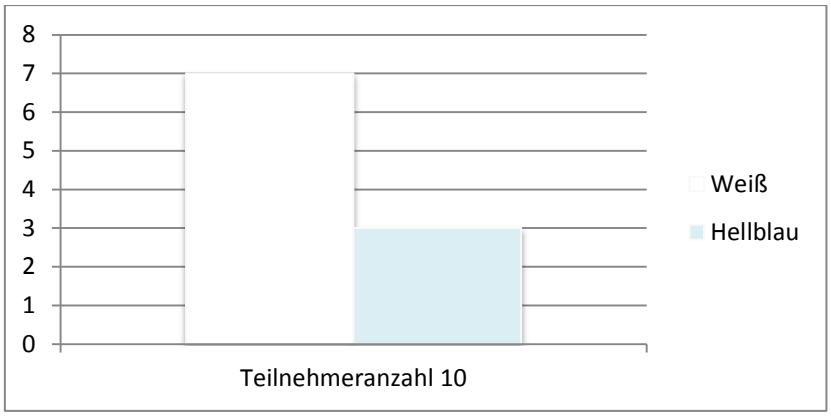

Abb. 41: Auswertung zu Frage 14

15. Wie bewerten Sie die farbliche Darstellung des Elements Feuer?

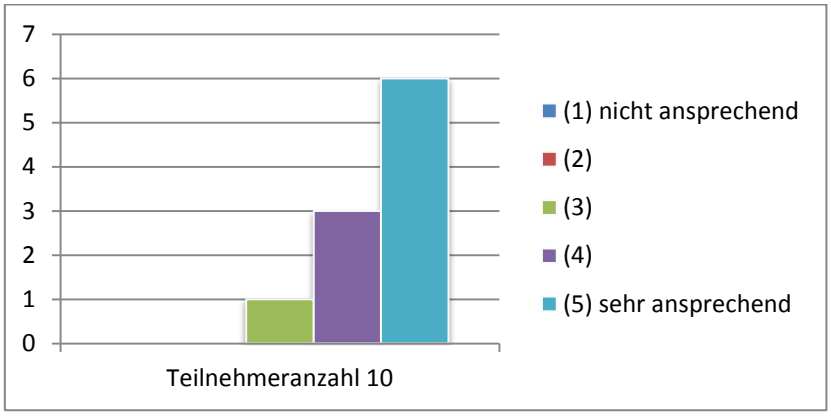

Abb. 42: Auswertung zu Frage 15

16. Wie bewerten Sie die farbliche Darstellung des Elements Wasser?

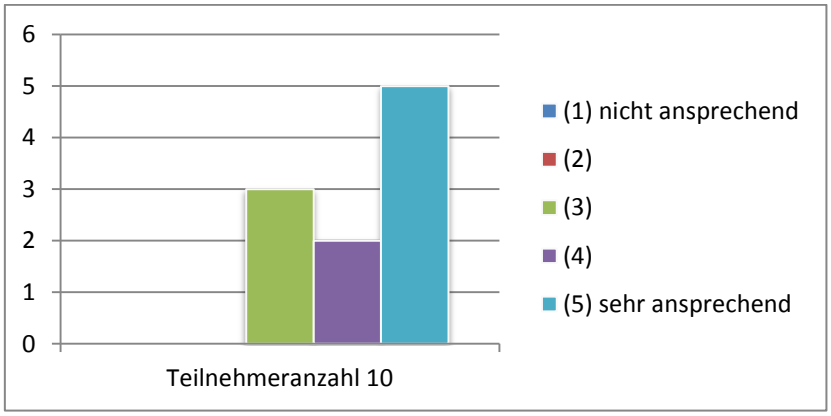

Abb. 43: Auswertung zu Frage 16

17. Wie bewerten Sie die farbliche Darstellung des Elements Wind?

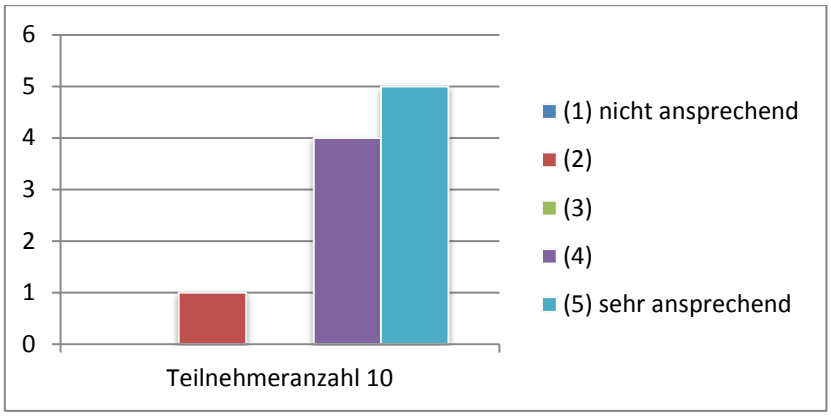

Abb. 44: Auswertung zu Frage 17

18. Wie bewerten Sie die farbliche Darstellung des Elements Erde?

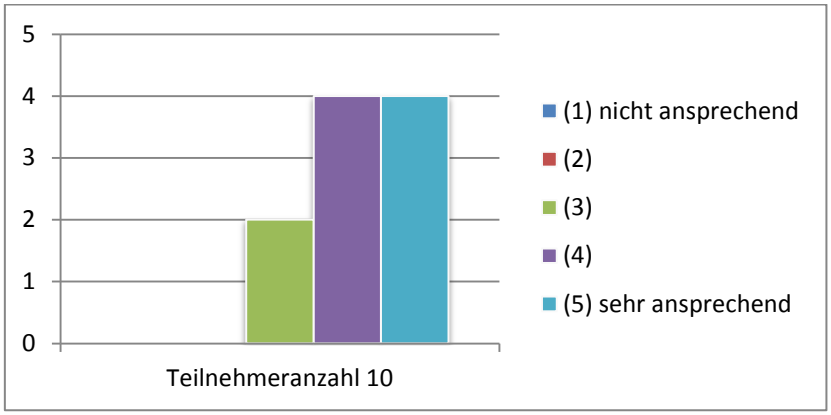

Abb. 45: Auswertung zu Frage 18

19. Haben Sie die natürlichen Farbkombinationen beim Spielen der Level bewusst wahrgenommen?

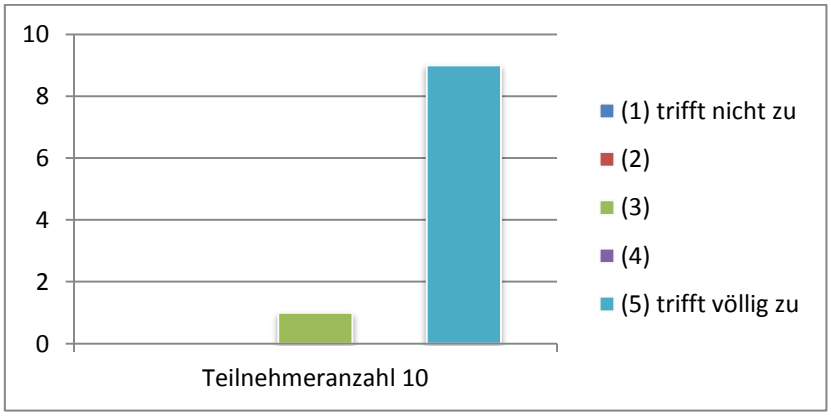

Abb. 46: Auswertung zu Frage 19

5.4.5 Farbwahrnehmung II

Zur Bewertung welcher Level den Testern im Gesamten am Besten gefallen hat, hat Feuer fünf Stimmen, Erde eine, Wasser drei und Wind eine bekommen.

20. Welcher Level hat Ihnen im Gesamten am Besten gefallen?

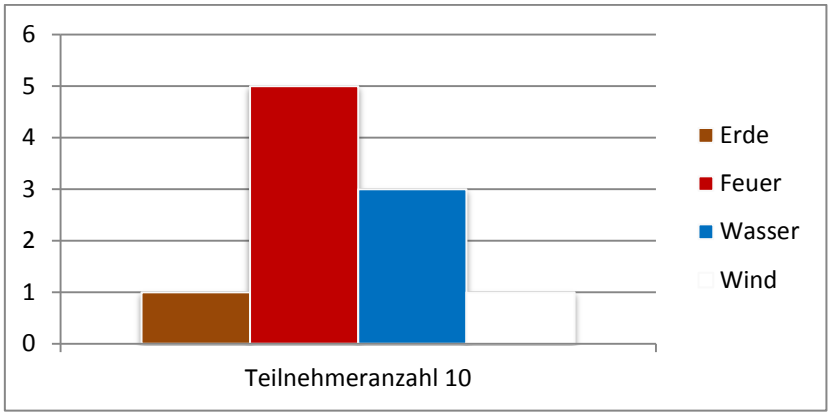

Abb. 47: Auswertung zu Frage 20

5.4.6 Farbwahrnehmung III

Den Teilnehmern fiel Rot fünfmal, Blau dreimal, Weiß einmal und Hellblau ein, als Sie an die Applikation zurück gedacht haben. Die Lieblingsfarben der Teilnehmer wurde wie folgt beantwortet, mit viermal Blau, einmal Gelb, einmal Schwarz, einmal Lila und dreimal Rot.

21. Welcher Farbe ist Ihnen in Erinnerung geblieben, wenn Sie an das Spiel denken?

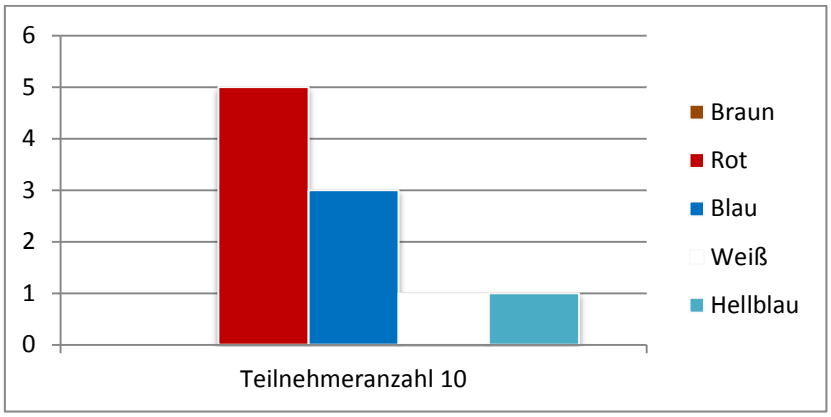

Abb. 48: Auswertung zu Frage 21

22. Was ist Ihre Lieblingsfarbe?

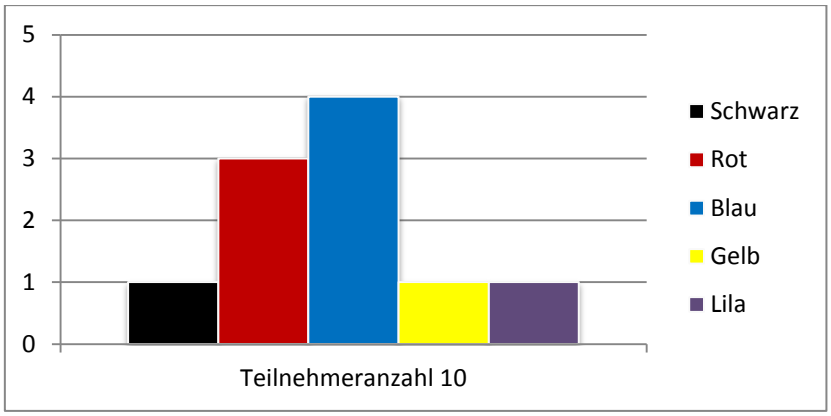

Abb. 49: Auswertung zu Frage 22

5.5 Bewertung der Ergebnisse

Im Folgenden Abschnitt werden zuerst die Sinnesblöcke bewertet und später die Forschungsfrage aufgegriffen.

In dem Bereich Benutzerfreundlichkeit sind drei Fragen gestellt. Der Spaßfaktor des Spiels ist mit 3.6 im Durschnitt von 5 möglichen Sternen bewertet und hinterlässt einen guten Eindruck. Fünfzig Prozent der Teilnehmer bewerten den Schwierigkeitsgrad des Spiel mit *angemessen* und weitere 50 Prozent mit *völlig angemessen.* Das ergibt einen Mittelwert von 4.5 von 5 Möglichen und spiegelt eine angemessene Zufriedenheit in der Schwierigkeit dar. Analoger Mittelwert ergab die Bewertung zur Dauer eines Level und zeigt dessen Angemessenheit. Der Spieler ist somit in der Lage durch seine Geschicklichkeit alle vorgegebenen Level zu überwinden und das Ziel zu erreichen. Außerdem kann damit kann ausgeschlossen werden, dass das Spielgefühl des Nutzers negativ beeinflusst wurde, durch einer zu hohen Schwierigkeit oder Langwierigkeit.

Der Bereich Mechanik enthält ebenfalls drei Fragen. Die Menüführung ist von 60 Prozent der Teilnehmer als *intuitive* und von 40 Prozent als *völlig intuitive* bewertet. Das Ergibt einen Mittelwert von 4.4 und somit ist das Menü selbsterklärend und kann klar gesteuert werden. Bis auf eine Ausnahme ist die Steuerbarkeit der Spielfigur von 90 Prozent der Tester als *neutral* bewertet. Diese erfüllt somit Ihren Zweck der Fortbewegung des Hauptcharakters und dessen meistern der verschieden Level. Somit kann ein kleiner Einflussfaktor beim Spielgefühl entstehen, da es immer zu ± Abweichungen kommen kann. Der unterschiedliche Aufbau der Level und dessen Einfluss auf das Spielgefühl ist von eins „trifft nicht zu“ bis fünf „trifft völlig zu“ von zehn Prozent der Tester als *nicht zu treffend*, von fünfzig Prozent als *neutral* und von vierzig Prozent als *zutreffend eingeordnet*. Somit hinterlässt es einen neutralen Eindruck bei den Testern. Dies kann Bedeuten, dass Sie die Frage nicht richtig Einordnen konnten, oder dass der unterschiedliche Aufbau der Spielobjekte auf den Grundplattformenobjekten keinen bedeuteten Einfluss gehabt hat. Dies unterstützt die spätere Bewertung zur Forschungsfrage, da die unterschiedliche Positionen, dessen Anzahl, dessen minimal abgewandelte Form der Spielobjekte in der Szene genau richtig angeordnet sind. Sie haben keinen Lerneffekt erzeugt aufgrund der unterschiedlichen Anordnung sind aber trotzdem identisch um ein gleiches Spielgefühl zu erzeugen.

Der Abschnitt Optik beinhaltet ebenfalls drei Fragen. Die visuelle Darstellung der einzelnen Levels ist von den Nutzern mit der Tendenz zu *ansprechend* eingestuft. Mit einem Mittelwert von 4.3 von 5 Möglichen, ist die Gestaltung und Farbkombination der Level sehr passend. Da es keine abschweifende Bewertung dies bezüglich gibt, sind zutreffende Welten entstanden und bieten eine gute Grundlange zur Bewertung. Das Menü sollte eine animierte Komponente enthalten und zeitlos sein. In einer Skala von eins „nicht ansprechend“ bis fünf „sehr ansprechend“ ist die visuelle Darstellung des Menüs von 20 Prozent mit *zwei*, von 20 Prozent mit *drei*, von 40 Prozent als *vier* und von 20 Prozent mit *fünf* bewertet. Dies ergibt einen Durschnitt von 3.6 und kann als neutral bewertet werden. Somit ist das Menü kein optisches Highlight sondern als dezent und zeitlos einzustufen. Außerdem fällt der Einflussfaktor weg, dass das Menü so heraussticht, das dessen Farbkomponenten die Forschungsfrage beeinflussen. Die visuelle Darstellung des gesamten Spiels ist mit 4.2 im Durschnitt sehr positiv ausgefallen.

Der erste Sinnesabschnitt zur Farbwahrnehmung stellt wichtige Frage. Unteranderem stellt er die entscheidende Farbe welche Farbe spezifisch in dem Level Erde, Feuer, Wasser und Wind in der Erinnerung geblieben ist. In der Literatur wird das Element Erde als Braun, wie Grün dargestellt, ebenfalls wird das Element Wind als Weiß, wie Hellblau dargestellt. Feuer und Wasser sind mit einer eindeutigen Farbe gekennzeichnet [ve] [ve1]. Den Testern ist im Level Erde die Farbe *Grün* viermal und *Braun* sechsmal in der Erinnerung geblieben. Für das Level Feuer ist zehnmal die Farbe *Rot* notiert und für das Level Wasser zehnmal die Farbe *Blau.* Im Level Wind haben sieben Teilnehmer die Farbe *Weiß* und drei Teilnehmer die Farbe *Hellblau* in der Erinnerung behalten. Somit ist die Farbkombination der Level auf den Punkt gebracht und erfüllt die Darstellung der Farbkomponenten eines Elements im jeweiligen Level und dessen Farbwahrnehmung. Bei den vier Folgenden Mittelwerten haben die Tester in einer Skalar von eins „nicht ansprechend“ bis fünf „sehr ansprechen“ ihre Bewertung abgegeben. Die Fragen beziehen sich spezifisch zur der farblichen Darstellung der einzelnen Elemente. Die Darstellung des Elements Feuer hat einen ist mit 4.5, Wasser mit 4.2, Wind mit 4.3 und Erde mit 4.2 bewertet. Dies unterstützt meine vorherige Aussage zur Farbwahrnehmung und dessen Darstellung. Neunzig Prozent der Teilnehmer haben die natürliche Farbkombination in den Level beim Spielen *völlig* wahrgenommen und zehn Prozent beurteilen dies mit *neutral*. Somit wurde das Element durch die Spielszenen erfolgreich vermittelt und bis auf eine Ausnahme klar wahrgenommen. Damit ist der Einflussfaktor, dass ein Element und dessen Szene nicht zugeordnet werden kann ausgeschlossen worden.

Der zweite Sinnesabschnitt zur Farbwahrnehmung hat eine Frage zu seinem persönlichen Favorit zwischen den Level. Zur Bewertung welcher Level den Testern im Gesamten am Besten gefallen hat, dabei hat Feuer fünf Stimmen bekommen, Erde Eine, Wasser Drei und Wind Eine. Der dritte Sinnesabschnitt zur Farbwahrnehmung fragt unter anderem persönliche Erinnerung zur der Anwendung ab. Bei der Frage welche Farbe in der Erinnerung geblieben ist, sobald Sie an die Anwendung decken haben, die Teilnehmer fünfmal Rot, dreimal Blau, einmal Weiß und einmal Hellblau notiert. Anschließend wird die Lieblingsfarbe der Probanden abgefragt um auszuschließen, dass Diese aufgeschrieben wurde und um einen eventuellen Zusammenhang zwischen Dieser und der Spielwahrnehmung zu erkennen. Die Lieblingsfarben der Teilnehmer ist beantwortet mit viermal Blau, einmal Gelb, einmal Schwarz, einmal Lila und dreimal Rot.

Im Folgenden wird Bezug auf die Forschungsfrage genommen, die lautet: „Inwiefern beeinflussen Farbkombinationen im Spiel eines Jump & Run Games, in Form der vier auftretenden Elemente, die menschliche Wahrnehmung und deren Auswirkungen auf das individuelle Spielgefühl.“ Zuvor wurden wichtige zusätzliche Einflussfaktoren die die menschliche Wahrnehmung beeinflussen können erkannt und ausgeschlossen oder reduziert. Es gibt eine Starke Überschneidung zwischen der

Lieblingsfarbe und der in Erinnerung gebliebenen Farbe der Teilnehmer. Dadurch kann nicht ausgeschlossen, werden das die Lieblingsfarbe aufgeschrieben worden ist. In Zahlen ausgedrückt übereinstimmen 70 Prozent der Lieblingsfarben mit den in der Erinnerung geblieben Farben überein und 30 Prozent sind konträr dazu. Auffallend ist, dass 90 Prozent des am Besten gefallenden Level mit der Farbe übereinstimmt, die den Teilnehmern in der Erinnerung geblieben ist, sobald Sie an das Spiel denken. Fünfzig Prozent der Teilnehmer gefällt das Level Feuer , Rot, am Besten und zugleich 50 Prozent der Teilnehmer erinnern sich an diese Farbe zurück. Das Gleiche gilt für 30 Prozent der Nutzer die das Level Wasser, Blau, am Besten gefällt und 30 Prozent kommt die Farbe Blau in Erinnerung. 10 Prozent denken an die Farbe Weiß und 10 Prozent gefällt das Level Wind, Weiß am Besten. Jedoch gibt es auch 10 Prozent denen eine andere Farbe in Erinnerung bleibt, Hellblau und 10 Prozent die das Level Erde, Braun am Besten fanden. Somit, in Kombination, 10 Prozent haben nicht mit der zurückgedachten Farbe und dem dazugehörigen Level übereinstimmt. In der „Farbenpsycholoie“ widmet man sich der Analyse der Effekte, die Farbtöne auf unser Verhalten haben. Laut einer Studie zur Wirkung von Farben im Marketingbereich bewerten 90 Prozent der Kunden ein Produkt unter anderem aufgrund dessen Farben. Weiterhin bekräftigen Wissenschaftler die Annahme das Farben einen wesentlichen Bestandteil dazu beitragen wie etwas wahrgenommen wird. Bei Webseiten bilden sich 42 Prozent der Kunden ihre Meinung basierend auf dem Design, bei dem die Farbe den größten Einflussfaktor wiederspiegelt. Umgekehrt hat eine schlechte Farbauswahl einen negativen Einfluss auf Kunden [fmw]. Auch in diesem Nutzertest ist festzustellen, dass die Welt die am Besten gefallen hat und wahrgenommen wurde, auch die zugehörige Farbe reproduziert beim Zurückerinnern. Da die Grundvoraussetzungen wie zuvor beschrieben geschaffen worden sind, wie das Beachten eines fast gleichen Spielgefühls, mit selben Objekten aber unterschiedlicher Positionierung und Anordnung. Einer neutralen Steuerung und einem nicht zu auffälligen Menü. Nur die das Herausstechen der Grafiken für die Elemente und dessen farbliche Umsetzung steht im Fokus. Abschließen ist festzuhalten das Farbkombinationen in einem *2D-Jump ’n’ Run* Spiels eine Auswirkung auf die menschliche Wahrnehmung haben und das individuell Spielgefühl gemäß der Empathie einer Farbe beeinflusst. Zusätzlich ist die Farbpalette der vier Elemente eine gute natürliche Basis gewesen um diese Frage zu untersuchen. Ausnahmen bestätigen die Regel, deswegen sind nicht alle davon beeinflusst, da der Fokus der Bewertung oder Wahrnehmung auf anderen Werten basieren kann, wie die der Steuerung.

5.6 Kritische Reflexion der eigenen Untersuchung

Kritsch zu Betrachten ist die geringe Teilnehmerzahl bei dem Nutzertest von zehn Personen. Um eine valide und genauere Aussage tätigen zu können, ist eine erheblich größere Teilnehmeranzahl erforderlich. Weiterhin ist es schwierig zu Beurteilen, ob die Teilnehmer unterbewusste Farbabspeicherungen abrufen sobald sie das Element lesen, die dazu abgespeicherte Farbe aufschreiben und nicht die tatsächlich wahrgenommene. Um das Ergebnis zu stärken muss ein Weiterer Nutzertest mit ungefähr 1000 Teilnehmern aufwärts durchgeführt werden.

6 Fazit und Ausblick

Die Folgenden Unterpunkte des Kapitels befassen sich mit der Entwicklung der Applikation *Element.* Dabei wird die Ausgangssituation zusammengefasst, ein Fazit gezogen und ein Ausblick gegeben.

6.1 Ausgangssituation

Ziel war es ein funktionierendes *2D -Jump ‘n‘ Run* Spiel zu erstellen für Android, welches auf dem Handy gespielt werden kann. Als das Spielgenre feststand, wurde sich für die Entwicklungsumgebung Unity entschieden. Zum Erstellen der zahlreichen Grafiken fiel die Wahl auf Adobe Photoshop. Für beide Softwareprogramme, wie für den Themenbereich, programmieren einer Applikation, bestanden keine Vorkenntnisse. Eine Forschungsfrage war nicht eingeplant hat sich aber aus den Möglichkeiten entwickelt. Anschließend soll der Nutzertest zur Evaluation und Bewertung dienen.

6.2 Fazit

Die Entwicklung einer interaktiven Spiele-Applikation unter Android war ein voller Erfolg. Das Erlernen der Grundlagen von Unity und Adobe Photoshop war anfangs wichtige Grundvoraussetzung für ein erfolgreiches Abschließen dieses Konzeptes der vorliegenden Arbeit. Es konnten wichtige Einblicke und Lernerfahrungen in der Modellierung von Grafiken in Photoshop erlangt werden. Weiterhin haben zahlreiche Misserfolge, wie Erfolge bei der Umsetzung mein Programmierwissen verbessert und mir eine Kompetenz in Unity verschafft. Nach ungefähr sechzig Prozentiger abgelaufener Restzeit für dieses Projekt, war es schwierig ein Konzept zu finden. Welches schon vorhandene Vorgaben in Form der Vier Elemente inkludiert und trotzdem eine realistische Forschungsfrage zulässt. Letztendlich konnte ein spannendes Konzept gefunden werden. Aufgrund der wenigen Erfahrung im Kompletten Themengebiet dieser Arbeit war es sehr aufwändig alle Grafiken zu erstellen, zu bearbeiten oder zu finden, da es nur begrenzte kostenlose Ressourcen gibt. Zusätzlich wurde die Aufgabe erschwert nun diese Welten und Elemente passend zu den Elementen zu modellieren. Damit die Forschungsfrage beantworten werden konnte, mussten auch alle Farben in den Spielszenen und Grafiken abgestimmt werden um eine realistische Farbwahrnehmung erzeugen zu können. Im Nachhinein hat dies in der Kombination des Entwickeln eines *2D -Jump ‘n‘ Run* Spiel sehr viel Aufwand und Zeit gekostet. Ich würde sagen das Umsetzten dieser Arbeit hat sich damit um ein Vieles verkompliziert. Die Joystick - Steuerung könnte etwas agiler sein, erfüllt aber völlig Ihren Zweck. Es wurden alle Konzeptionsvorgaben umgesetzt und in die Anwendung implementiert.

Die Anwendung wurde durchweck positiv bewertet, besonders die grafische Darstellung und die farbliche Komponente. Die Forschungsfrage kam zu dem Ergebnis, dass Farbkombinationen, dargestellt von den natürlichen vier Elementen, die menschliche Wahrnehmung und dessen Auswirkung auf das individuelle Spielgefühl beeinflusst, zutrifft. Insgesamt haben 90 Prozent der Teilnehmer diesen Effekt gezeigt, was sehr beeindruckend ist. Somit ist am Ende bewiesen das ein *2D -Jump ‘n‘ Run* Spiel für Nutzer mit einer guten Farbwahl beeinflusst und verbessert werden kann.

6.3 Ausblick

Die individuellen aufwändigen Hintergründe der vier Spielszenen, sind so konzipiert, dass aus jedem einzelnen ein eigenes *2D -Jump 'n' Run* Spiel daraus entwickelt werden kann. Weiterhin kann für jedes Element aufgrund des fortlaufenden Hintergrundes leicht weitere Spiellevel erstellt werden. Die gute und große Grundbasis dieses Projekt und der gelegte Fokus auf das Grafische und Farbliche ermöglichen es, in Zukunft weitere spezifische Elemente in das Spiel einzubauen. Da es noch viele Möglichkeiten gibt, wie weitere Gegner, die eine Art kleine KI besitzen und den Spieler anlaufen, sobald der erkannt wird. Des Weiteren können Gegner eingebaut werden die nur mit der Berührung über den Touchscreen eliminiert werden können. Es könne Eigenschaften dem Charakter hinzugefügt werden, die nur in dem jeweiligen Element funktionieren. Wie der Doppelsprung bei Wind, das Schießen bei Feuer, ein Schutzschild bei Erde und bei Wasser das Überwinden von Wasser. Die Joystick Steuerung kann verbessert und erweitert werde. Die Musik kann für jedes Level variieren und denn Spaßfaktor steigern. Es gibt zahllose Möglichkeiten zur Erweiterung und Verbesserung der Applikation. Die Applikation ist aufgrund der Ausgangslage nicht konkurrenzfähig auf dem aktuellen Spielemarkt. Eignet sich jedoch wunderbar für das Erlernen und Umsetzten weitere wichtiger Komponenten zum Beispiel das Verwenden von 3D Art in der 2D Spielszene.

7 Anhang

- Nutzertest

Nutzertest

Information:

Danke für Ihre Teilnahme an diesem Nutzertest, die Dauer beträgt ungefähr 40 Minuten. Der vorliegende Fragebogen dient zur Evaluierung meines entwickelten Jump'n'Run – Spiels für Android Smartphones im Zusammenhang meiner Forschungsfrage.

Bitte füllen Sie den folgenden Fragebogen erst nach Beendigung des kompletten Spiels aus.

Öffnen Sie die Anwendung und lesen Sie sich zuerst die Spielanleitung durch. Falls Fragen auftreten sollten, stellen Sie diese bitte vor dem Starten des ersten Levels. Teilen Sie mir bitte mit, wenn Sie alles verstanden haben und starten möchten.

Viel Spaß! Name: _____

Bewertung einer interaktiven Spiele-Applikation:

Benutzerfreundlichkeit					
	(1) = Stern	**(2) = Sterne**	**(3) = Sterne**	**(4) = Sterne**	**(5) = Sterne**
Wie bewerten Sie den Spaßfaktor des Spiels?	☐	☐	☐	☐	☐
	(1) = nicht angemessen	**(2)**	**(3)**	**(4)**	**(5) = völlig angemessen**
Wie bewerten Sie den Schwierigkeitsgrad des Spiels?	☐	☐	☐	☐	☐
Wie bewerten Sie die Dauer eines Levels?	☐	☐	☐	☐	☐

Mechanik					
	(1) = nicht intuitive	**(2)**	**(3)**	**(4)**	**(5) = völlig intuitive**
Wie bewerten Sie die Menüführung des Spiels?	☐	☐	☐	☐	☐

Wie bewerten Sie die Steuerbarkeit des Spiels?	☐	☐	☐	☐	☐

	(1) = trifft nicht zu	(2)	(3)	(4)	(5) = trifft völlig zu
Hat der unterschiedliche Aufbau der Level Ihren Spaßfaktor beeinflusst?	☐	☐	☐	☐	☐

Optik

	(1) = nicht ansprechend	(2)	(3)	(4)	(5) = sehr ansprechend
Wie bewerten Sie die visuelle Darstellung der einzelnen Levels?	☐	☐	☐	☐	☐
Wie bewerten die die visuelle Darstellung des Menüs?	☐	☐	☐	☐	☐
Wie bewerten Sie die visuelle Darstellung des gesamten Spiels?	☐	☐	☐	☐	☐
Wie bewerten Sie den fortlaufenden Hintergrund in den Levels?	☐	☐	☐	☐	☐

Farbwahrnehmung I

Welche Farbe ist Ihnen im Level Erde am Stärksten in der Erinnerung geblieben? _____

Welche Farbe ist Ihnen im Level Feuer am Stärksten in der Erinnerung geblieben? _____

Welche Farbe ist Ihnen im Level Wasser am Stärksten in der Erinnerung geblieben? _____

Welche Farbe ist Ihnen im Level Wind am Stärksten in der Erinnerung geblieben? _____

	(1) = nicht ansprechend	(2)	(3)	(4)	(5) = sehr ansprechend
Wie bewerten Sie die farbliche Darstellung des Elements Feuer?	☐	☐	☐	☐	☐
Wie bewerten Sie die farbliche Darstellung des Elements Wasser?	☐	☐	☐	☐	☐
Wie bewerten Sie die farbliche Darstellung des Elements Wind?	☐	☐	☐	☐	☐

Wie bewerten Sie die farbliche Darstellung des Elements Erde?	☐	☐	☐	☐	☐
	(1) = trifft nicht zu	**(2)**	**(3)**	**(4)**	**(5) = trifft völlig zu**
Haben Sie die natürlichen Farbkombinationen beim Spielen der Levels bewusst wahrgenommen?	☐	☐	☐	☐	☐

Farbwahrnehmung II

	Erde	**Feuer**	**Wasser**	**Wind**
Welcher Level hat Ihnen im Gesamten am Besten gefallen?	☐	☐	☐	☐

Farbwahrnehmung III

Welche Farbe ist Ihnen in Erinnerung geblieben, wenn Sie an das Spiel denken? _____

Was ist Ihre Lieblingsfarbe? _____

Anmerkungen:

Was fanden Sie nicht gut? _____

_____

_____

Literaturverzeichnis

[eae] Marko Gargenta. (2011, 1. Auflage). Einführung in die Android-Entwicklung, Kalifornien: O'Reilly

[dap] https://developer.android.com/guide/platform .- Letzter Zugriff: 20.05.2019

[taa] https://www.programmierenlernenhq.de/tutorial-android-activities-und-intents/ .- Letzter Zugriff: 20.05.2019

[ast] https://developer.android.com/studio/intro .- Letzter Zugriff: 20.05.2019

[lli] https://www.pro-linux.de/news/1/21708/intellij-idea-14-veroeffentlicht.html .- Letzter Zugriff: 20.05.2019

[liba] https://github.com/libgdx/libgdx/wiki/The-application-framework .- Letzter Zugriff: 25.05.2019

[libd] https://libgdx.badlogicgames.com/documentation/ .- Letzter Zugriff: 25.05.2019

[cs] https://web.archive.org/web/20121202194727/http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf .- Letzter Zugriff: 26.06.2019

[sjdk] https://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/ .- Letzter Zugriff: 26.06.2019

[sony] https://web.archive.org/web/19990915060909/http://www.sonicfoundry.com/news/ShowRelea-se.asp?ReleaseID=107 .- Letzter Zugriff: 05.07.2019

verweise später genauer

[apc] Sibylle Mühlke. (2013). Adobe Photoshop CS6: das umfassende Handbuch, Bonn: Galileo Press

[jnr] https://www.arte.tv/de/videos/078750-010-A/art-of-gaming/ .- Letzter Zugriff: 05.07.2019

[ssc] https://www.techopedia.com/definition/27153/side-scroller .- Letzter Zugriff: 05.07.2019

[smb] https://de.wikipedia.org/wiki/Super_Mario_Bros. .- Letzter Zugriff: 23.08.2019

[lim] https://de.wikipedia.org/wiki/Limbo_(Computerspiel) .- Letzter Zugriff: 23.08.2019

[tww] https://de.wikipedia.org/wiki/The_Whispered_World .- Letzter Zugriff: 23.08.2019

[ray] https://raymanpc.com/wiki/de/Rayman_Jungle_Run .- Letzter Zugriff: 25.08.2019

[ve] https://www.canstockphoto.de/vier-elemente-nat%C3%BCrlich-natur--40279404.html .- Letzter Zugriff: 25.08.2019

[ve1] https://www.istockphoto.com/de/vektor/vier-elemente-symbole-flach-stil-wasser-feuer-erde-luft-beschilderung-gm519164196-90395941 .- Letzter Zugriff: 30.09.2019

[fmw] https://99designs.de/blog/design-tipps/farben-marketing-werbung/ .- Letzter Zugriff: 30.09.2019

[luu] https://docs.unity3d.com/Manual/ExecutionOrder.html .- Letzter Zugriff: 26.06.2019

[bet] https://www.welt.de/wirtschaft/webwelt/article177071160/Android-und-iOS-Apples-und-Googles-Gegner-liegen-erstmals-bei-0-0-Prozent.html .- Letzter Zugriff: 26.06.2019

[pug] https://de.statista.com/statistik/daten/studie/547170/umfrage/umsatz-von-google-play-weltweit/ .- Letzter Zugriff: 26.06.2019

[alcy] https://developer.android.com/guide/components/activities/activity-lifecycle .- Letzter Zugriff: 05.07.2019

[lib] https://github.com/libgdx/libgdx/wiki/The-life-cycle .- Letzter Zugriff: 05.07.2019

[smb1] http://www.nintendolife.com/news/2015/08/mario_history_super_mario_bros_-_1985#enlarge-2 .- Letzter Zugriff: 10.09.2019

[lim1] https://static.fore.4pcdn.de/premium/Screenshots/4b/42/2129478-medium.jpg .- Letzter Zugriff: 10.09.2019

[tww1] https://en.wikipedia.org/wiki/The_Whispered_World#/media/File:The_Whispered -_ World_parallax_scrolling_sample_1.jpg .- Letzter Zugriff: 10.09.2019

[ray1] https://www.gamersglobal.de/screens/72459 .- Letzter Zugriff: 30.09.2019

[ggm] https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/ .- Letzter Zugriff: 30.09.2019