UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Entwicklung einer interaktiven Applikation unter Android

Bachelorarbeit

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

vorgelegt von

Christopher Jensen

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	A. K. Hebborn (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Mai 2015

Erklärung

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

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

I Zusammenfassung

In dieser Arbeit wird eine interaktive Applikation für das Android OS entwickelt. Bei der Applikation handelt es sich um ein Virtual Reality Spiel. Das Spiel gehört zum Ego-Shooter-Genre und spielt in einem Weltraumszenario. Durch den Einsatz eines Stereo Renderers ist es möglich das Spiel in Kombination mit einer Virtual Reality Brille zu spielen.

Abstract

In this thesis, an interactive application is developed for Android OS. The application is about a virtual reality game. The game is settled in the genre of first-person shooters and takes place in a space scenario. By using a stereo renderer, it is possible to play the game combined with virtual reality glasses.

II Inhaltsverzeichnis

1 Einleitung

Getrieben von der Virtual Reality als Zukunftstechnologie, wird in der vorliegenden Bachelorarbeit versucht, ein interaktives Spiel in einer virtuellen Umgebung zu entwickeln. Dabei kommen Virutal Reality Funktionalitäten zum Einsatz. Das Spiel wird für die Android Plattform entwickelt. Da Virtual Reality zunehmend an Bedeutung gewinnt und Android in der Mobilindustrie stark vertreten ist, wird die Kombination dieser beiden Bereiche zu einem interessanten Aspekt, welchen es im Rahmen dieser Bachelorarbeit zu ergründen gilt.

1.1 Zielsetzung der Arbeit

Der Schwerpunkt dieser Arbeit ist die Entwicklung einer Interaktiven Applikation unter Android. Dabei fiel die Entscheidung auf die Entwicklung eines Videospiels. Es wird ein Spielkonzept erarbeitet, dass auf dem Prinzip der Virtual Reality basiert. Die Wahl von Virtual Reality als Grundlage, wurde aufgrund der vielversprechenden Zukunft die dieses Gebiet bietet, getroffen.
Das enorme Tempo in dem sich die VR-Technologien weiterentwickeln wird andere Einstellungen gegenüber Kreativität, Innovation und Erfindungsreichtum mit sich bringen und darüber hinaus Einfluss auf die Unternehmenskultur haben. Auf der Basis dreidimensionaler Produktdaten ist Virtual Reality dabei, zu einem entscheidenden Erfolgsfaktor für die Optimierung der Produktentwicklung zu werden. Virtual Reality ist in der heutigen Zeit als ein essentielles Werkzeug im Umgang mit virtuellen Prototypen in verschiedensten Anwendungsfeldern etabliert. Es entwickelt sich damit zu einer Schlüsseltechnologie für erfolgreiche Unternehmen.

Das vorwiegende Ziel dieser Bachelorarbeit ist es, sich mit der Softwareentwicklung für die Android Plattform auseinanderzusetzen. Der Fokus liegt dabei auf der Entwicklung von Spielen. Auf der Grundlage eines OpenGL ES 2.0/3.0 Frameworks soll ein Spiel - Space Wars - entwickelt werden.

Das Spiel soll ein Shooter in einer Weltraumumgebung werden. Die Steuerung wird durch die Smartphone internen Sensoren realisiert. Dabei ist eine Kombination des Gyrosensors und des Beschleunigungssensors geplant. Der Gyrosensor bestimmt dabei die Orientierung im dreidimensionalen Raum und der Beschleunigungssensor präzisiert die Ergebnisse und verhindert Schwankungen bei den Sensorwerten. Das Spielkonzept soll eine simple Spielmechanik beinhalten. Zusätzlich wird ein Punktesystem entwickelt, um damit ein Spielziel zu realisieren. Dadurch bietet sich die Möglichkeit eine Bestenliste zu nutzen, in welcher die besten Spieler und ihre Punktestände gespeichert werden. Da das Ziel in dieser Bachelorarbeit nicht die Entwicklung eines marktreifen Spiels ist, wird sich auf ein Level Design

beschränkt. Ebenfalls werden passende Töne erstellt, um die Immersivität zu steigern. In dieser Bachelorarbeit werden viele herausfordernde Gebiete angeschnitten und dadurch die Grenzen der Spielentwicklung aufgezeigt.

Erweiterbarkeit und Vielseitigkeit sind wichtige Eigenschaften, die das Spiel erfüllen soll. Die Erweiterbarkeit ermöglicht das schrittweise Auf- und Ausbauen des ursprünglichen Spielkonzepts. Zur Nutzung einer Virtual Reality Brille wäre beispielsweise die Erweiterung des Renderers zu einem Stereo Renderer nötig. Die Vielseitigkeit sorgt dafür, dass viele Bereiche der Spielprogrammierung auf Android behandelt werden können.

Das Ergebnis dieser Arbeit wird ein Virtual Reality Spiel sein, welches zur Kategorie der Casual Spiele gehört und auf der Android Version 5.0 basiert.

1.2 Vorgehen

Zu Beginn der Arbeit wird ein Spielkonzept entwickelt, das die Rahmenbedingungen des Spiels abstecken soll. Dabei werden der grobe Ablauf des Spiels mit den Interaktionen des Spielers und das Spielprinzip definiert.

Während einer Einarbeitungszeit von 4 Wochen, werden durch diverse Android Tutorials Informationen über die Spielentwicklung gesammelt und der Aufbau des Codes von anderen Virtual Reality Spielen analysiert, um einen ersten Überblick über die Softwareentwicklung unter Android zu erhalten. Die Nutzung einer verfügbaren Android-Game-Engine wird ausgeschlossen, da das Lernziel im Rahmen dieser Bachelorarbeit die Erlangung von möglichst umfassenden Kenntnissen über die Android Softwareentwicklung ist. Dieses Lernziel ist bei der Nutzung einer fertigen Android-Game-Engine jedoch nicht realisierbar, da die Menge der nötigen Eigenarbeit im programmiertechnischen Bereich, im Vergleich zu der Entwicklung einer Applikation ohne die Nutzung einer Android-Game-Engine, gering ausfällt. Die gewählte Programmiersprache für die Entwicklung der Applikation ist OpenGL ES. Um den Umfang der Entwicklung einzugrenzen, wird ein Framework als Grundlage dienen, auf dem die Applikation aufgebaut wird. Dazu wird das Open Source Community Framework namens Rajawali benutzt, welches auf OpengGL ES 2.0/3.0 basiert. Das Framework wird als Basis für die weitere Programmierung benutzt.

Nach der Einarbeitung in das Rajawali-Framework, wird ein erster Prototyp entwickelt. Dieser wird in zwei Hauptklassen unterteilt. Eine Hauptklasse ist die GameActivity-Klasse, welche dauerhaft aktiv ist und für die Spiellogik und das Graphical User Interface zuständig ist. Die andere Hauptklasse ist die GameRenderer-Klasse, die für die Darstellung von Levels, Objekten, Animationen und für die Kollisionsabfragen zuständig ist. Neben der Einbindung der Grafiken und Audiodateien wird ein Punktesystem implementiert, welches Punktestände in einer Datenbank speichert.

2 Virtual Reality

Unter Virtual Reality versteht man die Simulation einer computergenerierten Welt und ihrer physikalischen Eigenschaften in einer Echtzeit computergenerierten interaktiven virtuellen Umgebung.[1] Das Ziel ist es eine scheinbar echte Welt zu erschaffen, in die der Betrachter eintauchen, sich in ihr bewegen und seine Fantasien und Vorstellungen umsetzen kann.[1] Virtual Reality bildet eine hochwertige Benutzerschnittstelle, die über Kopf- und Handbewegungen, über die Sprache oder den Tastsinn gesteuert wird.[1]

Die virtuelle Realität ist eine Simulation, bei der versucht wird durch Computergrafik eine realistisch erscheinende Umgebung zu erschaffen. Diese Umgebung ist keineswegs statisch, sondern sie reagiert auf Benutzereingaben. Zwischen Benutzer und der virtuellen Umgebung besteht also eine direkte Interaktion.

Benutzereingaben werden über Sensoren, beispielsweise innerhalb von Datenhandschuhen oder Datenhelmen, erfasst und in Steuerbefehle umgesetzt. Sie wirken sich direkt auf die virtuelle Umgebung aus. Dreht sich der Betrachter beispielsweise nach links, so ändert sich dadurch auch die Perspektive der virtuellen Umgebung, durch eine Verschiebung nach links. Ein weiteres Beispiel ist die Interaktion des Betrachters mit der virtuellen Spielwelt mit Hilfe eines Datenhandschuhs. So kann er nach einem virtuellen Gegenstand greifen und durch die haptischen Sensoren im Handschuh werden die Tastbewegungen umgesetzt. Durch diese Interaktionen taucht der Betrachter in die virtuelle Welt ein. Der Betrachter kann somit Teil der Aktion werden und kann darin seine Vorstellungen umsetzen.

Virtual Reality ist noch ein recht junges Wissenschaftsgebiet. Die rasante Weiterentwicklung dieses Gebietes wird unter anderem von den schnellen Fortschritten bei der zugrundeliegenden Hardware getrieben.

3 Grundlagen

Dieses Kapitel gibt einen kurzen Einblick in die Spielentwicklung, sowie in die unterschiedlichen Bereiche des Rajawali-Frameworks.

3.1 Game-Development

Im Prinzip besteht jede Applikation, die für die Android Plattform entwickelt wird, immer aus den gleichen Bausteinen. Die entsprechenden Module abstrahieren die Kommunikation mit dem zu Grunde liegenden Betriebssystem und umfassen meistens folgende Funktionen:

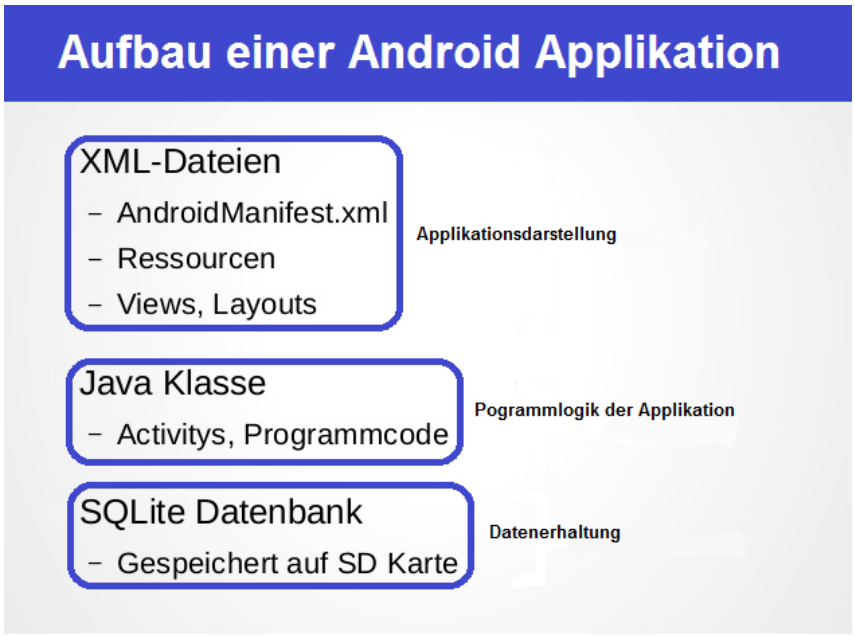

Abbildung 1: Aufbau einer Android-Applikation

- **XML-Dateien**
  Durch die XML-Dateien, wird das Layout der verschiedenen Fenster bestimmt, die in der Applikation genutzt werden. Zudem wird hier der Name einer Applikation festgelegt. Zusätzlich findet in den XML-Dateien die Deklaration der Java-Klasse statt, mit der die Applikation startet. Hier wird auch das minimale Android API Level definiert, welche die Applikation benötigt. Zudem werden in den XML-Dateien auch die Libraries bestimmt, welche die Applikation verwendet.

- **Java-Klassen**
  In den Java-Klassen befindet sich die gesamte Programmlogik. Es existiert mindestens eine Activity-Klasse, die dauerhaft aktiv ist und die Spiellogik festlegt. Zusätzlich wird hier der Renderer bestimmt, der für die Darstellung sämtlicher Inhalte auf dem Bildschirm verantwortlich ist. Normalerweise existieren darüber hinaus noch Java Klassen für jedes Objekt und jede Animation des Spiels. In manchen Fällen wird auch die Spiellogik in einer separaten Klasse definiert, welche in der Activity-Klasse geladen wird.

- **SQLite Datenbank**
  Daten, wie die Einstellungen der Applikation, Spielstände oder Highscores, legt Android in einer SQLite-Datenbank ab. Das zugehörige Datenbanksystem SQLite gehört zu den stabilsten und am meisten ausgereiften Werkzeugen, dass der Open-Source-Bereich zu bieten hat. Auch an Geschwindigkeit ist SQLite kaum zu schlagen. Zwar kann man in Android-Anwendungen auch mit einem anderen Datenbanksys-

tem arbeiten, doch gibt es keinen Grund dazu. Für die Arbeit mit SQLite stellt Android Klassen wie SQLiteDatabase zur Verfügung, die auf das System zugeschnitten sind.

Die Softwareentwicklung unter Android basiert hauptsächlich auf der Java Programmiersprache und das Android OS unterstützt Java durch diverse verfügbare Libraries, welche Android Softwareentwickler entlasten. Die Funktionen, die durch das Android Betriebssystem bereitgestellt werden, beinhalten qualitativ hochwertige Audio-und Videomedienformate.

Durch die Einführung neuer Plattform- und Softwareupdates entwickelt sich die Mobilindustrie ständig weiter, wodurch das gesamte Benutzererlebnis gefördert wird. Unter den großen mobilen Plattformen wie Blackberry, Windows Mobile, iPhone, Symbian OS und Android, ist Android zweifellos zu einem Hauptwettbewerber geworden, aufgrund der Unterstützung von Multi-tasking, einfacher Anpassungsmöglichkeiten und hoher Usability. Da das Android Betriebssystem eine Open Source Plattform ist, ist es kosteneffizienter als andere Plattformen.

Die hohe Nachfrage nach Android Applikationen hat zu einer erhöhten Nachfrage von Android Softwareentwicklern geführt.

3.1.1 Game-Loop

Das Herz aller Computerspiele stellt der Game Loop mit seinen unterschiedlichen Aufgaben dar. Etwas vereinfacht lässt sich der Game Loop in folgendem Zyklus darstellen.

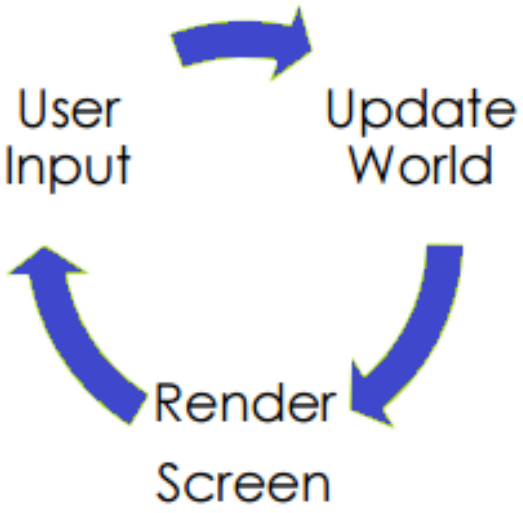

Abbildung 2: Game Loop

Die genauen Schritte des Game Loops können sich je nach Art des Spiels ein wenig unterscheiden. Folgende Schritte sind Teil des Game Loops in „Space Wars“:

- User Input entgegen nehmen
- Spiel Status gemäß Input aktualisieren
- Spiel Status gemäß Zeitverlauf aktualisieren
- Kollisionserkennung zwischen Objekten
- Objekte aktualisieren
- Anzeige gemäß des Spiel Status aktualisieren
- Anzeige rendern
- Toneffekte wiedergeben, falls eine Explosion stattfindet oder der Spieler getroffen wird

3.2 Rajawali-Framework

Das Rajawali-Framework ist ein Open Source Community Framework für die Softwareentwicklung unter Android und basiert auf OpenGL ES 2.0/3.0. Es ist zur Verwendung für normale Applikationen, sowie zur Verwendung für Live Wallpaper vorgesehen. Es wurden bereits viele Applikationen und Live Wallpaper mit dem Rajawali-Framework erstellt. Das Framework ist ein wachsendes Toolkit zur Entwicklung von 3D-Inhalten und wird ständig weiterentwickelt. Das Rajawali-Framework ist sehr umfangreich und bietet viele vordefinierte Klassen zur Entwicklung von 3D-Applikationen.

3.2.1 Komponenten

In diesem Kapitel werden die einzelnen Komponenten aus dem Rajawali-Framework erläutert, die bei der Entwicklung der Applikation im Rahmen dieser Bachelorarbeit verwendet werden.

Quaternion Die Quaternionen bilden ein 4D-Zahlensystem, vergleichbar mit dem 2D-Zahlensystem der Komplexen Zahlen, jedoch sind sie bei der Multiplikation nicht kommutativ. Sie werden in dieser Applikation zur Darstellung und einfachen Berechnung von Orientierungen im 3D-Raum verwendet, wobei sie hier deutlich anschaulicher sind als unitäre Matrizen. Eine Quaternion hat eine w-, x-, y- und z-Komponente. Die Klasse stellt Funktionen zur Addition, Subtraktion und Multiplikation mit anderen Quaternionen zu Verfügung, sowie eine Funktion zur Multiplikation mit einem Skalar.

Vector3 Vector3 ist eine einfache Implementation eines 3-dimensionalen Vektors. Die Klasse besitzt 3 Komponenten für die x-, y- und z-Koordinate vom Typ „Double“. Zudem stehen hier Funktionen zur Addition, Subtraktion, Multiplikation und Division zur Verfügung. Zusätzlich existieren Funktionen zur Skalierung, Rotation, Invertierung und Normalisierung des Vektors. Ebenfalls sind die Berechnung des Kreuzproduktes und die Distanz zu einem anderen Vektor möglich.

Matrix Die Matrix-Klasse besteht aus 4-mal 4-Spalte-Vektor-Matrizen in einer spaltenweisen Anordnung. Die Matrix-Klasse ist eine Erweiterung der Matrix-Klasse aus der Android eigenen Matrix-Klasse. Sie wurde so modifiziert, dass sie nun doppelt genaue Fließkommadaten unterstützt, im Gegensatz zu der Android eigenen Klasse, bei der nur einfach genaue Fließkommadaten genutzt werden. Die Funktionen dieser Klasse nutzen Matrizen und Vektoren im OpenGL ES-Format, die in 2D-Arrays gespeichert werden. Ebenfalls werden Funktionen zur Rotation, Skalierung und Verschiebung einer Matrix

zur Verfügung gestellt. Zusätzlich liegen auch Funktionen zum Multiplizieren, Invertieren und Transponieren von Matrizen vor.

LoaderAWD Die LoaderAWD-Klasse ist ein Object-Loader, der AWD Dateien laden kann. Im Rajawali-Framework werden Objekte standardmäßig als AWD-Datei gespeichert.

Transformable3D Transformable3D ist der Grundbaustein für alle Objekte, die in dieser Applikation verwendet werden. Jede Objektklasse implementiert diese Klasse und erweitert sie. Transformable3D besitzt einen 3D-Vektor für die Position, Rotation und Skalierung. Zudem existieren ein Quaternion für die Orientierung und ein LookAt-Vektor.

Light Die Licht-Klasse ist eine Erweiterung von Transformable3D. Hier werden Position des Lichts und Richtung des Lichtstrahls in einem Array des Typs „Double“ gespeichert. Ebenfalls wird hier die Lichtstärke gespeichert und festgelegt ob die Lichtquelle eine Punktlichtquelle, eine direktionale Lichtquelle oder ein Spotlight ist.

Texture In der Textur-Klasse kann ein Name gespeichert werden, sowie eine Bitmap, die entsprechend der Texturkoordinaten auf ein Objekt gemappt wird.

Material In der Material-Klasse können Werte für diffuse, spekulare und ambiente Beleuchtung festgelegt werden. Auch existiert hier eine Liste mit Lichtquellen.

Object3D Die Objekthierarchie wird durch eine Baumstruktur definiert. Jedes Objekt hat einen Elternknoten und eine Liste von Kindknoten. Jedem Objekt kann ein Name, ein Material und ein Hüllkörper zugewiesen werden. Außerdem kann man die Sichtbarkeit an- und ausschalten. Die Object3D-Klasse ist eine Erweiterung der Tranformable3D-Klasse. Sie hat eine Model-View-Matrix, eine Model-Matrix, eine Model-View-Projection-Matrix und eine Projection-Matrix, sowie eine Rotation-Matrix. Außerdem werden hier der Elternknoten und alle Kind-Knoten in einer Liste gespeichert. In der Object3D-Klasse kann man auch ein Material und einen Namen anlegen. Ebenfalls wurde hier eine Variable vom Typ „Integer“ implementiert, die in jedem Renderschritt erhöht wird, um zu ermitteln wie lange eine Instanz der Object3D-Klasse bereits gerendert wird.

Plane Die Plane-Klasse gehört zur Klasse der Primitiven und sie erweitert die Object3D-Klasse. Die Plane-Klasse beschreibt eine Ebene. Hier werden zusätzlich eine Höhe und eine Breite festgelegt, sowie ein Richtungsvektor, in dessen Richtung die Normale der Ebene zeigen soll.

Frustum Die Frustum-Klasse setzt sich aus 6 Plane-Objekten zusammen, die das Frustum bilden. Die Plane-Objekte werden aus der inversen Projection-View-Matrix gebildet.

Camera Die Camera-Klasse ist eine Erweiterung der Klasse Transformable3D. Hier befinden sich zusätzlich eine View-Matrix, Rotation-Matrix und Projection-Matrix vom bereits beschriebenen Matrix-Typ. Darüber hinaus existieren Variablen vom Typ „Double“ für die Near-Plane, Far-Plane und für das Sichtfeld, sowie eine Variable vom Typ „Frustum“ für das Kamera-Frustum. Zudem steht eine Funktion zur Verfügung, um das Frustum zu zeichnen.

HermiteSplines Die HermiteSpline-Klasse ermöglicht die Erzeugung von kubisch hermiteschen Splines. Dazu existiert in dieser Klasse eine Liste, in welcher mehrere Punkte in Form eines 3D-Vektors gespeichert werden können. Zwischen diesen Punkten wird dann eine Kurve interpoliert. Die Kurve besteht aus Segmenten zwischen den Punkten. Für jeden dieser Punkte besitzt der kubisch hermitesche Spline eine Tangente.

TerrainGenerator Durch die TerrainGenerator-Klasse ist es möglich quadratische Terrains zu generieren, wobei eine Bitmap als Tiefenkarte genutzt wird. Dabei nutzt der Generator die r-Komponente, des RGB-Kanals der Bitmap, als Temperatur und die g-Komponente als Tiefenwert. Das Terrain wird erzeugt, indem ein Canvas-Objekt angelegt wird und entsprechend der Informationen in der Bitmap transformiert wird. Ein Canvas-Element ist ein mit Höhen- und Breiten-Angaben beschriebener Bereich, in den per JavaScript gezeichnet werden kann.

Animation Die Animation-Klasse des Rajawali-Frameworks ist die Elternklasse aller Animationsklassen. Hier werden Bestandteile initialisiert, die in allen Animationen Verwendung finden. Dazu gehören Variablen für die Dauer einer Animation, die Dauer die eine Animation bereits aktiv ist, die Startzeit der Animation sowie die Verzögerung mit der sie beginnen soll und auch eine Variable für die Deltazeit. Jeder Animation kann ein Listener zugewiesen werden. Dieser überprüft dauerhaft den Zustand einer Animation.

Hierbei gibt es die Zustände Start, Stopp und Update. Durch den Listener können für jeden dieser Zustände weitere Funktionsaufrufe ausgeführt werden.

Spline-Animation Die Spline-Animation-Klasse ist eine Erweiterung der Animation-Klasse. Diese Animation nutzt Hermite Splines um ein Objekt entlang des Splines zu bewegen. Dabei wird entsprechend der Deltazeit der aktuelle Punkt auf der Kurve errechnet, an dem sich das Objekt befinden sollte und dorthin verschoben. Zusätzlich wird die Tangente an der Position des Punktes berechnet und als Look-At-Vektor des Objekts genutzt.

Translate-Animation Die Translate-Animation-Klasse ist eine Erweiterung der Animation-Klasse. Diese Animation wird für eine Bewegungsanimation eines Objekts von einem Punkt zu einem anderen genutzt. Dabei wird zwischen diesen beiden Punkten ein Vektor generiert. Entsprechend der Deltazeit wird bei jeder Aktualisierung der Animation der Punkt berechnet an dem sich das Objekt befinden sollte und an diesen Punkt verschoben.

Scale-Animation Die Scale-Animation-Klasse ist eine Erweiterung der Animation-Klasse. Sie wird zur Animation veränderlicher Größen genutzt. Die Animation benötigt einen Vektor mit drei Komponenten, in denen festgelegt wird, um welchen Wert eine Skalierung um die x-, y- und z-Achse stattfinden soll. Die Animation findet statt, indem in jedem Renderdurchgang die aktuelle Skalierung des zu animierenden Objektes abgefragt wird und ein Vektor zwischen der aktuellen Größe und der zu erreichenden Größe erzeugt wird. Entsprechend der Deltazeit werden mit diesem Vektor die Werte errechnet, die den aktuellen Werten der x-, y- und z-Komponenten hinzu addiert werden, um so die neuen Werte der Skalierung zu erhalten.

Scene Die Szene verarbeitet alle Inhalte einer 3D-Szene, aber überlässt es dem Renderer den OpenGL Kontext zu verarbeiten. Die Szene hat eine Liste mit Objekten, die der Benutzer füllen kann. Die Szene beinhaltet zusätzlich Listen für Animationen und Kameras, um sie jederzeit hinzufügen, entfernen, ersetzen oder wechseln zu können. Alle Objekte, die in der Liste vorhanden sind, werden gerendert und sind somit in der Szene sichtbar.

Renderer Der Renderer ist eine Erweiterung des GLSurfaceView Renderers. Der GLSurfaceView Renderer ist ein generisches Renderer Interface. Der Rajawali Framework Renderer ist für die Regelung des gesamten OpenGL Kontextes zuständig. Er erlaubt es, zu jeder Zeit Szenen hinzuzufügen, zu entfernen, ersetzen oder zu wechseln.

4 Game-Design

In diesem Kapitel wird die anfänglich definierte Spielidee weiter ausgearbeitet.

4.1 Spielkonzept

Die Spielidee besteht darin, dass der Spieler versuchen muss, gegnerische Raumschiffe durch Geschosse zu zerstören. Damit ein Level beendet werden kann, müssen die Lebenspunkte des Spielers auf null sinken. Die gegnerischen Raumschiffe werden regelmäßig, nachdem sie erschienen sind, einen Angriff abfeuern. Diesem kann man nicht ausweichen, was dazu führt, dass man einen Lebenspunkt verliert. Das Ziel ist es dabei möglichst schnell gegnerische Raumschiffe zu zerstören, bevor sie den Spieler angreifen können. Bei der Zerstörung eines gegnerischen Raumschiffes erhält der Spieler einen Punkt. Der Spieler soll so viele Punkte sammeln, wie es ihm möglich ist, bevor seine Lebenspunkte auf null sinken. Jedoch muss der Spieler ein gegnerisches Raumschiff in der Welt erst finden, da es entlang einer zufälligen Route fliegt. Am Ende der Runde werden die erlangten Punkte in einer Punkteübersicht eingespeichert, falls der höchste je erlangte Punktewert übertroffen wurde. Das Spielprinzip baut also auf dem Ehrgeiz eines Spielers auf, seine eigenen Leistungen oder die Leistungen anderer Spieler zu überbieten.

Grundsätzlich kann ein Level einige Sekunden bis hin zu vielen Minuten andauern. Die Bedienung ist intuitiv da über Sensoren gesteuert wird und lediglich ein Knopf existiert, der zum Abfeuern der Geschosse genutzt wird.

4.1.1 Schauplatz

Der Schauplatz des Geschehens ist die Oberfläche eines Asteroiden im Weltraum, da dies eine sinnvolle Umgebung für ein Spiel ist, bei dem es um Raumschiffe geht. Die Oberfläche des Asteroiden ist rötlich und zerklüftet und im Hintergrund sieht man den Weltraum. Der Hintergrund ist dabei permanent in Bewegung, um mehr Dynamik herzustellen, da sonst der gesamte Schauplatz zu statisch wirkt.

4.1.2 Eingabemethoden & Sensoreinbindung

Als Eingabemethode werden der Gyrosensor und der Beschleunigungssensor genutzt. Dabei wird die von Google bereitgestellte HeadTracker-Klasse verwendet. Die Klasse stellt eine View-Matrix zu Verfügung, in der die aktuelle Orientierung des Smartphones gespeichert ist. Zudem enthält die Klasse einen Sensor-Event-Listener, der permanent überprüft ob die Sensorwerte sich verändern. Wenn dieser Fall eintritt, wird die View-Matrix entsprechend aktualisiert. Darüber hinaus bietet diese Klasse eine Funktion, mit der man zu jeder Zeit die letzte aktuelle View-Matrix erhalten kann.

In der Applikation wird ein Objekt der Klasse HeadTracker im Renderer instanziiert. In jedem Renderdurchlauf wird dann die letzte aktuelle View-Matrix abgefragt und ihre Werte invertiert, um sie als Kameraorientierungskoordinaten zu verwenden.

4.1.3 Angriffswelle

Jede Angriffswelle besteht immer nur aus einem Raumschiff, welches an einer zufälligen Position in der Welt erscheint. Würde die Position an der ein Raumschiff erscheint nicht zufällig sein, würde die Herausforderung des Spiels verloren gehen, da ein Spieler nur noch Projektile an die Position feuern müsste, an der die Raumschiffe erscheinen, um eine sehr hohe Punktzahl zu erreichen. Zudem müsste der Spieler das gegnerische Raumschiff nicht mehr in der Welt suchen, was das Spielkonzept zerstören würde. In ersten Versuchen bestand eine Angriffswelle aus mehr als einem Raumschiff, was jedoch zu einem sehr anstrengenden beziehungsweise unangenehmen Bewegungsablauf bei der Suche nach mehreren Raumschiffen führte, da das schnelle anvisieren von Gegnern an verschiedenen Positionen über eine Sensorsteuerung sehr hektisch werden kann.

4.1.4 Physik

Da das Spielgeschehen im Weltraum stattfindet, gibt es keine physikalischen Regeln, wie zum Beispiel Schwerkraft, die eingehalten werden müssen. Die einzigen physikalischen Aspekte die beachtet werden müssen, sind die Kollisionen der Projektile mit den Raumschiffen. Da jedoch kein Objektverhalten bei einer Kollision berechnet werden muss, da nur eine Explosion an der Kollisionsstelle ausgelöst wird, sind in der gesamten Applikation keine physikalischen Berechnungen notwendig.

4.1.5 Schwierigkeitsgrad

Der Schwierigkeitsgrad im Spiel ist nicht veränderbar. Jedoch wird während des Spielverlaufs der Schwierigkeitsgrad kontinuierlich erhöht, da nach jeder verstrichenen Minute das Intervall, in dem die gegnerischen Raumschiffe erzeugt werden, verkürzt wird. Dies ist

wichtig um einerseits die Dauer einer Spielrunde zu begrenzen, da ein erfahrener Spieler bei einem gleichbleibenden Intervall, unter Umständen endlos eine Spielrunde fortführen könnte und andererseits, um das Spiel auch fordernd zu gestalten.

4.1.6 Grafik

Die Grafik ist eine recht simple, da sämtliche Szenen mit OpenGL ES gezeichnet werden und keine speziellen Shader-Effekte verwendet werden. Sämtliche Inhalte des Spiels werden durch Bitmap-Texturen, die auf Materialien gemappt werden dargestellt. Die Bitmaps haben dabei eine relativ geringe Auflösung von 800x600 abwärts, um das schnelle Laden der Texturen zu ermöglichen.

4.1.7 Töne

Bei jedem Erscheinen eines gegnerisches Raumschiffes, bei jedem Treffer den der Spieler erleidet und bei jeder Explosion wird eine passende Tondatei abgespielt. In vorherigen Prototypen der Applikation wurden noch Töne bei jedem Abschuss eines Projektils abgespielt, was jedoch auf Dauer zu einem geminderten Spielerlebnis führte, da es sehr Ablenkend war.

4.1.8 Belohnung

Als Belohnung erhält der Spieler nach jedem erfolgreichen Abschuss einen Punkt. Nach einer absolvierten Spielrunde, werden die Punkte eines Spielers mit dem höchsten je erzielten Punktestand verglichen. Falls seine Punkte den höchsten Punktestand übertreffen, wird sein Punktestand als neuer Highscore eingetragen.

4.1.9 Gameplay

Der Spieler findet sich auf einem Asteroiden im Weltraum wieder und er besitzt am Anfang 20 Lebenspunkte, die als farbige Balken am oberen Bildschirmrand angezeigt werden. Alle acht Sekunden erscheint ein gegnerisches Raumschiff, dessen Ziel es ist die Lebenspunkte des Spielers auf null zu senken. Wenn ein gegnerisches Raumschiff bereits acht Sekunden lang existiert, gibt es einen Schuss auf den Spieler ab, der dadurch einen Lebenspunkt verliert. Durch eine Berührung des Feuerknopfes am unteren rechten Bildschirmrand, kann ein Projektil abgefeuert werden. Falls dieses Projektil ein gegnerisches Raumschiff trifft, explodiert das Raumschiff. Durch die Zerstörung eines Raumschiffes erhält der Spieler einen Punkt zu seinem Punktestand hinzu, der im oberen linken Bildschirmrand angezeigt wird. Jede verstrichene Minute erhöht sich die Frequenz, mit der die gegnerischen Raumschiffe erscheinen. Falls die Lebenspunkte des Spielers auf null sinken, wird ein neues Fenster geöffnet, in welchem „Game Over“ steht. Danach wird wieder ein neues Fenster

geöffnet, in welchem eine Punkteübersicht angezeigt wird. In die Punkteübersicht werden die Highscores eingetragen.

5 Implementation

5.1 Testsystem

Das Gerät, auf welchem die Applikation getestet wird, ist das G3 Smartphone von LG Electronics. Das Smartphone ist mit einem Quadcore-Prozessor von Qualcomm ausgestattet. Jeder der Kerne besitzt eine Taktrate von 2,5 Gigahertz. Das Smartphone besitzt ebenfalls 2 Gigabyte Arbeitsspeicher. Das Display ist ein 5,5 Zoll großes IPS-Display, mit einer Auflösung von 2.560 x 1.440 Pixel. Genutzt wird eine Adreno 330 GPU. Das installierte Betriebssystem ist Android 5.0 Lollipop. Es gehört zu den aktuell Leistungsfähigsten Smartphones auf dem Markt und bietet somit genug Leistung zur Implementation der Applikation, die im Rahmen der Bachelorarbeit entwickelt wird.

5.2 Klassenunterteilung

Die Applikation ist in zwei Hauptklassen unterteilt. Eine GameActivity-Klasse und eine GameRenderer-Klasse. Die GameRenderer-Klasse ist dabei eine Erweiterung des Rajawali Framework Renderers. Darüber hinaus sind drei weitere Klassen für das Hauptmenü, der Game Over Benachrichtigung und der Highscore-Tabelle vorhanden. Zunächst beginnt die Implementation mit der Erstellung der GameActivity-Klasse. Dort wird festgelegt was bei dem Start der Applikation geschehen soll und zudem wird in der Activity-Klasse das Graphical User Interface angelegt, sowie die Spiellogik bestimmt.

5.3 Graphical User Interface

Zu Beginn wird eine Instanz der GLSurfaceView-Klasse initialisiert. Diese Klasse verwaltet die sichtbare Bildschirmoberfläche, die ein besonderer Teil des Speichers ist und in das View-System von Android eingefügt werden kann. Außerdem verwaltet GLSurfaceView ein EGL-Display, um OpenGL das Rendern auf eine Oberfläche zu ermöglichen. Ebenfalls akzeptiert die Klasse ein benutzerdefiniertes Renderer-Objekt, welches das eigentliche Rendering übernimmt. Weiterhin werden alle zu Android gehörenden Menü- und Navigationsleisten im SurfaceView ausgeblendet, sowie die Titelleiste der Applikation entfernt, um der Applikation den vollen Bildschirm zur Verfügung zu stellen. Zudem werden hier weitere Einstellungen vorgenommen, um zu verhindern, dass der Bildschirm nach einigen Sekunden Inaktivität abschaltet. In den nächsten Schritten wird das Graphical User Interface implementiert. Dazu wird ein von Android zur Verfügung gestelltes FrameLayout-Objekt genutzt. Dort ist es möglich grafische User-Elemente hinzufügen. Das FrameLayout-Objekt wird dann dem GLSurfaceView-Objekt hinzugefügt und befindet sich somit immer im Vordergrund der Applikation. Um das Abfeuern der Projektile zu ermöglichen, wird ein Knopf erzeugt, welchem zwei Texturen zugeordnet werden. Der

Knopf ist dabei, ein durch Android zur Verfügung gestelltes Button-Element, welchem eine Textur und ein Listener zugeordnet werden kann. Die Texturen sind dabei eine oranger Kreis und ein roter Kreis. Der Knopf hat standardmäßig die Farbe Rot und falls er gedrückt wird wechselt die Farbe auf Orange, indem eine neue Zuordnung der Knopftextur erfolgt, falls der Listener eine Betätigung des Knopfes wahrnimmt. Der Knopf wird dann in die rechte untere Ecke des FrameLayout-Objekts eingefügt.

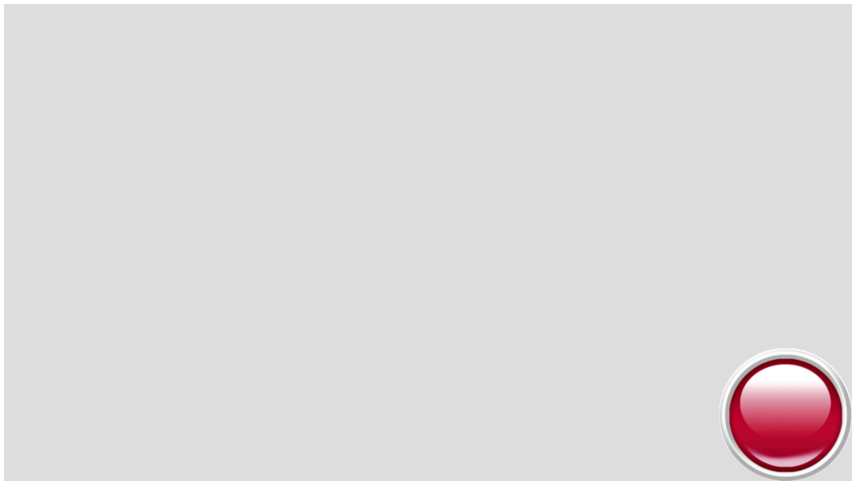

Abbildung 3: Knopf zum Abfeuern der Projektile falls nicht gedrückt

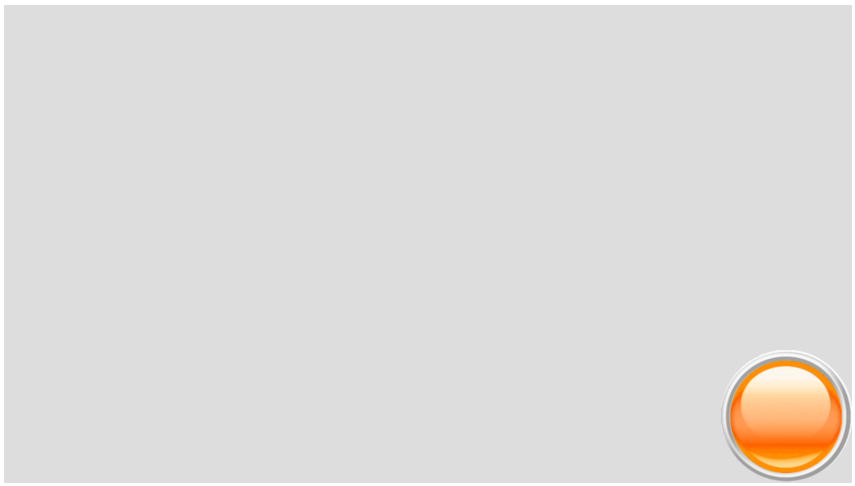

Abbildung 4: Knopf zum Abfeuern der Projektile falls gedrückt

Weiterhin wird eine Fadenkreuz-Textur mit transparentem Hintergrund im PNG-Format in die Mitte des FrameLayout-Objekts eingefügt um das Zielen zu ermöglichen.

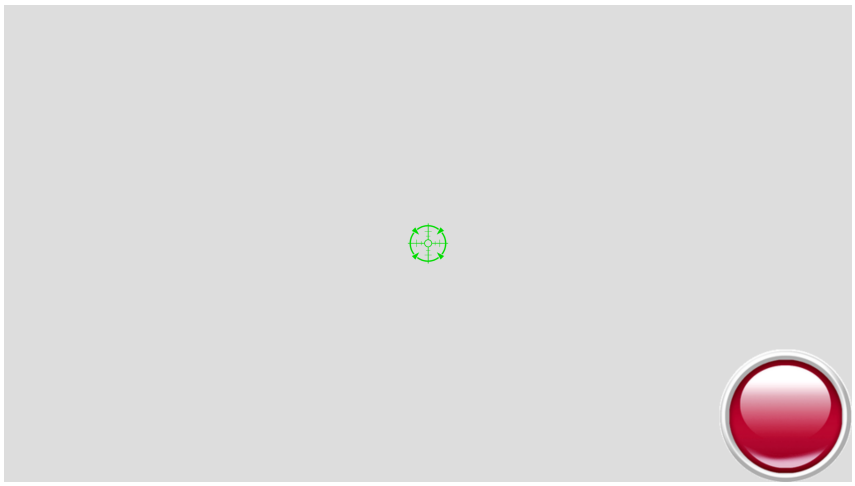

Abbildung 5: Fadenkreuz

Zur Anzeige der Lebenspunkte werden fünf Balken, mithilfe des von Android zur Verfügung gestellten LinearLayout-Objekts, in der Mitte des oberen Randes eingefügt und ihnen wird eine variable Farbe zugewiesen, die je nach Lebenspunktzahl wechselt. Das LinearLayout-Element wird normalerweise benutzt, um alle ihm hinzugefügten Kinder entweder horizontal oder vertikal innerhalb des LinearLayout-Elements zu ordnen. In diesem Fall jedoch wird dem LinearLayout nur eine Farbe zugewiesen. Bei voller Lebenspunktzahl sind alle Lebensbalken blau. Bei Verlust von Lebenspunkten wechselt die Farbe des rechten Lebensbalkens von Blau zu Grün zu Gelb zu Rot und schließlich zu Grau. Sobald ein Lebensbalken die Farbe Grau annimmt, verändert der benachbarte linke Lebensbalken die Farbe, bis schließlich alle Lebensbalken Grau sind.

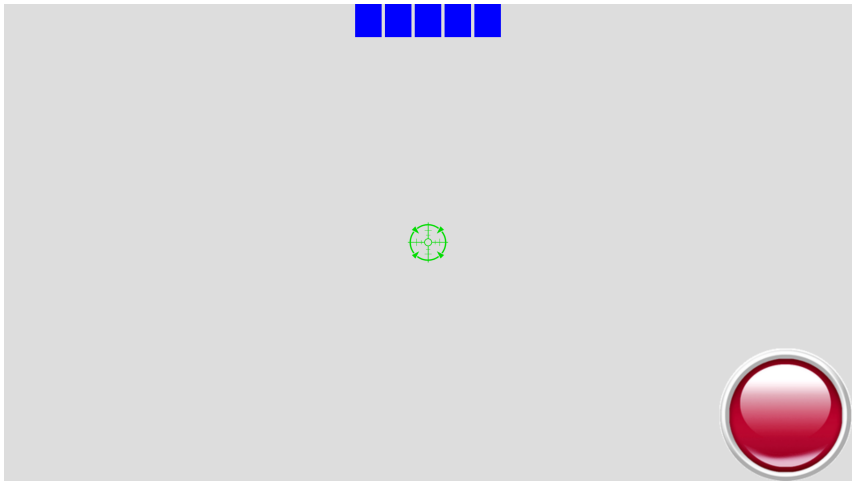

Abbildung 6: Lebensbalken bei vollen Lebenspunkten

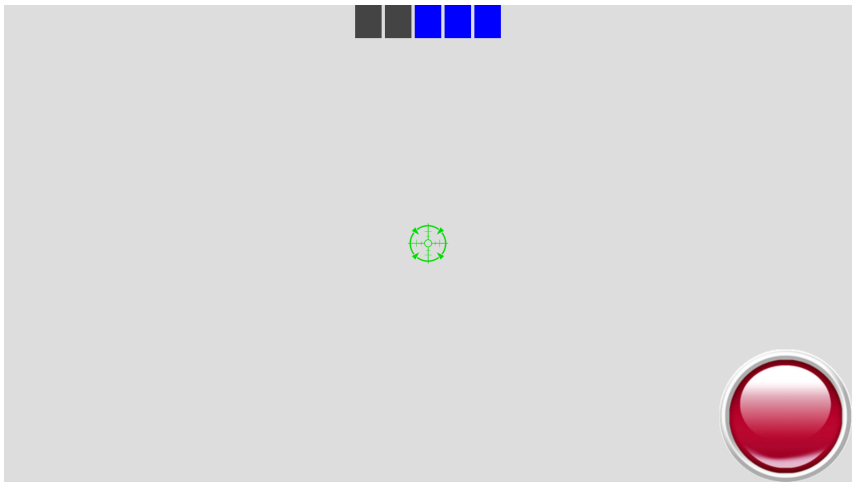

Abbildung 7: Lebensbalken bei bereits verlorenen Lebenspunkten

Der letzte Schritt bei der Implementierung des Graphical User Interface, ist die Anzeige der Punkte auf dem Display. Dazu wird ein TextView-Objekt in die obere linke Ecke des FrameLayouts eingefügt und mit einer String Variable belegt. Dieser Variable werden die erlangten Punkte durch eine Umwandlung in einen String zugewiesen. Das TextView-Element dient dazu, Text für den User darzustellen und dem User optional zu erlauben den Text zu editieren. Dabei ist das TextView-Element ein vollständiger Texteditor, jedoch ist die Klasse standardmäßig so konfiguriert, dass das Editieren von Text nicht erlaubt ist.

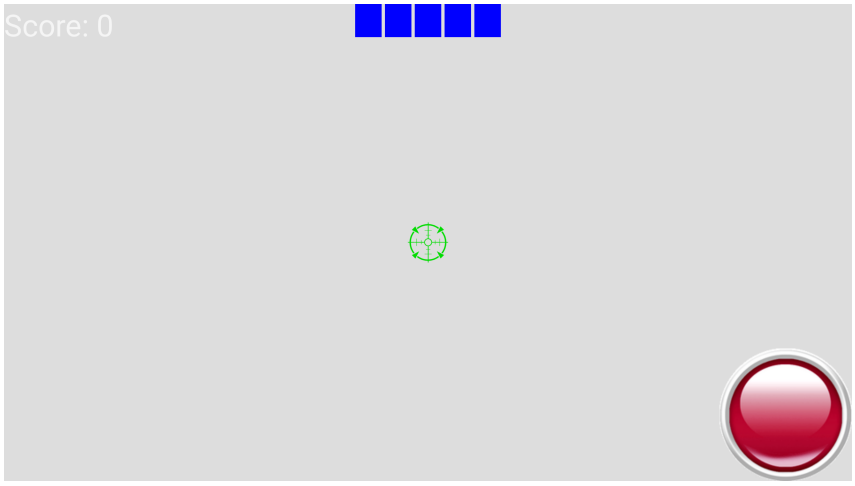

Abbildung 8: Punktanzeige

Der letzte Schritt bei der Erstellung der Activity-Klasse, ist die Implementierung der Spiellogik. Dieser Schritt wird in Kapitel 5.15 erläutert.

5.4 Headtracker

Als nächstes findet die Implementierung der GameRenderer-Klasse statt. Der GameRenderer ist eine Erweiterung des Rajawali-Framework Renderers. Im GameRenderer wird die gesamte Szene erzeugt und zwischengespeichert. Der erste Schritt bei der Implementierung des Renderers, ist die Initialisierung eines Headtrackers und eines Quaternions für die Kameraorientierung, sowie die Initialisierung der Kamera selbst, welche im Ursprung platziert ist. In jedem Renderschritt wird die View-Matrix des Headtrackers ausgelesen und die einzelnen Komponenten invertiert, um die Kameraorientierungskoordinaten zu erhalten. Diese werden anschließend als Quaternion abgespeichert.

5.5 Generierung der Szene

Im weiteren Verlauf werden Arrays für Objekte, Splines und Töne angelegt, um diese zwischenzuspeichern und damit die Performanz zu erhöhen, sowie eine Liste die alle zu rendernden Szenen beinhaltet. In dieser Applikation wird dabei nur eine Szene verwendet. Zusätzlich wird ein LoaderAWD-Objekt initialisiert, sowie Variablen, um den Punktestand und die Menge der aktiven gegnerischen Raumschiffe zu speichern. Im Renderer wird auch die gesamte Szene generiert. Dazu wird ein direktionales Licht angelegt und als Beleuchtungsmodell wird das vom Framework bereitgestellte Lambert Beleuchtungsmodell gewählt. Das Lambert Beleuchtungsmodell basiert auf dem Lambertschen Gesetz, welches die Winkelabhängigkeit der Lichtstärke auf einer Diffusen Oberfläche beschreibt. Je kleiner der Winkel des einfallenden Lichts zur Oberflächennormale eines Körpers ist, umso intensiver ist es. Nach der Einstellung des Lichts, wird ein Terrain mithilfe des vom Rajawali-Framework zur Verfügung gestellten Terrain-Generators erzeugt, indem aus einer Höhenkarte einer zerklüfteten Landschaft ein Terrain-Objekt generiert wird und der Szene als Knoten hinzugefügt wird. Dem Terrain werden zusätzlich ein grob strukturiertes Material für eine Asteroiden ähnliche Oberfläche, sowie eine Normalmap für eine korrekte Beleuchtung hinzugefügt. Dem Material wird darüber hinaus eine dunkelrote Farbe zugewiesen.

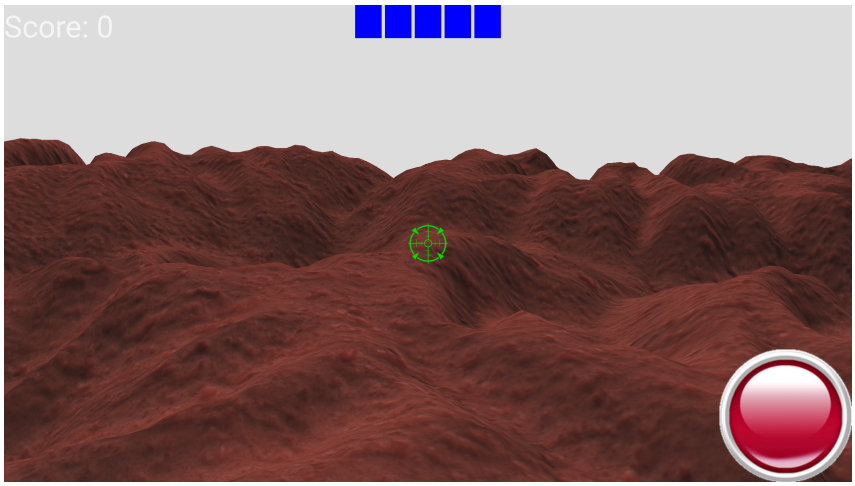

Abbildung 9: Terrain

Nach der Generierung des Terrains, wird die Skybox erstellt. Hierfür werden sechs Skybox-Texturen an ein Würfelprimitiv gemappt, welches das Terrain umgibt. Das Primitiv wird dann der Szene hinzugefügt.

Als nächstes werden die Pfade für die Raumschiffe erzeugt, indem die Hermite Splines des Rajawali-Frameworks genutzt werden. Dafür bilden 4 Punkte, wobei die x-, y- und z-Koordinate zufällige Werte erhalten, einen Hermite Spline. Insgesamt werden 50 Hermite Splines in einem Array zwischengespeichert, um die wiederholte Generierung von Splines zur Laufzeit zu verhindern und die Performanz der Applikation somit zu erhöhen. Der nächste Schritt ist die Initialisierung und die Zwischenspeicherung der Raumschiff-Objekte. Dazu wird über das LoaderAWD-Objekt die AWD-Datei des gegnerischen Raumschiff-Objekts geladen und als ein Object3D-Objekt gespeichert. Die Lambert Beleuchtungsmethode wird auch hier verwendet. Insgesamt werden 50 Raumschiffe in einem Array zwischengespeichert, um die Performanz weiter zu erhöhen. Jedem Raumschiff-Objekt wird eine Raumschiff-Textur und eine bisher nicht genutzte Spline-Animation zugewiesen.

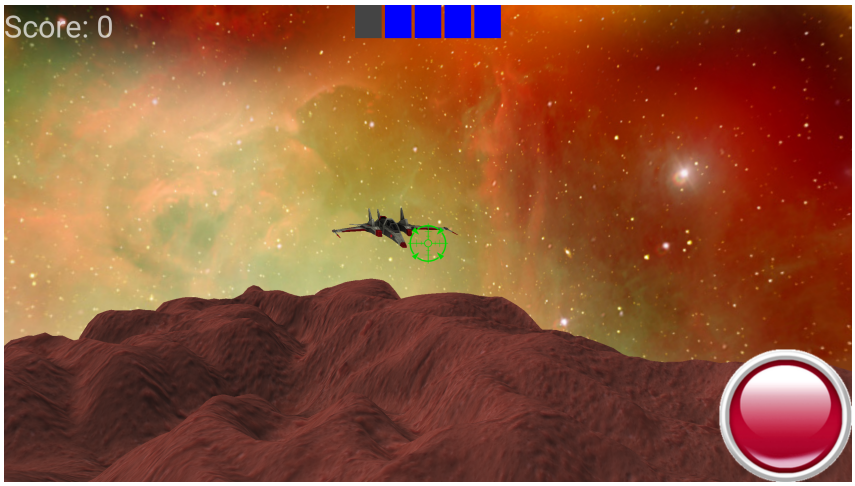

Abbildung 10: Gegnerisches Raumschiff

Die Spline-Animation nutzt dabei einen bisher nicht verwendeten zwischengespeicherten Pfad. Um sicherzustellen, dass ein Pfad nicht mehrfach genutzt wird, werden in einer Schleife die Pfade im Array der Reihe nach einer Splne-Animation zugewiesen. Die Pfade werden dann dem Raumschiff an derselben Position des Arrays, in welchem die Raumschiffe gespeichert sind, zugewiesen. Die AWD-Datei und die Raumschiff-Textur sind für jedes Raumschiff gleich, daher unterscheiden sie sich nicht voneinander.

5.6 Mediaplayer

Nach diesem Schritt findet die Initialisierung der Toneffekte statt. Hierbei wird der durch Android zur Verfügung gestellte Mediaplayer genutzt. Die Mediaplayer-Klasse kann dazu genutzt werden, um die Wiedergabe von Audiodaten, Videodaten und Streams zu kontrollieren. Explosionen und das Erscheinen gegnerischer Raumschiffe, sowie der Verlust von Lebenspunkten erhalten einen eigenen Toneffekt. Daher werden drei Mediaplayer initialisiert. Einem Mediaplayer wird ein passender Ton für Explosionen, einem ein Ton für das Erscheinen von Gegnern und dem letzten Mediaplayer ein Ton für den Lebenspunktverlust zugewiesen. Dadurch können nun zu jeder Zeit die Toneffekte abgespielt werden. Um ein Raumschiff in der Szene erscheinen zu lassen, wird eine Methode implementiert, mit der ein bisher nicht verwendetes Raumschiff aus dem Array als Knoten an die Szene übergeben wird. Das Raumschiff erhält dabei eine zufällige Startposition. Dabei wird auch der Ton, der das Erscheinen von Raumschiffen signalisiert, durch den Mediaplayer abgespielt. Damit sind alle Grundbausteine für die Applikation gelegt, womit die Implementation der Spiellogik und der Animationen beginnen kann.

5.7 Spiellogik

Für die Spiellogik wird in die GameActivity-Klasse ein Handler eingefügt, der mit einer Verzögerung von 500 Millisekunden überprüft, ob die Lebenspunkte bei null liegen. Falls dies der Fall ist wird die Game-Over-Klasse ausgeführt, die in Kapitel 5.12 erläutert wird. Wenn es nicht der Fall ist, wird eine Variable vom Typ „Float“ inkrementiert, um zu ermitteln wie lange das Spiel bereits aktiv ist. Sobald acht Sekunden verstrichen sind und die Lebenspunkte mehr als null betragen, wird die Funktion zum Erzeugen eines Raumschiffes durch den Handler aufgerufen. Zusätzlich wird bei jedem Aufruf des Handlers der Punktestand aktualisiert, indem der Inhalt der Punktestand-Variablen im Renderer durch den Inhalt der Stringvariablen des TextView-Objektes im FrameLayout, welches die Punkte anzeigt, ersetzt wird. Nach 60 Sekunden wird die Zahl erhöht, um welche die Variable zur Überprüfung der Aktivitätsdauer des Spiels inkrementiert wird, sodass der Schwierigkeitsgrad des Spiels angehoben wird. Darüber hinaus wird eine Variable vom Typ „Integer“ implementiert, welche die Lebenspunkte speichert. Diese Variable wird vom Handler auf ihren Wert überprüft, um die Farbe der Lebensbalken entsprechend anzupassen und festzustellen wann das Spiel endet. Damit gegnerische Raumschiffe den Spieler angreifen können, wird im Renderer eine Variable vom Typ „Boolean“ eingefügt. Diese gibt an, ob ein Raumschiff lange genug aktiv ist, um dem Spieler einen Lebenspunkt abzuziehen. Im Renderer wird dafür ebenfalls, in jedem Frame, der Timer jedes aktiven Raumschiffes erhöht und zeitgleich überprüft, ob der Timer eines Raumschiffes einem Vielfachen von 480 entspricht. Ist dies der Fall wird die Variable auf Wahr gesetzt. Dadurch wird angegeben, dass der Spieler nun einen Lebenspunkt verlieren muss. Dies funktioniert jedoch nur, solange das Spiel mit 60 Frames pro Sekunde ausgeführt wird, da sonst nicht sichergestellt werden kann, dass acht Sekunden vergangen sind. Um den Lebenspunktverlust zu realisieren, wird im Handler der GameActivity-Klasse zusätzlich überprüft, welchen Inhalt die Boolean-Variable hat. Falls die Variable Wahr ist, werden die Lebenspunkte um eins gesenkt und die Variable wieder auf Falsch gesetzt und zusätzlich wird durch den Mediaplayer ein Ton ausgelöst, wodurch der Spieler gewarnt wird, dass er einen Lebenspunkt verloren hat. Darüber hinaus wird ein visueller Effekt eingefügt, welcher ausgelöst wird falls der Spieler einen Lebenspunkt verliert. Dabei wird neben dem Abspielen des Tons eine Textur in der Mitte des Bildschirms eingeblendet. Die Textur soll an einen Blutspritzer erinnern. Diese wird als Bitmap-Datei in ein ImageView-Objekt geladen und über einen Zeitraum von zwei Sekunden in der Bildschirmmitte dargestellt. Nach Ablauf der zwei Sekunden wird das ImageView-Objekt ausgeblendet. Das ImageView-Objekt wird dabei von Android zur Verfügung gestellt. Es wird genutzt, um beliebige Bilder anzuzeigen, wie beispielsweise ein Symbol. Die ImageView-Klasse kann Bilder aus verschiedenen Quellen laden.

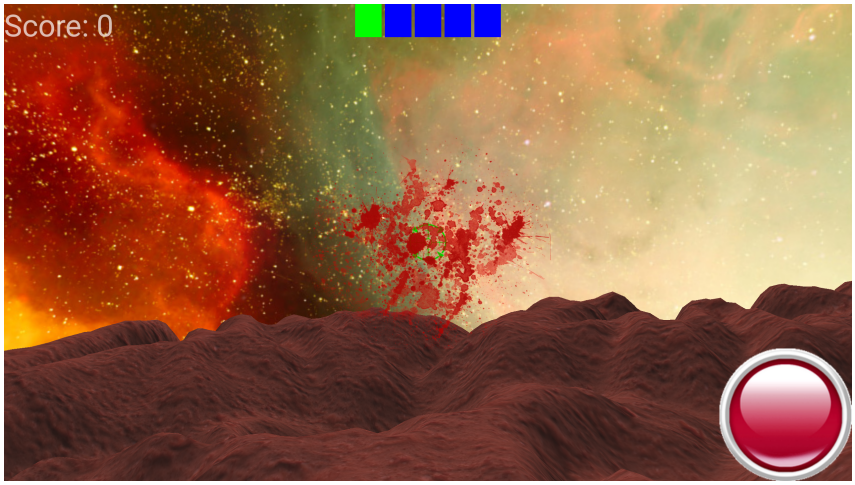

Abbildung 11: Treffereffekt

5.8 Animationen

Zur Umsetzung der Animationen werden die Spline-Animation, Scale-Animation und Translate-Animation des Rajawali-Frameworks genutzt. Jede dieser Animationen wird ebenfalls ein Listener zugeordnet, um weitere Funktionsaufrufe bei jedem Update und am Ende einer Animation ausführen zu können. Die Animation der Raumschiffe wird durch eine Spline-Animation realisiert. Dazu wird, bei der Erzeugung eines Raumschiffes in der Szene, ein zufälliger zwischengespeicherter Pfad, welcher noch nicht verwendet wird, ausgewählt und dem Raumschiff als Animation zugewiesen. Dabei wird die Animation auch in der Szene registriert. Die Animation wird in einer endlosen Schleife wiederholt, bis das Raumschiff, dem sie zugewiesen wurde, zerstört wird. Falls dies passiert, werden jeweils der Raumschiff-Knoten und die Animation aus der Szene gelöscht. Die Animation der Projektile wird durch eine in den Renderer implementierte Funktion ausgelöst. Die Funktion wird bei jeder Betätigung des Knopfes zum Abfeuern der Projektile aufgerufen. In dieser Funktion wird zuerst der aktuelle Look-At-Vektor der Kamera berechnet, indem er aus der dritten Zeile der inversen View-Matrix der Kamera extrahiert wird. Die einzelnen Komponenten werden zusätzlich invertiert. Dadurch wird von Raumkoordinaten zu Weltkoordinaten übersetzt, sodass der Look-At-Vektor in Weltkoordinaten vorliegt. Dann wird ein Punkt auf dem Look-At-Vektor nahe der Far-Plane der Kamera bestimmt. Zu diesem Punkt soll das Projektil fliegen. Die Umsetzung geschieht, indem ein Primitiv in Form einer Sphäre aus dem Rajawali-Framework generiert wird und ihm eine rote Farbe zugewiesen wird. Die Sphäre wird ebenfalls mit der Lambert Beleuchtungsmethode beleuchtet. Im nächsten Schritt wird der Sphäre eine Translate-Animation zugewiesen, wobei die Kameraposition der Startpunkt und der vorher gewählte Punkt im Look-At-

Vektor der Endpunkt ist. Dann wird die Sphäre der Szene, als Knoten, hinzugefügt und die Animation in der Szene registriert. Nach Abschluss der Animation, wenn die Sphäre den Endpunkt erreicht hat, werden die Animation und die Sphäre wieder aus der Szene gelöscht. Die Scale-Animation findet bei der Animation der Explosionen Verwendung. Hierzu wird bei einer Kollision eines Projektils mit einem Raumschiff ein Primitiv, in Form einer Plane, erzeugt. Diesem Primitiv wird dann eine Explosionstextur zugewiesen. Die Plane wird dann durch die Zuweisung einer Scale-Animation innerhalb von zwei Sekunden um das Fünffache in alle Richtungen vergrößert. Darüber hinaus wird die Plane dann der Szene hinzugefügt und die Animation ebenfalls in der Szene registriert. Zusätzlich wird dabei, durch den Mediaplayer, ein Explosionston ausgelöst. Nach Abschluss der Skalierung werden die Plane und die Animation aus der Szene gelöscht.

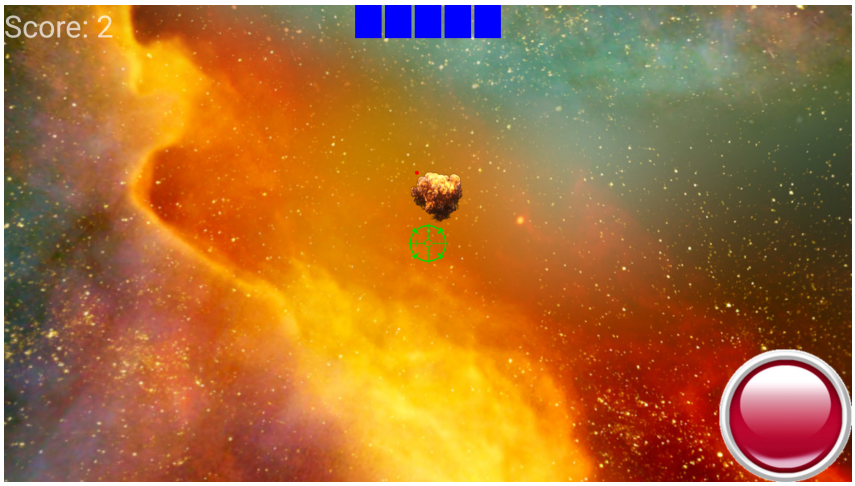

Abbildung 12: Explosion eines gegnerischen Raumschiffs

5.9 Kollisionserkennung

Die Translate-Animation wird zusätzlich für die Kollisionserkennung verwendet. Da die Anzahl der in der Szene dargestellten Objekte sehr begrenzt ist, werden keine Hüllkörper implementiert. Stattdessen wird innerhalb der Translate-Animation, in jedem Aktualisierungsschritt abgefragt, ob die Distanz zwischen der Sphäre und einem aktiven Raumschiff einen Wert von 1,5 unterschreitet. Dies geschieht indem das Array, in welchem die Raumschiffe zwischengespeichert sind, in einer Schleife durchlaufen wird. Dabei wird an jeder Position des Arrays, die Distanz zwischen dem Raumschiff, an dieser Arrayposition und der Sphäre überprüft. Falls die Distanz kleiner oder gleich 1,5 sein sollte, findet eine Kollision statt, Dann wird die Explosionsanimation ausgelöst und zusätzlich das betroffene Raumschiff und seine zugewiesene Animation, sowie die betroffene Sphäre und seine zugewiesen Animation aus der Szene entfernt. Diese Implementation der Kollisionserken-

nung wurde gewählt, da die Anzahl an gleichzeitig in der Szene dargestellten Objekte so gering ist, sodass bei dieser Methode der Kollisionserkennung keine Verringerung der Performanz auftritt, während der Aufwand der Implementation geringer ausfiel, als bei der Implementation von Hüllkörpern.

5.10 Menü

Im Folgenden findet die Implementation des Menüs, der Spielregelanzeige, der GameOver-Klasse und der Bestenliste statt. Für das Menü wird eine neue Activity-Klasse erstellt. In dieser wird, wie auch in der GameActivity-Klasse, die Navigations- und Menüleiste von Android deaktiviert. Zusätzlich wird die horizontale Ausrichtung der Darstellung auf dem Display erzwungen. Im nächsten Schritt werden drei Knöpfe untereinander positioniert. Jeweils einen zum starten des Spiels, einen zur Darstellung des Spielprinzips und einen, um die Applikation wieder zu schließen. Jedem dieser Knöpfe ist ein Listener zugewiesen worden, der die Knöpfe auf eine Betätigung überprüft. Sobald dies der Fall ist, werden weitere Funktionen ausgeführt. Im Fall des Knopfes zum Starten des Spiels, wird ein Intent gestartet, über den die GameActivity-Klasse aufgerufen wird. Ein Intent ist eine abstrakte Beschreibung einer Operation, die durchgeführt werden soll. Es kann verwendet werden, um eine Aktion zu starten. Im Wesentlichen wird ein Intent zum Ausführen von Activity-Klassen genutzt. Man kann sich einen Intent als Kleber zwischen Activity-Klassen vorstellen. Es ist im Grunde eine passive Datenstruktur, die eine abstrakte Beschreibung einer auszuführenden Aktion beinhaltet. Bei der Betätigung des Knopfes zur Anzeige der Spielregeln, wird ebenfalls ein Intent gestartet, durch den die Activity-Klasse für die Spielregeln ausgeführt wird. Bei der Betätigung des Knopfes zum Schließen des Spiels, wird die gesamte Applikation beendet.

Abbildung 13: Hauptmenü

5.11 Spielregeln

In der Spielregel-Klasse werden nur die Menüleiste und die Navigationsleiste von Android deaktiviert, sowie die horizontale Ausrichtung der Darstellung auf dem Display erzwungen. Die Anzeige der Spielregeln wird über eine XML-Datei erreicht, die zu einem späteren Zeitpunkt erläutert wird. Zusätzlich wird ein Knopf in die untere rechte Ecke eingefügt, durch dessen Betätigung eine Rückkehr in das Hauptmenü möglich ist.

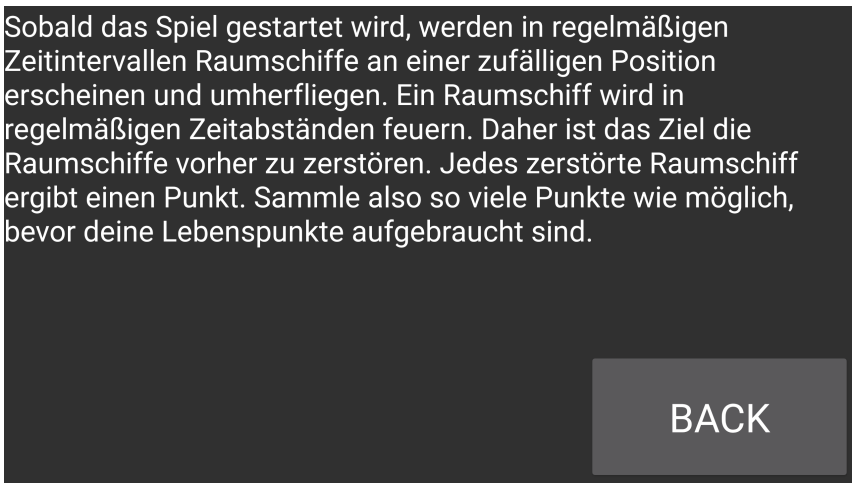

Abbildung 14: Spielregeln

5.12 Spielende

Die Implementation der GameOver-Klasse ist mit der Implementation der Hauptmenü-Klasse weitestgehend identisch. Hierbei handelt es sich ebenfalls um eine Activity-Klasse. Darin werden nur die Menüleiste und die Navigationsleiste von Android deaktiviert, sowie die horizontale Ausrichtung der Darstellung auf dem Display erzwungen. Die Anzeige des Textes wird über eine XML-Datei erreicht. Zusätzlich wird ein Knopf in die untere rechte Ecke eingefügt, durch den die Bestenliste aufgerufen werden kann.

Abbildung 15: Game-Over Anzeige

5.13 Bestenliste

Zur Anzeige des erzielten und des höchsten Punktestandes wird eine neue Activity-Klasse erstellt. Die Menüleiste und die Navigationsleiste von Android werden auch hier deaktiviert, sowie die horizontale Ausrichtung der Darstellung auf dem Display erzwungen. Diese Klasse nutzt SharedPreferences um den Punktestand zu speichern. Mit SharedPreferences können Werte gespeichert werden. Diese sind so lange gespeichert, bis sie überschrieben werden oder die Applikation deinstalliert wird. Genau das wird für den Highscore benötigt. Dazu werden drei Methoden in der GameActivity-Klasse benötigt:

```
1         public int readHighscore() {
2             SharedPreferences pref = getSharedPreferences("GAME", 0);
3             return pref.getInt("HIGHSCORE", 0);
4         }
```

Listing 1: Code-Snippet: GameActivity.java

Hierbei wird ein Objekt des Typs SharedPreferences angelegt, der die SharedPreferences aus „GAME“ liest. Wenn dort kein Wert vorhanden ist, soll 0 genommen werden. Diese Methode hat als Rückgabewert einen Integer-Wert, der dem Highscore entspricht.

```
1         public void writeHighscore(int highscore) {
2             SharedPreferences pref = getSharedPreferences("GAME", 0);
3             SharedPreferences.Editor editor = pref.edit();
4             editor.putInt("HIGHSCORE", highscore);
5             editor.commit();
```

```
6           }
```

Listing 2: Code-Snippet: GameActivity.java

Diese Methode erstellt einen Editor der das Objekt „pref“ bearbeitet und dort den Integer-Wert Highscore einfügt. Danach muss noch ein Commit-Befehl ausgeführt werden, um den Wert zu übergeben.

```
1           public void compareScore() {
2               if (mRenderer.score > readHighscore()) {
3                   writeHighscore(mRenderer.score);
4               }
5           }
```

Listing 3: Code-Snippet: GameActivity.java

Diese Methode soll den neu errungenen Punktestand mit dem alten Highscore vergleichen. Wenn der neue Punktestand höher ist als der Alte, soll der Wert überschrieben werden. Da über SharedPreferences nur die höchste Punktzahl gespeichert werden soll, ist der folgende Code vor dem Aufruf der Highscore-Activity nötig.

```
1           compareScore();
2           Intent i = new Intent(getApplicationContext(), GameOver2.class);
3           i.putExtra("score", mRenderer.score);
```

Listing 4: Code-Snippet: GameActivity.java

Hierbei wird zunächst die erreichte Punktzahl mit dem Rekord verglichen und dem nächsten Intent als eine Zusatzinformation übergeben. Die übergebene Zusatzinformation ist ein String-Wert mit dem Namen „score“ und besitzt den Wert des Punktestands der Renderer-Klasse. Zuletzt wird die GameOver-Klasse als neuer Intent gestartet. In der GameOver-Klasse wird die erreichte Punktzahl lediglich als Zusatzinformation bei der Ausführung der Highscore-Klasse übergeben. Durch die Highscore-Klasse wird nun die erreichte Punktzahl, die als Zusatzinformation erhalten wird, auf dem Bildschirm dargestellt. Zusätzlich wird auf dem Bildschirm, über der erreichten Punktezahl, die höchste jemals erreichte Punktezahl dargestellt. Dazu ist auch in dieser Klasse die Methode zum Auslesen des Highscores erforderlich, um diesen darstellen zu können. Die Informationen werden durch zwei TextView-Objekte dargestellt.

```
1           int score = this.getIntent().getExtras().getInt("score");
```

```
2
3           TextView tvScore1 = (TextView) findViewById(R.id.yourScore);
4           tvScore1.setText("Your score is: " + Integer.toString(score));
5           TextView tvHighscore1 = (TextView) findViewById(R.id.highscore);
6           tvHighscore1.setText("Highscore is: " + Integer.toString(
                readHighscore()));
```

Listing 5: Code-Snippet: Highscore.java

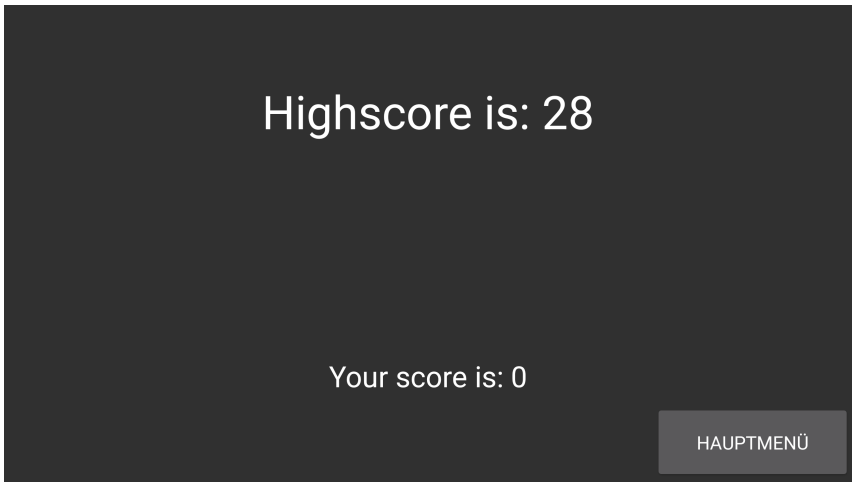

Abbildung 16: Bestenliste

5.14 Android Manifest

Jede Android Applikation besitzt eine AndroidManifest.xml Konfigurationsdatei. Diese Datei enthält essentielle Informationen über die Applikation, welche vom Android System benötigt werden, um diese zu starten und wunschgemäß zu verwalten. Unter anderem sind folgende Informationen im Manifest enthalten:

- Java Package der Applikation (dient im System als ID für die Applikation)
- Deklaration der Applikations-Komponenten (Activities, Services usw.)
- Berechtigungen, die von der Applikation benötigt werden (z.B. Internetzugang)
- Deklaration von benötigten Hard- und Software Features (Multitouch, Kamera usw.)
- Benötigtes API Level (min. SDK Version)
- Third-Party Libraries
- Deklaration des Applikationsicons

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="vr"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="16"
9          android:targetSdkVersion="19" />
10
11     <application
12         android:allowBackup="true"
13         android:icon="@drawable/ic_launcher"
14         android:label="@string/app_name">
15         <activity
16             android:name="GameActivity"
17           android:screenOrientation="landscape"
18             android:label="@string/app_name">
19         </activity>
20         <activity
21             android:name="GameOver"
22             android:screenOrientation="landscape"
23             android:label="@string/game_over">
24         </activity>
25         <activity
26             android:name="Scoring"
27             android:screenOrientation="landscape"
28             android:label="@string/scoring">
29         </activity>
30         <activity
31             android:name="Rules"
32             android:screenOrientation="landscape"
33             android:label="@string/rules">
34         </activity>
35         <activity
36             android:name="MainMenu"
37             android:screenOrientation="landscape"
38             android:label="@string/main_menu">
39             <intent-filter>
40                 <action android:name="android.intent.action.MAIN" />
41                 <category android:name="android.intent.category.DEFAULT"
                       />
42                 <category android:name="android.intent.category.LAUNCHER
                       " />
43             </intent-filter>
44         </activity>
45     </application>
46
47 </manifest>
```

Listing 6: Code-Snippet: AndroidManifest.xml

Wie bereits beschrieben, besteht das Spiel nur aus wenigen Activity-Klassen. Da auch ansonsten keine speziellen Konfigurationen nötig sind, ist das Manifest kurz und übersichtlich. Das als Applikationsicon verwendete Bild zeigt dabei ein Raumschiff in einem Weltraumszenario und passt somit zum Konzept des Spiels.

Abbildung 17: Applikationsicon

5.15 XML Dateien

Ein Layout definiert die optische Struktur für eine Benutzerschnittstelle, wie beispielsweise das User Interface für eine Tätigkeit oder Widgets. Im Rahmen dieser Bachelorarbeit wird das Layout jeder einzelnen Klasse, mit Ausnahme der GameActivity-Klasse, durch XML-Dateien deklariert. Android bietet ein einfaches XML-Vokabular, durch welches ein Layout definiert werden kann. Der Vorteil ein User Interface in XML zu deklarieren ergibt sich durch die Trennung der Präsentation der Applikation vom Code, der das Verhalten steuert. Die Deklaration des User Interface befindet sich in einer externen XML-Datei, wodurch das User Interface modifiziert werden kann ohne den Quellcode zu verändern und neu kompilieren zu müssen. Zusätzlich wird das Debuggen von Problemen, durch die Nutzung von XML, vereinfacht, da die Struktur des User Interface einfacher zu visualisieren ist.

5.15.1 Menü

In der zum Menü gehörigen XML-Datei wird zunächst ein Hintergrundbild festgelegt, in welchem eine Weltraumszenerie zu sehen ist. Danach werden drei Knöpfe deklariert. Jeweils einen zum starten des Spiels, zur Anzeige der Spielregeln und zum Schließen der Applikation. Die Opazität der Knöpfe wird reduziert, um den Hintergrund durch die Knöpfe hindurch sehen zu können.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
3      android:layout_width="match_parent" android:layout_height="
           match_parent"
4      android:background="@drawable/background">
5
6      <Button
7          android:layout_width="300dp"
8          android:layout_height="100dp"
9          android:text="Start Game"
10         android:id="@+id/button"
11         android:textSize="30dp"
12         android:alpha="0.6"
13         android:layout_alignParentTop="true"
14         android:layout_alignParentLeft="true"
15         android:layout_alignParentStart="true" />
16
17     <Button
18         android:layout_width="300dp"
19         android:layout_height="100dp"
20         android:text="Rules"
21         android:id="@+id/button2"
22         android:layout_centerVertical="true"
23         android:layout_centerHorizontal="true"
24         android:textSize="30dp"
25         android:alpha="0.6" />
26
27     <Button
28         android:layout_width="300dp"
29         android:layout_height="100dp"
30         android:text="Exit"
```

```
31          android:id="@+id/button3"
32          android:layout_alignParentBottom="true"
33          android:layout_centerHorizontal="true"
34          android:textSize="30dp"
35          android:alpha="0.6" />
36  </RelativeLayout>
```

Listing 7: Code-Snippet: main-menu.xml

5.15.2 Game-Over

In der zur Game-Over-Klasse gehörigen XML-Datei wird lediglich ein TextView-Element mit dem Text Game-Over deklariert. Zusätzlich wird der Knopf deklariert, durch welchen ein Benutzer zum Highscore gelangt.

```
1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
2       xmlns:tools="http://schemas.android.com/tools"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       tools:context="${packageName}.${activityClass}">
6
7       <TextView
8           android:layout_width="wrap_content"
9           android:layout_height="wrap_content"
10
11          android:text="@string/game_over"
12          android:textSize="50sp"
13          android:textStyle="bold"
14          android:id="@+id/game"
15          android:layout_alignParentTop="true"
16          android:layout_alignParentLeft="true"
17          android:layout_alignParentStart="false"
18          android:layout_alignParentBottom="true"
19          android:singleLine="false"
20          android:gravity="center"
21          android:minWidth="300dp"
22          android:layout_alignParentRight="true"
23          android:layout_alignParentEnd="true" />
24
25      <Button
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:text="Weiter"
29          android:id="@+id/button5"
30          android:layout_alignParentBottom="true"
31          android:layout_alignParentRight="true"
32          android:layout_alignParentEnd="true"
33          android:minWidth="200dp"
34          android:minHeight="100dp" />
35
36  </RelativeLayout>
```

Listing 8: Code-Snippet: game-over.xml

5.15.3 Highscore

In der zur Highscore-Klasse gehörigen XML-Datei wird ein TextView-Element für den höchsten jemals erzielten Punktestand deklariert. Zusätzlich wird ein TextView-Element

zur Anzeige des erzielten Punktestands und ein Knopf zur Weiterleitung zum Hauptmenü deklariert. Der Inhalt der TextView-Elemente wird zur Laufzeit bestimmt.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:layout_width="match_parent" android:layout_height="
        match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Highscore"
        android:id="@+id/highscore"
        android:minWidth="300dp"
        android:gravity="center"
        android:textSize="35dp"
        android:minHeight="160dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Your_Score1"
        android:id="@+id/yourScore"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:minHeight="160dp"
        android:minWidth="300dp"
        android:gravity="center"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hauptmenue"
        android:id="@+id/button6"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:minHeight="60dp"
        android:minWidth="150dp" />

</RelativeLayout>
```

Listing 9: Code-Snippet: highscore.xml

5.15.4 Game

Die zur GameActivity-Klasse gehörende XML-Datei ist leer, da das Layout hier, wie bereits in Kapitel 5.3 beschrieben, durch Programmcode umgesetzt wird.

6 Erweiterung

Um die Spielerfahrung immersiver zu gestalten, wird die Applikation, die im Rahmen der Bachelorarbeit entwickelt wird, erweitert. Ziel ist es, dass die Applikation in Kombination mit dem Google Cardboard genutzt werden kann. Das Google Cardboard ist eine Virtual Reality Brille aus Pappe. Als Bildschirm wird ein Smartphone genutzt, welches am Cardboard befestigt wird. Dazu musste der GameRenderer zu einem Stereo Renderer erweitert werden, sowie das Layout sämtlicher Klassen an das Google Cardboard angepasst werden.

6.1 Stereo Renderer

Der Stereo Renderer ist eine Erweiterung des Rajawali-Framework Renderers. Dabei ist die GameRenderer-Klasse nun eine Erweiterung des Stereo Renderers, statt wie bisher eine Erweiterung des Rajawali-Framework Renderers. Im Stereo Renderer werden zunächst zwei Kameras initialisiert. Eine für das linke Auge und eine für das rechte Auge. Die Werte für die vordere und hintere Clippingebene, sowie die Werte für das Sichtfeld werden aus der Kamera, die im Spielrenderer angelegt wird, übernommen. Um das Bild separat für das linke und rechte Auge zu rendern, muss der Pupillenabstand beachtet werden. Dazu werden die Kameras um ein bestimmtes Offset entlang der x-Achse verschoben. Die linke Kamera wird um die Hälfte des Offsets nach links verschoben und die rechte Kamera wird um die Hälfte des Offsets nach rechts verschoben. Zusätzlich werden zwei Texturen genutzt, in welche die gesamte Szene, zum einen aus der Sicht der Kamera für das linke Auge und zum anderen aus der Sicht der Kamera für das rechte Auge, hineingerendert wird. Die Breite der Texturen wird auf die Hälfte der Breite des Viewports gesetzt und die Höhe wird auf die Höhe des Viewports gesetzt. Dies geschieht, da der Bildschirm nun in zwei Hälften geteilt wird. Zudem wird auch die Projection-Matrix der Kameras mit der halbierten Breite des Viewports neu definiert. Desweiteren werden zwei screenfilling Quads initialisiert, in denen die Texturen angezeigt werden. Die Breite der Quads wird halbiert, sodass jedes Quad eine Bildschirmhälfte abdeckt. Das Quad für das linke Auge wird in der linken Bildschirmhälfte positioniert und das Quad für das rechte Auge wird in der rechten Bildschirmhälfte positioniert. Nun wird in jedem Renderschritt zunächst die Kameraorientierung aktualisiert und eine Referenz auf die aktuelle Szene gespeichert. Dann wird die referenzierte Szene aus der Sicht der linken Kamera in die Textur für das linke Auge gerendert. Danach wird die aktive Kamera gewechselt und aus der Sicht der rechten Kamera in die Textur für das rechte Auge gerendert. Zuletzt wird noch einmal ein Renderaufruf durchgeführt ohne in eine Textur zu rendern, um eine Referenz auf die aktualisierte Szene zu erhalten. Der Renderloop startet dann von neuem.

6.2 Layout

Im weiteren Verlauf musste das Graphical User Interface bzw. das Head-up-Display des Spiels und das Layout sämtlicher Klassen für die Nutzung des Google Cardboard angepasst werden. Dabei werden neue Layouts definiert, ohne die alten zu überschreiben, um dem Benutzer zu ermöglichen, zwischen stereoskopischem Rendering und normalem Rendering zu wählen. Alle Elemente des Head-up-Displays werden dupliziert und jeweils an die beiden Bildschirmhälften angepasst. Der Knopf zum Abfeuern der Projektile wird entfernt. Stattdessen werden die Projektile nun durch eine Betätigung des Magnetschalters, welcher sich an der linken Seite des Google Cardboards befindet, abgefeuert. Um herauszufinden wann der Magnetschalter betätigt wird, stellt Google eine Activity-Klasse zur Verfügung, die eine Erweiterung der Android eigenen Activity-Klasse ist. In dieser Klasse existiert eine Methode die immer aufgerufen wird, falls der Magnetschalter betätigt wird. Daher ist es nötig alle Activity-Klassen der Applikation von der von Google zur Verfügung gestellten Activity-Klasse erben zu lassen, statt wie bisher von der Android eigenen Activity-Klasse. Damit kann nun jederzeit ermittelt werden, wann der Magnetschalter betätigt wird, da die dazugehörige Methode aufgerufen wird. Bei jedem Aufruf dieser Methode wird nun die Funktion zum Abfeuern der Projektile aufgerufen.

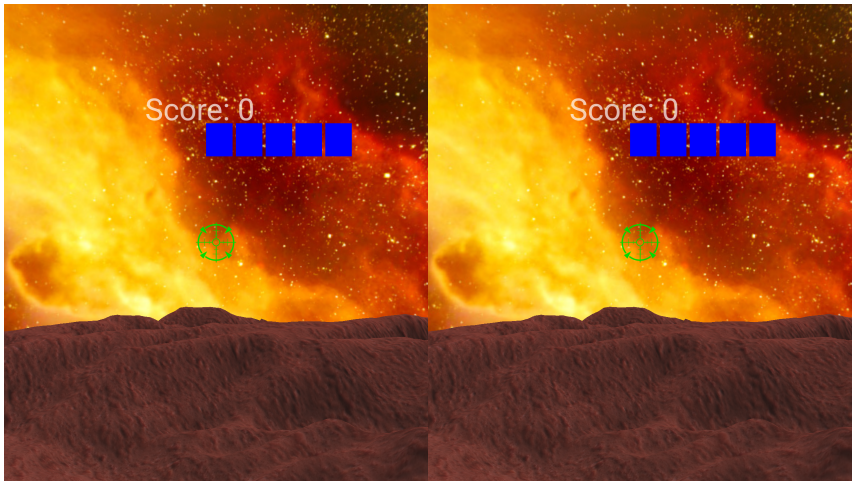

Abbildung 18: Nutzung des Stereo Renderers

Alle Elemente der Game-Over-Klasse und der Highscore-Klasse werden ebenfalls dupliziert und jeweils an die einzelnen Bildschirmhälften angepasst. Die Knöpfe die in beiden Klassen enthalten sind, werden entfernt. Stattdessen werden nun die Aktionen, die durch die Knöpfe ausgelöst wurden, durch eine Betätigung des Magnetschalters ausgelöst.

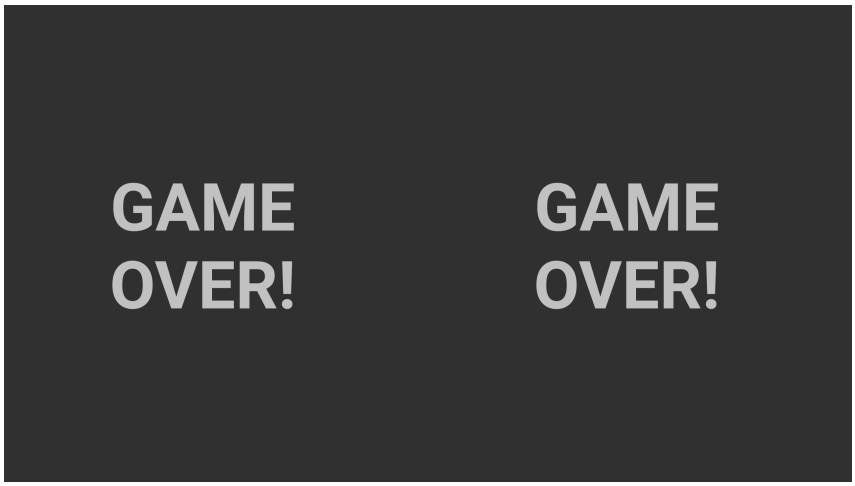

Abbildung 19: Angepasste Game-Over Anzeige

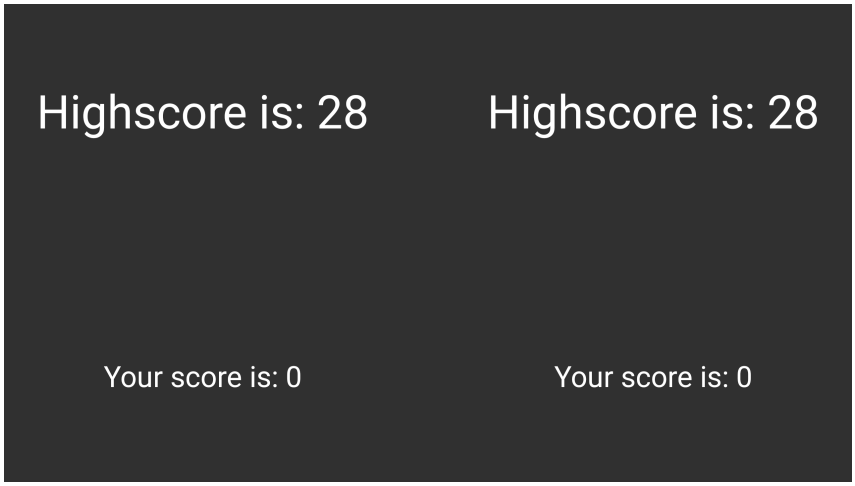

Abbildung 20: Angepasste Bestenliste

Das Layout des Hauptmenüs bleibt weitestgehend unverändert. Lediglich ein Knopf zum Starten im Google Cardboard Modus wird hinzugefügt. Daher hat der Benutzer nun die Wahl ob er die Applikation mit dem Google Cardboard oder ohne nutzen möchte. Je nach Wahl werden entweder die an das Google Cardboard angepassten Layouts und der Stereo Renderer oder die nicht angepassten Layouts und der normale Renderer geladen.

Abbildung 21: Angepasstes Hauptmenü

7 Analyse

Um herauszufinden, wie Nutzer das Produkt bewerten, welches im Rahmen der Bachelorarbeit entstanden ist, wird ein Nutzertest durchgeführt. Der Aufbau und die Ergebnisse des Nutzertests werden in diesem Kapitel behandelt.

7.1 Aufbau der Analyse

Der Nutzertest umfasst drei Seiten. Auf diesen sollen die Probanden die Applikation bewerten, nachdem sie eine Spielrunde absolviert haben. Dazu werden hauptsächlich Bewertungsskalen eingesetzt. Die Skalen reichen von eins bis sechs, wobei eins ein negatives Bewertungsurteil und sechs ein positives Bewertungsurteil ist. Dabei wird die Applikation in Verbindung mit dem Google Cardboard genutzt. Zuerst müssen die Probanden ihre bisherigen Erfahrungen im Bereich Virtual Reality angeben. Weiterhin sollen alle Aspekte des Google Cardboard und das Virtual Reality Erlebnis bewertet werden. Zusätzlich müssen die Probanden das Spiel selbst beurteilen. Insgesamt müssen 17 Fragen beantwortet werden. Die Probanden haben am Ende des Nutzertests die Möglichkeit Anmerkungen im Bezug zur Applikation zu machen.

7.2 Ergebnisse

Der Nutzertest wurde mit insgesamt zehn Probanden durchgeführt. Nach der Auswertung des Nutzertests ergaben sich folgende Ergebnisse:

Die Mehrheit der Probanden hat bereits Erfahrungen im Bereich Virtual Reality gesammelt. Auf einer Skala von eins bis sechs ergibt sich ein durchschnittlicher Erfahrungswert der Probanden von 3,3.

Abbildung 22: Nutzertestauswertung zu Frage 1

Acht von zehn Probanden haben ihre Erfahrungen mit der Oculus Rift gesammelt. Nur wenige haben bereits mit anderen Virtual Reality Geräten Kontakt gehabt.

Abbildung 23: Nutzertestauswertung zu Frage 2

Die meisten der Probanden fanden die Spielerfahrung sehr immersiv. Auf einer Skala von eins bis sechs liegt der Mittelwert bei 4,5.

Abbildung 24: Nutzertestauswertung zu Frage 3

Viele der Tester haben, bei der Nutzung des Google Cardboard, das Sichtfeld als zu klein empfunden. Sieben von zehn Personen hätten lieber ein größeres Sichtfeld gehabt.

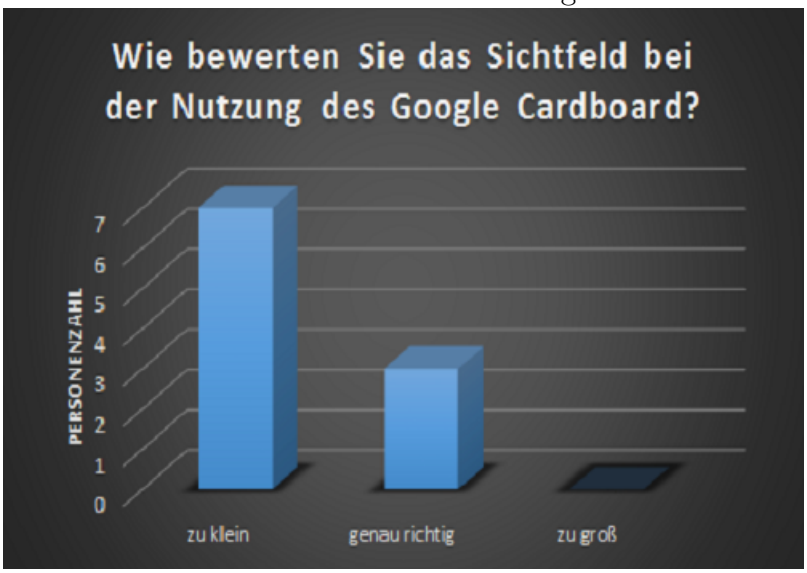

Abbildung 25: Nutzertestauswertung zu Frage 4

Die Meinungen über den Tragekomfort des Google Cardboard waren geteilt. Auf einer Skala von eins bis sechs beträgt der Mittelwert 3,5.

Abbildung 26: Nutzertestauswertung zu Frage 5

Die Sensorgenauigkeit der Applikation wurde von allen Probanden als sehr gut empfunden. Der Mittelwert beträgt hier 5,2.

Abbildung 27: Nutzertestauswertung zu Frage 6

Die Mehrheit der Probanden war mit dem Bewegungsanteil zufrieden, welcher zum Spielen des Spiels nötig ist. Acht von zehn Personen fanden den Bewegungsanteil genau richtig.

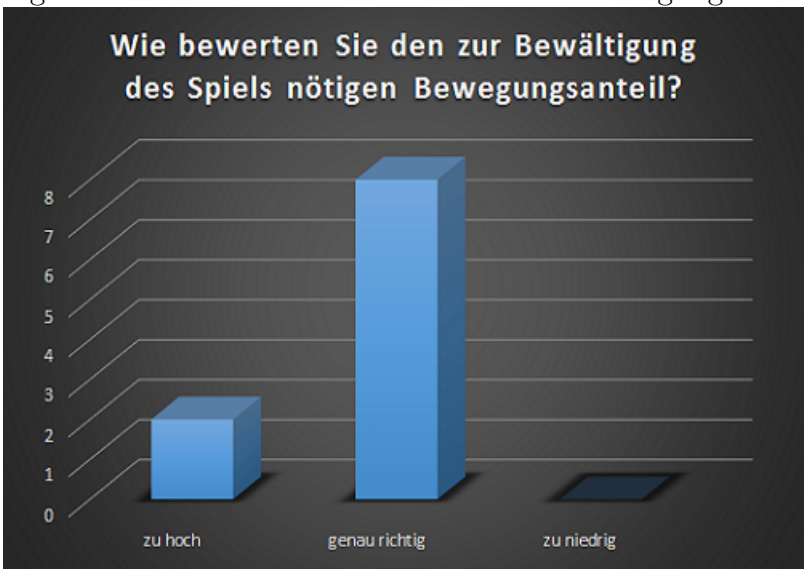

Abbildung 28: Nutzertestauswertung zu Frage 7

Das Layout des Menüs wurde von den Testern eher positiv bewertet. Der Mittelwert beträgt hier 4,2.

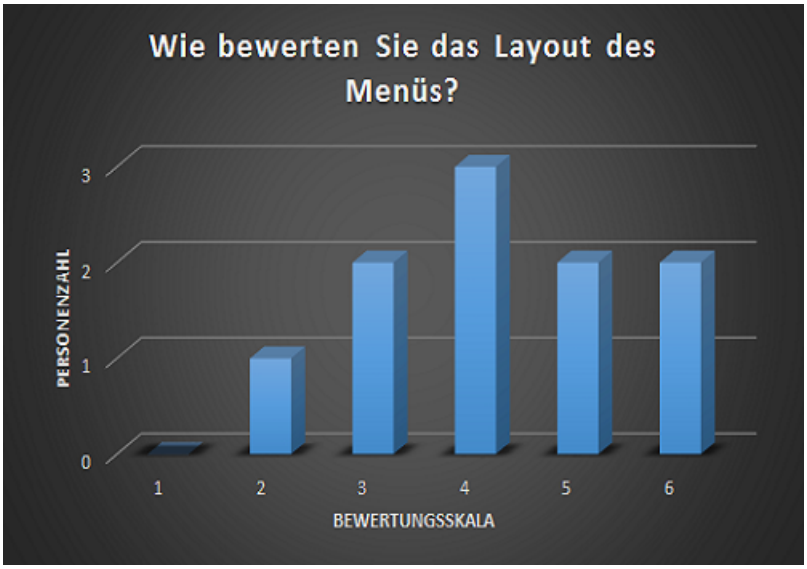

Abbildung 29: Nutzertestauswertung zu Frage 8

Die Meinungen der Probanden über das Head-up-Display waren neutral. Bei der Auswertung des Nutzertests ergab sich hier ein Mittelwert von 3,8.

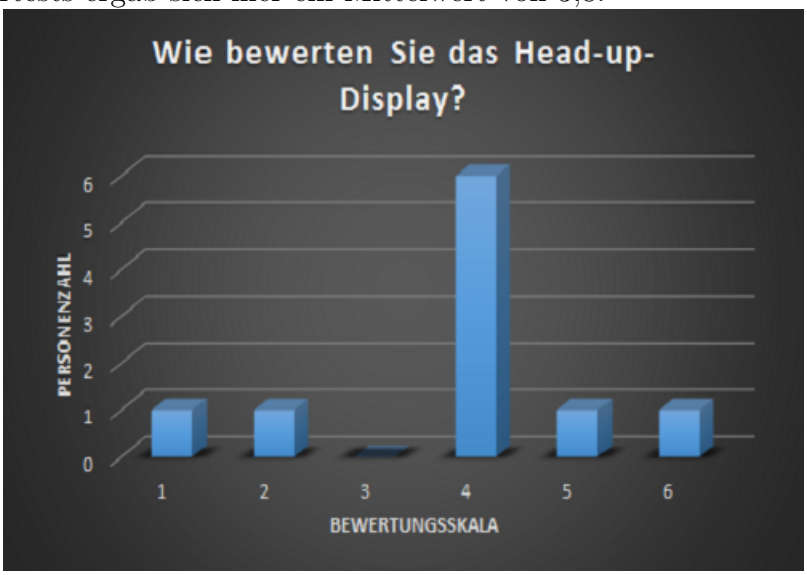

Abbildung 30: Nutzertestauswertung zu Frage 9

Der Schwierigkeitsgrad des Spiels wurde von den Testern mehrheitlich positiv bewertet. Sechs von zehn Testern fanden den Schwierigkeitsgrad genau richtig.

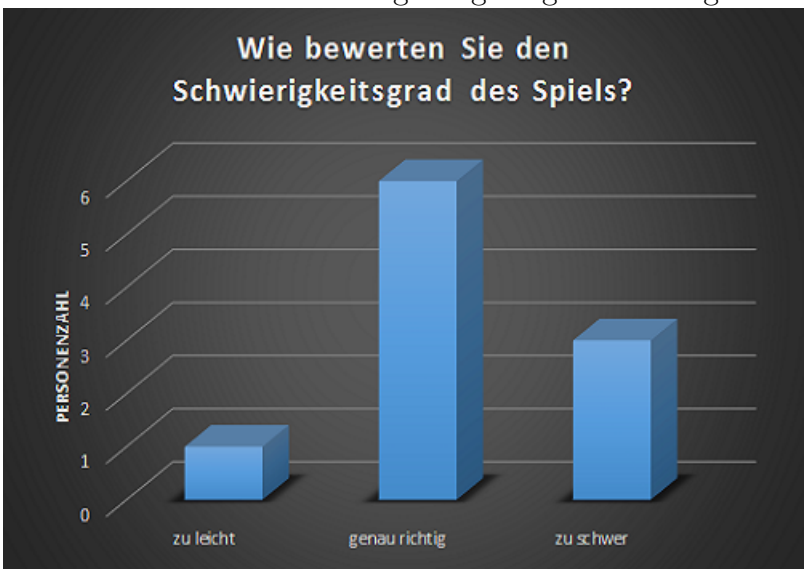

Abbildung 31: Nutzertestauswertung zu Frage 10

Die Bewertung der Grafik des Spiels fiel auch positiv aus. Der Mittelwert beträgt hier exakt 4.

Abbildung 32: Nutzertestauswertung zu Frage 11

Das Weltraumszenario wurde von den Probanden als sehr gut empfunden. Hier beträgt der Mittelwert 4,3.

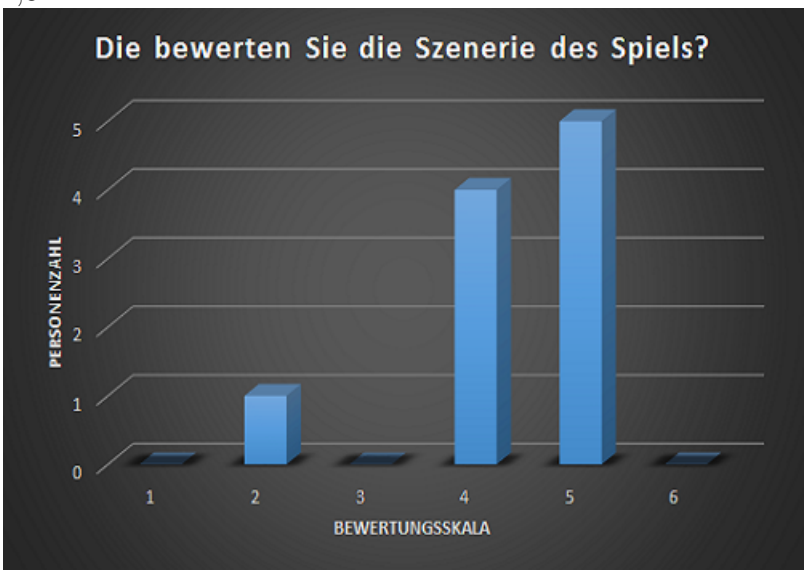

Abbildung 33: Nutzertestauswertung zu Frage 12

Die Animationen des Spiels sind mehrheitlich positiv bewertet worden. Die Auswertung ergab einen Mittelwert von 4,1.

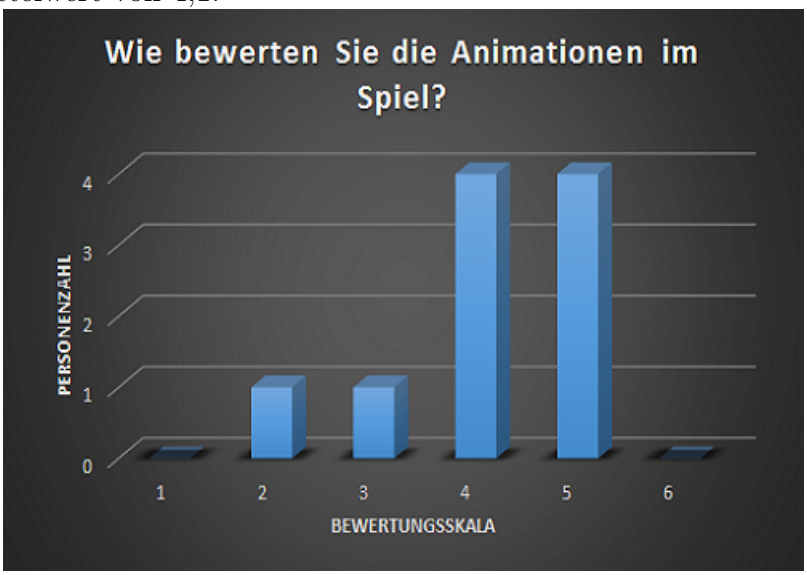

Abbildung 34: Nutzertestauswertung zu Frage 13

Die Tester haben die Bewegungsgeschwindigkeit der Raumschiffe größtenteils positiv empfunden. Acht von zehn der Tester bewerteten die Bewegungsgeschwindigkeit als genau richtig.

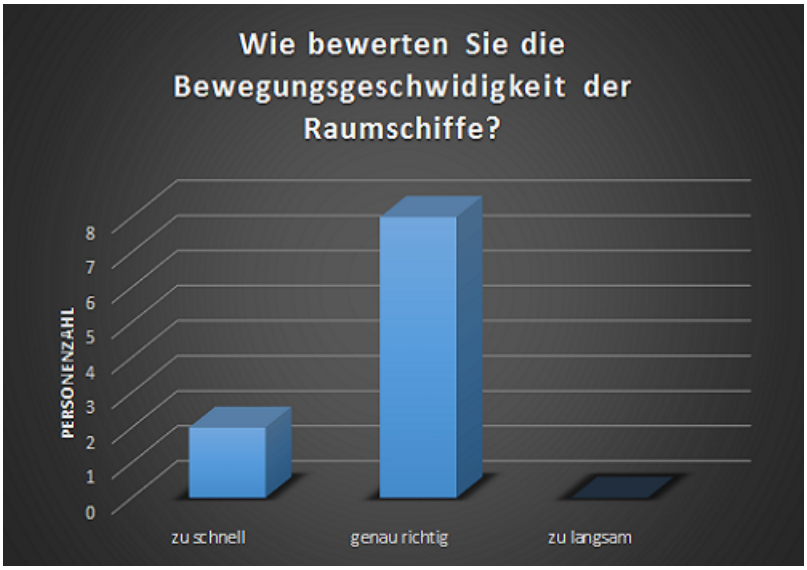

Abbildung 35: Nutzertestauswertung zu Frage 14

Bei der Bewertung der Bewegungsgeschwindigkeit der Projektile waren die Meinungen der Probanden jedoch geteilt. Die Hälfte der Probanden fand die Geschwindigkeit genau richtig und die andere Hälfte gab an, das ihnen die Geschwindigkeit zu langsam ist.

Abbildung 36: Nutzertestauswertung zu Frage 15

Die Umsetzung der Bestenliste wurde eher neutral bewertet. Der Mittelwert liegt hier bei 3,6.

Abbildung 37: Nutzertestauswertung zu Frage 16

Das Spielprinzip hingegen wurde sehr positiv bewertet. Der errechnete Mittelwert beträgt in diesem Fall 4,7.

Abbildung 38: Nutzertestauswertung zu Frage 17

7.3 Diskussion

Insgesamt sind die erzielten Ergebnisse sehr zufriedenstellend. Dass die Tester die Immersion während ihrer Spielerfahrung als sehr gut empfunden haben, fällt positiv auf. Es gelingt also mit einer sehr kostengünstigen Virtual Reality Lösung ein immersives Spielerlebnis zu kreieren. Leider fällt bei dem Google Cardboard Licht ein, was die Immersion ein wenig stört. Durch den niedrigen Preis des Google Cardboard, sind die darin befindlichen Linsen minderwertig. Dies führt dazu, dass der Bildschirm des Smartphones nicht vollständig gesehen werden kann. Das erklärt auch die schlechten Bewertungen des Sichtfeldes. Der Nutzertest ergab ebenfalls, dass der Tragekomfort des Google Cardboard sehr niedrig ist, da das Cardboard aus Karton mit scharfen Kanten besteht. Die Meinungen über die Genauigkeit der Sensoren waren sehr gut. Der von Google bereitgestellte Headtracker ist also eine sehr gute Lösung, um die Steuerung im Virtual Reality Bereich zu realisieren. Die Bewertungen der einzelnen Aspekte des Spiels waren mehrheitlich positiv. Durch eine Beschleunigung der Bewegungsgeschwindigkeit der Projektile kann die Spielerfahrung weiter verbessert werden. Die Umsetzung der Bestenliste müsste geändert werden. Eine Highscore-Tabelle mit der Möglichkeit seinen Namen und seine erreichten Punkte einzutragen, wäre die bessere Option.

8 Schlussbetrachtung

Im folgenden Kapitel werden die Arbeiten und die daraus erworbenen Erkenntnisse abschließend zusammengefasst. Es werden Erfolge und Misserfolge der Arbeit aufgezeigt und Verbesserungsmöglichkeiten vorgeschlagen. Anschließend werden Möglichkeiten zur zukünftigen Entwicklung und Weiterführung des erstellten Prototyps dargelegt.

8.1 Ausgangslage

Ausgangslage war die Aufgabe eine interaktive Applikation unter Android zu konzipieren und umzusetzen. Dabei bestand die Herausforderung darin, sich in die Softwareentwicklung unter dem Android Betriebssystem einzuarbeiten und die Möglichkeiten und Grenzen kennen zu lernen und zu analysieren.

8.2 Fazit

Lösungsansatz war die Nutzung eines OpenGL ES Framworks. Um möglichst viel über die Softwareentwicklung unter Android zu lernen, wurde das Open Source Community Rajawali-Framework gewählt, da hier eine Arbeit nahe am Code möglich war und dadurch ein größerer Lernerfolg eintreten konnte. Das Framework wurde als Grundbaustein genutzt und darauf aufbauend wurde das Spiel Space Wars entwickelt. Es baut auf dem ursprünglichen Spielkonzept (Kapitel 4.1) auf und alle Ansätze konnten daraus umgesetzt werden. Das Spiel hat einen Casual Stil und ist deshalb von der Menüführung intuitiv und bei den meisten Spielern bekannt. Das Grafik-Konzept lag im Rahmen dieser Bachelorarbeit nicht im Fokus und daher wurden keine zusätzlichen Shader zur Aufbesserung der Grafik entwickelt, wodurch die Grafik sehr simpel ist. Das Punktesystem soll den Ehrgeiz der Spieler anregen und dadurch das Spiel etwas fesselnder gestalten. Dank des Google Cardboard Software Development Kits gestaltete sich die Einbindung der Smartphone Sensoren als unkompliziert. Insgesamt wurden alle im Spielkonzept gesetzten Ziele erreicht und ein gelungenes Spiel entwickelt. Durch die zusätzlich eingebrachte Kompatibilität mit dem Google Cardboard, wurde die anfangs gestellte Aufgabenstellung sogar übertroffen. Durch die Auseinandersetzung mit dem Source Code des Rajawali-Frameworks und durch die Realisierung des Projekts im Rahmen dieser Bachelorarbeit war es möglich, eine umfassende Einsicht in die Softwareentwicklung unter Android zu erhalten, wodurch das angestrebte Lernziel erfüllt wurde. Aufgrund der umfangreichen Auseinandersetzung mit dem Rajawali-Framework, sind viele Fehler im Source Code entdeckt worden, die der Entwicklung durch die Community geschuldet sind, da das Framework ein Open Source Community Projekt ist. Daher sind Fehler im Source Code unvermeidlich. So ist beispielsweise keine vollständige Threadsicherheit innerhalb des Frameworks gewährleistet,

wodurch einzelne Komponenten von verschiedenen Programmbereichen gleichzeitig genutzt werden und es somit zu Berechnungsfehlern kommt. Daher ist der entwickelte Prototyp des Spiels nicht marktreif. Bei der zukünftigen Entwicklung von Spielen für Android wird daher auf die Nutzung eines Open Source Frameworks verzichtet und stattdessen ein kommerzielles Framework genutzt. Die Entwicklung eines eigenen Frameworks wäre auch denkbar. Jedoch war die Nutzung des Rajawali-Frameworks lohnenswert, um eine umfassende Einsicht in die Softwareentwicklung unter Android zu erhalten und somit das gewünschte Lernziel im Rahmen dieser Bachelorarbeit zu erreichen.

8.3 Ausblick

Wie bereits erwähnt, kommt es durch die fehlende Threadsicherheit innerhalb des Rajawali-Frameworks zu Berechnungsfehlern. Durch die Behebung dieses Problems, würde das entwickelte Spiel marktreif werden. Zusätzlich wäre die Einbindung von Shader-Effekten zur Verbesserung der Grafik ein interessanter Aspekt, da die Kapazitäten aktueller Smartphones durch den Prototypen bisher nicht ausgereizt werden. Durch den Nutzertest wurden ebenfalls einige Anmerkungen zur Applikation gesammelt. Die Umsetzung der Anmerkungen würde das Spiel weiter optimieren und interessanter gestalten. So blenden beispielsweise die Schüsse beim Abfeuern. Eine Veränderung der Startposition der Projektile würde dies verhindern. Darüber hinaus kann die Spannung des Spiels erhöht werden, indem verschiedene Gegner-Raumschiffe mit verschiedenen Eigenschaften implementiert werden. Auch das Einführen von Endgegnern in das Spiel wäre eine Option. Eine Positionsanzeige der Gegner bei einem Treffer durch einen Pfeil oder auch eine Minimap wäre eine sinnvolle Erweiterung des Spiels. Zurzeit wird ein festes Offset von 0,6 für den Pupillenabstand bei dem stereoskopischen Rendering verwendet. Die Möglichkeit das Offset durch den Spieler selbst anpassen zu lassen wäre praktisch.

9 Literatur

[1] Dörner, R. and Broll, W. and Grimm, P. and Jung, B., *Virtual und Augmented Reality (VR / AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*, Springer Berlin Heidelberg, eXamen.press, 9783642289033, 2014.

[2] Gargenta, M., *Learning Android*, O'Reilly Media, 9781449307240, 2011.

[3] Mednieks, Z. and Dornin, L. and Meike, G.B. and Nakamura, M., *Android-Programmierung*, O'Reilly, 9783955611415, 2013.

[4] Burton, M. and Franken, G., *Android Application Development For Dummies*, Wiley, –For dummies, 9781118417454, 2012.

[5] James, D., *Android Game Programming For Dummies*, Wiley, –For dummies, 9781118235997, 2012.

[6] Simon, J., *Head First Android Development*, O'Reilly Media, Incorporated, Head First Series, 9781449393304, 2012.

[7] Annuzzi, J. and Darcey, L. and Conder, S., *Introduction to Android Application Development: Android Essentials*, Pearson Education, Developer's Library, 9780133477337, 2013.

[8] Dennis Ippel, Rajawali,
`http://www.rozengain.com/blog/category/rajawali/`

[9] Emil Persson, Humus,
`http://www.humus.name/index.php?page=Textures`

[10] Space Telescope Science Institute (STScI), Hubble,
`http://hubblesite.org/newscenter/archive/releases/2006/01/`

[11] Wikimedia Commons, Wiki,
`http://commons.wikimedia.org/wiki/File:Orion,_battle_spaceship.jpg`

Anhang

Nutzertest

Kreuzen Sie für jedes nachfolgend genannte Element die Zahl rechts an, die Ihrer Meinung nach am besten dessen Qualität zum Ausdruck bringt. Wählen Sie die Zahl entsprechend der angegebenen Skala aus

1. **Wie viel Erfahrung haben Sie bereits mit Virtual Reality?**
   Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
keine	◯	◯	◯	◯	◯	◯	sehr viel

2. **Mit welchen Geräten haben Sie bereits Erfahrungen im Bereich Virtual Reality gesammelt?**
   Wählen Sie alle zutreffenden Antworten aus.

   - ☐ Keine Erfahrung
   - ☐ Google Cardboard
   - ☐ Samsung Gear VR
   - ☐ Durovis Dive
   - ☐ Carl Zeiss VR
   - ☐ Oculus Rift
   - ☐ Sonstiges: .....

3. **Wie immersiv war Ihr Spielerlebnis?**
   Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
gar nicht immersiv	◯	◯	◯	◯	◯	◯	sehr immersiv

4. **Wie bewerten Sie das Sichtfeld bei der Nutzung des Google Cardboard?**
   Markieren Sie nur ein Oval.

   - ◯ zu klein
   - ◯ genau richtig
   - ◯ zu groß

5. **Wie bewerten Sie den Komfort des Google Cardboard?**
   Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr unbequem	◯	◯	◯	◯	◯	◯	sehr bequem

6. **Wie bewerten Sie die Sensorgenauigkeit des Spiels?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

7. **Wie bewerten Sie den zur Bewältigung des Spiels nötigen Bewegungsanteil?**
Markieren Sie nur ein Oval.

- ◯ zu hoch
- ◯ genau richtig
- ◯ zu niedrig

8. **Wie bewerten Sie das Layout des Menüs?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

9. **Wie bewerten Sie das Head-up-Display?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr unübersichtlich	◯	◯	◯	◯	◯	◯	sehr übersichtlich

10. **Wie bewerten Sie den Schwierigkeitsgrad des Spiels?**
Markieren Sie nur ein Oval.

- ◯ zu leicht
- ◯ genau richtig
- ◯ zu schwer

11. **Wie bewerten Sie die Grafik des Spiels?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

12. **Wie bewerten Sie die Szenerie des Spiels?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

13. **Wie bewerten Sie die Animationen im Spiel?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

14. **Wie bewerten Sie die Bewegungsgeschwindigkeit der Raumschiffe?**
Markieren Sie nur ein Oval.

- ◯ zu schnell
- ◯ genau richtig
- ◯ zu langsam

15. **Wie bewerten Sie die Bewegungsgeschwindigkeit der Projektile?**
Markieren Sie nur ein Oval.

- ◯ zu schnell
- ◯ genau richtig
- ◯ zu langsam

16. **Wie bewerten Sie die Umsetzung der Bestenliste?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr schlecht	◯	◯	◯	◯	◯	◯	sehr gut

17. **Wie bewerten Sie das Spielprinzip?**
Markieren Sie nur ein Oval.

	1	2	3	4	5	6	
sehr langweilig	◯	◯	◯	◯	◯	◯	sehr spannend

18. **Haben Sie noch weitere Anmerkungen?**

.....

.....

.....

.....

.....