UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Entwicklung eines Jump'n Run-Spiels mit Sensorensteuerung

Bachelorarbeit

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

vorgelegt von

Thomas Vorberger

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

Zweitgutachter: Anna Katharina Hebborn, M.Sc.
(Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Juli 2014

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.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

Ort, Datum Unterschrift

Zusammenfassung

In dieser Bachelorarbeit wird die Frage behandelt, ob ein Jump'n Run-Spiel mit Sensorensteuerung für Android-Geräte sinnvoll ist. Hierzu wurde ein Spiel entwickelt, das in unterschiedlichen Level einmal mit und ohne Sensoren steuerbar ist. In einer zweiten Version wird das Spiel komplett anhand von Sensoren gesteuert, damit man später die Steuerungen vergleichen kann. Es wird erklärt, wie das Spiel geplant, entworfen und untersucht wurde. Zudem wird geprüft, ob es schon Spiele mit Sensorensteuerung generell gibt. Die Engine, mit der das Spiel entwickelt wurde, wird ebenfalls vorgestellt. Abschließend erfolgt die Auswertung eines dafür ausgearbeiteten Nutzertests über die Spieltauglichkeit des Spiels hinsichtlich der Steuerung.

Abstract

In this bachelor thesis, the question of whether or not a jump'n run game with sensor control for android devices is useful, is handled. To this end, a game was developed, which is once controlled with and without sensors at different levels. In a second version, the game is completely controlled by means of sensors, so that the controls can later be compared. It is explained how the game was planned, designed and investigated. In addition, it is checked whether games with sensor control already exist. The engine, which was used to developed the game, is also introduced. Finally, the evaluation is carried out for an elaborated user test on the playability of the game in terms of control.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Ein Smartphone ist für viele Leute heutzutage nicht mehr wegzudenken. Es wird immer mehr in den Alltag integriert. Ob nun zum Surfen, Chatten oder auch nur als Zeitmesser, ein Blick auf das Gerät lohnt sich immer mehr. Im Schnitt schaut ein Smartphone User 80 Mal am Tag auf sein Handy, so überrascht es nicht, dass es auch zum Zeitvertreib genutzt wird, wie zum Beispiel zum Spielen [nss]. Und das wurde auch erkannt: 92 Prozent der Menschen in Deutschland, die gelegentlich Computerspiele spielen, greifen immer häufiger zu ihrem Smartphone, so dass es zu einem konkurrenzfähigem Gaming Device herangewachsen ist. [WWWb]. Gründe hierfür sind unter anderem Gelegenheitsspiele, sogenannte *Casual Games*. Sie zeichnen sich durch leichtes Erlernen des Spiels aus und bieten schnelle Erfolgserlebnisse sowie kurzweilige Unterhaltung [cg]. Auch die Anzahl der Smartphones steigt weiterhin an. Im Jahr 2017 sollen 80 Prozent der Mobiltelefone in Europa Smartphones sein [msh]. Sich mit der Entwicklung von Mobile Games zu beschäftigen macht also durchaus Sinn und wird zunehmend profitabler.

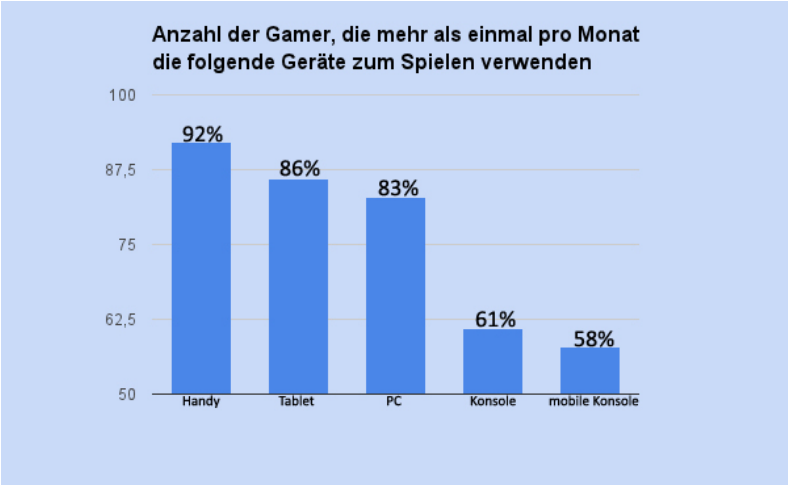

Abbildung 1: Anzahl der Gamer, die mehr als einmal pro Monat die folgenden Geräte zum Spielen verwenden [mz cg]

1.2 Aufgabenstellung

Die Aufgabe dieser Ausarbeitung ist die Untersuchung und Entwicklung eines Jump'n Run Spiels auf einem Android-Gerät unter Verwendung von dessen Sensoren. Die bei dem Smartphone mitgelieferten Sensoren (hier nur die Berücksichtigung des Beschleunigungssensors) sollen bei der Steuerung der Spielfigur eine wesentliche Rolle spielen. Neben der eher üblichen Touch-Steuerung soll nun untersucht werden, inwieweit eine Steuerung durch Lageveränderung des Smartphones bei Jump'n Run Spielen sinnvoll ist.

Zudem wird eine zweite Variante in die App integriert, die permanent die Steuerung mittels Sensoren verwendet.

1.3 Aufbau der Arbeit

Um ein Spiel mit Sensorensteuerung zu entwickeln muss sich zuallererst mit den Sensoren eines Smartphones auseinandergesetzt werden.

In Kapitel zwei werden diese vorgestellt, erläutert und in Beispielen beschrieben, für welchen Zweck sie verwendet werden. Das nächste Kapitel befasst sich mit der Untersuchung wie Jump'n Run Spiele auf Android-Geräten bisher gesteuert werden. Es wird ebenso untersucht, ob es schon Spiele mit Sensorensteuerung außerhalb des Jump'n Run-Genres gibt. Abschließend werden Jump'n Run Spiele vorgestellt, die sich ähnlich wie das zu entwickelnde Spiel in der Steuerung verhalten. Kapitel vier stellt das Konzept für das Spiel vor. Es wird erklärt, mit welchen Ideen, mit welcher Story und mit welchen Vorstellungen an das Spiel herangegangen wird und wie diese technisch umgesetzt werden. Hierzu werden das Betriebssystem Android und die für die Entwicklung des Spiels relevante AndEngine vorgestellt. Kapitel fünf beschreibt die Programmierung des Spiels in einzelnen Schritten und wie die Steuerung der Spielfigur mit und ohne Sensorensteuerung umgesetzt wurde. In Kapitel sechs wird anschließend anhand eines Nutzertest untersucht, welche Steuerungsmöglichkeit im Spiel mehr Anklang gefunden hat, um Schlüsse daraus zu ziehen, ob das Spiel mit Sensorensteuerung bei Jump'n Run Spielen sinnvoll ist. Abschließend werden die Ergebnisse vorgestellt und eine Wertung präsentiert.

2 Grundlagen

2.1 Smartphonesensoren

Smartphones sind heutzutage mit vielen Sensoren ausgestattet. Neben Kamera und Mikrofon besitzen sie einige Messgeräte, die physikalische oder chemische Eigenschaften in analoge elektrische Signale umwandeln [sen].

Nachfolgend werden die gängigsten Sensoren in einem Smartphone aufgelistet und erläutert.

Beschleunigungssensor. Dieser wird eingesetzt, um die Bewegungen des Smartphones zu erfassen, beispielsweise zur Umschaltung des Displays zwischen Hoch- und Querformat [bms]. Die Kräfteverschiebung beim Kippen wird an drei Achsen gemessen, der x-, y- und z-Achse des Smartphones (siehe Abbildung 2). Das Koordinatensystem bildet ein Rechtssystem, wobei die x-Achse die Ausrichtung nach links/rechts, die y-Achse nach oben/unten und die z-Achse, die zu dem Betrachter des Displays zeigt, die Ausrichtung nach

hinten und vorne angeben. Die Bewegung wird anhand eines Siliziumstabes an jeder Achse gemessen. Ändert dieser seine Position, zum Beispiel durch Kippen des Smartphones, so verändert sich auch die elektrische Kapazität innerhalb des Sensors, womit man Aussagen über die Höhe der Beschleunigung entlang einer Achse machen kann.

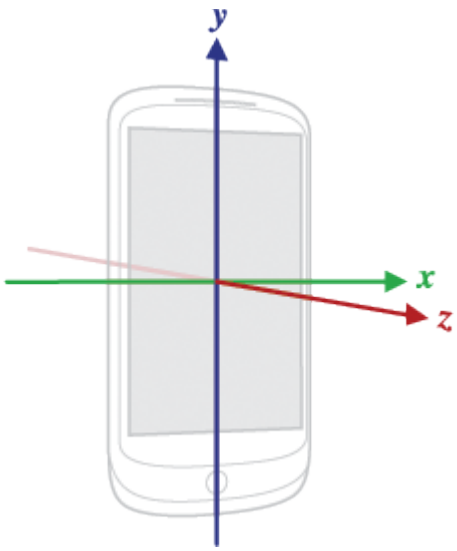

Abbildung 2: Koordinatensystem eines Smartphones (Quelle: http://developer.android.com/reference/android/hardware/SensorEvent.html)

Helligkeitssensor. Der Sensor misst das Licht in der Umgebung, so dass die Helligkeit des Displays entsprechend angepasst werden kann. Neben der Intensität des Lichtes misst er zudem auch die Farbtemperatur. Das Display kann somit kontrastreichere und farbsättigendere Bilder anzeigen [zss14].

Gyroskop. Es bestimmt im Gegensatz zum Beschleunigungssensor die Lage des Smartphones über zusätzliche Schwingungsbewegungen aufgrund der Corioliskraft auf einem sich drehenden Kreisel. Dieser Kreisel ist heute so klein, dass er auf einen winzigen Chip passt [pwc] [comSD].

Global Positioning System (GPS). Eine im Smartphone eingebaute Antenne empfängt Signale von mehreren Satelliten, um die genaue Position des Smartphones zu bestimmen. Das GPS wird als Sensor mit aufgelistet, da es der Hardware mitteilt, in welcher Umgebung sich eine Person gerade befindet [zss14].

Magnetometer. Der Chip misst jeweils entlang der x-, y-, und z-Achse die Stärke und Richtung des Erdmagnetfeldes mittels des Hall-Effekts. In diesem Hall-Sensor wird eine Platte von Strom durchflossen, wobei ein senkrecht zur Flussrichtung verlaufendes Magnetfeld die Ausgangsspannung des Sensors verändert. So kann es Angaben über Richtung und Flussdichte des Magnetfeldes geben. Anwendungsgebiete findet der Sensor im Kompass oder zur Erkennung von Stromleitungen in Wänden.

Näherungssensor. Mittels Infrarotstrahlen kann dieser Sensor erkennen, ob das Smartphone gerade ans Ohr gehalten wird. Ist dies der Fall, werden der Bildschirm und die Berührungssteuerung abgeschaltet, um ungewollte Funktionen während des Telefonierens zu unterbinden. Zudem erkennt der Infrarotstrahl Wischgesten der Hände.

Touchscreen. Auch das Display selbst ist ein Sensor, der jede Bewegung und indirekte Berührungen registriert. Durch das Berühren des Bildschirms verändert sich das Signal der elektrischen Leitungen unter dem Glas, womit die Position und Dauer der Berührung berechnet werden kann [zss14-2].

Barometer. Dieser misst den Luftdruck in der Umgebung und bestimmt für das Assisted Global Positioning System (A-GPS) den ungefähren Höhenwert über dem Meeresspiegel. Außerdem kann dieser zusammen mit dem Beschleunigungssensors als Schrittzähler eingesetzt werden [nw Gs4].

Temperatursensor. Er dient zur Messung der Luftfeuchtigkeit und der Umgebungstemperatur. Zudem kann auch die Temperatur der Hardware selbst gemessen werden, um zum Beispiel Angaben über die CPU-Temperatur zu geben [enSen].

Für das zu entwickelnde Jump'n Run Spiel wird der Beschleunigungssensor zur Steuerung der Spielfigur eingesetzt, da er exakt das Kippen des Smartphones erkennt und sich somit die Position der Spielfigur verändern lässt.

3 Untersuchung Jump'n Run und deren Steuerung

In dieser Arbeit soll ein Jump'n Run Spiel mit Sensorensteuerung entwickelt werden. Dazu muss zunächst definiert werden, was genau aus einem Spiel ein Jump'n Run Spiel macht und welche Steuerungsmöglichkeiten es bisher gibt, damit sie über Sensoren erweitert werden können. Zusätzlich soll untersucht werden, ob es bereits ähnliche Ansätze gibt.

3.1 Jump 'n Run Spiele im Allgemeinen

Jump'n Run Spiele charakterisieren sich hauptsächlich über eine Spielfigur, die laufend und springend durch einen oder mehrere Level geführt werden. Darüber hinaus müssen Hindernisse überwunden werden, Punkte oder Gegenstände eingesammelt und Gegnern ausgewichen oder gar vernichtet werden, wobei es hierfür mehrere Abwandlungen oder Kombinationen gibt. So kann ein Jump'n Run Spiel ein Geschicklichkeitsspiel oder ein Kampfspiel sein, bei dem der Gegner vernichten werden muss, oder sogar beides. Man unterscheidet zwischen 2D und 3D Jump'n Run Spielen. In 2D wird die Spielfigur und Landschaft in der isometrischen Perspektive von der Seite gesehen. Erwartungskonform wird der Spieler horizontal von links nach rechts gesteuert, wobei es auch Spiele mit vertikaler Laufrichtung gibt, bei denen man über Leitern oder Plattformen nach oben gelangen muss. Bei Spielen in 3D bewegt sich die Spielfigur dreidimensional im Raum in der Zentralprojektion [jnr], was anfangs zu Schwierigkeiten bei der Koordination von Sprüngen führte [dgdj].

3.2 Jump'n Run Spiele und deren Steuerung auf Smartphones

Längst haben sich die Jump'n Run Spiele auch auf Smartphones mit Android-Betriebsystem verbreitet. Große Titel wie *Super Sonic* oder *Rayman* findet man genauso auf den Smartphones wie auf Spielekonsolen, auf denen sie ihren Ursprung hatten. Die Frage ist nun, wie die Spieleentwickler die Steuerung der Spielfigur auf den Smartphones umgesetzt haben. Zusätzliche Controller oder Tastaturen sind nicht sehr weit verbreitet, also muss die Steuerung mit dem Spiel über den Touchscreen verwirklicht werden. Einige Entwickler lassen als Lösungsansatz virtuelle Controllertasten auf dem Display anzeigen, über die man das Spiel steuert. Hierzu werden links und rechts im unteren Bereich des Displays jeweils ein virtueller Steuerknüppel angezeigt, der via Touchevents angetippt oder kreisen gelassen werden kann. Der Spieler bekommt das Gefühl das Smartphone selbst sei ein Controller (siehe Abbildung 3). Der linke Touchbereich steuert die Laufrichtung, während der rechte für den Sprung ausgerichtet ist. So kann man während eines Sprunges immer noch die Richtung variieren, um alle Münzen zu erreichen. Die Steuerung kann etwas gewöhnungsbedürftig sein und bedarf etwas Übung. Über die Tatsache, dass mit ständigem Positionieren der Daumen die Sicht behindert wird, kann noch hinweggesehen werden, da das Geschehen meist oberhalb der Steuerelemente stattfindet.

Abbildung 3: Sonic4 - Episode II: virtuelle Steuerknüppel als Controller

Eine weitere Art der Steuerung ähnelt der gerade vorgestellten Version. Das Spiel *Hungry Shark Evolution* handelt von einem Haifisch, der nur überleben kann wenn er genug Fische frisst. Der Spieler ist also gezwungen im Ozean auf Nahrungssuche zu gehen. Im Gegensatz zu herkömmlichen Jump'n Run Spielen kann sich der Hai in alle Richtungen bewegen und kommt somit auch ohne Plattformen aus. Da aber Fische eingesammelt und giftigen Quallen ausgewichen werden müssen, erfüllt es dennoch die Eigenschaften eines Jump'n Run Spiels. Der Jump erfolgt über einen Touch auf das Display, welcher den Hai einen Schub gibt um auch Vögel über der Wasseroberfläche zu fressen. Das Spiel verwendet auch wie bei *Sonic* einen virtuellen Steuerknüppel, der aber per Touch an beliebiger Stelle platziert werden kann. Dieser ist so lange an der Stelle fixiert bis man den Finger vom Touchscreen nimmt (siehe Abbildung 4). Hebt der Spieler wieder den Finger vom Display, so verschwindet auch der Knüppel. Dadurch kann aber die Sicht auf das Spiel durch den eigenen Finger ungewollt verdeckt werden.

Abbildung 4: Hungry Shark Evolution: Der Spieler kann selbst bestimmen, wo sein Touchbereich sein soll

Die am häufigsten vorkommende Art der Steuerung ist das gänzliche Weglassen von Controller-Elementen. Das Laufen der Spielfigur erfolgt automatisch. Die Figur läuft also konstant von links nach rechts, bis das Level beendet wurde. Die charakteristische Eigenschaft Springen erfolgt über ein kurzes einmaliges Berühren des Displays, so dass der Spieler sich nur auf Sprünge konzentrieren muss. Das Ausweichen von Hindernissen steht hierbei im Vordergrund. Abbildung 5 zeigt einen Auszug aus dem Spiel *Rayman Fiesta Run*, bei dem nur mittels Sprüngen die fliegenden Objekte eingesammelt werden müssen.

Abbildung 5: Rayman Fiesta Run: Timen von Sprüngen mittels Berühren des Displays, während sich die Figur von alleine vorwärts bewegt

3.3 Bekannte Spiele mit Sensorensteuerung

Der Smartphone Hersteller HTC brachte mit seinem Smartphone HTC Touch Diamond ein Spiel namens *Teeter* heraus, welches die Funktionen des Smartphones im Einklang mit dessen Sensoren präsentieren sollte. Das Spiel ist eine digitale Umsetzung eines Holzlabyrinthes, bei dem eine Metallkugel durch das Labyrinth mittels Kippen der Plattform manövriert werden muss, ohne in die Löcher zu fallen. In *Teeter* wird die Kugel durch Kippen des Smartphones bewegt.

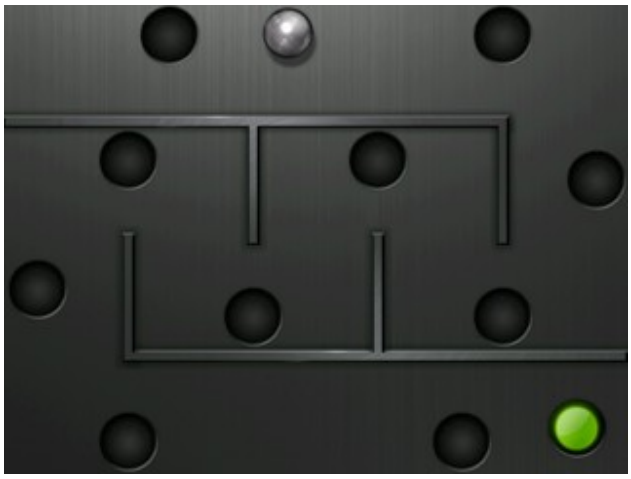

Abbildung 6: Manövrieren einer Kugel durch ein Labyrinth mittels Kippen im Spiel Teeter (Quelle: http://www.worldofppc.com/reviews /item/htc-touch-diamond)

Ein weiteres Spiel, bei dem Sensoren zur Steuerung eingesetzt werden ist *Traffic Racer*. Hier muss ein Auto durch den Verkehr gelenkt werden und Autos überholt oder auswichen werden. Jeweils links und rechts am unteren Bildschirmrand befinden sich Gaspedal und Bremse, die mit dem Finger betätigt werden müssen. Die Lenkung erfolgt ausschließlich über das Kippen des Smartphones nach links oder rechts. Punkte erhält der Spieler, wenn dieser so knapp wie möglich ein Auto überholt und es somit schneidet (siehe Abbildung 7). Da das Smartphone horizontal gekippt in der Hand liegt, bekommt der Spieler so den Eindruck, als wäre das Smartphone das Lenkrad. Die Pedale können so einfach mit den Daumen erreicht werden.

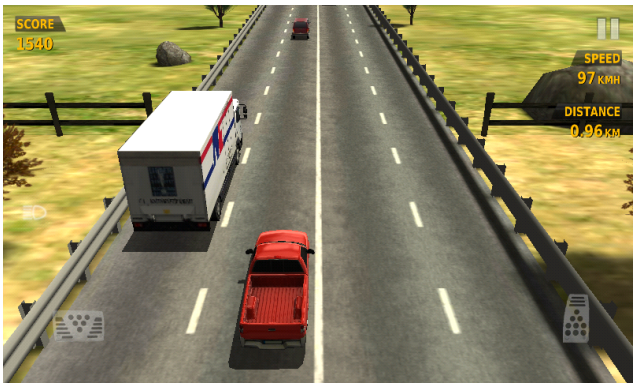

Abbildung 7: Traffic Racer, Lenkung mittels Kippen nach links und rechts

3.4 Jump'n Run Spiele mit Sensorensteuerung

Es gibt also bereits einige Spiele, die Sensoren zum Steuern verwenden. Interessant für diese Arbeit ist nun, ob es schon Jump'n Run Spiele mit Sensorensteuerung gibt. Tatsächlich gibt es welche. Ein bekanntes ist *Doodle Jump*. Es ist ein Plattform Game, bei dem man eine Figur vertikal über Plattformen durch permanentes Springen nach oben befördern muss. Die Steuerung erfolgt wie bei *Traffic Racer* durch den Beschleunigungssensor, um die Spielfigur nach links oder rechts durch Kippen zu steuern (siehe Abbildung 8).

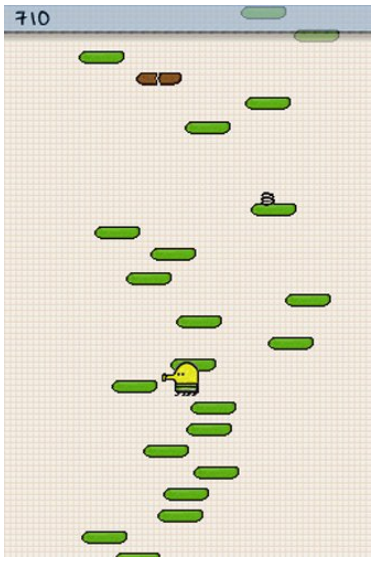

Abbildung 8: Doodle Jump: Kippsteuerung nach links oder rechts

Auch im 3D Bereich gibt es etliche Spiele, wie zum Beispiel *Temple Run*. *Temple Run* ist wie der Name schon sagt ein Runner Game, das heißt die Spielfigur rennt kontinuierlich nach vorne. Man muss somit Hindernissen durch Springen oder Ducken ausweichen. Über einen Wisch nach oben springt der Protagonist bei *Temple Run* in die Höhe. Um die Laufposition des Spielers zu ändern, muss man sein Handy nach links oder rechts kippen (siehe Abbildung 9 und 10). Eine Wischbewegung nach links oder rechts lässt eine Richtungsänderung um 90 Grad zu.

Abbildung 9: Temple Run, links rechts Bewegung, über einen Wisch zurRichtungsänderung

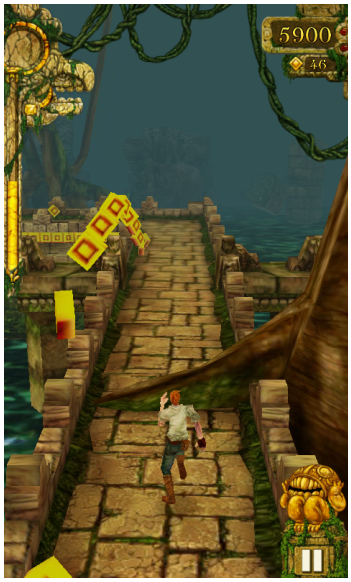

Abbildung 10: Temple Run, Wisch nach oben zum Springen

Ein weiteres Spiel in dieser Rubrik ist *Banana Kong*. Größtenteils spielt sich dieses Jump'n Run wie ein Spiel ohne Sensorensteuerung. Die Spielfigur, in diesem Fall ein Affe, läuft automatisch von links nach rechts durch das Level. Die Sprünge werden wie bei den meisten Jump'n Runs über Touchbefehle realisiert. Ziel ist es, Punkte durch Einsammeln von Bananen zu erzielen (siehe Abbildung 11). Während des Spielens gibt es aber durch Verwenden eines Power-Ups eine Möglichkeit ein neues Level zu erreichen. Der Spieler befindet sich nun in einer Unterwasserwelt, in der die Steuerung sich zusätzlich verändert. Die Figur wird jetzt durch Kippen des Smartphones gesteuert (siehe Abbildung 12).

3.5

Abbildung 11: Banana Kong ohne Sensorensteuerung

Abbildung 12: Banana Kong mit Sensorensteuerung, Kippen auf und ab

3.5 Fazit

Es gibt somit durchaus einige Spiele, die Sensoren zur Steuerung der Spielfigur miteinbeziehen. Auch bei Jump'n Runs ist eine Steuerung durch Kippen des Smartphones bereits realisiert, jedoch eher dezent eingesetzt worden. Die Schwierigkeit liegt hierbei, den Spieler nicht all zu sehr vom Geschehen durch hektische Bewegungen abzulenken.

Das zu entwickelnde Spiel sollte am Besten eine bereits bekannte Steuerung enthalten, um generell die Akzeptanz von Sensorensteuerung zu untersuchen, sowie neue Bewegungsmöglichkeiten anbieten, um die Grenzen bei der Benutzung aufzuzeigen. Für die bekannte Steuerung dient das Spiel *Banana Kong* als Vorlage, da der Einsatz von Sensoren elegant gelöst wurde und man immer noch die Steuerung von Jump'n Runs ohne Sensoren hat. Dadurch lassen sich besser Unterschiede in der Steuerung untereinander untersuchen. Eine neue Bewegungsart soll in einer zweiten Version des Spiels vorkommen, die gänzlich den Beschleunigungssensor für die Bewegung der Spielfigur nach links oder rechts einsetzt. So lässt sich auch zwischen den zwei Versionen die Steuerung und deren Akzeptanz bei den Spielern vergleichen. Es muss auch überlegt werden, ob das zu entwickelnde Spiel in 2D oder 3D entwickelt werden soll. Die Anfänge hatten die Jump'n Run Spiele in der Zweidimensionalität, wobei heutzutage die dreidimensionale Welt kaum wegzudenken ist und sich dadurch neue Möglichkeiten eröffnen. Trotzdem soll das Spiel in 2D entwickelt werden, um bei den Wurzeln des Jump'n Runs zu bleiben. Außerdem könnten sich durch die Dreidimensionalität Schwierigkeiten bei der Steuerung und Kontrolle der Spielfigur ergeben, welche durch unkoordinierte Sprünge den Schwierigkeitsgrad des Spiels unnötig erhöhen würde.

4 Konzeption

4.1 Spielidee

Die Idee ist ein klassisches Jump'n Run Spiel zu entwickeln, bei dem eine Spielfigur durch ein Level geführt, Gegenstände eingesammelt und Hindernissen ausgewichen werden muss. Hierzu wird die Welt zweidimensional von der Seite betrachtet. Die Spielfigur in dem Spiel ist ein Pinguin, der in jedem Level Fische einsammeln muss, die er unter anderem über Plattformen oder Kisten mittels Sprünge erreichen kann. Als Hindernisse dienen Abgründe, in die der Pinguin hineinfallen kann oder Bomben, die nicht berührt werden dürfen.

Da in dieser Arbeit die Steuerung mittels Beschleunigungssensor bei Jump'n Run Spielen getestet werden soll, muss ein Unterschied zu herkömmlichen Steuerung erkennbar sein. Hierzu wechselt die Steuerung von Level zu Level, um später den Sinn von Sensorensteuerung zu untersuchen. Des Weiteren

wurde auch eine zweite Version des Spiels entwickelt, bei der das gleiche Spiel komplett anhand von Sensoren gespielt wird.

Das Spiel setzt sich aus insgesamt vier Level zusammen. Bei dem ersten Level erkennt man deutlich die Grundzüge eines Jump'n Run Spiels. Es gibt Plattformen, Graben und Kisten als Hindernisse sowie Fische zum Einsammeln. In Level zwei macht sich die Einsetzung des Beschleunigungssensor bemerkbar, da hier die Spielfigur durch das Kippen des Smartphones gesteuert wird. In diesem Spielabschnitt befindet sich die Spielfigur Unterwasser und muss Fische fressen und Bomben ausweichen. (vgl. Abbildung 13). Im darauffolgendem Level befindet sich der Pinguin wieder an Land, wo sich das Szenario vom ersten Level wiederholt. Am Ende angekommen fliegt der Pinguin im viertel Level an einem Luftballon hängend durch die Luft und muss fliegende Fische einsammeln, wobei die Steuerung wieder wie in Level zwei mit Hilfe der Sensorsteuerung erfolgt.

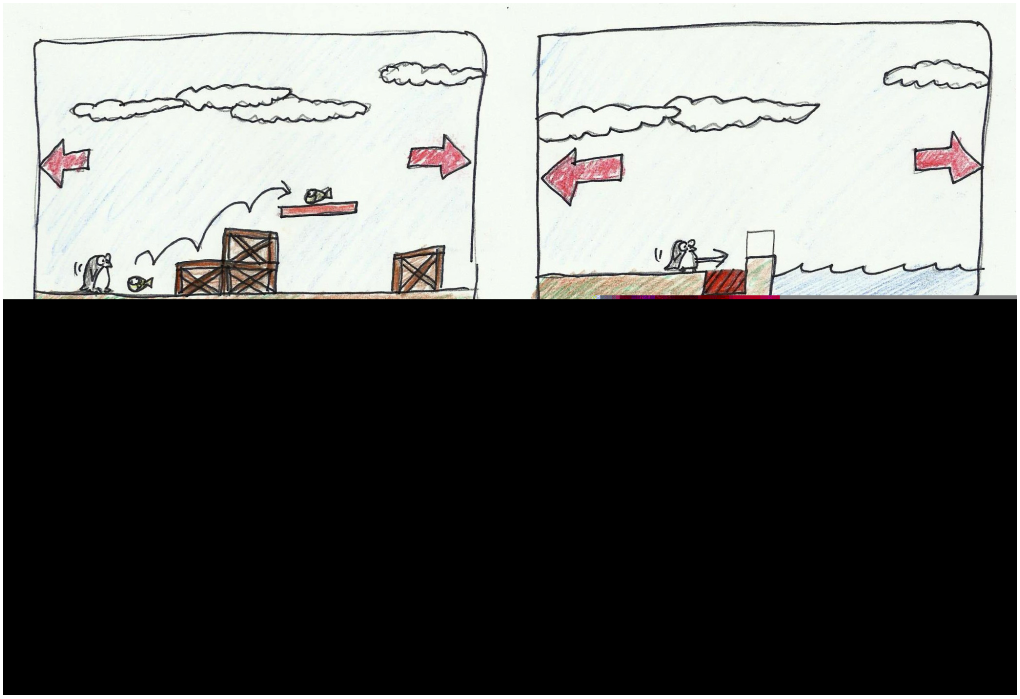

Abbildung 13: Skizzierung Leveldesign und Steuerung von PinguTime

4.2 Story

Das Jump'n Run Spiel trägt den Namen „PinguTime“. Wie der Name schon vermuten lässt, ist der Hauptcharakter ein Pinguin, der in einer sich je nach Level verändernden Welt Fische sammelt und sich auf die Reise macht, um die Fähigkeit des Fliegens zu erlernen. Der tapfere Held kämpft sich auf einer Insel über tiefe Schluchten und spitzen Stacheln hindurch, taucht in den Weiten des Ozeans ein und weicht gefährlichen Wasserbomben aus, bis er am Ende am Südpol auf einen Heißluftballon trifft, der ihm zu der Fähigkeit

des Fliegens verhilft. Um gestärkt die schwierige Reise anzugehen, muss er sich von Fischen ernähren, die er auf dem Weg findet. Bei seinem Glück trifft er während seines Fluges mit den Heißluftballon auch auf fliegende Fische, die auch in der Luft seinen Appetit stillen werden.

4.3 Konzept zur Umsetzung

In diesem Abschnitt wird erklärt, wie das Spiel technisch umgesetzt werden soll. Hierzu werden alle wichtigen Elemente der Spieleprogrammierung vorgestellt, die verwendet wurden. Dazu gehört das Betriebssystem, die Engine, sowie die Soft- und Hardware.

4.3.1 Android

Android ist ein auf Linux basierendes Betriebssystem für Smartphones, Netbooks und Tablets. Es ist eine Open-Source Software, die von Google im Jahre 2008 entwickelt wurde [and]. Durch die Tatsache, dass der Quelltext frei zugänglich ist, lassen sich anders als bei Apple schnell und einfach Anwendungen für Android entwickeln und zügig im eigenen Android Market namens Google Play veröffentlichen [andVN]. Zudem erweitert sich der Marktanteil von Android zunehmend: Wie in Abbildung 14 ersichtlich besitzt Android im Jahr 2013 einen Marktanteil von 78,6 Prozent, während hingegen das Betriebssystem iOS von Apple Marktanteile immer weiter verliert.

Smartphonemarkt 2013 laut IDC (Verkaufszahlen in Mio. Stück)

Betriebssystem	Verkaufszahlen 2012	Marktanteil	Verkaufszahlen 2013	Marktanteil	Veränderung zum Vorjahr
Android	500,1	69,0 %	793,6	78,6 %	58,7 %
iOS	135,9	18,7 %	153,4	15,2 %	12,9 %
Windows Phone	17,5	2,4 %	33,4	3,3 %	90,9 %
Blackberry	32,5	4,5 %	19,2	1,9 %	-40,9 %
Andere	39,3	5,4 %	10,0	1,0 %	-74,6 %
Gesamt	725,3	100,0 %	1009,6	100,0 %	39,2 %

Abbildung 14: Smartphonemarkt 2013 (Quelle: http://www.zdnet.de/88184149/idc-android-erhoeht-anteil-smartphonemarkt-auf-fast-80-prozent/)

In dieser Arbeit wird mit diesem Betriebssystem gearbeitet und eine App hierfür entwickelt, da es auch durch persönliche Erfahrung eine Vielzahl von Vorteilen mit sich bringt.

4.3.2 AndEngine

Die AndEngine ist ein spezielles Framework für die Spieleentwicklung in 2D unter Android. Es wurde von Nicolas Gramlich entwickelt und ist Open-Source und somit frei zugänglich. Sie bietet verschieden Funktionalitäten einer Laufzeitumgebung, unter anderem eine Grafik- und Physik-Engine, Soundsystem, Steuerung sowie eine Datenverwaltung. Dabei benutzt sie die Hardwarebeschleunigung Open GL ES [VAEPr12]. Der Vorteil dieser Engine ist es, dass sowohl Anfänger als auch fortgeschrittene Programmierer mühelos mit ihr umgehen können und sie alles mit sich bringt, was für die Spieleentwicklung für Android-Geräte in 2D von Nöten ist [ae2D]. Sollen Spiele in 3D entworfen werden, so müsste eine andere Engine dafür verwendet werden.

4.3.3 Android SDK

Da das Spiel in 2D entwickelt werden soll, wird hierfür die Software Android Software Developer Kit (Android SDK) verwendet, mit dem man in der Programmiersprache Java mit Hilfe des Plugins Android Development Tools (ADT) für die Entwicklungsumgebung Eclipse Apps programmieren kann. Das ganze bietet Google auf seiner Developer-Seite [andsdk] als Bundle an, welches Eclipse mit dem ADT-Plugin, den Android SDK Tools und einem eigenen Emulator Android Virtual Device (AVD) enthält.

Das Spiel soll vorerst für ein Smartphone programmiert werden, um das Spiel an eine breitere Masse zu verteilen, da momentan mehr Menschen Smartphones als Tablets besitzen. Für dieses Spiel wurde das Smartphone Samsung Galaxy S II GT-I9100 als Entwicklersmartphone verwendet. Es besitzt einen Dual-Core-Prozessor mit 1,2Ghz-Taktung, 1 GB Ram, sowie ein 4,27“ großes Display, welches für das Spiel vollkommen ausreichend ist.

5 Implementierung

5.1 Entwurf einer Rahmen-Applikation

5.1.1 Activity

Für das Spiel wird zunächst einmal ein Benutzerinterface benötigt. Dieses wird mit einer Activity-Klasse realisiert, die eine Interaktion zwischen Spieler und Smartphone ermöglicht. Mit ihr kann ein Fullscreen-Fenster gestartet werden und Befehle gegeben werden, was mit der App geschehen soll, wenn zum Beispiel der Zurück-Button betätigt wird. Im Normalfall schließt sich die App oder es geht ein Fenster weiter zurück. Die zu entwickelnde Rahmen-App soll zunächst ein Startbildschirm starten [act].

Mit Hilfe der AndEngine kann von dem Interface BaseGameActivity Gebrauch gemacht werden. Die Activity erbt von ihr hilfreiche Methoden, mit denen beim Starten der App Ressourcen wie Texturen, Regionen oder Sprites erstellt werden können. Abbildung 15 zeigt, wie sich so ein Spiel zusammensetzt. Die BaseGameActivity verwaltet die Kamera, Engine, Scenes, Entities wie Sprites oder Texturen. In den folgenden Kapiteln wird näher darauf eingegangen [fgt2].

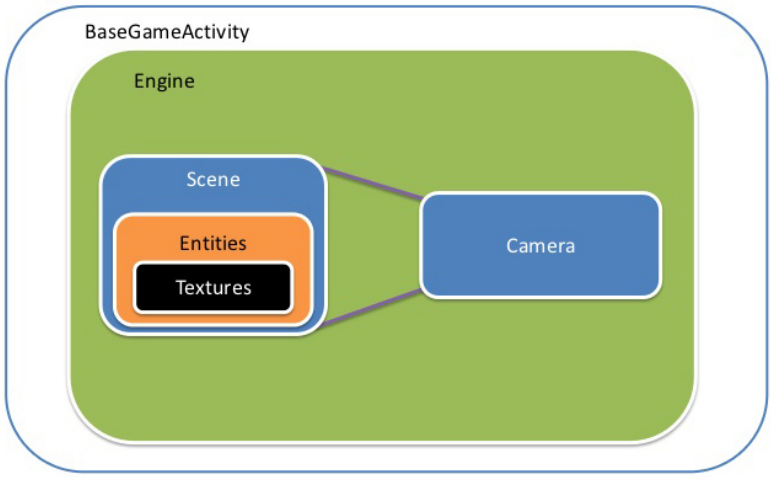

Abbildung 15: Aufbau eines Spiels(Quelle: http://de.slideshare.net/ANDLABSMUNICH/andengine)

5.1.2 Sprites

Ein Sprite ist ein zweidimensionales Bitmap an Stelle der angegebenen x, y Koordinaten. Eine Erweiterung ist der Tiled Sprite oder Animated Sprite, mit denen sich Animation erzeugen lassen [spr]. Da in einem Spiel logischerweise sehr viele Texturen und Sprites dargestellt werden sollen, ist es ratsam einen ResourcesManager zu implementieren. Er verwaltet die zu erzeugenden Entities um Übersicht und Kontrolle zu wahren, da eventuell nicht alle Texturen zu einem bestimmten Zeitpunkt verwenden werden sollen. Der ResourcesManager besitzt also viele einzelne load-Methoden mit Inhalten, die zusammengehören und geordnet aufrufbar sind [fgt3].

5.1.3 Scenes

Sollen verschiedene Grafikoberflächen in dem Spiel existieren, beispielsweise ein Menü, Ladebildschirm oder unterschiedliche Leveldesigns, kann das mit Scenes realisiert werden. In einer Scene lassen sich Hintergrund, Sprites, Buttons oder Text neu implementieren. Wird ein neues Erscheinungsbild benötigt, so kann einfach zwischen den Scenes gewechselt werden [scen]. Gerade bei der Implementierung eines Spiels können mit Scenes neue Level mit unterschiedlichen Charakterwerten geladen werden.

Für *PinguTime* wird zunächst eine abstrakte Klasse BaseScene implementiert, die Methoden enthält, die bei allen Scenes identisch sind. Sie enthält eine Methode createScene, die alle wichtigen Methoden zur Erstellung dieser Szene ausführt, sowie eine Methode onBackKeyPressed, die entscheidet, was bei der Zurück-Taste an der Hardware geschieht. Zusätzlich besitzt sie eine Methode disposeScene, die beim Wechsel zu einer anderen Scene die Ressourcen wie Sprites wieder entbindet. Eine weitere Methode ist getSceneType, die den Szenentyp zurückgibt, damit der Entwickler weiß in welcher Scene er sich gerade befindet. Dies ist notwendig, da für die Umschaltung der Scenes eine Hilfsklasse SceneManager implementiert wird. In dieser wird mit switch/case Statements die jeweiligen Scenes gesetzt und ruft die erforderlichen Ressourcen über den ResourcesManager auf [fgt4].

5.1.4 PhysicWorld

Der erste Versuch war die Implementierung einer Scene namens GameScene, die erst einmal eine Figur, eine Plattform und einen Hintergrund darstellen soll. Die AndEngine stellt hierfür eine Klasse PhysicWorld bereit, mit der sich einen Gravity-Parameter setzen und die Figur mit der Umgebung verbinden lässt. Sie dient als eine Art Container für alle Objekte in der PhysicWorld. Erstellt man einen Sprite *ground* in einer angegeben Farbe kann man nun über die Klasse PhysicsFactory der AndEngine die Methode createBoxBody aufrufen, die den Sprite als Gegenstand in der Spielewelt einbindet. Sie erstellt so also einen Body in der PhysicWorld, der mit der Umgebung interagiert [pht]. Wird die Spielfigur irgendwo oberhalb einer erstellten Plattform gesetzt, so fällt diese beim Starten der GameScene mit dieser Physik auf die Plattform und bleibt dort stehen. Genauso können Quadrate erstellt werden, die als Hindernisse benutzt werden können, da die Spielfigur nicht durch diese hindurchgehen kann. Die Steuerung des Spielers sollte nach dem Konzept, wie in Abbildung 13 dargestellt, mittels Buttons realisiert werden. Beim Drücken des Buttons mit dem Finger soll sich die Spielfigur nach links oder rechts bewegen können. Beim ersten Test sollten sich durch Klicken der Buttons die Farbe der Plattform ändern, um erste Funktionen zu

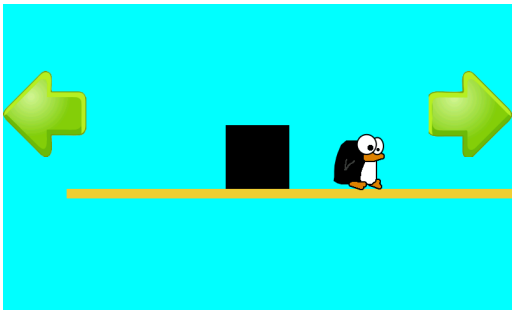

Abbildung 17: Rahmen-App vor Betätigung des Buttons

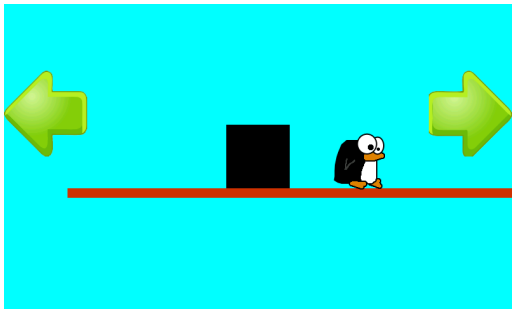

Abbildung 16: Rahmen-App bei Betätigung des Buttons

testen. Die ersten Ansätze eines Spiels wurden realisiert: Ein Spieler, Plattform, Hindernis und Hintergrund.

5.2 Ausbau der Rahmen-Applikation

5.2.1 Startbildschirm

Weiterführend wurde eine neue Scene hinzugefügt: die CoverScene. Sie zeigt ein Startbildschirm für das Spiel. Wird die App gestartet, erscheint das Spiellogo, welches nach 3 Sekunden die GameScene startet. Somit kann der Spieler genau nachvollziehen, welche App er gerade gestartet hat, falls er sich dessen nicht genau bewusst war.

Abbildung 18: Startbildschirm PinguTime

5.2.2 Loading Screen

Um Wartezeiten für das Laden der GameScene zu verkürzen, wurde auch eine LoadingScene eingebunden, die beim Starten der GameScene aufgerufen wird und sobald die GameScene-Ressourcen fertig geladen wurden die GameScene über die Methode

```
onTimePassed(final TimerHandler pTimeHandler) { ... }
```

via eines TimeHandlers startet. Die LoadingScene zeigt einfach nur einen „Loading...“-Text an, um den Spieler darüber zu informieren, dass die Spieldateien momentan geladen werden [fgt10].

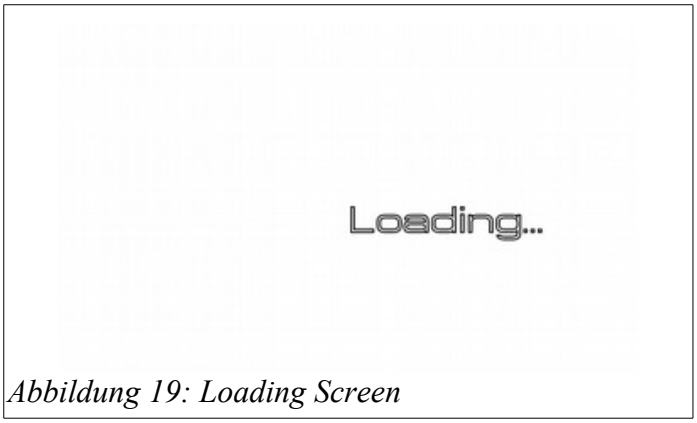

Abbildung 19: Loading Screen

5.2.3 Animated Player

Um mehr Realismus in das Spiel zu bekommen wurde die Spielfigur animiert. Wie schon in Kapitel 5.1.2 erwähnt ist ein AnimatedSprite eine Abwandlung eines Sprites. Dieser bekommt anstatt einer BitmapTextureRegion eine TiledTextureRegion zugewiesen, die mehrere Texturen in einem Bild verarbeiten kann. Hierzu muss die Spalten- und Reihenzahl angegeben werden, damit die verschiedenen Texturen gefunden und nacheinander abgespielt werden können (siehe Abbildung 20).

Abbildung 20: Player-Textur für Animation

Mit

```
pinguSprite.animate(PLAYER_ANIMATE, 0 ,1, true);
```

wird die Animation gestartet. Diese Methode wird ausgeführt, wenn der Spieler nach dem Laden des Levels auf das Display drückt, um das Spiel zu starten. Der Pinguin bewegt sich nun animiert in fort.

5.2.4 Fische

Wie jedes Jump'n Run Spiel braucht der Protagonist auch etwas zum einsammeln. Bei einem Pinguin bietet es sich an dafür Fische zu nehmen. Die Fische sind ebenso Sprites wie der Pinguin selbst. Um einen Aufmerksamkeitseffekt hinzubekommen, wurde der Fisch mit einem Pumping-Effekt versehen, das heißt, der Fisch bläht sich auf und nimmt dann

wieder ab. Dies geschieht über einen EntityModifier mit Hilfe eines Loops und einem ScaleModifier. In einer Schleife skaliert er so immer wieder die Größe des Fisches um einen float Wert 1.3f:

```
registerEntityModifier(new LoopEntityModifier( new ScaleModifier
                        (1, 1, 1.3f)));
```

Da jetzt ein Startbildschirm, eine Loading-Funktion und ein Spielrahmen vorhanden sind, können schönere Grafiken eingebunden und die Funktionen des Spielers wie Laufen und Springen erweitert werden.

Da das Spiel auf einem 4,27“-Display entwickelt wurde, stellte sich schnell heraus, dass die Buttons links und rechts einfach zu viel Spielfläche verdecken, wenn die Finger auf diese gelegt werden. Bei einem Jump-and-Run-Spiel ist es beim Spielverlauf wichtig zu wissen, was vor einem liegt, damit der Spieler auf Hindernisse oder Gruben rechtzeitig reagieren kann. Nach dieser Erkenntnis wurde die Steuerung geändert: Der Pinguin läuft kontinuierlich von alleine nach vorne und ein Sprung erfolgt durch einmaliges Tippen auf das Display. Dadurch bleibt die gesamte Spielwelt auf dem Display ersichtlich.

5.3 Steuerung der Spielfigur ohne Sensorensteuerung

Mit dem PhysicsConnector wird der Body der Spielfigur automatisch geupdatet und reagiert sofort auf dessen Veränderungen und Updates. Wie schon erwähnt soll die Spielfigur von alleine laufen. Damit beim Starten des Levels der Pinguin nicht gleich los rennt, beginnt das Spiel erst nach einmaligem Berühren des Displays. Eine boolean Variable *canRun* wird somit auf true gesetzt. Solange dieser Wert gesetzt ist, wird mit der Methode onUpdate(float pSecondsElapsed) des PhysicsConnectors die Lineare Beschleunigung des Player-Bodys konstant auf einen Wert gesetzt. Die Spielfigur läuft nun von alleine vorwärts. Sinkt der y-Wert der Figur unter einen bestimmten Wert, zum Beispiel bei einem Fall in eine Grube, so wird die Methode onDie() aufgerufen, die einen „Game over“-Text anzeigt und das Level neu startet. Mit der Methode *onSceneTouchEvent(Scene pScene, TouchEvent pSceneEvent)* lassen sich Aktionen über Displayberührungen steuern. So wird hier abgefragt, ob das Display schon einmal berührt wurde und das Spiel somit starten soll. Des Weiteren bedeutet im else-Fall, dass eine Berührung auf dem Display die Spielfigur springen lässt. Hierfür wird einfach die Höhe des Pinguins in der linearen Beschleunigung um einen konstanten Wert erhöht, so dass die Spielfigur anfängt zu springen und über die Physikengine wieder von alleine fällt.

Abbildung 21: Screenshot von Level 1: laufender Pinguin sammelt Fische im Dschungel

5.4 Steuerung anhand von Sensoren

Für die Steuerung der Spielfigur wurde ein neues Level erstellt, das ganz zu der Steuerung mittels Kippen des Smartphones nach oben und unten passt. Am Ende des ersten Levels springt der Spieler in das Wasser. Ab hier beginnt das zweite Level. Der Spieler schwimmt nun unter Wasser und kann sich so frei auf und ab bewegen während der Pinguin sich konstant vorwärts bewegt. Die Höhe der Spielfigur wird anhand der Sensordaten des Beschleunigungssensors bestimmt [senso].

5.4.1 Auslesen von Sensordaten

Das Auslesen der Sensordaten funktioniert über den SensorEventListener und den SensorManager. Über den SensorManager hat man Zugriff auf alle Sensoren, die ein Smartphone mit sich bringt. Nicht alle Smartphones haben die gleichen Sensoren eingebaut, deshalb ist es sinnvoll vorher anzufragen, ob der gewünschte Sensor überhaupt vorhanden ist. Der SensorManager bringt eine Liste der verfügbaren Sensoren mit sich, mit der gewünschte Sensoren ausgewählt werden können.

Auf den Beschleunigungssensor kann mit

```
sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER).get(0)
```

zugegriffen werden. Dies geschieht in der Hilfsklasse AccelerometerHelper, welche das Interface SensorEventListener erweitert. In Codeauszug 1 sieht man den Konstruktor des AccelerometerHelpers, in dem zuerst der SensorManager angelegt wird. Danach wird über die SensorList geprüft, ob ein Accelerometer vorhanden ist. Ist kein Sensor dieser Art vorhanden, würde

die Anfrage null zurückgeben. Nach Anlegen eines Accelerometers *mAccelerometer* kann über den registerListener Daten des Sensors empfangen werden [pm].

```
public AccelerometerHelper (Activity activity){
        mSensorManager = (SensorManager)activity.getSystemService
        (Context.SENSOR_SERVICE);
        if(mSensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER).size()!=0)
        {
                mAccelerometer = mSensorManager.getSensorList
                (Sensor.TYPE_ACCELEROMETER).get(0);
                mSensorManager.registerListener
                (this,mAccelerometer,SensorManager.SENSOR_DELAY_GAME);
        }
}
```

Codeauszug 1: Anlegen des SensorManagers in der Klasse AccelerometerHelper

Das Interface SensorEventListener bietet mehrere Methoden an, wie zum Beispiel die Methode *onSensorChanged(SensorEvent event)*, mit der über einen Event auf die jeweiligen Sensorwerte zugegriffen werden kann. Wie schon in Kapitel 2.1 erläutert, besitzt der Beschleunigungssensor drei float Werte x,y und z. Der Wert x wäre bei dem Spiel die aktuelle Kippausrichtung nach links oder rechts (hier MOVE genannt), der y-Wert dementsprechend die Kippausrichtung nach oben oder unten (vergleiche TILT). Im Codeauszug 2 werden die x und y Werte des Beschleunigungssensors in MOVE und TILT gespeichert.

```
public void onSensorChanged(SensorEvent event){

        TILT = event.values[0];
        MOVE = event.values[1];
}
```

Codeauszug 2: Abrufen von x,y-Werte des Beschleunigungssensors

5.4.2 Umsetzung auf Spielfigur

Im Unterwasserlevel steuert der Spieler den Pinguin durch Kippbewegung des Smartphones auf und ab, während er automatisch konstant vorwärts schwimmt. Soll nun die Höhe der Spielfigur durch Kippen verändern werden, so wird auf den x-Wert (TILT) des Accelerometers aus der AccelerometerHelper-Klasse zugegriffen und in den y-Wert der Linearen Beschleunigung eingefügt. Durch Multiplikation des Wertes lässt sich die Intensität regulieren. Der x-Wert der LinearVelocity ist eine Konstante, die

den Pinguin automatisch mit konstanter Geschwindigkeit vorwärts schwimmen lässt. Wird nun die lineare Beschleunigung der Spielfigur in einer onUpdate(float pSecondsElapsed)-Methode gesetzt, so verändert sich bei jeder Lageänderung des Smartphones auch der y-Wert des Pinguins.

```
playerbody.setLinearVelocity(8, AccelerometerHelper.TILT*10);
```

Bei einer Laufbewegung in x-Richtung der Spielfigur muss der float-Wert MOVE in der AccelerometerHelper-Klasse der LinearVelocity hinzugefügt werden. Er erhöht/verringert durch Kippen des Smartphones nach rechts/links die Geschwindigkeit des Pinguins. Dieser Effekt wird bei Game 2 benutzt und dient zum Vergleich der Sensorsteuerung bei der Laufbewegung.

```
playerbody.setLinearVelocity(AccelerometerHelper.MOVE*5,
            playerbody.getLinearVelocity().y);
```

5.5 Kollision

Die AndEngine liefert schon eine eigene Kollisionsabfrage. Kollisionen können so einfach mithilfe von if-Abfragen gemanagt werden. Trifft ein Sprite auf eine andere Instanz, so kann einfach implementiert werden, was danach passieren soll. Im Falle einer Kollision des Spielers mit einem Fisch, erhöht sich der Score um 1 und der Fisch verschwindet.

```
protected void onManagedUpdate (float pSecondsElapsed) {
       super.onManagedUpdate(pSecondsElapsed);

       if (pinguSprite.collidesWith(this)){
              addToScore(1);
              this.setVisible(false);
              this.setIgnoreUpdate(true);
       }
}
```

Codeauszug 3: Kollisionsabfrage mit einem Fisch

5.6 Levelloader

Für das Erstellen und Laden von Level wurde der LevelLoader aus [fgt11] entnommen. Er verwendet die AndEngine Klasse SimpleLevelLoader, die XML-Dateien laden und parsen kann. Über einen SAX-Parser läuft er über die XML-Datei und sucht nach vorher definierten Tag-Attributen anhand von if-else-Bedingungen und liest bei Übereinstimmung deren x- und y-Werte aus und erstellt mit der zu dem Tag gehörigen Textur einen Sprite. So kann in der XML-Datei beliebig viele Plattformen, Fische oder Bomben an jeder beliebigen Stelle generiert werden. So können weitere Level generiert werden.

5.7 Zusammenfassung und Ausblick

Das Spiel enthält nun zwei Versionen unterschiedlicher Steuerung, die im Hauptmenü unter Game 1 und Game 2 ausgewählt werden können. Der Spieler wird in beiden Fällen durch vier identische Level geschickt und sammelt dabei Punkte durch die Aufnahme von Fischen. Die Punktzahl wird rechts oben im Bild angezeigt. Im Code wird ein Highscore hochgezählt, der aber noch nicht in der App zum Beispiel durch den vorhandenen Menüpunkt Highscore zur Anzeige gebracht wird. Genauso fehlen noch Randbegrenzungen in den Level selbst. So ist es noch möglich, über einen Sprung in Kombination mit der Sensorensteuerung in Game 2 am Anfang des Levels über eine Rückwärtsbewegung über den Levelrand hinaus zu gelangen. Im Unterwasserlevel muss der Kollisionsbereich der Bomben noch besser angepasst werden, da für die runden Bomben eine quadratische Textur gewählt wurde, die bei dessen Berührung das Level neu startet. So kann man im Spiel beim knappen Ausweichen der Bomben dennoch an die Ecken der Texturen gelangen. Dies sollte bei der Weiterentwicklung des Spiels noch verbessert werden. Auch die Sprünge selbst könnten noch über eine bessere Sprungkurve und der Verwendung eines Double-Jumps durch mehrmaliges Klicken auf das Display ergänzt werden. Bei der Installation des Spiels auf Geräten ab einer Displaygröße von 5“ kann es vor allem bei Nexus Geräten zu Texturfehlern in Level 1 kommen, da das Spiel für das Samsung Galaxy S2 mit einer Auflösung von 800x480 entwickelt wurde und Probleme beim Laden von Texturen auftreten können. Manche Geräte konnten zudem das letzte Level nicht starten. Diese Kompatibilitätsprobleme müssen nach dieser Arbeit noch weiter untersucht und behoben werden. Das Spiel lief bisher auch auf einem Samsung Galaxy S4 immer noch einwandfrei.

6 Test des Spielspaßes und Akzeptanz der Steuerung

Nach der Implementation des Spieles muss nun getestet werden, ob die Steuerung mittels Sensoren für das Jump'n Run-Spiel sinnvoll ist. Hierzu werden Informationen benötigt, wie die beiden Spiele generell bei den Usern ankommen und wie sie sich mit den Steuerungen zurecht finden. Getestet wird hierbei auch das Flow-Erleben während des Spiels, wobei der Fokus auf dem Vergleich beider Varianten mit deren Einsetzung von Sensoren liegt.

6.1 Flow-Erleben

„Das Flow-Erleben ist ein ganzheitliches Gefühl, das erlebt wird, wenn man ganz in der Aktivität aufgeht“ ([CM85], S. 59). Es wurde im Jahre 1975 vom Psychologen Mihaly Csikszentmihalyi beschrieben und befasst sich mit dem Phänomen, warum Menschen bei Tätigkeiten, die als positiv empfunden werden, in einen Flow-Zustand geraten, bei dem sie eins mit der Handlung und dem Bewusstsein sind. Voraussetzung dafür ist, dass man die Tätigkeit

auch durch die eigenen Fähigkeiten meistern kann. Deshalb gerät man häufig beim Spielen in einen Flow-Zustand [flowk].

Die Untersuchung des Flow-Erlebens ist also ein wesentlicher Bestandteil beim Testen eines Spiels. Ist ein Spieler im Flow-Zustand, so stellt sich dieser neuen Herausforderungen und ist gewillt, diese eigens zu bewältigen. Die Messung des Flows erfolgt in diesem Test über einen Fragebogen [flow] [RVE03].

6.2 Testvorbereitung und Ablauf

Getestet werden das Spiel und dessen Steuerungsmöglichkeiten mit möglichst vielen Testpersonen unterschiedlichen Alters. Um gute Ergebnisse zu bekommen, sollten sowohl junge als auch ältere Personen das Spiel testen, die auch unterschiedlich gut im Umgang mit Smartphones oder Spielen generell sind. Untersucht werden soll das Flow-Erleben während des Spiels, das Spiel an sich und hauptsächlich das Zurechtkommen mit der Steuerung anhand von Sensoren innerhalb von Game 1 und im Vergleich mit Game 2. Dafür wurde ein Fragebogen angefertigt, der zusätzlich auch als Online-Formular abgerufen werden kann. Dadurch lässt sich dieser leichter verteilen und auswerten. Das Online-Formular wurde mit Google Forms unter [Gforms] erstellt. Wird der Fragebogen online verschickt, so bekommen alle Testpersonen zusätzlich einen Link zu der Installationsdatei, sowie eine Beschreibung und Anleitung des Spiels, welche sie zuvor durchlesen sollen. Alle anderen erhalten ein Testgerät mit dem vorinstallierten Spiel. Bevor diese das Spiel durchspielen, müssen auch sie eine genaue Beschreibung und Anleitung für das Spiel aufmerksam durchlesen (siehe Anhang Nutzertest PinguTime), um Informationen darüber zu erhalten, um was es in diesem Spiel geht und wie es gespielt wird. Danach sollen sie zuerst Game 1 und dann Game 2 bis zum Ende durchspielen und abschließend den Fragebogen schriftlich ausfüllen. Wichtig ist den Testpersonen dabei nicht über die Schulter zu schauen, sondern sie alleine das Spiel durchspielen zu lassen, da sie sonst unter Druck oder Anspannung stehen würden und die Messergebnisse dabei verfälscht werden können [msu].

6.2.1 Fragebogen

Der Fragebogen bedient sich der Fünf-Punkte-Skala von Likert. Er enthält für die Fragen fünf Antwortmöglichkeiten von einer Aussage, die von „trifft zu“ bis „trifft nicht zu“ abgestuft werden. Der Fragebogen setzt sich aus vier Teilen zusammen: Zuerst wird neben allgemeinen Angaben über Alter und Geschlecht die Häufigkeit der Nutzung von Smartphones und speziell von Spielen abgefragt, um die Testpersonen besser differenzieren und einschätzen zu können. Außerdem bekommt man so Eindrücke, welche Vorkenntnisse die Personen vor dem Spielen bereits besitzen. Der zweite Teil befasst sich mit der Untersuchung von Game 1 von *PinguTime*. Es werden Eindrücke abgefragt, wie die Tester das Spiel im Ganzen finden, wie die

Sensorensteuerung zum Spiel passt und ob sie stets die Kontrolle der Spielfigur behielten. Darüber hinaus müssen Angaben gemacht werden, ob durch das Kippen des Smartphones die Sicht beeinträchtigt wird. Dies ist ausschlaggebend, da sich daraus den Sinn über Sensorensteuerung bei Jump'n Run Spielen erkennen lässt. Da sich die Steuerung bei Game 1 von Touch zu Kippen jeweils nach einem Level ändert, soll herausgefunden werden, welche Steuerung besser zum Spiel passt und welches Level am leichtesten ist. Ebenso sollen die Sprünge und der Schwierigkeitsgrad des Einsammelns der Fische bewertet werden. Abschließend kann in einem dafür vorgesehenem Textfeld geschrieben werden, was an dieser Version gut und nicht gut war und welche Verbesserungsvorschläge die Testperson hat. Im Dritten Abschnitt des Fragebogens werden die gleichen Aspekte von Game 2 wie bei Game 1 behandelt. So kann man besser die zwei Varianten miteinander vergleichen und Unterschiede im Spielerleben erkennen. Der letzte Teil enthält Fragen über beide Spiele zusammen, unter anderem welches Spiel einem im Vergleich besser gefällt oder welche Steuerung besser zum Spiel gepasst hat.

6.3 Auswertung

Den Fragebogen haben insgesamt 18 Testpersonen im Alter von 15 bis 65 Jahren beantwortet. Unter ihnen waren 11 männliche und 7 weibliche Personen (siehe Abbildung 17) mit einem Altersdurchschnitt von 29,6 Jahren.

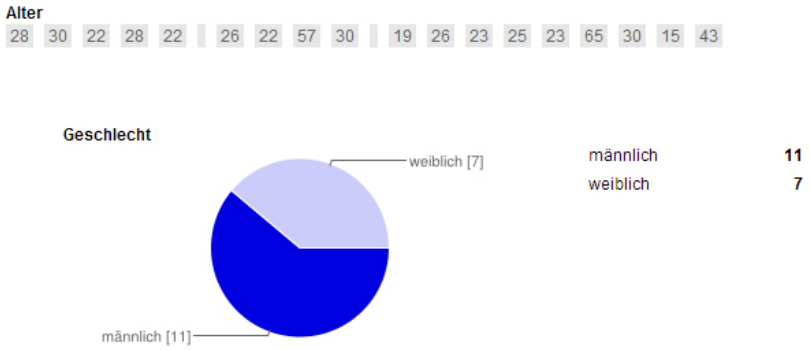

Abbildung 22: Alter und Geschlecht der Testpersonen

Die erste Frage nach den persönlichen Angaben von Abbildung 17 handelte von der Häufigkeit der Nutzung von Smartphones. Diese wurde von 72 Prozent der Teilnehmer mit sehr oft beantwortet. Nur zwei Leute nutzen es gelegentlich. Die Frage über das Spielverhalten auf dem Smartphone fiel nicht so deutlich aus: Nur 22 Prozent spielen sehr oft auf ihrem Handy, die meisten der Einschätzungen liegen bei einem Wert von 2 und 3 (vgl. Abbildung 18).

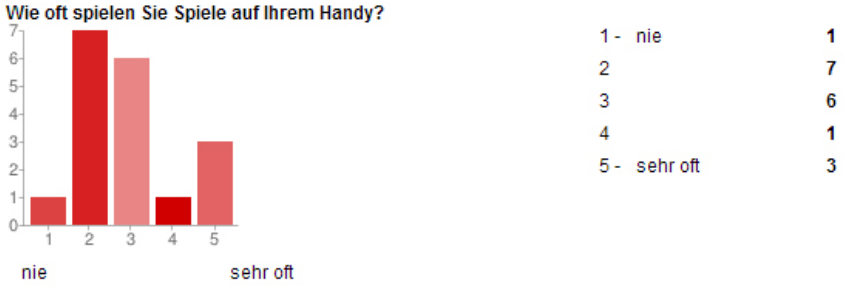

Abbildung 23: Spielverhalten der Testpersonen auf dem Smartphone

89 Prozent geben an, dass sie schon einmal Spiele mit Sensorensteuerung gespielt haben. Nur zwei der Testpersonen haben noch keine Erfahrungen damit gemacht. Zum Spiel selbst geben sie an, dass ihnen Game 1 besser gefallen hat als Game 2 (siehe Abbildung 19). Die Mehrheit hat bei Game 1 einen Wert von 5 angegeben, wobei sich bei Game 2 die Mehrheit eher bei 4 ansiedelt. Die Bestätigung gibt die Aussage im Abschnitt vier des Fragebogens, Game 1 sei besser als Game 2: Die Hälfte der Teilnehmer stimmten dem zu, während hingegen 33 Prozent dagegen stimmten und der Rest unentschlossen war und sich für einen Mittelwert entschied.

Abbildung 24: Vergleich der Sympathie von Game 1 und Game 2

Die Kippbewegung im zweiten und vierten Level von Game 1 hat nach den Angaben bei 83 Prozent gut bis sehr gut gepasst. Nur zwei Leute kamen nicht so gut damit zurecht und werteten die Steuerung mit 2. Bei Game 2 passte die Steuerung etwas weniger mit 78 Prozent mit einer fast ausgeglichenen Anzahl von einer Wertung bei 4 und 5 (vgl. Abbildung 20).

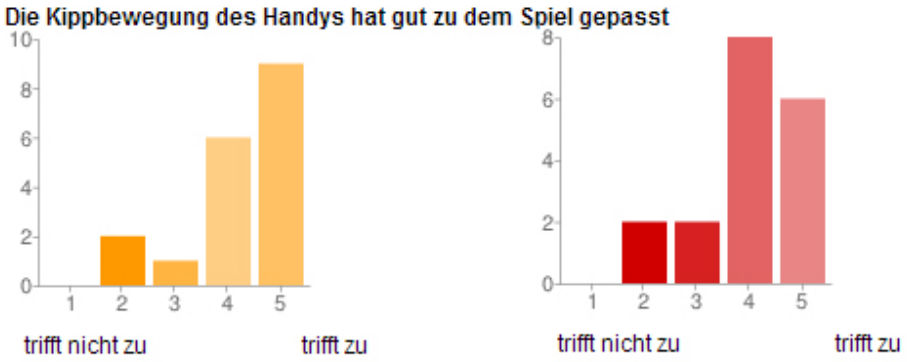

Abbildung 25: Kippbewegung in Relation zum Spiel bei Game 1 (links) und Game 2

Die Kontrolle über den Pinguin behielten in Game 1 44 Prozent der Teilnehmer, die eine Wertung von 5 angaben, wobei 3 Leute Schwierigkeiten hatten und einen Wert von 2 und 3 wählten. Bei Game 2 hatten deutlich mehr Personen die Kontrolle über den Pinguin, insgesamt zwei Drittel der Tester. Zwei der übrigen Testpersonen hatten wesentliche Schwierigkeiten und werteten mit jeweils 1 und 2, während der Rest teils-teils die Kontrolle behielten.

Die Aussage, dass man oft ein Level wieder von vorne beginnen musste, wurde sehr unterschiedlich gepunktet. Hier waren bei Game 1 alle Abstufungen mit 2 bis 5 Personen besetzt. Die meisten waren bei 1 und 4 vertreten, mit einer Person weniger bei der 3. Die Werte 2 und 5 waren am wenigsten vertreten. Bei Game 2 war es schon etwas deutlicher: ein Drittel der Teilnehmer mussten nicht oft ein Level wiederholen, das zweite Drittel musste teils-teils ein Level neu starten und nur eine Person musste oft von Neuem beginnen.

Die Sicht auf das Display wurde bei Game 1 zu gleichen Teilen von 39 Prozent gar nicht und eher doch eingeschränkt (vgl. Abbildung 21), während bei einer Person gänzlich die Sicht erschwert wurde. 3 Personen hatten nur leichte bis mittlere Einschränkungen. Bei Game 2 hatten wieder 39 Prozent keine Beeinträchtigungen, jedoch verteilen sich die Wertungen zwischen 3 und 5, die eine Sichtfeldblockade feststellen konnten.

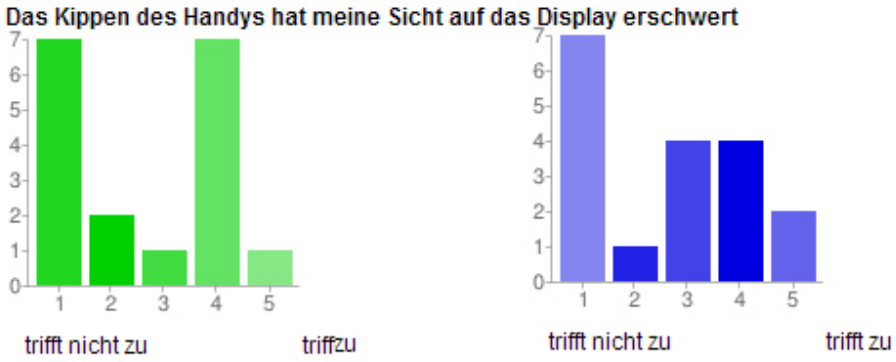

Abbildung 26: Sichteinschränkung bei Game 1 (links) und Game 2

Spaß hat das Spiel Game 1 so gut wie allen gemacht. Die Werte liegen hier nur bei 4 und 5. Bei Game 2 legen sich deutlich weniger auf die Bestwertung fest. Die klare Tendenz liegt hier mit 67 Prozent bei 4, nur einer wählt die 3, während der Rest auf der 5 liegt.

Die Frage nach dem leichtesten Level waren bei Game 1 die beiden Level mit Sensorensteuerung, bei Game 2 war es mit Abstand der Luftballon-Level, und auf Platz 2 das erste Level. Die Fische waren bei Game 1 mittelmäßig schwierig einzusammeln, während hingegen bei Game 2 die Hälfte keinerlei Probleme hatten.

Die Sprünge waren für die Hälfte der Testpersonen teils-teils schwierig zu Handhaben. Nur drei Personen fanden die Sprünge bei Game 1 leicht bis sehr leicht. Anders als bei Game 2: Hier fanden 78 Prozent der Tester die Sprünge leicht bis sehr leicht. Nur ein Tester lag bei einem Wert von 2.

Die Tabelle 1 zeigt eine Zusammenfassung der Dinge, die die Testpersonen bei Game 1 gut oder schlecht fanden, sowie einige Verbesserungsvorschläge, die sie in einem Fließtext auf dem Fragebogen eintragen konnten. So ist zum Beispiel positiv, dass das Genre Jump'n Run auch durch die Sensorensteuerung erhalten bleibt und der Wechsel zwischen der Sensorensteuerung und der normalen Touch-Bedienung gut passt. Das Spiel sei herausfordernd und auch das Leveldesign überzeuge. Weniger gut fanden die Testpersonen auftretende Kompatibilitätsprobleme bei den Online-Testern und die Sprünge im Spiel selbst, die etwas zu träge und nicht erwartungskonform wirken. Weiter könnten die Level länger sein und speziell im vierten Level hindernisreicher sein. Die Verbesserungsvorschläge basieren meist auf den negativen Aspekten des Spiels, die hier weiter ausgebaut, beziehungsweise verbessert werden sollen.

Gut	Schlecht	Verbesserungsvorschläge
Wechsel zwischen Kippen und Touch	Sprungbewegung träge	Sprünge verbessern
klassisches Jump'n Run-Feeling	Bombenradius zu hoch	Bewegungsneigung beliebig je nach Sitzposition
abwechslungsreiches Gameplay	Kompatibilitätsprobleme	Musik, Sound, Highscore einbinden
Kippsteuerung macht Spaß - direktere Steuerungsmöglichkeiten	Nicht alle Fische sind erreichbar (Frustration)	Begrenztes Leben
Erfordert Timing bei Sprüngen	Bewegungsneigung basiert auf waagerechter Haltung	
Herausfordernd – nicht zu schwer und nicht zu einfach	Zu kurze und wenige Level	
Motiviert zum Einsammeln der Fische	Mehr Herausforderungen	
Unterschiedliche Level	Doublejump fehlt	
Schnelles Erlernen der Steuerung sorgt für Erfolgserlebnis	Letztes Level zu leicht (keine Hindernisse)	

Tabelle 1: Zusammenfassung der Game 1-Kritiken und Verbesserungsvorschläge

Tabelle 2 zeigt die positiven und negativen Punkte, die während des Spielens von PinguTime hervorgingen. Positiv wäre hier, dass die Geschwindigkeit nun mittels der horizontalen Neigung des Smartphones steuerbar sei. Dies würde zu einer noch genaueren Steuerung des Spielers führen und man könne so Hindernisse leichter umgehen. Zudem könne man im Gegensatz zu Game 1 rückwärts laufen und so Fehler korrigieren. Das mindere den Schwierigkeitsgrad und fühle sich ungewohnt und eher unnatürlich an. Verbesserungen des Spiels wären dann die Optimierung der Sprünge, sowie ein Einbau eines Timers, der den Schwierigkeitsgrad somit erhöhen soll.

Gut	Schlecht	Verbesserungsvorschläge
Geschwindigkeit anpassbar	Steuerung schwieriger	Kürzere Sprünge
Leichtes Umgehen der Fallen	Man kann aus dem Level fliegen	Sprünge durch vertikales Kippen
Exakte Steuerung der Spielfigur möglich	Sprünge können mittels Sensorensteuerung abgebremst oder beschleunigt werden, dadurch leichteres Gameplay	Steife Sprünge ohne abbremsen
Rückwärts laufen möglich	Fühlt sich unnatürlich und ungewohnt an	Schwierigkeitsgrad erhöhen
Sprünge leichter kontrollierbar	Zu einfach	Einbau eines Timers
Leicht für Anfänger	Zu einseitige Hindernisse	
Alle Fische sind nun erreichbar		
Sensorensteuerung machen das erste und dritte Level spannender		

Tabelle 2: Zusammenfassung der Game 2-Kritiken und Verbesserungsvorschläge

7 Ergebnis und Bewertung

Zunächst lässt sich sagen, dass die Testgruppe hinsichtlich des Alters ein breites Spektrum bietet, da sowohl junge als auch ältere Personen vertreten sind. Auch die Verteilung der Geschlechter ist mit 61 zu 39 Prozent zufriedenstellend. Bei der ersten Frage ist auffallend, dass 72 Prozent der Testpersonen sehr häufig ihr Smartphone nutzen. Die Testpersonen sind somit mit dem Umgang eines Smartphones bestens vertraut und man kann daraus schließen, dass sie weniger Probleme mit neuen Steuerungsmechanismen haben werden. Ein Vorteil, der sich bei der Evaluation bemerkbar machen könnte, oder aber ein Nachteil, da sie schon einiges getestet oder bereits Sensorenfunktionen benutzt haben und eine gewisse Erwartungshaltung annehmen. Trotz des häufigen Gebrauchs spielen die Testpersonen eher selten Spiele auf ihrem Smartphone. Somit ist es nicht überraschend, dass sie auch laut Fragebogen weniger Jump'n Run Spiele spielen. Die Frage ist nun, wie sie das Jump'n Run Spiel mit der Sensorensteuerung annehmen und wie sie damit zurecht kommen. Immerhin haben schon 80 Prozent der Personen Erfahrungen bei Spielen mit Sensorensteuerung gesammelt. Die Ergebnisse zeigen, dass beide Versionen für gut empfunden wurden. Im direkten Vergleich lassen sich Schlüsse ziehen, welche Steuerung zu dem Spiel besser passt. Die Variante Game 1 schnitt in der Einzelwertung besser als Game 2 ab. Es herrschte hier eine Umverteilung von der Wertung 5 auf die 4. Im direkten Vergleich wird es sogar noch deutlicher: 50 Prozent der Befragten fanden Game 1 besser als Game 2. Nur

ein Drittel fand Game 2 besser. Der wesentliche Unterschied der beiden Spiele war der Zeitpunkt des Verwendens der Sensorsteuerung. Während Game 1 es dem Spieler nur in zwei Level ermöglicht hat sich frei und ungestört durch die Luft und Unterwasser im Raum zu bewegen, konnte man bei Game 2 immer die Spielfigur mittels Sensoren steuern. In Game 1 lief der Pinguin von alleine nach vorne, so dass die Konzentration mehr auf der Richtungsänderung über das Kippen lag.

Auffallend ist aber, dass die Steuerung von Game 2 mehr Spaß gemacht hat und die Steuerung laut den Testpersonen viel exakter war, trotz der Tatsache, dass Game 1 besser ankam. Ein Grund dafür ist das Zusammenspiel der Level, der Story und der Steuerung von Game 1. Läuft man steif durch das erste Level bis zum Ende und taucht dann in das Wasser ein, wechselt die Steuerung und die Bewegungen werden über den Beschleunigungssensor geschmeidiger. Genauso wenn der Pinguin an Land den Luftballon einsammelt und sich dann in der Luft befindet. Man spürt schon den Unterschied und erkennt, dass der Pinguin sich Unterwasser oder in der Luft wohler fühlt. Die Story und die Realisierung dieser über die Level scheinen mehr Anklang zu finden als die direkte Steuerung der Spielfigur. Vielleicht ist es auch die Abwechslung selbst, die das Spiel interessanter macht. In Game 2 ist es zwar möglich, alle Fische zu erreichen und jedem Hindernis gekonnt auszuweichen, jedoch mindert es auch den Schwierigkeitsgrad des Spiels, auch wenn die Steuerung spaßiger scheint. Die Sprünge können durch diese Steuerung besser koordiniert werden, jedoch lassen sie auch Bewegungen nach hinten zu oder katapultieren den Spieler nach vorne um gleich mehrere Hindernisse zu umgehen. Dies macht das Spiel zu einfach, zudem man auch wieder zurück bewegen kann, falls man einen Fisch nicht eingesammelt hatte. Abbildung 22 zeigt, welche Level den Probanden jeweils am leichtesten fielen. Wie bereits erwähnt fanden die Testpersonen bei Game 1 die Level mit der Sensorensteuerung leichter; am Leichtesten natürlich in beiden Fällen der Luftballon Level, da er keinerlei Hindernisse besitzt. Überraschend ist nun, dass bei Game 2 das Unterwasser Level nicht mehr an zweiter Stelle so wie bei Game 1 steht. Das bestätigt die Annahme, dass die Kippsteuerung und die Sprünge zusammen das Level zu einfach machen.

Abbildung 27: Vergleich Game 1 und Game 2: Welches Level ist am leichtesten

8 Fazit

Die meisten Kritikpunkte basieren auf Bugs in dem Spiel, wie zum Beispiel der Sprung des Pinguins. Würde man eine konkrete Sprungkurve einbinden, anstatt nur die Höhe zu verändern, hätte man das Problem der weiten Sprünge bei Game 2 nicht. Auch kann man noch mit der Steuerung aus dem Level fliegen, da die Begrenzungen nach hinten fehlen. Diese Probleme lassen sich relativ schnell beheben. Erhöht man auch den Schwierigkeitsgrad der Level durch mehr Hindernisse und Gegner, hat das Spiel Game 2 mit der Sensorensteuerung durchaus Potenzial. Wie aus den Verbesserungsvorschlägen der Probanden aus Tabelle 1 und 2 entnommen könnte noch ein Highscore und eine Lebensbegrenzung eingebaut werden, um die Suchtgefahr des Spiel zu erhöhen. Auch Game 1 hätte eine Chance, wenn man die Sprünge weiter optimieren würde. Dann müsste erneut getestet werden, welches Spiel nun wirklich das bessere ist.

9 Literaturverzeichnis

[CM85] Csikszentmihalyi, M. (1985): Das Flow-Erlebnis: Jenseits von Angst und Langeweile: im Tun aufgehen, Stuttgart: Klett-Cotta

[RVE03] Rheinberg, F. , Vollmeyer R. (2003). Die Erfassung des Flow-Erlebens.
Online-URL: http://www.psych.uni-potsdam.de/people/rheinberg/messverfahren/Flow-FKS.pdf
Aufgerufen am: 29.06.2014

[nss] Neuer Suchtfaktor Smartphone.
Online-URL: http://www.n-tv.de/wissen/App-soll-Alarm-schlagen-article12296266.html
Abgerufen am: 12.05.2014

[mzcg] Infografik Casual Gaming in Deutschland.
Online-URL: http://www.mobilezeitgeist.com/2013/06/03/infografik-casual-gaming-in-deutschland/
Abgerufen am: 12.05.2014

[cg] Casual Game.
Online-URL: http://de.wikipedia.org/wiki/Casual_Game
Abgerufen am: 18.05.2014

[msh] Mobile Shopping heute - der Omnikonsument.
Online-URL: http://www.mobile-zeitgeist.com/2014/05/15/mobile-shopping-heute-der-omnikonsument/
Abgerufen am: 20.05.2014

[ae2D] AndEngine Android 2D OpenGL Game Engine.
Online-URL: http://www.andengine.org/blog/
Abgerufen am: 11.04.2014

[VAEPr12] Henrik Voß: AndEngine Präsentation – Eine 2D-Spiel-Engine für Android, GDG Bremen, 2012.
Online-URL: http://bremengtug.googlecode.com/svn/presentations/2012_02_06/And%20Engine/AndEnginePr%C3%A4sentation.pdf
Abgerufen am: 30.05.2014

[jnr] Jump 'n' Run.
Online-URL: http://de.wikipedia.org/wiki/Jump_%E2%80%99n%E2%80%99_Run
Abgerufen am: 26.05.2014

[dgdj] Die Geschichte des Jump-n-Runs.
Online-URL: http://www.4players.de/4players.php/screenshot_list/Spielkultur/9161/Screenshots/45368/0/Die_Geschichte_der_Jump-n-Runs.html
Abgerufen am: 26.05.2014

[bms] So erkennt ein Smartphone jede Bewegung.
Online-URL: http://www.connect.de/ratgeber/bewegungsmessung-von-smartphones-1169755.html
Abgerufen am: 22.05.2014

[sen] Sensor.
Online-URL: http://www.itwissen.info/definition/lexikon/Sensor-sensor.html
Abgerufen am: 22.05.2014

[zss14] Mächtige Sensoren.
Online-URL: http://www.zeit.de/digital/mobil/2014-05/smartphone-sensoren-iphone-samsung
Abgerufen am: 22.05.2014

[pwc] Corioliskraft.
Online-URL: http://www.physik.wissenstexte.de/coriolis.htm
Abgerufen am: 22.05.2014

[comSD] Stimmgabelprinzip Drehratensensor.
Online-URL: http://commons.wikimedia.org/wiki/ File:Stimmgabel prinzip_Drehratensensor.png
Abgerufen am: 22.05.2014

[zss14-2] Mächtige Sensoren.
Online-URL: http://www.zeit.de/digital/mobil/2014-05/smartphone-sensoren-iphone-samsung/seite-2
Abgerufen am: 22.05.2014

[nw Gs4] Galaxy S4: Samsung erklärt die Sensoren des Superphones.
Online-URL: http://www.netzwelt.de/news/95620-galaxy-s4-samsung-erklaert-sensoren-superphones.html
Abgerufen am: 22.05.2014

[enSen] Feuchte- und Temperaturmessung in Samsung Galaxy S4.
Online-URL: http://www.elektroniknet.de/automation/sensorik/artikel/97936/
Abgerufen am 22.05.2014

[and] Android.
Online-URL: http://www.gruenderszene.de/lexikon/begriffe/android
Abgerufen am: 30.05.2014

[andVN] Android-Betriebsystem: Vor- und Nachteile.
Online-URL: http://www.tonline.de/ratgeber/technik /handy/id_46530318/ android-betriebssystem-vor-und-nachteile.html
Abgerufen am: 29.05.2014

[spr] Sprites.
Online-URL: http://www.matim-dev.com/sprites.html
Abgerufen am: 20.04.2014

[senso] Sensors Overview.
Online-URL: http://developer.android.com/guide/topics/sensors/sensors_overview.html
Abgerufen am: 28.05.2014

[scen] Scene.
Online-URL: https://developer.android.com/reference/android/transition/Scene.html
Abgerufen am: 20.04.2014

[pht] PhysicsTutorial.
Online-URL: http://www.andengine.org/forums/physics-box2dextension/physics-tutorial-for-those-having-trouble-t8816.html
Abgerufen am: 27.04.2014

[andsdk] Android SDK.
Online -URL: http://developer.android.com/sdk/index.html
Abgerufen am: 15.03.2014

[Gforms] Google Forms.
https://docs.google.com/forms/create
Abgerufen am: 15.06.2014

[fgt11] Loading Level from XML.
Online-URL: http://www.matim-dev.com/full-game-tutorial---part-11.html
Abgerufen am: 26.05.2014

[act] Activity.
Online-URL: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle
Abgerufen am: 12.05.2014

[fgt2] creating Activity.
Online-URL: http://www.matim-dev.com/full-game-tutorial---part-2.html
Abgerufen am: 2.03.2014

[fgt3] ResourcesManager.
Online-URL: http://www.matim-dev.com/full-game-tutorial---part-3.html
Abgerufen am: 4.03.2014

[fgt4] Scene management.
Online-URL: http://www.matim-dev.com/full-game-tutorial---part-4.html
Abgerufen am: 4.03.2014

[fgt10] Creating loading scene.
Online-URL: http://www.matim-dev.com/full-game-tutorial---part-10.html
Abgerufen am: 5.03.2014

[pm] Player movement.
Online-URL: http://perle-development.com/tutorials/andengine-tutorial-04-player-movement-02/
Abgerufen am: 5.05.2014

[flowk] Komponenten des Flow-Erlebens.
Online-URL: http://www.flow-usability.de/flowkomponenten.htm
Abgerufen am: 20.05.2014

[flow] Übersicht über den Flow-Begriff.
Online-URL: http://www.flow-usability.de/flow.htm
Abgerufen am: 20.05.2014

[msu] Mobile Usability Tests.
Online-URL: http://www.usabilityblog.de/2010/08/mobile-usability-tests-was-sollte-man-beachten/
Aufgerufen am: 3.06.2014

10 Anhang

Nutzertest zu PinguTime:

Worum geht es?

PinguTime ist ein Jump'n Run Spiel, bei dem ein Pinguin Fische einsammeln muss. Hierfür müssen diese mittels Sprüngen eingesammelt und Hindernissen wie Kisten, Gruben, Stacheln oder Bomben ausgewichen werden. Das Spiel besteht aus vier Leveln. Danach kehrt das Spiel wieder zurück ins Hauptmenü. Wenn du stirbst, fängt das Level wieder von vorne an.

Es gibt 2 verschiedene Versionen von PinguTime. Im Hauptmenü kannst du zwischen diesen Versionen wählen, indem du auf **Game 1** oder **Game 2** klickst.

Steuerung:

Game 1

Level 1 und 3:

Das Spiel startet bei **jedem** Level erst nach **einmaligem** Klicken auf das Display. Der Pinguin läuft dann selbstständig nach vorne. Durch erneutes Klicken springt der Pinguin nach oben. Einen „Doppel-jump“ gibt es nicht. Du musst also genau überlegen, wann du den Pinguin springen lässt.

Level 2 und 4:

Halte das Smartphone so flach vor dir, als würde es auf einem Tisch vor dir liegen. Du musst also von oben herab das Spiel betrachten. Durch **Kippen** des Handys nach vorne oder hinten steuerst du den Pinguin nach oben oder unten. So kannst du ihn geschmeidig Unterwasser oder in der Luft Fische einsammeln lassen.

Game 2

Tippe einmal auf den Bildschirm, damit das Spiel startet. Du kannst jetzt über **Kippen** des Smartphones nach links oder rechts den Pinguin nach vorne/rückwärts bewegen. Der Sprung erfolgt genauso wie bei **Game 1** durch Berühren des Displays. Je steiler du das Handy hälst, desto schneller bewegt sich der Pinguin.

Anmerkung:

Sei vorsichtig bei den Unterwasserbomben, sie haben auch wenn sie rund sind scharfe Kanten. Komme ihnen lieber nicht zu nahe!

Spiele nun Game1 und danach Game 2 durch. Anschließend füllen Sie bitte den beiliegenden Fragebogen aus! Viel Spaß!

Alter: _____

Geschlecht : ☐ weiblich ☐ männlich

Bitte kreuzen Sie zu jeder Aussage an, was am besten auf Sie zutrifft:

	nie		gelegentlich		Sehr oft	Weiß ich nicht
1. Wie oft nutzen Sie ihr Smartphone?	☐	☐	☐	☐	☐	☐
2. Wie oft spielen Sie Spiele auf ihrem Handy ?	☐	☐	☐	☐	☐	☐
3. Wie oft spielen Sie Jump'n'Run-Spiele?	☐	☐	☐	☐	☐	☐

	Nicht so sehr		Geht so		Sehr gerne	Weiß ich nicht
4. Wie sehr mögen Sie Jump'n'Run-Spiele ?	☐	☐	☐	☐	☐	☐

	Ja	Nein	Weiß ich nicht
5. Ich habe schon mal Spiele mit Sensorensteuerung(Kippen) gespielt.	☐	☐	☐

Fragen zu Game 1

	Trifft nicht zu		Teils-teils		Trifft zu	Weiß ich nicht
1. Mir hat das Spiel **Game 1** sehr gut gefallen	☐	☐	☐	☐	☐	☐
2. Die Kippbewegung des Handys hat gut zu dem Spiel gepasst	☐	☐	☐	☐	☐	☐
3. Ich hatte stets die Kontrolle über den Pinguin	☐	☐	☐	☐	☐	☐
4. Ich musste oft ein Level von vorne beginnen	☐	☐	☐	☐	☐	☐
5. Das Kippen des Handys hat meine Sicht auf das Display erschwert	☐	☐	☐	☐	☐	☐
6. Das Spiel macht Spaß	☐	☐	☐	☐	☐	☐
7. Ich würde das Spiel auch in meiner Freizeit spielen	☐	☐	☐	☐	☐	☐
8. Ich würde auch Geld für das Spiel bezahlen	☐	☐	☐	☐	☐	☐
9. Das Spiel hat Lust auf mehr gemacht	☐	☐	☐	☐	☐	☐

	Level 1 Dschungel	**Level 2** Unterwasser	**Level 3** Eis	**Level 4** Luftballon	Alle gleich
Welches Level fiel Ihnen am leichtesten?	☐	☐	☐	☐	☐

	Sehr schwer		mittelmäßig		Sehr leicht	Weiß ich nicht
1. Wie schwer fanden Sie die Level?	☐	☐	☐	☐	☐	☐
2. Wie schwer war es die Fische einzusammeln ?	☐	☐	☐	☐	☐	☐
3. Wie schwer fielen Ihnen die Sprünge?	☐	☐	☐	☐	☐	☐

Was fanden Sie an **Game 1** gut?

Was fanden Sie an **Game 1** nicht gut?

Was könnte man noch am Spiel an **Game 1** verbessern ?

Fragen zu Game 2

	Trifft nicht zu		Teils-teils		Trifft zu	Weiß ich nicht
1. Mir hat das Spiel **Game 2** sehr gut gefallen	☐	☐	☐	☐	☐	☐
2. Die Kippbewegung des Handys hat gut zu dem Spiel gepasst	☐	☐	☐	☐	☐	☐
3. Ich hatte stets die Kontrolle über den Pinguin	☐	☐	☐	☐	☐	☐
4. Ich musste oft ein Level von vorne beginnen	☐	☐	☐	☐	☐	☐
5. Das Kippen des Handys hat meine Sicht auf das Display erschwert	☐	☐	☐	☐	☐	☐
6. Das Spiel macht Spaß	☐	☐	☐	☐	☐	☐
7. Ich würde das Spiel auch in meiner Freizeit spielen	☐	☐	☐	☐	☐	☐
8. Ich würde auch Geld für das Spiel bezahlen	☐	☐	☐	☐	☐	☐
9. Das Spiel hat Lust auf mehr gemacht	☐	☐	☐	☐	☐	☐

	Level 1 Dschungel	**Level 2** Unterwasser	**Level 3** Eis	**Level 4** Luftballon	Alle gleich
Welches Level fiel Ihnen am leichtesten?	☐	☐	☐	☐	☐

	Sehr schwer		mittelmäßig		Sehr leicht	Weiß ich nicht
1. Wie schwer fanden Sie die Level?	☐	☐	☐	☐	☐	☐
2. Wie schwer war es die Fische einzusammeln ?	☐	☐	☐	☐	☐	☐
3. Wie schwer fielen Ihnen die Sprünge?	☐	☐	☐	☐	☐	☐

Was fanden Sie an **Game 2** gut?

Was fanden Sie an **Game 2** nicht gut?

Was könnte man noch am Spiel an **Game 2** verbessern ?

Vergleich Game 1 und Game 2

	Trifft nicht zu		Teils-teils		Trifft zu	Weiß ich nicht
1. Mir hat das Spiel **Game 1** besser gefallen	☐	☐	☐	☐	☐	☐
2. Mir hat das Spiel **Game 2** besser gefallen	☐	☐	☐	☐	☐	☐
3. Kippsteuerung bei Jump'n Run Spielen hat mir generell nicht gefallen	☐	☐	☐	☐	☐	☐
4. Die Steuerung von **Game 2** war viel besser	☐	☐	☐	☐	☐	☐
5. Die Sicht auf das Spiel wurde bei **Game 2** mehr behindert	☐	☐	☐	☐	☐	☐
6. Das Spiel **Game 2** hat mehr Spaß gemacht	☐	☐	☐	☐	☐	☐
7. Die Kippsteuerung hat in beiden Fällen nicht zu dem Spiel gepasst	☐	☐	☐	☐	☐	☐
9. Der Wechsel der Steuerung in **Game 1** hat sehr gut gepasst	☐	☐	☐	☐	☐	☐

Ich fand Game ☐ 1 ☐ 2 besser, weil...

11 Screenshots