Bitte deutlich leserlich ausfüllen!

Deckblatt einer

w i s s e n s c h a f t l i c h e n Bachelorarbeit

Vor- und Familienname	Matrikelnummer
Johannes Wolfgruber	01331048
Studienrichtung	Studienkennzahl
Elektrotechnik-Toningenieur	V 033 213

Thema der Arbeit:

Learn to Listen - Die Entwicklung eines Audio Games

Angefertigt in der Lehrveranstaltung: Computermusik und Multimedia 02
(Name der Lehrveranstaltung)

Vorgelegt am: ……
(Datum)

Beurteilt durch: ……..
(Leiter/-in der Lehrveranstaltung)

Learn to Listen
Die Entwicklung eines Audio Games

Bachelorarbeit

von

Johannes Wolfgruber

im Rahmen des Seminars

Computermusik und Multimedia 02

Betreuer

Dipl.-Ing. Johannes Zmölnig

Graz, den 19. September 2017

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Thema des Seminars „Computermusik und Multimedia 02“ im Sommersemester 2017 lautete *Audio Games*. Im Rahmen des Seminars sollte ein Konzept, sowie eine Umsetzung als „Proof of Concept“ erarbeitet werden. Die Wahl des Mediums sowie die verwendete Programmiersprache war dabei den Seminarteilnehmern selbst überlassen.

Zunächst sollte eine Hintergrundrecherche zu vorhandenen *Audio Games* durchgeführt werden. Die Teilnehmer konnten so einen Überblick und erste Inspirationen für die eigenen Spiele erhalten. Danach sollten eigene Konzepte erarbeitet und präsentiert werden. Durch eine Einführung in die Grundlagen des Game Designs erhielten die Seminarteilnehmer Anhaltspunkte und wichtige Informationen. Erst danach wurde ein Plan für die Implementierung erstellt und dann auch umgesetzt.

Der so entstandene Prototyp des Spiels - *Learn to Listen* - ist eine Mischung aus Jump’n’Run - Game und Lernspiel. Ziel des Spiels ist es innerhalb der vorgegebenen Zeit durch das Labyrinth ins Ziel zu kommen und dabei soviele Punkte wie möglich zu erzielen. Die Punkte werden mithilfe der Gehörbildungsaufgaben und Aufgaben des Seashoretests, welche über das Spielfeld verteilt sind, gesammelt. Die Aufgaben können gelöst werden, können jedoch auch übersprungen werden, um Zeit zu sparen. Zudem gibt es verschlossene Türen, die geöffnet werden können, indem die drei zufällig ausgewählten Aufgaben richtig gelöst werden. Hierdurch erhält der Spieler Bonuspunkte und eine Abkürzung zum Ziel.

In dieser Arbeit wird zunächst der Begriff *Audio Game* näher erläutert. Danach wird allgemein der Prozess des Game Designs erklärt. Über eine kurze Einführung in das verwendete Medium und die Programmiersprache wird schließlich die Vorgehensweise bei der Implementierung des Spiels dargelegt. Hierbei wird auf das Design der Benutzeroberfläche, die eigentliche Programmierung des Spiels, sowie das Sounddesign eingegangen.

Im Folgenden wird aufgrund der leichteren Leseart ausschließlich die männliche Form verwendet.

2 Audio Games

2.1 Definition

Unter *Audio Games* versteht man ganz allgemein Computerspiele, die anstatt über visuelles Feedback über auditives Feedback verfügen. Die Zielgruppen waren zu Beginn hauptsächlich blinde und visuell eingeschränkte Personen. Es ist wichtig, dass die *Audio Games* über ein auditives Interface verfügen, welches das Spielen komplett ohne grafischer Oberfläche ermöglicht [17, 21].

Audio Games wurden zunächst vorwiegend von Amateuren und blinden Programmierern entwickelt. Mit der Zeit zeigten jedoch immer mehr Menschen Interesse, so zum Beispiel Mobile Game Entwickler oder Mainstream Gamer. Dies zeigt, dass obwohl *Audio Games* hauptsächlich für blinde Menschen entwickelt wurden und werden, es trotzdem möglich ist eine soundbasierte Unterhaltung für jeden zu schaffen. Zusätzlich ergeben sich neue, interessante Wege für das Gameplay. Einer dieser Vorteile ist die räumliche Unabhängigkeit (Einsatz von 3D-Audio ermöglicht eine 360° - Interaktion) im Vergleich zu gewöhnlichen Videospielen, bei denen der Spieler immer an den Bildschirm gebunden ist. Hinzu kommt, dass die benötigte Rechenleistung bei *Audio Games* meist wesentlich geringer als bei Videospielen ist und somit mobile Geräte ein prädestiniertes Medium darstellen [17, 19, 21].

2.2 Geschichte der Audio Games

Das erste Audio Game erschien 1974 von Atari und nannte sich „Touch Me“. Hierbei handelte es sich nicht um ein Computerspiel, sondern ein eigenständiges Gerät in einem Arcade Gehäuse [21].

Abbildung 1: Handheld Version von Touch Me, 1978. [21]

Das Spiel gibt über eine Reihe von Lichtern und einen Lautsprecher gleichzeitig eine tonale und visuelle Sequenz aus, die der Spieler dann reproduzieren muss. Durch die auditive und visuelle Komponente war das Spiel gleichermaßen für Sehende und Blinde Menschen geeignet [15, 21].

Bevor es Betriebssysteme mit grafischer Oberfläche gab, waren textbasierte Systeme verbreitet. Diese Systeme waren für visuell eingeschränkte Menschen mit Hilfe von Text-to-Speech Software zugänglich. Dadurch wurden frühe textbasierte Computerspiele für alle erreichbar gemacht. Erst mit der Einführung von grafischen Benutzeroberflächen und damit einhergehend Videospielen, entstand eine Lücke zwischen visuell nicht eingeschränkten und blinden Spielern. Amateurentwickler begannen daraufhin, Videospiele als *Audio Games* zu adaptieren. So entstanden nach und nach Spiele, die sich immer weniger an Videospielen orientierten und dafür immer mehr eigene und neue Wege des Gameplays erschlossen [21].

Über die letzten Jahre gab es eine rasante Entwicklung im Bereich der Audio Chips und 3D - Soundengines für Computerspiele. Entwickler sowie Gamer legen immer mehr Wert auf hochwertig produzierte „Soundscapes“ in Videospielen. Trotz dieser Tendenzen spielt der Sound im Vergleich zur Grafik in kommerziellen Videospielen immer noch eine untergeordnete Rolle, eröffnet aber sehbehinderten Menschen zunehmend umfangreichere Spielemöglichkeiten [17].

3 Game Design

3.1 Definition eines Spiels

Um ein Spiel zu entwickeln, muss zunächst der Begriff des Spiels definiert werden. Im Englischen werden dazu die zwei verschiedenen Wörter „game“ und „play“ gegenübergestellt. Hierbei gibt es zwei Betrachtungsweisen, wobei die erste „play“ und die zweite „game“ als Überbegriff des jeweils anderen sieht [20].

Sieht man „game“ als Unterkategorie von „play“, werden alle Spiele welche vorgegebenen Regeln folgen, zum Beispiel „Mensch ärgere dich nicht“ oder „Völkerball“, als „game“ bezeichnet. Als „play“ wiederum wird zusätzlich jede andere Art von Spielen definiert, zum Beispiel das Spielen von Kindern auf einem Spielplatz [20].

Die zweite Betrachtungsweise ist konzeptioneller als die erste und nimmt das Spielen („play“) eines Spiels als eine der vielen Möglichkeiten ein Spiel („game“) zu erleben und zu untersuchen. So gesehen ist „play“ also eine Unterkategorie von „game“ [20].

Eine eindeutige Definition des Begriffs Spiel ist nicht einfach. Salen & Zimmerman bedienen sich hierzu bei verschiedenen Quellen und gelangen zu folgender Definition:

> „A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.“ [20]

> Ein Spiel ist ein System, in welchem sich Spieler an einem künstlichen Konflikt, welcher durch Regeln definiert ist und in einem quantifizierbaren Ergebnis resultiert, beteiligen. [1]

Der Begriff *System* beschreibt in diesem Zusammenhang im weitesten Sinne ein Gruppe von Elementen, welche interagieren, untereinander verwandt und voneinander abhängig sind. Ein oder mehrere *Spieler* interagieren hierbei aktiv mit diesem System. *Künstlich* bedeutet eine Abgrenzung des Spiels vom *realen Leben*, sowohl zeitlich als auch räumlich. Alle Spiele beinhalten hierbei einen *Konflikt*, sei es zwischen dem Einzelnen und dem Spiel oder zwischen mehreren

[1] Übersetzung: Johannes Wolfgruber

Spielern. Der *Konflikt* ist ein zentraler Bestandteil von Spielen. Wichtig hierbei sind die *Regeln*, welche eine Struktur vorgeben und definieren, was erlaubt ist und was nicht. Außerdem besitzt ein Spiel immer ein *quantifizierbares Ergebnis*, sei es ein Gewinner oder ein erreichter Punktestand am Ende eines Spiels. Dies unterscheidet das „game“ vom „play“ und gibt ein festes Ende vor [20].

3.2 Definition eines digitalen Spiels

Wenn man ein Spiel als ein System betrachtet, spielt das physische Medium des Spiels eine große Rolle, stellt jedoch nicht das gesamte System dar. Die digitale Technologie sollte hierbei als ein Teilelement des größeren und komplexeren Systems betrachtet werden [20].

Es gibt vier Haupteigenschaften, die digitale von analogen Spielen unterscheiden. Diese können auch in analogen Spielen auftreten, in digitalen Spielen sind sie jedoch stärker verwurzelt:

- **Es gibt eine unmittelbare aber eingeschränkte Interaktivität**
  Einer der interessantesten Punkte an Computerspielen ist das unmittelbare, interaktive Feedback, welches die digitale Plattform liefert. Das Spiel kann hierbei in Echtzeit und dynamisch auf die Eingaben des Spielers reagieren. Trotzdem sind der Interaktivität Grenzen gesetzt, welche sich aus den jeweils vorhandenen Ein- und Ausgabegeräten ergeben. Bei üblichen Computern wären dies Maus und Tastatur als Eingabegeräte sowie Bildschirm und Lautsprecher als Ausgabegeräte.

- **Es findet eine Manipulation von Informationen statt**
  Digitale Medien können als Maschinen betrachtet werden, welche Informationen speichern und manipulieren. Computerspiele nutzen dabei viele Arten von Daten, wie zum Beispiel Text, Bilder, Video, Audio, 3D - Inhalte und andere Arten von Informationen. Diese Vielzahl verschiedenster Daten wird von digitalen Spielen genutzt und verändert. Das funktioniert bei analogen Spielen nicht auf diese Art und Weise. Um ein typisches Brettspiel zu spielen, muss mindestens einer der Spieler zuvor die Regeln gelesen und verstanden haben. Bei einem Computerspiel hingegen, können die Regeln häufig während des Spiels erlernt werden. Zusätzlich verbergen digitale Spiele meist Informationen, um sie anschließend gezielt preiszugeben. Die-

ses Prinzip wird in vielen Strategiespielen angewandt und nennt sich „Fog of War“. Hierbei ist ein Teil des Spielfeldes zunächst verdeckt und muss vom Spieler erkundet werden.

- **Es gibt komplexe, automatisierte Systeme**
  Ein Computer kann Aufgaben automatisieren. Im Gegenzug zu nicht digitalen Spielen, bei denen jeder Schritt von den Spielern ausgeführt werden muss, werden bei Computerspielen gewisse Abläufe vom Programm ausgeführt und das Spiel vorangetrieben, ohne eine Eingabe des Spielers zu erhalten. Das kann allerdings auch zur Folge haben, dass der Spieler nicht mehr nachvollziehen kann, was gerade passiert ist.

- **Es gibt eine vernetzte Kommunikation**
  Viele moderne Computerspiele verfügen über die Option einer Kommunikation per Netzwerk, sei es per Chat oder Headset. Einen Kontext für Kommunikation bieten sowohl digitale als auch nicht digitale Spiele, wobei bei digitalen Spielen über weite Distanzen kommuniziert werden kann und so mit vielen anderen Spielern ein sozialer Raum geteilt werden kann.

Die Grundeigenschaften eines Spieles und die Herausforderungen beim Entwickeln eines Spiels bleiben jedoch unabhängig vom verwendeten Medium bestehen [20].

3.3 Richtlinien für Game Design

Nun sollen aus diesen Vor- und Nachteilen von digitalen Spielen Richtlinien zur Eintwicklung eines Computerspiels abgeleitet werden. Chris Crawford stellt hierzu eine Reihe von Punkten vor [16]:

- **Go with the grain**
  Es muss vermieden werden, den Computer für Aufgaben zu nutzen, für die er nicht ausgelegt ist. Stattdessen muss das Programm so geschrieben werden, dass die Maschine Funktionen ausführt, die gut ausgeführt werden können, um so die maximale Performance zu erhalten.

- **Don't transplant**
  Es soll vermieden werden, existierende Spiele ins Digitale zu konvertierten. Stattdessen sollen neue Spielkonzepte und Möglichkeiten entdeckt und genutzt werden.

- **Design around the I/O**
  Die größte Einschränkung eines Computers ist nicht die Leistung um komplexe Aufgaben zu berechnen, sondern die Ein- und Ausgabegeräte. Es ist daher wichtig das Game Design auf diese zuzuschneiden.

- **Keep it clean**
  Es ist wichtig, dem Spiel eine klare Struktur zu geben, um damit eine breite Masse an Spielern anzusprechen. Je mehr sich der Entwickler zu sehr und in zu vielen Details verliert, umso weniger Spieler werden erreicht.

- **Store less and process more**
  Ein Computer ist in erster Linie kein Speichermedium, sondern ein Rechner. Um die Stärke des Computers zu nutzen und den größten Freiraum im Game Design zu haben, sollten wenige Informationen gespeichert werden und im Gegenzug mehr Berechnungen stattfinden.

- **Maintain unity of design effort**
  Ein Spiel muss designed werden, der Computer jedoch programmiert. Das sind zwei verschiedene Aufgabengebiete. Beide müssen jedoch im Einklang stehen, um die bestmöglichen Entscheidungen treffen zu können.

Ein weiteres Kernkonzept im Gamedesign ist das iterative Design. Das bedeutet, dass wichtige Design - Entscheidungen durch das Spielen und Testen lassen des Spiels oder Prototypen getroffen werden. Dieser Prozess ist enorm wichtig, da es niemals möglich ist, das Spielerlebnis sowie den Spielspaß genau vorherzusagen [20].

Dieser Prozess wurde auch bei der Entwicklung von *Learn to Listen* durchlaufen. Zunächst stand eine Grundidee fest, welche zu einem Prototypen implementiert wurde. Danach wurde das Spiel immer wieder gespielt und dabei Aspekte verändert oder hinzugefügt. Wichtig in diesem Zusammenhang ist zudem, das Spiel von Außenstehenden spielen und testen zu lassen, um ein Feedback zu

erhalten. Außenstehende entdecken häufig Dinge, die dem Entwickler selbst nicht aufgefallen wären.

4 Learn to Listen - Das Spielkonzept

4.1 Grundidee

Das Genre von *Learn to Listen* ist eine Mischung aus einem 2D - Jump'n'Run - Game und einem Lernspiel. Der Spieler muss dabei durch eine Art mehrstöckiges Labyrinth navigieren und auf dem Weg ins Ziel verschiedene Aufgaben lösen. Das Übergeordnete Ziel des Spiels ist dabei, innerhalb der vorgegebenen Zeit das Ziel zu erreichen und durch das Lösen der Übungen so viele Punkte wie möglich zu erreichen.

4.2 Die verschiedenen Aufgaben

Bei den Aufgaben, die in den verschiedenen Leveln versteckt sind, handelt es sich um Übungen aus dem Bereich der Gehörbildung. Es gibt klassische Gehörbildungsaufgaben, sowie Aufgaben aus dem Seashoretest.

4.2.1 Gehörbildungsaufgaben

Es gibt vier klassische Typen von Gehörbildungsaufgaben mit unterschiedlichem Schwierigkeitsgrad. Diese sind den Aufgaben der Gehörbildungssoftware „Ear Master Pro“ nachempfunden [6]:

- **Intervalle unterscheiden**
  Es werden zwei benachbarte Intervalle hintereinander abgespielt (zum Beispiel eine kleine und eine große Terz). Der Spieler muss erkennen, ob das zweite Intervall größer oder kleiner als das Erste war. Bei richtiger Lösung erhält er 100 Punkte.

- **Tongeschlecht erkennen**
  Es wird entweder ein Dur-, Moll-, verminderter oder übermäßiger Dreiklang abgespielt. Der Spieler muss erkennen, um welchen Dreiklang es sich dabei handelt. Wird die Aufgabe korrekt beantwortet, erhält er 150 Punkte.

- **Umkehrungen erkennen**
  Es wird ein Dreiklang abgespielt. Dieser ist entweder in Grundstellung, in der ersten Umkehrung oder in der zweiten Umkehrung. Der Spieler

muss hören, um welche Umkehrung es sich handelt und erhält bei richtiger Lösung 200 Punkte.

- **Intervalle erkennen**
  Es werden zwei Töne hintereinander abgespielt. Der Spieler muss erkennen, welches Intervall zwischen den Tönen liegt. Hierbei sind alle Intervalle innerhalb einer Oktave möglich. Bei richtiger Antwort erhält er 500 Punkte.

4.2.2 Seashoretest

Der Seashore Test umfasst eine Reihe musikpsychologischer Aufgaben, welche die Musikalität der Probanden messen. Entwickelt wurde der Test von Carl Emil Seashore im Jahr 1919 [1]. Bei *Learn to Listen* wurden drei verschiedene Typen von Aufgaben aus dem Seashoretest verwendet:

- **Höher oder tiefer**
  Es werden zwei Töne mit $500Hz$ und $508Hz$ hintereinander abgespielt. Der Spieler muss erkennen ob der zweite Ton höher oder tiefer als der Erste war. Bei richtiger Lösung erhält er 100 Punkte

- **Lauter oder leiser**
  Es werden zwei Töne hintereinander abgespielt, wobei zwischen den Tönen eine Pegeldifferenz von $1dB$ vorliegt. Der Spieler muss hören, ob der zweite Ton lauter oder leiser war als der Erste. Wird die Aufgabe korrekt beantwortet, erhält er 100 Punkte.

- **Kürzer oder länger**
  Es werden zwei unterschiedlich lange Töne hintereinander abgespielt. Der Spieler muss benennen, ob der zweite Ton länger oder kürzer war als der Erste. Auch hierfür erhält er bei richtiger Beantwortung 100 Punkte.

4.3 Gameplay

Zur Navigation im Spielfeld gibt es Pfeilbuttons. Als Navigationshilfe dienen dabei die verschiedenen Gamesounds. Zudem können immer nur Buttons betätigt werden, wenn die gewählte Bewegung ausführbar ist. Hierdurch weiß der Spieler zum Beispiel, wenn er sich am Rand des Spielfeldes befindet. Zum Anhören

und anschließenden Beantworten der Aufgaben befindet sich in der Mitte der Bedienfläche ein Aufgabenbutton. Dieser lässt sich nur bedienen, wenn der Spieler sich auf einem Aufgabenfeld befindet, welches er zuvor noch nicht beantwortet hat.

Für jedes Level gibt es eine vorgeschriebene Zeit, innerhalb der das Ziel erreicht werden muss. Sobald der Timer nur noch 10 Sekunden anzeigt, ertönt jede Sekunde ein Warnton. Läuft die Zeit ab bevor der Spieler im Ziel ist, ist das Spiel vorbei. Erreicht er aber das Ziel bevor die Zeit abgelaufen ist, so wird die verbliebene Zeit als Bonuspunkte, jeweils mit dem Faktor 10, angerechnet.

Zusätzlich zu den Aufgabenfeldern kann der Spieler auch auf eine verschlossene Tür stoßen. Um diese zu öffnen, müssen drei zufällig ausgewählte Aufgaben richtig gelöst werden. Nach erfolgreicher Beantwortung erhält der Spieler zusätzlich 500 Punkte und die Tür öffnet sich. Dies bietet nun eine Abkürzung zum Ziel, welche genutzt werden kann, falls die Zeit knapp wird.

Neben dem normalen Modus gibt es zusätzlich einen Tutorial - Modus. Dieser bietet die Option, das Spiel ohne Zeitbegrenzung zu spielen. Allerdings kann in diesem Modus kein neuer Highscore erreicht werden.

5 Das Smartphone als Plattform

5.1 Die Wahl der Plattform

Durch die Einführung des iPhone von Apple im Jahr 2007 wurde ein Wendepunkt auf dem Smartphone Markt markiert. Es entstanden weitere mobile Betriebssysteme. Seit Herbst 2011 ist Android das am häufigsten installierte Betriebssystem auf Smartphones [12]. Im dritten Quartal 2016 konnte ein Marktanteil von 87,5 Prozent erreicht werden [3].

Es gibt eine Reihe von Gründen, Spiele für eine mobile Plattform, im speziellen für Android zu entwickeln. Android ist eine offene Plattform. Das bedeutet geringere Beschränkungen bei der Entwicklung von Apps und Spielen. Wie zuvor schon erwähnt, ist Android die am schnellsten wachsende mobile Plattform. Dies wiederum bedeutet, dass man hier die meisten potentiellen Spieler erreichen kann [18].

Seit Mai 2013 stellt Google die Entwicklungsumgebung Android Studio auf Basis von IntelliJ IDEA bereit. Diese wurde zum Programmieren von *Learn to Listen* verwendet.

5.2 Android Grundlagen

Um eine Android App zu programmieren benötigt man das Android SDK. Das Android SDK ist ein Framework, in welchem Apps und Spiele mit der Programmiersprache Java entwickelt werden können. Die Android SDK Tools kompilieren den Source Code und alle eingebundenen Dateien und Layouts zu einer `.apk` - Datei. Dieses „Android Package“ enthält alle benötigten Dateien und kann auf einem Android Gerät installiert werden [4, 7].

Eine Android App besteht aus verschiedenen sogenannten Activities. Eine Activity ist die Schnittstelle für die Interaktion mit dem Spieler. Sie repräsentiert einen Bildschirm mit grafischer Oberfläche. Eine Activity wird als Subklasse der Klasse `Activity` implementiert. Um zwischen den verschiedenen Activities und somit auch zwischen verschiedenen Benutzeroberflächen umzuschalten wird ein `Intent` benötigt. Durch ein solches Objekt wird eine andere Activity gestartet und der Benutzer kann so zwischen den verschiedenen Oberflächen umschalten. Hierbei können zusätzlich Werte aus einer Activity in eine sonst unabhängige

Activity übernommen werden [4].

Für das korrekte Funktionieren einer App ist der Lebenszyklus einer Activity von entscheidender Bedeutung:

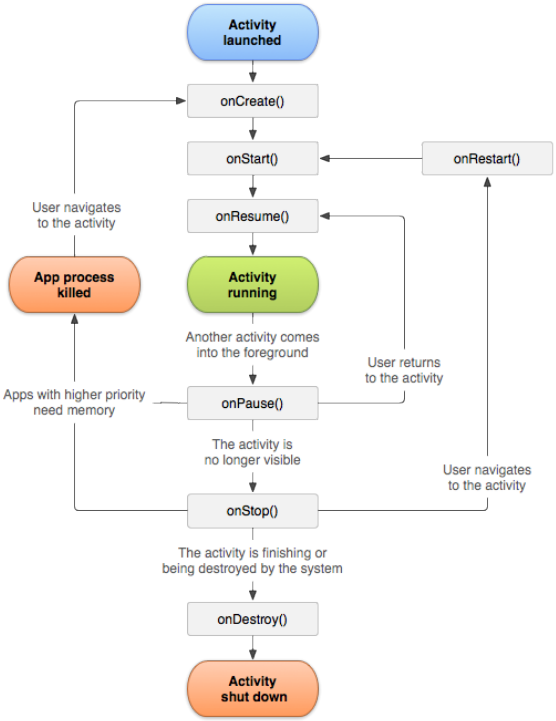

Abbildung 2: Der Lebenszyklus einer Activity, Quelle: [13]

Beim Start einer Activity wird zu allererst immer die `onCreate()` - Methode ausgeführt. Hier wird das Layout geladen und alle Bedien- und Oberflächenelemente initialisiert. Die anderen Methoden aus Abbildung 2 dienen dazu, für alle möglichen Bedarfsfälle einen Einstiegspunkt in die Activity zu gewährleisten. Wird zum Beispiel ein Dialog gezeigt, so springt die Activity in die `onPause()` - Methode und nach Beenden des Dialogs in die `onResume` - Methode. `onCreate()` kann nur einmal beim ersten Starten der Activity, beziehungsweise wenn die Activity zerstört und neu gestartet wird, aufgerufen werden [13].

Eine Android App besteht zusätzlich zum Source Code aus anderen Resourcen wie zum Beispiel Bilder, Audio-Dateien und Layouts. Diese Resourcen befinden sich im `res` - Ordner. Jede Datei in diesem Ordner erhält eine ganzzahlige ID mit der die jeweilige Resource programmatisch aufgerufen werden kann. Bestandteile wie das Layout, Animationen, Strings oder Farben werden als XML-Dateien ebenfalls in diesem Ordner gespeichert.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/root_constraint_layout_game"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBackground">

    <ImageButton
        android:id="@+id/left_button"
        android:layout_width="120dp"
        android:layout_height="100dp"
        android:layout_marginBottom="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:scaleType="fitCenter"
        app:layout_constraintBottom_toTopOf="@+id/down_button"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/up_button"
        app:srcCompat="@drawable/ic_keyboard_arrow_left_background"
        tools:ignore="ContentDescription" />
```

Abbildung 3: Ausschnitt aus der Layout Datei für die `GameActivity.class`

6 Implementierung des Spiels

6.1 Das Layout der App

Learn to Listen besteht aus drei verschiedenen Bildschirm Layouts. Nachdem der Spieler die App startet erscheint zunächst ein Splash-Screen.

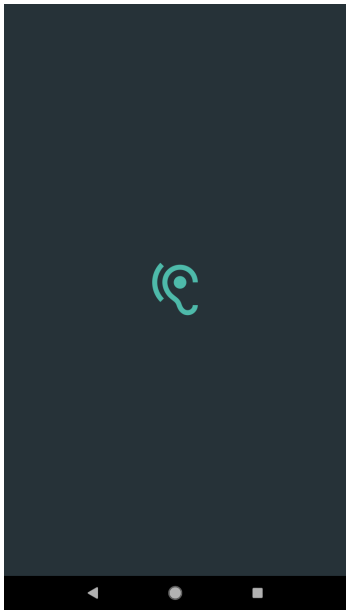

Abbildung 4: Der Splash Screen während des Ladens

Sobald alle Sounds und Views geladen wurden, befindet der Spieler sich im Hauptmenü:

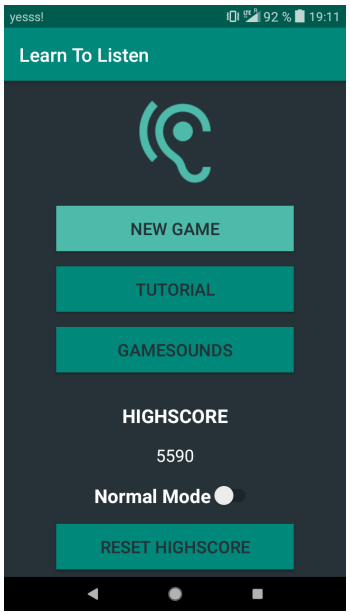

Abbildung 5: Das Hauptmenü

Das Layout in Abbildung 5 besteht aus folgenden Komponenten:

- **NEW GAME**
  Startet ein neues Spiel im gewählten Modus.
- **TUTORIAL**
  Zeigt eine Beschreibung des Spiels und der Regeln.
- **GAMESOUNDS**
  Lässt den Spieler die verschiedenen Gamesounds vorhören und kennenlernen.
- **HIGHSCORE**
  Zeigt den aktuellen Highscore an.
- **MODE SWITCH**
  Lässt den Spieler zwischen normalem Modus und Tutorial-Modus umschalten.
- **RESET HIGHSCORE**
  Setzt den Highscore auf 0 zurück.

Nach Betätigen des „NEW GAME“ - Buttons wird ein neues Spiel gestartet und dem Spieler wird das Spiellayout präsentiert:

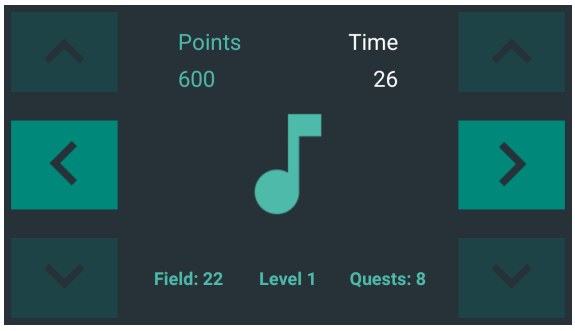

Abbildung 6: Das Layout während des Spiels

Das Layout in Abbildung 6 ist das komplexeste Layout. Das Spiel wird im Querformat gespielt um eine bessere Haptik zu erhalten. Zusätzlich fühlt sich das Smartphone so mehr als Game Controller an.

Am linken und rechten Bildschirmrand befinden sich die Pfeilbuttons zum navigieren. Hierbei können je nach belieben auf beiden Seiten gleichermaßen die Hoch- und Runter- Buttons benutzt werden. Links oben wird der aktuelle

Punktestand angezeigt, rechts oben die verbleibende Zeit in Sekunden. Am unteren Rand erfährt der Spieler, wieviele Aufgaben sich noch in diesem Level befinden und wie nahe er sich dem Ziel befindet. Die „Field“ - Anzeige ist dabei so zu verstehen: Je kleiner die Zahl, desto näher befindet der Spieler sich am Ziel. In der Mitte des Touchscreens befindet sich ein Image - Button. Dieser verändert sich abhängig von der Position des Spielers im virtuellen Spielfeld:

Aufgabenfeld

Das Notensymbol zeigt an, dass der Spieler sich auf einem Aufgabenfeld befindet. Um die Aufgabe zu hören und den Antwort - Dialog angezeigt zu bekommen, kann das Symbol gedrückt werden.

Aufgabenfeld richtig beantwortet

Dieses Symbol zeigt an, dass der Spieler eine Aufgabe korrekt gelöst hat oder auf ein bereits richtig beantwortetes Feld geht.

Aufgabenfeld falsch beantwortet

Dieses Symbol zeigt an, dass der Spieler eine Aufgabe falsch beantwortet hat oder sich auf einem Feld befindet, auf dem bereits eine Aufgabe falsch beantwortet wurde.

Leiter nach oben

Dieses Symbol zeigt an, dass der Spieler sich auf einem Feld mit der Möglichkeit ein Stockwerk nach oben zu gehen befindet.

Leiter nach unten

Dieses Symbol zeigt an, dass der Spieler sich auf einem Feld mit der Möglichkeit ein Stockwerk nach unten zu gehen befindet.

Verschlossene Tür

Dieses Symbol zeigt an, dass der Spieler auf eine verschlossene Tür stößt. Das Schloss kann mehrfach gedrückt werden, um die drei Aufgaben zum Öffnen der Tür zu lösen.

Geöffnete Tür

Dieses Symbol zeigt an, dass der Spieler eine Tür entriegelt hat oder auf eine zuvor schon geöffnete Tür stößt.

Wenn der Spieler ein Level beendet hat, erscheint ein weiteres Layout. Hierbei ergeben sich die Optionen über die beiden Buttons entweder das nächste Level zu spielen oder ins Hauptmenü zurückzukehren. Zusätzlich wird eine Statistik angezeigt. Der Spieler erfährt somit seinen aktuellen Punktestand und wieviele Aufgaben richtig, falsch oder nicht beantwortet wurden.

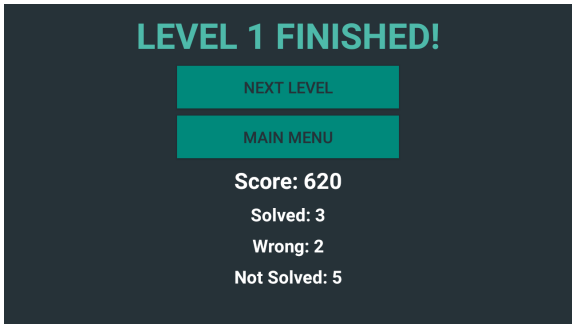

Abbildung 14: Das Layout nach Beendigung eines Levels

Bei Gameover, beziehungsweise wenn das Spiel durchgespielt wurde, wird ein ähnlicher Bildschirm gezeigt, jedoch ohne die Möglichkeit das nächste Level zu starten. Der Spieler muss ins Hauptmenü zurückkehren und das Spiel von vorne beginnen.

6.2 Das Design des Layouts

Für das Design der App wurden Teile des von Google entwickelten Design Standards *Material Design* verwendet [8]. Alle verwendeten Icons bedienen sich ebenfalls dieses Standards und können frei von der Material Design Website heruntergeladen und verwendet werden [9]. Die Farben und deren Zusammenstellung wurde mit einem Farbtool für Material Design erstellt [5]. Auch hier

gibt es eine XML-Resource-Datei, in welcher alle verwendeten Farben definiert werden:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#00897B</color>
    <color name="colorPrimaryDark">#005B4F</color>
    <color name="colorAccent">#4EBAAA</color>
    <color name="colorBackground">#263238</color>
</resources>
```

In der App werden abgesehen von Schwarz und Weiß nur diese vier Farben verwendet:

Primäre Farbe
Primäre dunkle Farbe
Akzentfarbe
Hintergrundfarbe

6.3 Leveldesign

Jedes Level besteht aus mehreren Stockwerken, mit jeweils mindestens einer Möglichkeit ins nächsthöhere Stockwerk zu gelangen. Man startet an der linken Wand im untersten Stockwerk.

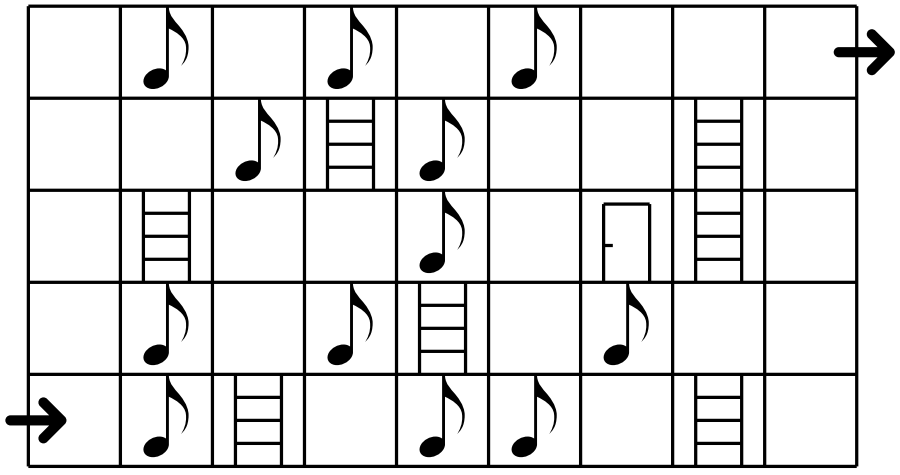

Abbildung 15: Leveldesign für Level 3

Die Abbildung 15 zeigt den Aufbau eines Levels. Die Noten stehen für Aufgabenfelder, die Leitern bieten Wege nach oben und unten und die Tür kann geknackt werden, um schneller ans Ziel zu gelangen.

In Android Studio werden die Level als String-Arrays in der XML-Resource-Datei gespeichert. Jeder Index des Arrays entspricht dabei einem Feld aus Abbildung 15. In jedem Index steht ein Key-Wert, der definiert, was auf dem jeweiligen Feld geschieht. Der XML-Code passend zu Abbildung 15 in `strings.xml` sieht wie folgt aus:

```
<string-array name="LEVEL_THREE">
    <item>LEFT_W</item>
    <item>INVERSION</item>
    <item>LADDER</item>
    <item>EMPTY</item>
    <item>HIGH_LOW_SEASHORE</item>
    <item>MAJOR_MINOR</item>
    <item>EMPTY</item>
    <item>LADDER</item>
    <item>RIGHT_W</item>
    <item>LEFT_W</item>
    <item>INTERVAL</item>
    <item>EMPTY</item>
    <item>INVERSION</item>
    <item>LADDER</item>
    <item>EMPTY</item>
    <item>INT_HIGH_LOW</item>
    <item>EMPTY</item>
    <item>RIGHT_W</item>
    <item>LEFT_W</item>
    <item>LADDER</item>
    <item>EMPTY</item>
    <item>EMPTY</item>
    <item>INTERVAL</item>
    <item>EMPTY</item>
    <item>DOOR</item>
    <item>LADDER</item>
    <item>RIGHT_W</item>
    <item>LEFT_W</item>
    <item>EMPTY</item>
    <item>INT_HIGH_LOW</item>
    <item>LADDER</item>
    <item>MAJOR_MINOR</item>
    <item>EMPTY</item>
    <item>EMPTY</item>
    <item>LADDER</item>
    <item>RIGHT_W</item>
    <item>LEFT_W</item>
    <item>MAJOR_MINOR</item>
    <item>EMPTY</item>
    <item>INTERVAL</item>
    <item>EMPTY</item>
    <item>HIGH_LOW_SEASHORE</item>
    <item>EMPTY</item>
    <item>EMPTY</item>
    <item>FINISH</item>
</string-array>
```

Abbildung 16: XML-Code für Level 3

Die Bedeutung der Key Werte der verschiedenen Indizes:

EMPTY: Leeres Feld

FINISH: Zielfeld

LADDER: Leiter nach oben

DOOR: Verschlossene Tür

LEFT_W: Links befindet sich eine Wand

RIGHT_W: Rechts befindet sich eine Wand

INTERVAL: Aufgabenfeld mit Intervallübung

INVERSION: Aufgabenfeld mit Umkehrungsübung

INT_HIGH_LOW: Aufgabenfeld mit Übung zum Intervalle unterscheiden

MAJOR_MINOR: Aufgabenfeld mit Tongeschlechtsübung

HIGH_LOW_SEASHORE: Aufgabenfeld mit Seashore Tonhöhenübung

LONG_SHORT_SEASHORE: Aufgabenfeld mit Seashore Tonlängenübung

LOUD_QUIET_SEASHORE: Aufgabenfeld mit Seashore Lautstärkenübung

6.4 Java Implementierung

6.4.1 Das Hauptmenü

Die App ist in vier Activities unterteilt, welche jeweils von der Android-Klasse `AppCompatActivity` abstammen. Nach dem starten befindet man sich in der `MainMenuActivity.class`. Hier werden zunächst alle Sounds mithilfe der `Sound`-Klasse initialisiert (siehe Kapitel Sounddesign). Danach werden die Buttons initialisiert und über einen `OnClickListener` die jeweilige Aktion implementiert:

In Abbildung 17 ist das Klicken des „NEW GAME“-Buttons implementiert. Zunächst wird der Sound für ein neues Spiel abgespielt. Danach wird ein neuer Intent erstellt. Dieser wird dann in die Methode `startActivityForResult` übergeben. Diese Methode ermöglicht später die Übergabe des Punktestands aus der `GameActivity.class` in die `MainMenuActivity.class`, wenn der `requestCode == 1` ist. Dies wird in der `onActivityResult` Methode überprüft. Wird ein gültiger Wert übergeben und der Punktestand ist höher als der vorherige Highscore, so wird der neue Highscore im `TextView` gespeichert und in die `SharedPreferences` geschrieben. Somit wird sichergestellt, dass der Wert auch nach dem Beenden der App weiter besteht.

```
Button mNewGameButton = (Button) findViewById(R.id.new_game_button);
        mNewGameButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                sSound.playSound(sSound.getSoundIDnewgame(), VOL_GAME);
                //start GameActivity.java
                Intent newGameIntent = new Intent(MainMenuActivity.this,
                        GameActivity.class);
                startActivityForResult(newGameIntent, 1);
            }
        });
```

Abbildung 17: Button zum Starten eines neuen Spiels

6.4.2 Die GameActivity

In der `GameActivity.class` wird zunächst das Layout auf Vollbild und Querformat gestellt. Danach werden die ganzen zuvor geladenen Sounds in die `GameActivity` übernommen. Zusätzlich wird geprüft, welcher Modus im Hauptmenü gewählt wurde. Danach wird die Methode `startLevel()` aufgerufen. Hier werden alle Level aus der Datei `strings.xml` eingelesen und in String-Arrays gespeichert. Hat der Spieler im Hauptmenü den normalen Modus gewählt, so wird ein Countdown-Timer initialisiert. Anschließend werden die Pfeilbuttons initialisiert und implementiert. Durch das Drücken der Pfeilbuttons wird der Index des Arrays entsprechend geändert. Die links/rechts - Buttons erniedrigen, beziehungsweise erhöhen den Index jeweils um eins. Die hoch/runter - Buttons erhöhen, beziehungsweise erniedrigen den Index jeweils um die Anzahl der Felder die es pro Reihe im jeweiligen Level gibt. Bei jedem Klicken wird nach dem Ändern des Index die Methode `switchKey()` aufgerufen. Hierbei wird der Wert des aktuellen Feldes übergeben. Die `switchKey()` - Methode besteht aus einer switch - Abfrage, die zwischen den verschiedenen Key - Werten der Felder unterscheidet und entsprechende Aktionen ausführt.

Zunächst überprüft die Methode `checkForLadder()`, ob eine Leiter nach unten vorhanden ist. Abhängig davon wird der passende Sound abgespielt sowie die entsprechenden Pfeilbuttons aktiviert. Danach wird das Icon in der Mitte der Bedienoberfläche zu einem Notensymbol geändert. Das Handy vibriert kurz und das Notensymbol blinkt auf. Über den `OnClickListener` des Notensymbols

```
case "INTERVAL":
    if(checkForLadder()) {
        mSound.playSound(mSound.getSoundIDexerciseDown(), VOL_GAME);
    } else {
        mSound.playSound(mSound.getSoundIDExercise(), VOL_GAME);
    }
    mIcon.setImageResource(R.drawable.ic_music_note_accent);
    mIcon.startAnimation(mAnimationBlinking);
    mVibrator.vibrate(100);
    mClickID = 0;
    mIcon.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(mClickID == 0) {
                mRandInterval = intervalIDs[randNumber(intervalIDs.length)];
                mSound.playSound(mRandInterval, VOL_EXERCISE);
                DialogFragment mDialog = new IntervalDialogFragment();
                mDialog.show(getFragmentManager(), "DialogFragment");
                mClickID = 1;
            }
        }
    });
    fieldCounter();
    break;
```

Abbildung 18: Switch Case für den Key „INTERVAL“

wird der Klick des Aufgabenbuttons implementiert. Hier wird zunächst aus allen Aufgaben des entsprechenden Aufgabentypen zufällig eine Aufgabe ausgewählt und danach abgespielt. Danach öffnet sich ein `DialogFragment`. Der Spieler kann nun seine Antwort auswählen und bestätigen. Durch das Bestätigen wird in der GameActivity die Methode `onUserSelectValue()` aufgerufen und die Antwort übergeben. Um zu überprüfen ob die Antwort richtig oder falsch ist, wird diese über eine if-Abfrage mit dem zuvor abgespielten Sound verglichen. Wurde die Aufgabe richtig beantwortet, ertönt der entsprechende Sound und das Symbol in der Mitte der Oberfläche zeigt ein Häckchen. Zusätzlich werden die Punkte zu den Gesamtpunkten gezählt.

Erreicht der Timer die 10 Sekunden Marke, wird jede Sekunde ein Warnton ab-

gegeben, um den Spieler darauf aufmerksam zu machen. Läuft die Zeit ab öffnet sich der Gameover - Dialog und der Spieler kann seine Statistik einsehen und ins Hauptmenü zurückkehren. Erreicht er das „FINISH“ - Feld vor Ablauf der Zeit, wird zunächst die Restzeit mit 10 multipliziert und zu den Gesamtpunkten addiert. Danach wird das Level um eins erhöht und überprüft, ob sich der Spieler schon im letzten Level befunden hat. Danach wird ein Dialog geöffnet, in dem der Spieler das nächste Level starten oder ins Hauptmenü zurückkehren kann. Das Betätigen des „NEXT LEVEL“ - Buttons führt wiederum die `startLevel()` - Methode mit dem nächsthöheren Level aus.

6.4.3 Die Sound Klasse

In der Klasse `Sound.class` werden alle Sounds initialisiert und über Getter-Methoden an die Activities übergeben. Zunächst wird hierbei ein `Soundpool`-Objekt erstellt. Danach wird jedem Wave-File aus dem Resources Ordner `raw` eine eigene Sound - ID zugewiesen und diese an den Soundpool übergeben. Um Zugriff auf die Sound - IDs in den verschiedenen Activities zu erhalten, wird für jede ID eine Getter - Methode erstellt, zum Beispiel:

```
soundIDminorThirdC3 = mSoundPool.load(context, R.raw.minor_third_root_c3, 1);

int getSoundIDminorThirdC3() {
    return soundIDminorThirdC3;
}
```

Abbildung 19: Die Sound-ID der kleinen Terz mit Grundton C3 und die Getter-Methode

Die Initialisierung der Sounds muss nur einmal beim Starten der App durchgeführt werden. Das Sound - Objekt wird dabei als `static` - Objekt erstellt und wiederum über eine Getter - Methode an die anderen Activities übergeben.

6.5 Sounddesign

Das Sounddesign für *Learn to Listen* wurde in der Ableton Live 9 Suite [2] erstellt. Hierbei wurden die Plug-ins Massive [10] und Kontakt mit der Sample - Library „The Gentleman“ [14] von Native Instruments verwendet. Die Aufgaben aus dem Seashoretest wurden dabei unbearbeitet von der Website des IEM

übernommen [11].

Für die Gehörbildungsaufgaben wurden Klaviersamples aus der oben genannten Kontakt Library verwendet. Um bei jeder Aufgabe den gleichen Klang zu erhalten, wurde immer die selbe Velocity verwendet. Zusätzlich haben alle Übungen die gleiche Länge. Das Klavier ist ein übliches Instrument für die Gehörbildung und wird auch im Gehörbildungsunterricht sowie in Programmen wie „EarMaster“ [6] verwendet.

Für die verschiedenen Gamesounds wurde der VST - Synthesizer Massive verwendet. Dieser Wavetable - Synthesizer eignet sich sehr gut, um den erwünschten 8-bit Retro - Gamesound zu erstellen. Hierzu wurden meist Puls- oder Sägezahnwellen als Grundlage genommen. Diese werden mit Amplituden- und Tonhöhenmodulation versehen und leicht verzerrt. Zum Schluss wurde noch ein Bitcrusher verwendet, um die Bittiefe zu verkleinern und so den 8-bit Charakter hervorzuheben.

7 Konklusio

Im Rahmen des Seminars „Computermusik und Multimedia 02“ wurde ein Prototyp eines *Audio Games* entwickelt. Hierbei wurden alle Schritte des Game Designs durchlaufen und am Ende entstand ein fertiger Prototyp, der als Proof-of-Concept dient. Der anfängliche Plan, *Learn to Listen* komplett für sehbehinderte Menschen zugänglich zu machen, stellte sich als schwierig heraus. Durch die fehlende Haptik des Touchscreens eines Smartphones müssen andere Wege gewählt werden um ein Spiel oder eine App auf mobilen Geräten barrierefrei zu gestalten. So könnte die Text-To-Speech Funktionalität beispielsweise für die Navigation in den Menüs gewählt werden. Etwas schwieriger und aufwändiger gestaltet sich dies dann im Spiel selbst. Hier könnte man beispielsweise 3D - Audio zur Orientierung innerhalb des Spielfeldes verwenden. Unter Berücksichtigung dieser Gesichtspunkte, wurde der Prototyp nicht gänzlich für blinde Menschen zugänglich gemacht. Durch die sehr einfache Steuerung und die auditiven Hinweise im Spiel sollte es jedoch Spielern mit stark eingeschränktem Sehvermögen möglich sein, *Learn to Listen* zu spielen. Als weiterer Schritt wurde das „Gamesounds“ - Menü mit Text-To-Speech Funktion integriert. Das bedeutet, der Spieler bekommt bei einmaligem Betätigen eines „play“ - Buttons die Art des Sounds vorgesagt, bei nochmaligem Betätigen wird der Sound abgespielt.

In der Weiterentwicklung dieses Prototypen müssten noch einige Schritte und Optimierungen vorgenommen werden, bevor das Spiel für den Endnutzer freigegeben werden kann. Hierzu gehören zum Beispiel die Entwicklung weiterer Level und Schwierigkeitsgrade. Der Prototyp enthält bisher nur drei verschiedene Level, welche leicht in der Schwierigkeit steigen. Durch eine höhere Anzahl an Leveln könnten in den ersten Leveln zum Beispiel leichtere Intervalle abgefragt werden, um nach und nach schwierigere hinzuzunehmen. Zusätzlich könnten sich die Level noch mehr in ihrer Größe verändern.

Eine weitere Herausforderung in der Entwicklung des Prototypen stellt der Speicherbedarf dar. Da jede Aufgabe als einzelne Wave - Datei eingebunden wird und für jeden Aufgabentypen mehrere Grundtöne zur Verfügung stehen, benötigt die App sehr viel Speicherkapazität (aktuell ca. $150MB$). Dies hält potentielle Spieler eher vom Downloaden der App ab. Ein Lösungsansatz wäre, die Audio-Engine so zu gestalten, dass nur einzelne Töne als Wave - Dateien

eingebunden werden. Diese einzelnen Töne müssten in Echtzeit zu den jeweils benötigten Aufgaben kombiniert werden. Eine besondere Schwierigkeit wäre hierbei sicherlich das richtige Timing beim Zusammenspiel der Töne.

Insgesamt bot das Seminar einen interessanten Einblick in die Entwicklung eines Spiels, sowie die Programmierung einer App.

Literatur

[1] *A Dictionary of Psychology*. http://www.oxfordreference.com/view/10.1093/oi/authority.20110803100450451. – [Online; Stand: 12.07.2017]

[2] *Ableton Live*. https://www.ableton.com/. – [Online; Stand: 19.09.2017]

[3] *Android (Betriebssystem)*. https://de.wikipedia.org/wiki/Android_(Betriebssystem). – [Online; Stand: 22.07.2017]

[4] *Application Fundamentals | Android Developers*. https://developer.android.com/guide/components/fundamentals.html. – [Online; Stand: 22.07.2017]

[5] *Color Tool - Material Design*. https://material.io/color/#!/?view.left=0&view.right=0&primary.color=00897B. – [Online; Stand: 15.07.2017]

[6] *EarMaster 6 Professional: Software für Gehörbildung und Vom-Blatt-lesen: Lektionen und Übungsinhalte*. http://www.klemm-music.de/lernen/earmaster/earmaster_pro/lektionen.php. – [Online; Stand: 12.07.2017]

[7] *Introduction to Android | Android Developers*. https://developer.android.com/guide/index.html. – [Online; Stand: 22.07.2017]

[8] *Material Design*. https://material.io/. – [Online; Stand: 09.06.2017]

[9] *Material Icons - Material Design*. https://material.io/icons/. – [Online; Stand: 15.07.2017]

[10] *Native Instruments Massive*. https://www.native-instruments.com/de/products/komplete/synths/massive/. – [Online; Stand: 19.09.2017]

[11] *Seashore Test Aufgaben, Institut für elektronische Musik*. https://iem.kug.ac.at/lehre/elektrotechnik-toningenieur/zulassung/zulassungspruefung-bachelor-elektrotechnik-toningenieur/seashore-test.html. – [Online; Stand: 16.06.2017]

[12] *Smartphone*. https://de.wikipedia.org/wiki/Smartphone#Geschichte. – [Online; Stand: 22.07.2017]

[13] *The Activity Lifecycle | Android Developers*. https://developer.android.com/guide/components/activities/activity-lifecycle.html. – [Online; Stand: 23.07.2017]

[14] *The Gentleman*. https://www.native-instruments.com/de/products/komplete/keys/the-gentleman/. – [Online; Stand: 19.09.2017]

[15] Atari Museum: *The Atatri Touch Me*. http://www.atarimuseum.com/videogames/dedicated/touchme.html. – [Online; Stand: 06.07.2017]

[16] Crawford, C. : *The Art of Computer Game Design by Chris Crawford*. 1982. – ISBN 0881341177

[17] Friberg, J. ; Gärdenfors, D. : Audio games: new perspectives on game audio. In: *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology* (2004), S. 148–154. ISBN 1581138822

[18] James, D. G.: *Android game programming for dummies*. Wiley, 2013. – ISBN 9781118235997

[19] Röber, N. ; Masuch, M. : Playing Audio-Only Games - A Compendium of interacting with virtual, auditory worlds. In: *Digital Games Research Conference*, 2005

[20] Salen, K. ; Zimmerman, E. : Rules of Play: Game Design Fundamentals. (2004). ISBN 0262240459

[21] Wikipedia: *Audio Games*. https://en.wikipedia.org/wiki/Audio_game. – [Online; Stand: 06.07.2017]

Johannes Wolfgruber

(Name in Blockbuchstaben)

01331048

(Matrikelnummer)

Erklärung

Hiermit bestätige ich, dass mir der *Leitfaden für schriftliche Arbeiten an der KUG* bekannt ist und ich diese Richtlinien eingehalten habe.

Graz, den …….

…….
Unterschrift der Verfasserin/des Verfassers