PipeHype

Android App zur Lautstärkemessung und -interpretation

Mobile Systeme
Sommersemester 2014

Virak Oum
1901577

Thomas Jäger
2076956

Inhaltsverzeichnis

1 Einleitung....3
1.1 Ursprüngliche Idee.... 3
1.2 Überarbeitete Idee.... 3
2 Anforderungen....4
3 Klassenübersicht....5
3.1 MainActicity.java.... 5
3.2 Anleitung.java.... 5
3.3 Selection.java.... 6
3.4 Record.java.... 7
3.4.1 Spieloberfläche.... 9
3.5 Voegel.java.... 10
3.6 DB.java.... 10
4 Layout....11
5 Probleme....12
5.1 Verbessertes Vorgehen.... 12
6 Ausblick....13

1 Einleitung

1.1 Ursprüngliche Idee

Die Ursprüngliche Idee war es eine „Pfeiftrainer“-App mit dem Namen „PypeHype“ für Android Smartphones zu programmieren, mit welcher Benutzer das Pfeifen üben können.

Die App sollte einen vom Benutzer erzeugten Pfeifton über das Mikrofon aufnehmen, und dessen Lautstärke und Frequenz auslesen. Mit Hilfe dieser Funktionen sollte es dem Benutzer möglich sein, gezielt Töne der Tonleiter auszuwählen, um diese zu üben. Es sollte eine Anzeige geben die dem Benutzer ein Feedback über seinen aktuellen Ton gibt, damit er weiß, ob er momentan über oder unter dem zu übenden Ton liegt.

Zusätzlich sollte die Lautstärke gemessen und ausgegeben werden.

1.2 Überarbeitete Idee

Die Überarbeitete Idee verzichtet auf eine Frequenzanalyse, da deren Implementierung den zeitlichen Umfang dieses Projekts sprengen würde, was allerdings leider erst nach Beginn der Recherche nach Lösungen klar wurde.

Stattdessen beschränkt es sich auf die Aufnahme und Ausgabe der Lautstärke. Diese Lautstärkemessung des Pfeiftons wird als kleines Spiel verpackt ausgegeben.

In diesem Spiel schlüpft der Spieler in die Rolle eines männlichen Vogels. Es geht darum, eine bestimmte Zeit einen Pfeifton zu halten, welcher eine Mindestlautstärke erreichen und halten muss um gewertet zu werden. Hält der Benutzer den geforderten Pfeifton lange genug, lockt er durch sein Pfeifen eine „Vogeldame“ an, welche durch einen abgespielten Sound, sowie eine visuelle Anzeige in Form einer RatingBar dargestellt wird.

Insgesamt gibt es drei verschiedene Schwierigkeitsstufen, welche sich getrennt voneinander spielen lassen.

2 Anforderungen

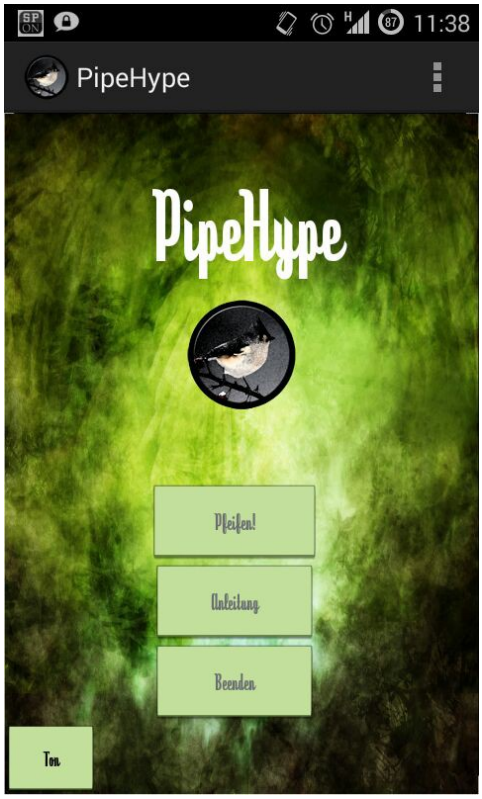
- **Hauptmenü**
  - Weiterleitung zur Levelauswahl.
  - Weiterleitung zur Anleitung.
- Anleitung soll hinreichend erklärend sein, damit Benutzer die App/das Spiels verwenden können.
- **Levelauswahl**
  - Die drei Level lassen sich auswählen.
  - Weiterleitung zur Spieloberfläche.
- **Spieloberfläche**
  - Ein vom Benutzer erzeugter Pfeifton wird live aufgenommen und ausgewertet.
  - Lautstärke soll mit Hilfe einer ProgressBar wiedergegeben werden.
  - Gesammelte Punkte (Vogeldamen) sollen visuell mit Hilfe einer RatingBar , sowie mit einem Soundeffekt dargestellt werden.
  - Drei Level mit merkbar unterschiedlichem Schwierigkeitsgrad sind spielbar.
  - Spiel soll neugestartet und beendet werden können.

3 Klassenübersicht

3.1 MainActicity.java

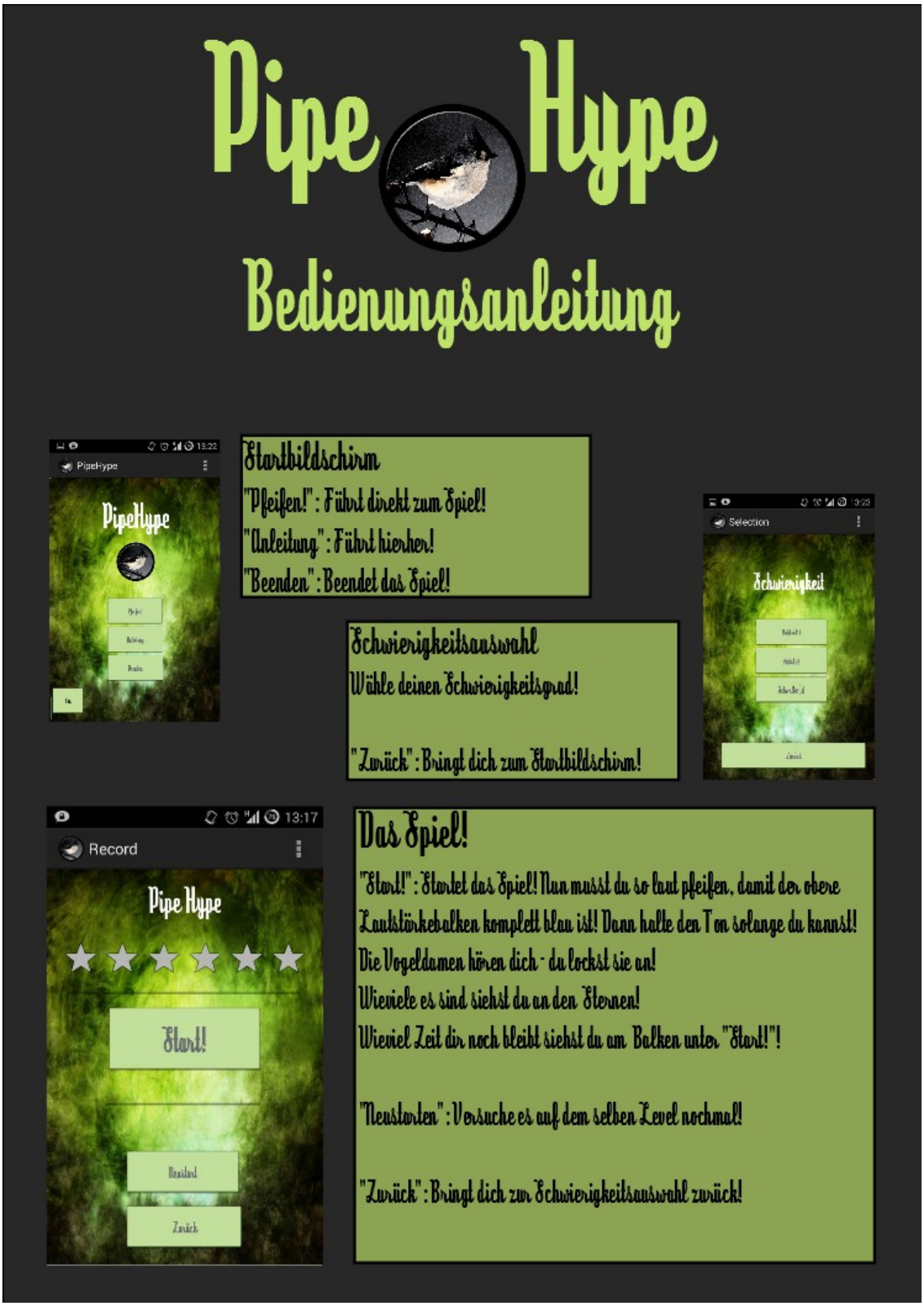
Die MainActicity.java stellt in dieser App das Hauptmenü dar. Sie dient als Ausgangspunkt zur Navigation durch das Programm und besteht aus:

- ToggleButton button_Sound: Durch Klicken dieses Buttons kann der Benutzer den Hintergrundsound deaktivieren/aktivieren.
- Button button: Leitet den Benutzer zur Klasse Selection.java weiter.
- Button button_anl: Leitet den Benutzer zur Anleitung.java weiter.
- Button button_schließen: Schließt die App.
- Ein Mediaplayer, welcher den Hintergrundsound abspielt und mit dem ToggleButton button_Sound die Lautstärke auf 0 und wieder auf 0.1 setzten kann
- Außerdem erzeugt die Mainacticity.java ein globales Objekt der Klasse Voegel.java. Dieses wird erst in der Record.java verwendet, wird allerdings bereits hier erstellt, da sie unter Anderem den Ton des gesamten Programms steuert.



3.2 Anleitung.java

Diese Activity enthält die Bedienungsanleitung und einen Button welcher den Benutzer zum Hauptmenü zurückführt.



3.3 Selection.java

Diese Activity dient zur Auswahl der Schwierigkeitsstufe und besteht aus:

- **Buttons**:
  - level1, level2, level3: Leiten den Benutzer zur gewünschten Schwierigkeitsstufe weiter.
  - menu: Leitet den Benutzer zum Hauptmenü zurück.
- **Methoden**:
  - goToLevel1(), goToLevel2(), goToLevel3(): Diese Methoden werden durch die oben genannten Buttons entsprechend aufgerufen und bauen ein neues Objekt der Klasse Record.java. Sie übergeben diesem die Werte für den gewählten Schwierigkeitsgrad. Dies ist zum Einen der entsprechenden String mit den Werten „1“ - „3“, zur Anzeigen des Levels in der Record.java. Zum Anderen den Wert, welcher bestimmt wie lange der Pfeifton gehalten werden muss. Er ist in ms angegeben. Diese beiden Werte werden im Bundle „Level“ gespeichert und übergeben.

3.4 Record.java

Diese Klasse beinhaltet die Spieloberfläche und ist die wichtigste Klasse des Programms.

- **Buttons/Bars:**

- button_Close: Leitet den Benutzer zum Selection.java zurück.
- button_Restart: Startet das aktuelle Level neu. Der Button wird erst „klickbar“ wenn der ToggleButton button_Start betätigt wurde.
- ToggleButton button_Start: Startet die Aufnahme und damit das Spiel. Solange er aktiv ist läuft die Aufnahme. Er wird inaktiv wenn die Spielzeit abgelaufen ist oder wenn der Benutzer ihn erneut anklickt. Er trägt im aktivierten Zustand die Aufschrift „Abbrechen!“ und im inaktiv „Start!“ Wird er manuell in den inaktiven Zustand versetzt, wird das Spiel abgebrochen und die bis dahin gesammelte Anzahl der Vogeldamen ausgegeben. Läuft die Spielzeit ab, passiert dies ebenfalls.

- **Methoden:**
  - restart(): Diese Methode startet das aktuell gespielte Level neu. Dazu beendet sie das Objekt Voegel voegel, stoppt das Objekt DB db, ließt die übergebenen Werte für das aktuelle Level aus der Selection.java ein, startet die Record.java neu und schließt die aktuell aktive Instanz von Record.java (Also quasi sich selbst). Sie wird durch einen Klick des Buttons button_Restart aufgerufen.
  - Close(): Schließt die aktuelle Instanz von Record.java, indem sie wie die Methode restart() arbeitet, aber keine neue Instanz der Record.java startet, sondern von Selection.java. Auslöser ist ein Klick auf den Button button_Close.
  - HandleMessage(): Diese Methode enthält das „Herz“ der App. Sie sorgt durch Aufrufe von Methoden aus der DB.java und der Voegel.java dafür dass zum einen der Lautstärkewert eingelesen und berechnet wird, die eroberten Vogeldamen hochgezählt werden und die entsprechende Anzahl ausgegeben wird. Außerdem sorgt sie dafür dass die Zeit korrekt runtergezählt wird und beendet das Spiel nach deren Ablauf. Ausgelöst wird sie durch eine Message der Runnable runnable.

  Runnable runnable(): Sendet messages zur Aktivierung an die HandleMessage(), solange die Zeit > 0 ist und der ToggleButton button_Start aktiv ist. Zwischen jeder Message macht sie 100 Millisekunden Pause, um eine korrekte Zeit- und Punkteberechnung zu ermöglichen.

3.4.1 Spieloberfläche

- Button „Start!“ = button_Start.
- Button „Neustart“ = button_Restart
- Button „Zurück“ = button_Close

Die RatingBar zeigt die bereits eroberten Vogeldamen.

Die ProgressBar unter der RatingBar zeigt die aktuelle Lautstärke des Pfeiftons an. Ist der Balken voll blau gefüllt, ist der Ton laut genug.

Die ProgressBar unter dem button_Start zeigt die verbleibende Zeit an.

3.5 Voegel.java

Diese Klasse besitzt keine eigene Oberfläche und dient zur Berechnung und Rückgabe der aktuell angelockten „Vogeldamen."

- **Methoden:**
  - voegelAddieren(): Diese Methode erhöht die Zahl der Integer Variablen anzahlVoegel bei jedem Aufruf um 1.
  - getVoegel(): Gibt den aktuellen Wert der Variablen anzahlVoegel mit einem String zurück.
  - vogelSound(): Erstellt einen Mediaplayer und gibt ein Vogezwitschern aus. Ist der ToggleButton button_Sound in der MainActicity.java deaktiviert setzt sie die Lautstärke des Mediaplayers auf 0, ist er aktiv auf 0.3.

3.6 DB.java

Diese Methode besitzt ebenfalls keine Oberfläche. Sie dient zur Aufnahme des Pfeiftons über das Mikrofon und berechnet daraus die Lautstärke.

- start(): In dieser Methode wird der MediaRecorder initialisiert und gestartet.
- stop(): Der MediaRecorder wird gestopt und released.
- getAmplitude(): Die Amplitude des eingehenden Tons wird ausgelesen.
- getAmplitudeEMA(): Diese Methode berechnet mit Hilfe der Amplitude den Dezibelwert und gibt ihn zurück.

4 Layout

Beim Layout wurde entschieden, jeweils als Layout-Container das „RelativeLayout“ -Element zu verwenden und in dieses weitere Layout-Modelle einzubinden. Durch die Verschachtelung der Layouts ist es uns möglich gewesen die einzelnen Elemente in gleichmäßiger Relation zueinander zu setzen und die Anordnung für die meisten Screens kompatibel zu machen. Außerdem bietet es die Möglichkeit, eine freie Gestaltung hinsichtlich der Button- und Textobjekte zu gewährleisten indem man die Objekte gruppiert anordnen lässt. Da der Fokus darin liegt, dem User eine klare Menüführung zu bieten, wurden jeweils die wichtigen Bedienelemente zentral positioniert, so dass man diese sofort erkennen kann.

5 Probleme

Eines der Hauptprobleme während der Arbeit an PipeHype war das Aufsetzen/Verwalten eines Github-Repositories. Immer wieder zerschoss es uns das Repository durch Versionskonflikte, obwohl wir bereits Erfahrung damit hatten – jedoch nur mit Java, nicht mit Android. Eines der größten Probleme war dass nach beinahe keinem Pullvorgang die Libraries problemlos eingebunden wurden konnten und auch die Lösung dieses nur manuell zu lösenden Problems nicht immer dieselbe war.

Zusätzlich erschwert wurde uns die Arbeit dadurch, dass wir zum einen Teil Windows, zum anderen Teil Linux verwenden.

So mussten wir sehr viel Zeit opfern um andere Github-Tools zu testen, Repositories neu aufzusetzen.

Probleme ergaben sich natürlich auch durch unsere, bereits zu Beginn dieser Dokumentation beschriebenen Fehleinschätzungen was den Umfang und die Schwierigkeit unseres ursprünglichen Vorhabens angeht. Leider war es uns wie oben beschrieben nicht möglich dieses Ziel umzusetzen.

5.1 Verbessertes Vorgehen

Um zukünftig Verzögerungen und Probleme zu vermeiden, sollten wir uns frühzeitiger und präziser über den Umfang und Arbeitsaufwand, welcher zur Umsetzung von Ideen notwendig ist, informieren. Bei Problemen wie dem Aufsetzen und Verwalten des Repositories müsste man früher externe Hilfe hinzuziehen.

Eine klarere, koordiniertere Arbeitsstruktur kann helfen Zeit und Aufwand zu sparen und einen einfacheren zukünftigen Ausbau der Anwendung vereinfachen.

6 Ausblick

Im Großen und Ganzen sind wir mit dem Ergebnis dieses Projekts zufrieden. Wir haben es zwar nicht geschafft unser ursprüngliches Ziel zu erfüllen, haben aber rechtzeitig umgelenkt um dennoch ein akzeptables Ergebnis, wenn auch etwas abgewandelt, fertigstellen zu können.

Auch konnten wir Erfahrung sammeln und viel über die Android Programmierung lernen, da dies unser erstes Android Projekt war.

Wir möchten in Zukunft an dieser App weiterarbeiten. Man könnte das Spiel in viele Richtungen verfeinern und erweitern. Zum Beispiel durch eine schönere, ansprechende grafische Darstellung, dem Einbauen mehrerer Schwierigkeitsstufen, die dan auch miteinander verknüpft werden, und somit eine Art „Kampagne“ darstellen. Ebenfalls denkbar ist eine eigenen grafische Spieloberfläche in einer GameEngine (Was allerdings ein weitaus größeres Unterfangen wäre.)

Ein Highscore ist ebenso denkbar, wie eine Internetanbindung um diesen mit anderen Spielern/Freunden vergleichen zu können.

Mit genug Zeit wird es auch möglich sein unser ursprüngliches Ziel, das Auslesen der gepfiffenen Frequenz, umzusetzen und in das Spiel zu integrieren. Dann könnte die Schwierigkeit nicht nur in der zu erreichenden Lautstärke bestehen, sondern ebenfalls durch die gepfiffene Tonhöhe gesteuert werden. Dies macht das Spiel facettenreicher, interessanter und spannender.