Creating a hearing aid app for smartphones

Frederik Kammel

2018

Contents

[1. Motivation and introduction 2](#_Toc506684670)

[2. Overview over hearing impairment 2](#_Toc506684671)

[3. Overview over the operating principles of a hearing aid 2](#_Toc506684672)

[3.1. Volume compression 2](#_Toc506684673)

[3.2. Amplification algorithms 2](#_Toc506684674)

[4. Sound latency 2](#_Toc506684675)

[5. Introduction to the Android application framework 3](#_Toc506684676)

[5.1. Basic framework 3](#_Toc506684677)

[5.2. Android Native Development Kit 3](#_Toc506684678)

[Literature 5](#_Toc506684679)

# Motivation and introduction

Elder people generally suffer from hearing loss. To solve the issue, most of them buy hearing aids. Such devices are often expensive and involve regular hearing care. On the other hand, most people already own a smartphone which is equipped with a microphone and the possibility to connect headphones. Although the acoustics of the microphones in smartphones are by far worse than the acoustics of professional grade hearing aids, smartphones are often cheaper and easier to set up than traditional hearing aids which might represent a good first step for many people in the world of hearing aids.

This paper focuses on the creation of the HearingAid app, an Android app that allows users to record the smartphone’s microphone input, amplify and play it back in real time over the user’s headphones. While being aware of the limitations imposed by the device hardware and by the Android platform, the app aims to be as good as a professional hearing aid on supported devices.

# Overview over hearing impairment

Hearing loss affects every patient differently. While most patients are affected as a sign of aging, some patients acquire their impairment through genetical disorders. In any case, hearing aids can only offer help in case of partial hearing loss, meaning that the patient is still capable to hear sounds that exceed a certain threshold in volume. While all humans have that so called absolute threshold of hearing, which describes the minimum volume a sound must have to be perceived, patients who suffer from hearing loss have a threshold which is significantly higher than the one of the average human being (Kießling 2001).

While the absolute threshold of hearing (ATH) describes the lower bound of the hearing range of a human being, the threshold of pain describes the level of loudness at which the patient feels pain, thus representing the higher bound of the hearing range. In general, hearing impaired patients maintain their threshold of pain while their ATH rises (Kießling 2001). The actual value of both, the ATH and the threshold of pain vary from person to person and also depend on the frequency of the sound (Kießling 2001).

# Overview over the operating principles of a hearing aid

## Volume compression

In general, hearing aids aim to amplify the sound a patient would hear. Most relevant sounds (speech, music, warning signals) are in a frequency range between 500 and 6000 Hz (Kießling 2001). Due to the above stated fact that the patients ATH rises while the threshold of pain doesn’t, hearing aids perform volume compression (a large range of volume is mapped on a smaller range of volume). Sounds below the patient’s ATH are amplified while making sure that no sound exceeds the threshold of pain. On the other hand, hearing aids must preserve the perception of volume (louder sounds must be perceived louder). That means that hearing aids cannot simply amplify all sounds by the same amount (risk to exceed the threshold of pain) and cannot amplify all sounds to the same volume neither (perception of volume would be lost) (Kießling 2001). To accomplish this, hearing aids generally know their user’s ATH and threshold of pain and calculate the amount by which a sound must be amplified for each sound individually depending on its frequency and volume.

## Amplification algorithms

# Sound latency

While the volume by which a sound is amplified hugely impacts the user’s comfort, sound latency does even more. Users will start to notice sound delays if the latency is more than 10 ms (Superpowered Inc. 2017) and will get seriously distracted at higher latencies. According to our own field measurements, latencies of more than 50 ms make conversations impossible. While one might still be able to hear the conversation partner, it will become difficult for the user to speak since he will hear his own voice delayed by the very same latency. Due to this, hearing aid hardware and algorithms must be built to process sound in 10 ms or less. This is easy for hearing aid manufacturers as they are constructing their own hardware and have the budgets for more expensive hardware, engineered for the exclusive purpose of sound processing. On smartphones, this is even a bigger issue than the acoustics of the microphone as smartphone hardware is built for general purpose and not necessarily optimized for low latency. Also, smartphone CPUs have several tasks at once (responding to calls, responding to user input, execution of other apps running in the background) while hearing aid hardware solely processes the sound.

# Introduction to the Android application framework

## Basic framework

Apps on the Android platform are generally built using the Java language. This comes with many advantages: Developers do not need to worry about memory management since Java is garbage collected. Because Java runs on the so-called Java Virtual Machine (JVM)[[1]](#footnote-1), it is cross platform which removes the effort of compiling a program for every CPU architecture available on the market.

The advantages aside, the Java platform also brings several disadvantages. Firstly, the JVM slows the program down by design as it must translate every instruction of the virtual CPU to instructions for the real CPU. Furthermore, as Java is mostly Just-in-time compiled (JIT compiled), the JIT compiler also takes up time to compile. Thirdly, when the garbage collector becomes activated, the application is halted for several milliseconds until the garbage collector finishes its work.

All those factors lead to applications written in the Java language to be slower. While this does not matter in usual usage cases[[2]](#footnote-2), it does in the case of this app as it directly impacts sound latency, which is a crucial factor of success as mentioned above.

## Android Native Development Kit

As there are many other use cases which require low latency, several application developers have requested Google (the main manufacturer of Android) to allow execution of so-called native code. Native code are instructions that are executed directly on the device’s CPU instead of being executed on the JVM. This eliminates the JVM software layer but also removes all the advantages that come with the use of the JVM. Usually, native code is written in C++ and compiled into assembler code for each platform that the app will be able to run on.

Google made interoperability between native code and Java code easy by offering the so-called Android Native Development Kit (NDK). The NDK offers all tools required for native development (compiler, framework classes). To transfer information between native code and Java code, the NDK utilizes a concept that is already present in Java, the Java native interface (JNI). The latter allows information (commands, strings, numbers, …) to be transferred from Java to native code and vice versa. In general, all code that affects the view of an Android application is written in Java while backend code is written in C++. A controller class written in Java is responsible for the transfer of data and commands from and to C++.

# UI Design

## Main screen

Generally, the app’s UI design follows Google’s Material Design Guide (Google LLC 2014). The main screen of the app has one major task: Start and stop the hearing aid functionality. As such, it consists of only one Play button which transforms itself into a Pause button when the hearing aid is switched on. Also, users of the app are likely to be aged as hearing loss is often caused by aging. Thus, users might also be able to have issues with their vision, which is why the button is orange in front of a white background.

Apart from allowing the user to switch the hearing aid on and off, the main screen also displays various messages to the user:

* The message of the day (MOTD): If we have important news to tell the users, we can publish them on a so-called RSS feed which resides on a remote server. The app retrieves this feed and displays an abstract of the news to the user in case new news were found. If the user wishes to read the entire message, he may click a ‘Read more’ button or dismiss the message if he is not interested in reading it. The MOTD is displayed as a so-called non-modal bottom sheet.
* A warning if the user’s hardware does not support low latency sound processing. The operating categorizes device hardware by its audio performance: So-called pro-audio devices must guarantee a maximum sound latency of 20 ms and low-latency devices must guarantee a maximum sound latency of 45 ms (Superpowered Inc. 2017). If the app detects that it is run on a device which does not respect the pro-audio or low-latency specification, a corresponding message is displayed to the user that he might experience audio latencies of 20/45 ms or more. This message is also displayed as a non-modal bottom sheet.

## Settings screen and hearing test

# Architecture of backend code

## Sound processing core

The reasons for sound latency can be divided into two separate groups: Latency induced by the device hardware and operating system and latency induced by the app. The former cannot be influenced by application developers why the latter can. Therefore, this section focuses on latency induced by the app.

Firstly, all unnecessary software layers must be removed as each software layer adds latency. For that reason, sound processing code is written in C++ to avoid the JVM software layer. To simplify access to audio data, the Superpowered Audio SDK was leveraged which claims to be the “Fastest Mobile Audio Engine for Games, VR, Music and Interactive Audio Apps” (Superpowered Inc. 2017).

Audio latency induced be the app is solely caused by the fact that all operations performed during the processing of the sound take time. Firstly, some operations require the microphone to record several samples of audio before even being able to begin processing (like e. g. the Fourier transformations that are used to apply the equalizer). But other operations (like memory access) also take time. Hence, every operation must be evaluated and if any operation is found to be not necessary, it must be removed. All remaining operations must be designed in a way that optimizes the time it takes to execute them. For example, operations should use pointers to access memory to avoid unnecessary copies of objects.

Literature

Google LLC (2014): Material Design. Online verfügbar unter https://material.io/, zuletzt geprüft am 17.03.2018.

Kießling, Jürgen (2001): Strategien zur Anpassung von Hörgeräten. In: *EINBLICKE - Forschungsmagazin der Carl von Ossietzky Universität Oldenburg* 2001 (33). Online verfügbar unter http://www.presse.uni-oldenburg.de/einblicke/33/7kiessling.pdf, zuletzt geprüft am 20.01.2018.

Superpowered Inc. (2017): What Developers Can and Cannot Do to Lower Android Audio Latency. The Android Audio Low Latency Primer to Android’s 10ms Problem. Superpowered Blog. Online verfügbar unter http://superpowered.com/android-audio-low-latency-primer, zuletzt geprüft am 20.01.2018.

1. A software layer that emulates a virtual CPU and translates the instructions of the virtual CPU to instructions for the real hardware CPU. [↑](#footnote-ref-1)
2. Users won’t matter if their navigation app takes 5.0 seconds to calculate a route or 5.1 seconds [↑](#footnote-ref-2)