ulm university universität ulm

Faculty for Engineering, Computer Science and Psychology
Institute of Databases and Information Systems

Implementation and evaluation of a mobile web application for auditory stimulation of chronic Tinnitus patients

Bachelor thesis

Submitted by:
Fabian Henkel
fabian.henkel@uni-ulm.de

Reviewer:
Prof. Dr. Manfred Reichert
Supervisor:
Dipl.-Inf. Marc Schickler

2015

Abstract

Tinnitus is a prevalent disease that mainly states a big mystery to all kinds of scientific faculties and causes enormous costs due to further research. An initial assumption of the disease was the coherence of Tinnitus with a worse spatial hearing ability of the patient. With the assistance of mobile devices, it is the aim of this thesis to realize a mobile web application that allows it to draw conclusions that might support this theory in an easy available and ambulant way. The application is created as a game that has its focus on spatial hearing. The thesis depicts the used Application Programming Interfaces and names possible improvements.

Contents

1 Introduction

According to a research study from 2006, about 60% of Germans of all age were at least temporarily suffering under Tinnitus or deafness during their lifetimes. There are also studies concerning only young people: A survey under pupils brought up, that about 60% of 580 probands already had to deal with Tinnitus. Globally, the number of people suffering from Tinnitus ranges from 5% to 15% and most frequently can be determined in the older male population. Altogether, it can be said that Tinnitus is an omnipresent disease, with which almost everyone comes in touch [42, 25].

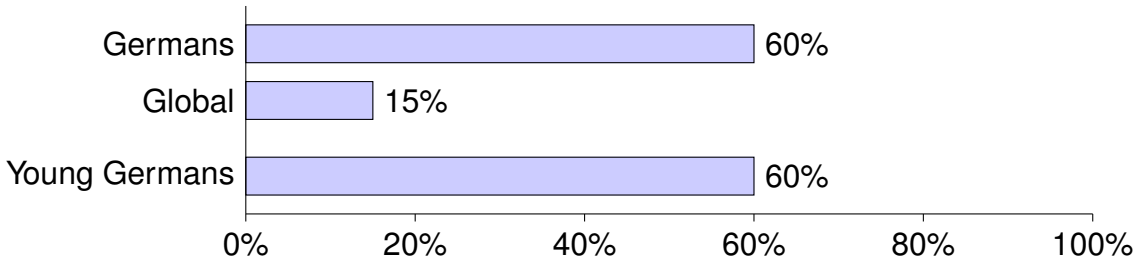


Figure 1.1: Tinnitus statistics: regarding young German study participants, Germans in general and the global number of diseased persons.

Temporarily or persistently appearing Tinnitus seems to be triggered for example by acoustic or workplace stress. Tinnitus can be classified into forms that are caused by other environmental causes or diseases, but there are still forms that do not have any perceptible reasons. Altogether, Tinnitus is mostly not sufficiently investigated and states an interdisciplinary challenge for all kinds of scientific faculties [42, 25].
As a result, this thesis should contribute to the investigation of this disease. This should be realized by the help of latest technical developments, the fact that almost every person can access a smartphone. Patients should now be able to use these devices to better draw conclusions about their diseases.

1.1 Motivation

This Bachelor thesis should realize a web application that gives the opportunity to examine the Tinnitus disease in a better way. As a coherence between spatial sound recognition and the Tinnitus disease was assumed in the initial meetings with psychologist Winfried Schlee, the main focus of this work is the playback of spatialized sound sources. The cooperation is caused by his brought knowledge in the Tinnitus research that lead among other things to the Track your Tinnitus project [21, 3, 36]. To reach the best possible coverage in realizing this application on all popular mobile platforms, there are three other Bachelor theses realized on the mobile platforms Android, iOS and Windows Phone. This should result in Tinnitus related studies that could operate in an ambulant way, without resulting in enormous costs caused by special experimental designs. The hypothesis of the coherence between Tinnitus and reduced spatial localization abilities should be supported by these works.

1.2 Aim of this thesis

The objective of this web application is to be a highly accessible possibility which can imitate spatial audio. Like presented in [41] about so called *"audio games"*, the idea is to create an application that mainly uses sound for the user-interaction. The screen should only have a minor role. To make the application interesting for both, children and older people, it was created like a safari game in which the user is a photographer. Throughout the game, various animal sounds appear auditory around the user who is wearing headphones to better recognize the spatial effect. The user rotates with the device until the animal appears to be right in front. Now the screen is touched to take a photo of the animal. With the elapsed time that was needed to find the supposed position of the animal, as well as the difference in the animal's and the user's orientation, it is intended to draw conclusions about the user's hearing abilities. This should reveal potential correlations between the Tinnitus disease and the hearing abilities.
As the final mobile application has to deal with limited technical resources (like slow network connections) but also has advanced technical possibilities (like the sensors for orientation detection) compared to stationary devices, this thesis shows up weaknesses of the used technologies, depicts the realized web application and compares the requirements with the reached. Despite of these challenges, modern mobile de-

vices have the necessary equipment to run new types of business applications that conquer the limited technical resources and make use of the possibilities [35].

1.3 Structure of this thesis

The further thesis is structured as follows: chapter 2 deals with the technical and psychophysiological backgrounds the whole thesis is based on. Chapter 3 names all the requirements that were set for the final web application. How the application got conclusively implemented is named in chapter 4. Some examples of usual interactions between the user and the application can be found in the subsequent chapter 5. A comparison of the requirements and the implementation is found in chapter 6. The survey that was performed is depicted and evaluated in chapter 7. Problems, their improvements as well as a final statement are given in chapter 8. After this follows chapter 9 which covers the acknowledgements.

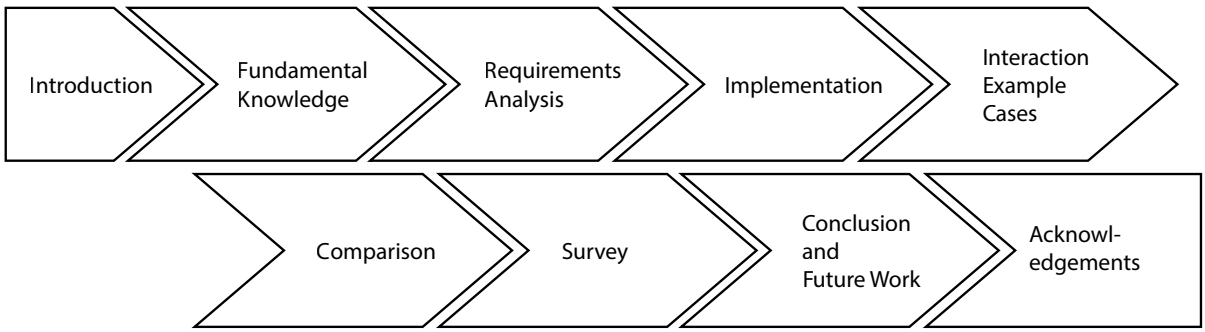


Figure 1.2: Structure of the thesis

2 Fundamental Knowledge

The following chapter deals with the background of the used technologies as well as the psychophysiological process of spatial hearing. The technical background refers to the highest possible accessibility of the application on mobile devices. As a result, the most popular operating systems and browsers are summarized.

2.1 Mobile Web Application

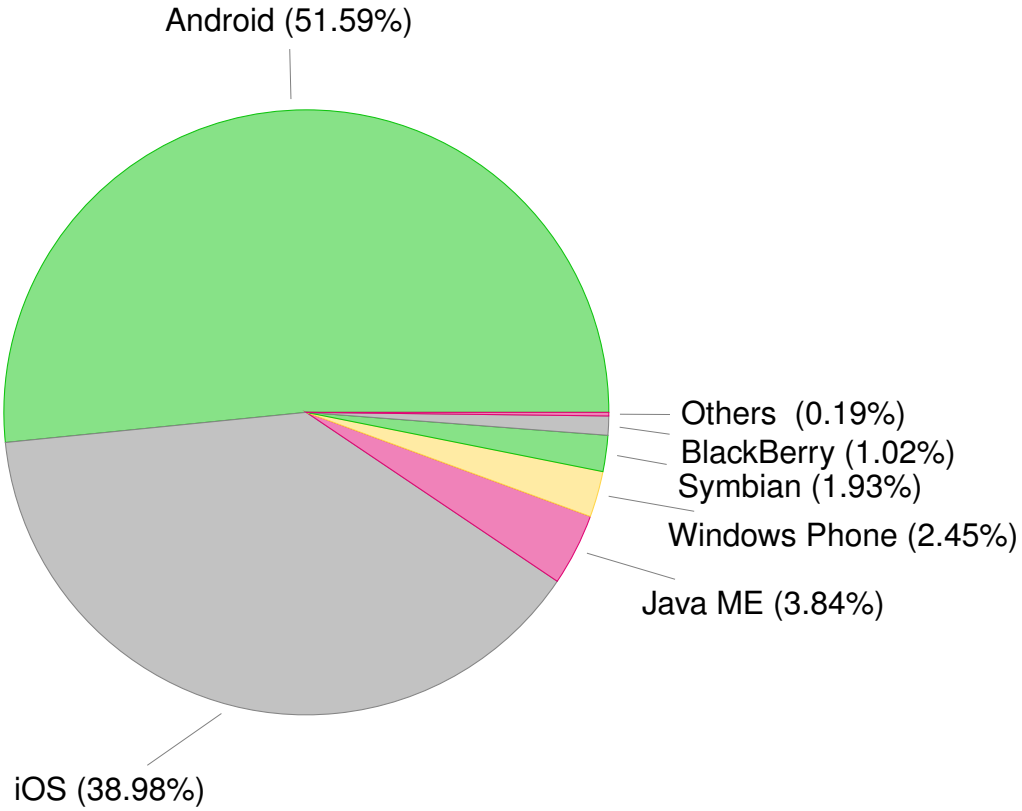


Figure 2.1: Mobile/ Tablet Operating System Market Share (May 2015) [31]

As the application should be accessible from a wide range of mobile devices, it was decided to create a solution that is not depending on the operating system of a mobile device. So, next to the applications for the popular operating systems Android, iOS and Windows Phone this web application should be realized in the scope of this work. As

shown in chart 2.1 about 94% of the globally used mobile devices run on the above mentioned operating systems. This application aims mainly at the remaining 6% of devices that cannot be supported by the other applications. In this context there arises the question, if these partly technical obsoleted devices are capable of dealing with the application and the used Application Programming Interfaces (APIs). In the conclusion there is given an outlook how also desktop devices could be supported.

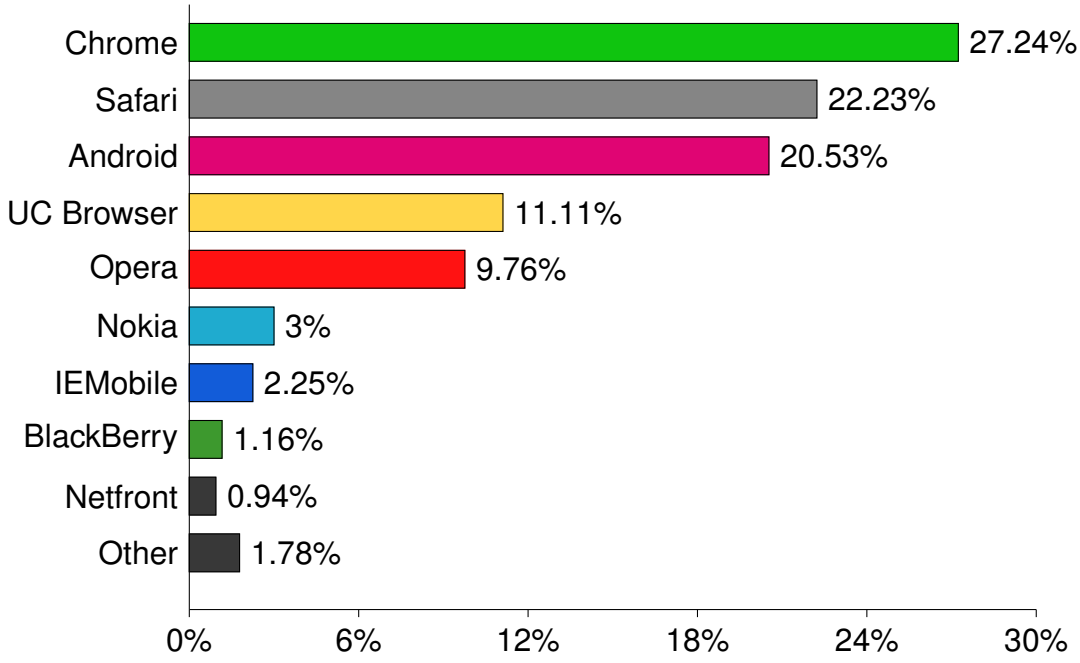


Figure 2.2: Mobile Browser Stats: Top 9 Mobile Browsers from May 2014 to May 2015 [19]

An analysis of the mobile browser distribution brought up, that most users are using the mobile Chrome browser, followed by Safari and the Android browser. As shown in chart 2.2, supporting these three browsers can cover 70% of the market share of mobile devices. Indeed, it is unclear if all of the devices that run a supported browser belong to the 6% of devices that cannot run a native application. Nevertheless, a better support could be reached and even users that are able to run native applications might prefer the web solution. In the implementation part of this work a further analysis of the available APIs of these browsers is presented. To reach a look that is optimized for touch inputs, the JavaScript framework jQuery mobile was used. Buttons and popups can be used due to this framework. This enables the creation of native-application-like user-interfaces [13].

2.2 Sound Localization

This paragraph should depict the process of sound localization against the psychophysiological background. The spatial recognition of sound sources located not directly in front of the listener's head (at 0°), depends mostly on the so called Interaural Time Difference (ITD). ITD is caused by a longer way that the acoustic noise has to take to one ear, if the sound source is not located directly in front of the listener. As the pinna and the head act like a filter for specific bandwidths, these bands are cued in different strengths. Being formed individually, every head has its own Head Related Transfer Function (HRTF) that describes the alteration of the sonic waves by the head and pinna. These different bandwidths are called Blauert's directional bands. The cue of these bandwidths, that is caused by the HRTF, is called Interaural Level Difference (ILD). As shown in figure 2.3, the polar angle α is described by the angle between the horizontal line and the line which connects the sound source and the listener's head, the lateral angle β describes the difference in the user's orientation and the orientation of the sound effect. The basic distinction within a lateral angle of -90° (left) and 90° (right) is performed mostly by the ITD. But even in this basic form of sound localization, the ILD plays an important role. The polar angle α as well as the differentiation if the sound source is located in front or behind the listener is fully ILD dependent [26, 5, 4].

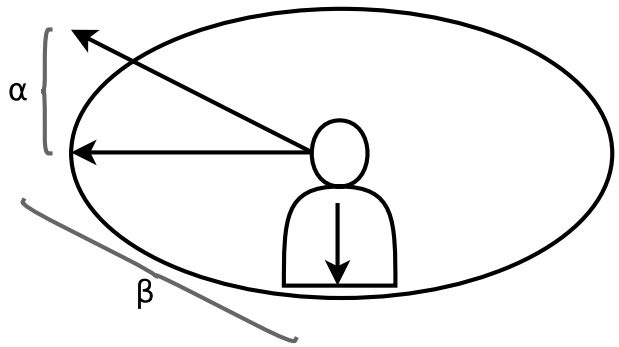


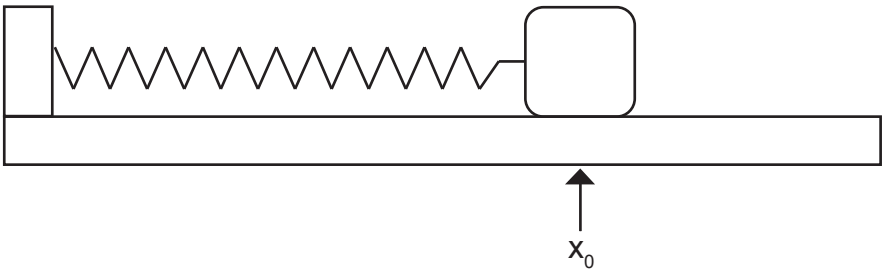
Figure 2.3: Auditory Angles

A very primitive form of sound localization can be achieved by simply cueing the audio volume on one ear. Here, the ear that is headed more towards the sound source is the one with the higher audio volume. The Blauert's directional bands are not taken into account. It is simply assumed that all of these bands are cued with the same intensity. A more advanced version of sound localization can be achieved by using band filters in order to manipulate Blauert's bands with different intensities. These ideas will be discussed in the implementations chapter of the Sound API.

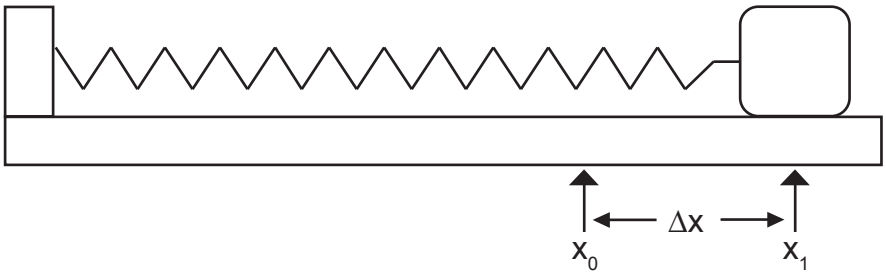
2.3 Device Position Sensors

Thinking of device position sensors may lead to beliefs of masses attached to a spring or compass needles that point northwards. But as we are in the 21st century, these imaginations have nothing in common with reality. The following three paragraphs will take a closer look at the distinctive device position sensors that almost every modern smartphone has build-in: accelerometer, gyroscope and magnetometer. These sensors are for example used by the Device Orientation API and are essential for this application feature.

2.3.1 Accelerometer



(a) ... with no acceleration.



(b) ... with acceleration.

Figure 2.5: A Basic Spring-Mass System Accelerometer

The Accelerometer measures roughly said the acceleration relative to free fall. This is realized along the three main axis X, Y and Z. It can be visualized by graphics 2.5. This basic accelerometer measures the deflection of the spring with the attached mass by measuring the δx value. If three of these are arranged parallel to the main axis, every change in orientation can be realized and measured. This makes it possible for the device to change for example from portrait to landscape mode when rotating the device.

Obviously there are no solid metal balls attached to springs inside the smartphone.

These Micro Electro Mechanical Systems (MEMS) are nowadays rather realized with a seismic mass made of silicon that swings in between the sensor which measures the deflection [27, 23, 18].

2.3.2 Gyroscope

A gyroscope by contrast is used to measure spins of the device. These cannot be measured very accurately by the accelerometer. The basic gyroscope works with a rotating disk that is not depending on any rotation axis. Due to the gyroscopic forces, the rotating disk is almost in the same state, after the device changed its orientation. These differences can be measured and so the corresponding orientation change is computed.
Mobile devices use a variation of this sensor that uses Piezo-elements which are oscillating. A modified device orientation results in measurable changes in the electromagnetic field that lead to a computation of the new orientation [27, 24].

2.3.3 Magnetometer

After gyroscope and accelerometer were introduced as sensors for determining the direction of movement of a device, the magnetometer is introduced. This can simply be compared to a compass that points northward.
Technically seen, this sensor measures the strength of the magnetic field of earth and adapts a coordinate system to it. So, this rather takes the orientation relative to the ground into account [27].

For computing the correct device orientation like in the Device Orientation API, all of these three sensors are used to approximate the real orientation and position of a mobile device.

3 Requirements Analysis

The following chapter should name and depict the requirements which emerged during the first phase of the meetings for the applications. They are mainly based upon recommendations and necessaries which were determined in initial conversations. Marc Schickler and Rüdiger Pryss of the Institute of Databases and Information Systems at the University of Ulm and psychologist Winfried Schlee took part together with the application team that was formed by students. Schickler, Pryss and Schlee could already raise their profile in previous Tinnitus-related works like the Track your Tinnitus project [21, 3]. As Tinnitus disease should be investigated in a better way, these requirements aim on certain aspects that were observed under Tinnitus patients. As named in the introduction, spatialized sound playback, device orientation and time measurement are the most important parameters to investigate.

3.1 Functional Requirements

This paragraph aims at introducing the functional basics the application should have to investigate Tinnitus in a better way. As the application should be realized as a game, playing back sounds in a spatial context, detecting orientation-change events, measuring the needed time and saving the acquired results are the main functional requirements.

3.1.1 Game

The application should have the structure and main functionality of a game application. There should be a main menu where basic settings can be made. These settings should concern the number of appearing sounds and whether there is a background sound. For the survey there should be a text input field to enter the current user ID and finally there should be a possibility to set the number of rounds to play. As described in the introduction, the actual game should mainly work with auditory signals rather than

visual ones. So, for each round that was set as number of rounds in the main menu, a defined number of sounds should appear in an auditory way. The user is wearing headphones to not be disturbed by any background noises and to accurately realize the auditory changes. Now the user has to rotate the device in a horizontal way in order to make the spawn points change their spatial positions. The effect should convey the illusion of real sound sources that stay on the same position, even if the user rotates. As soon as the users think that they are heading towards the currently searched spawn point, there should be a possibility to touch the screen in order to save the result. As a background task, the time between the spawn of the sound and the user-interaction should be measured. The user should receive a visual feedback in form of a rendered photo that shows the difference between the user's heading and the orientation of the spawn point. There should also be displayed the needed time, as well as a photo of the corresponding animal. After playing the predefined number of rounds, the user should be redirected back to the main menu.

3.1.2 Orientation Detection

As most devices have built-in orientation sensors (such as gyroscope, accelerometer or compass), it should be possible to use these sensors to determine a change in orientation. So, if the users hold the device in their hands and turn on the spot, this change in direction should result in events that should have an influence on the other game features. For example the spatial sound should be affected by these orientation-change events. It should also be possible to determine the difference between the point that the user is heading to and the real orientation of the spawn point. This feature should be powerful in a way that it allows real-time processing, besides be reliable and allow it to detect changes as accurate as to measure changes correct to one degree.

3.1.3 Spatial Sound

The spatialized sound feature should convey the illusion of randomly distributed sound sources that seem to stay on the same position, even if the users rotate with their devices. The perceived sound should make it possible to clearly determine the current localization of the spawn point. Together with the currently described orientation-detection, an orientation-change event should result in real-time changes of the sound localization. Wearing headphones, the users should be able to adjust their orientation

by rotating the devices. This should allow them to make orientation adjustments until they point towards the supposed target direction.

3.1.4 Time Measurement

As it is an important issue for the investigation of Tinnitus to know more about potentially occuring problems in sound localization, time measurement should also be present. As a result, it may be possible to draw conclusions about a relation between the disease and a worse spatial sense of hearing. In this context, time measurement should start every round at the spawn of a new sound effect. The user-interaction should state the end of every measurement. These values should be correct to one decimal place.

3.1.5 Saving Results

All the acquired data should be saved reliable in a persistent way. This saved data should be accessible at every point of time. The data should be stored in a standardized format like the comma-separated values (CSV) format [39].

3.2 Non-Functional Requirements

The non-functional requirements are depicted in this paragraph. As the main focus of this application lies on the real-time processing of sound, these requirements mostly concern the performance of different game modules. Other basic requirements of an application with user-interaction like user-feedback or a non-freezing screen are also named.

3.2.1 Performance

The performance requirements refer to the main application on the one hand and the modules related to the sound-localization and orientation-detection on the other hand. This means in particular that the application should have an interface that reacts within a decent time without freezing at any point of time. Additionally the sound-localization and orientation-detection modules have to be able to process the sound in real-time.

No perceivable delay should be present. Resources should be loadable within an acceptable time, depending on the current network connection.

3.2.2 Feedback

In general, the users should get a feedback for every action they perform.
As the users are playing a safari game when using this application, they should receive a visual feedback of the animal they took a photo of. This should happen in form of a popup that shows the animal, the difference between the target and the user's heading and finally the time that was needed to take the photo. At the end of a round, a conclusion of the acquired data should be shown. As well as the visual feedback, auditory feedback should be given to the user while rotating.

3.2.3 Accuracy

Next to the time that the user needed to take a photo of the animal, the difference in direction should be measured in an accurate way. To be more specific: It should be possible to measure the time correct to 0.1 seconds and the differences in direction correct to one degree.

3.2.4 Availability

As named in the introduction, the application should be a highly accessible web application. Optimally, it should cover all of these devices, which are not natively supported by the mobile platforms Android, iOS and Windows Phone. So, the most possible accessibility should be reached. For that purpose, it is necessary that the used APIs are supported by a wide range of browsers. It is also important, that the application can be used at every point of time.

4 Implementation

The following chapter should give an overview of the implementation of the final application. In addition to that, ideas that came up during the development-process as well as improvements and their causes will be demonstrated. In the first section the general coding paradigm, the development pattern and the project architecture are depicted. The architecture is focusing only the most important components of each module. This is followed by a section for the server-side as well as the client-side implementation. Both depicts the used frameworks and APIs. The application is available on `http://soda.dbisuulm.de/`. For the development-process, an Android phone (version 5.0.2) with the mobile Chrome browser (version 45) was used.
The browser support of the used APIs is rated as can be seen in table 4.1

Support Levels		
Full feature support	Only partly supported	Not supported

Table 4.1: Support Levels Explanation: color code for the 'Can I Use'-tables for certain APIs

4.1 General

The next paragraph names the used programming-paradigm and design pattern for the application structure. These standards were followed as far as possible to create an easy to use framework that can be maintained and extended in an uncomplicated way.

4.1.1 Coding Standards

The implementation follows the SOLID principles (see table 4.2) and is designed, as far as possible, with the MVC pattern (see table 4.3). As a result, the JavaScript, that is responsible for the functionality of the application, is implemented in an object-oriented

5 SOLID Principles	
Single Responsibility Principle	A class should have one, and only one, reason to change.
Open Closed Principle	You should be able to extend a classes behavior, without modifying it.
Liskov Substitution Principle	Derived classes must be substitutable for their base classes.
Interface Segregation Principle	Make fine grained interfaces that are client specific.
Dependency Inversion Principle	Depend on abstractions, not on concretions.

Table 4.2: The Five SOLID Principles [30]. Robert C. Martin describes how the object-oriented class design should look like in these five SOLID principles.

manner. Based on an article about writing efficient and fast Javascript [32], it was always tried to write the client-side Javascript code in the most efficient way.

By using the SOLID principles and the MVC pattern, a maintainable and extendable application could be created.

MVC Pattern	
Model	Contains the data and logic of the application.
View	Responsible for user-interactions and displaying the content provided by the model
Controller	Acts as a link between the Model and the View.

Table 4.3: The MVC Pattern [34]. The MVC pattern describes how the written code can be improved by making it more editable and extendable.

4.1.2 Main Architecture and the role of JavaScript

As mentioned before, the application is fully client-side and only uses the server to fetch resources, which can be seen in figure 4.1. Because of the used programming

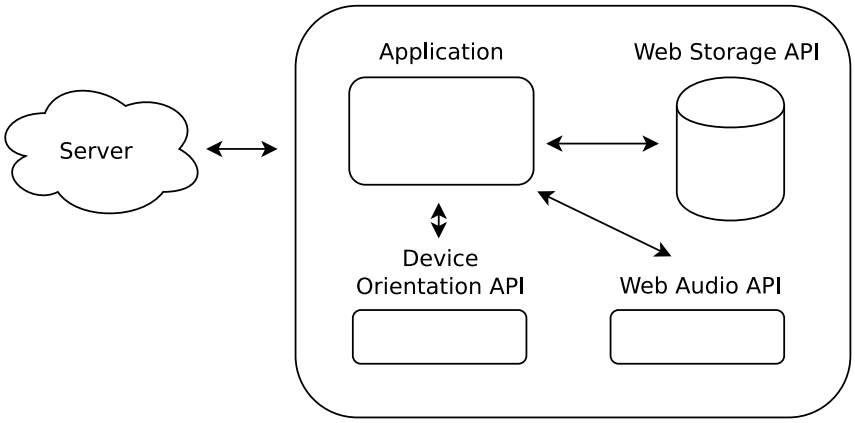


Figure 4.1: Main Architecture

language being JavaScript, the browser states the programming environment. The figure above shows basically, how the APIs, the application and the server interact.
With exceptions, ECMAScript 5 (better known as JavaScript) can be interpreted by most browsers [10] and thus is the perfect possibility to provide client-side functionality. First of all, one of the advantages of a script language like JavaScript is that the interpretation of the code happens at run-time. Additional time for compiling is not necessary at all. There is also a dynamic type system and a prototype-based object model which allows very easy and flexible programming [16, 29].
As the implementation is event-based, there is an object ActionHandler that is a centralized controller for the performed actions of the application. For that purpose, the ActionHandler communicates mainly with the Web Storage API [22], the Web Audio API [2] and the Device Orientation API [6].
In this connection the Web Storage API is responsible for saving and retrieving the acquired data of the game or of the settings-menu. The Device Orientation API is incorporated when changes in the devices orientation are detected or the current orientation is needed. The Web Audio API is responsible for the spatialized playback of the spawning animal sounds.

4.1.3 Code Structure

The following paragraph takes a closer look at the structure of the main application part. Figure 4.1 only shows how the application interacts with the used APIs. Now, the modules that use the APIs are observed more closely. The most important objects of the modules are shown in figure 4.2.

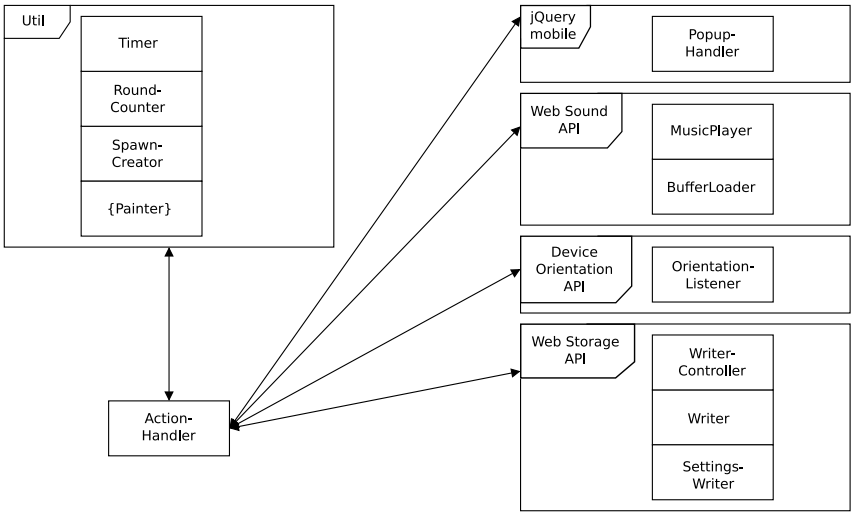


Figure 4.2: Code Structure

An important module that is mostly essential for the application-like view of the website is the jQuery mobile module. Especially the popup feature should be observed in the examples part. Another module that is indispensable for the smooth operation of the application is the Utils module. All objects that perform background tasks or provide other information for other objects are located here. The objects that underlie the modules jQuery mobile, Web Storage, Web Audio and Web Orientation are further depicted in the implementation parts of the APIs or frameworks they belong to.

4.2 Server-Side Implementation

As it was decided to store all the survey data directly on the device, the server can get along without a database. How this decision came up is discussed in the implementation part of the Web Storage API.

As the server was implemented, the only requirements that came up for it were routing and a possibility to determine if the device is a mobile one or not. The Apache server can check for mobile devices out of the box by altering the RewriteRules in the .htaccess file. In the same manner, all incoming requests could be mapped to a single index.php file where these requests are processed and mapped to the corresponding resources.

But it was decided to use the Slim micro-framework [28] for the routing purpose, as well as the mobiledetect framework [17] for the mobile devices detection because it seemed more flexible and comfortable to use.

The first idea was to make the website also available for users of non-mobile devices.

Because it could not be realized within the limitations of this work, this feature is not available now. This is discussed in the conclusion part of this work. As a result, visitors of the website that use a device that is not detected as a mobile one are redirected to an error page.

4.3 Client-Side Implementation

It follows a section about the used APIs and frameworks which made the client-side functionality possible. All of them were mainly chosen according to a device support as high as possible. So, the application should cover most of the devices that are not supported by the similar theses for the mobile platforms Android, iOS and Windows Phone.

4.3.1 jQuery Mobile

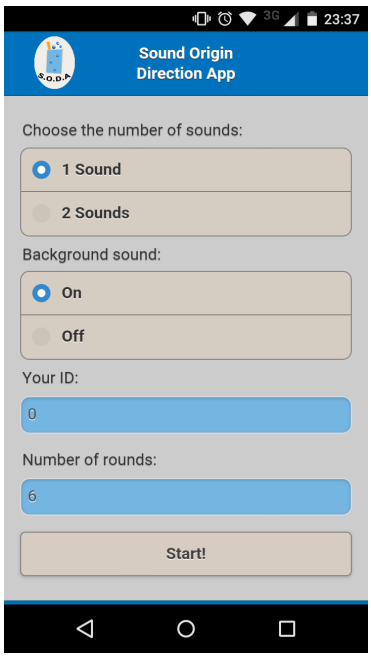


Figure 4.3: Screenshot, displaying the main-menu

The ability to provide a real cross-platform application is given with the jQuery mobile framework. After its browser support [14], there are only few mobile browsers that are not fully supported. According to that a more precise statistics of mobile browsers [8] than the one mentioned in the introduction part of this work, less than 1% of the browsers are not rated grade 'A' in the jQuery mobile browser support. This means

they support every feature. A part of these non-grade-'A'-rated browsers are rated grade 'B', which means, they do not provide Asynchronous JavaScript and XML requests (Ajax-requests) [20].
Apart of these deprecated devices, it could be assumed that, at least theoretically, about 99% of the mobile devices could use this framework without deduction. The framework was mainly used to create a user-interface that looks like a native application and not like a website. The framework offers a broad range of user-interface-components that are designed for the touch-based user-interaction: for instance buttons, input fields and more general - pages with sliding transitions. Figure 4.3 displays for example, how these elements look like in the implemented version. As can be seen in figure 4.4, popups were used to give the users feedback of achievements they made. The framework does not only supply a good-looking front end. In fact, interesting features such as preloading content were used as well.

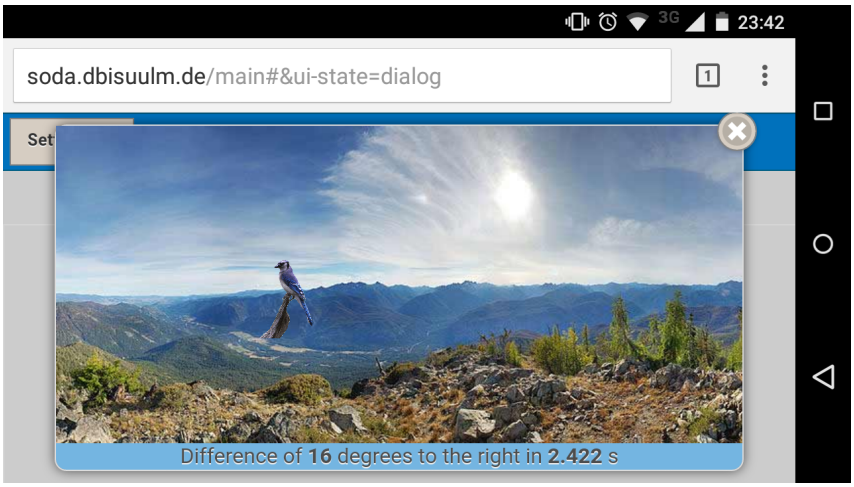


Figure 4.4: Screenshot, displaying a popup with the results of the user-interaction

4.3.2 Web Audio API

This paragraph depicts the development-process of one of the main application modules, namely the sound module [2]. As mentioned earlier, this module is responsible for the illusion of spatialized sound. Before using the Web Audio API, it was the initial idea to create the spatial character by using the ILD and ITD, which where mentioned in the introduction chapter. For this purpose, the stereo sounds were converted to mono sounds with the audio software Audacity in order to reduce potential spatial impressions. First of all, a version of the application was created that calculated a audio

volume cue for the ear that is averted from the sound source. It was simply assumed that each of Blauert's directional bands is cued with the same intensity.

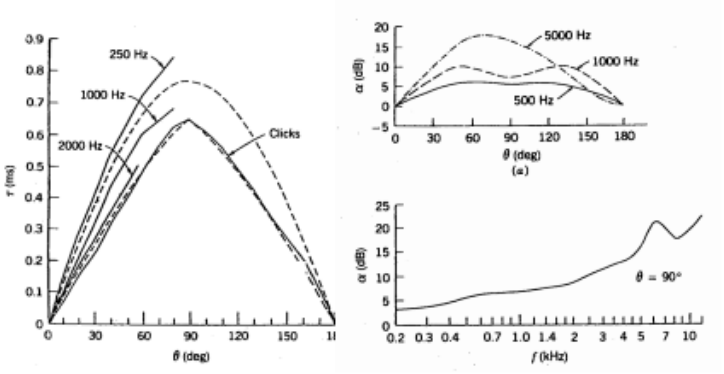


Figure 4.5: ITD (in ms) as a function of the lateral angle for sinusoids of three frequencies and clicks ([26] on page 24, figure 3)

The values of the ITD provided in figure 4.5 show, that the maximum delay on one ear may be at about 0.8 ms if the sound source is at an angle of 90°. These small values could not be reproduced within this primitive reproduction. The approached function that should simulate the hyperbolic cueing level of the ear that is aversed from the sound source can be seen in figure 4.6. A big problem of this approach is that the user is not able to differ if the sound comes rather from the front (0° to 90°) or back (90° to 180°) direction. This is because a sound source located at e.g. 30° to the user's right, technically is the same as a sound located at 150°. A differentiation is only possible after the user realized that a rotation of the device has another result as suspected. Undoubtedly, this problem is not acceptable. Next to this temporarily insoluble problem, there was also trouble with the performance. The calculation in real-time was not possible at all with this version of the application. Nevertheless, this primitive approach imitated the spatial sound in an amazingly good way.

The problems with the approach from above lead to a second idea: The usage of a HRTF for the individual manipulation of specific bands [26] had to be implemented in a performant way. These so called Blauert's directional bands can be manipulated by the implementation of bandpass filters, assuring variously dampened frequency bands. As it was mentioned in the introduction chapter, every head is different and so the usage of averaged values for the HRTF is necessary. Due to the fact that there are various libraries or APIs that give the possibility to play spatialized audio on the web, it was decided to not implement it manually, because it would have gone beyond the limits of

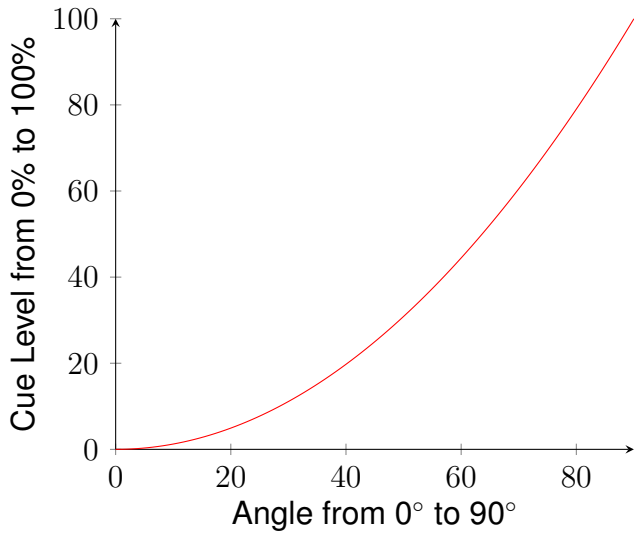


Figure 4.6: Cueing level for the ear aversed from the sound source with the approached function $f(x) = x^2/81$

this work. Instead, different possibilities as the WebAL library [43] or the Web Audio API [2] were compared. Although, an analysis of the possible usage of the Web Audio API (see table 4.4) brought disillusioning results, it was the best choice to make. The API itself works on a lower level, uses Assembly, C or C++ code, according to the browser implementation. The API is very performant but it must not be forgotten that the API still is in the development process and no standard yet.

IOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile
8.4	8	44	10	30
Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	
44	40	11	9.9	

Table 4.4: Can I Use Web Audio API: supported browsers [11]

The Web Audio API has a graph-based schema that works within an AudioContext. This interface holds a list of AudioNodes and their connections. Each of the nodes performs a specific processing step with the audio data. The most minimal version can be implemented by initializing and connecting the SourceNode and the DestinationNode. In every case, the SourceNode receives the audio data by an AudioBuffer object that should only be used for sounds no longer than one minute. By contrast, the Des-

tinationNode can be seen as a hardware-connected node that outputs the sound as the user finally perceives it. For adding additional effects to the sound that is provided by the SourceNode, the corresponding node simply has to be inserted in the graph between the Source- and DestinationNodes. A preceding node is effected by the subsequent ones. In this application, especially the PannerNode and GainNode will be observed [2].

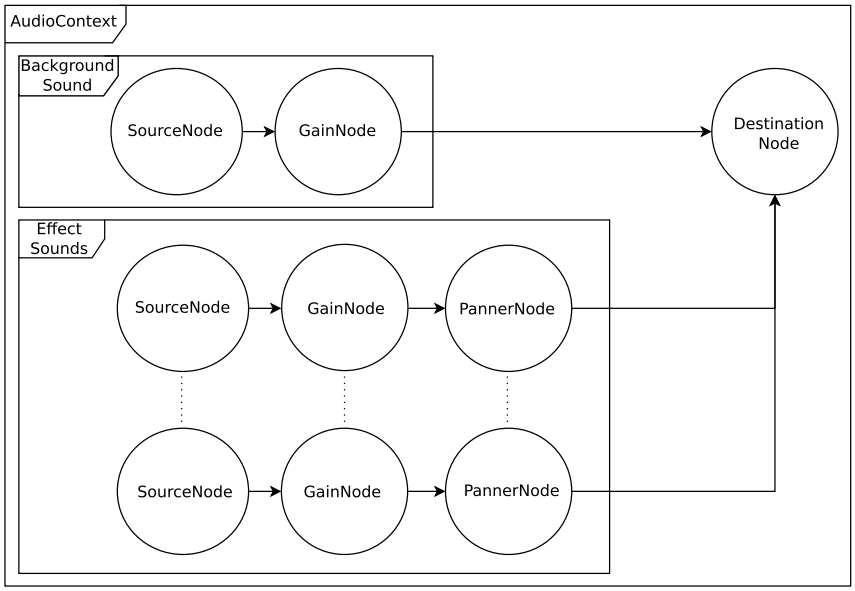


Figure 4.7: Web Audio API Structure

As shown in figure 4.7, there are SourceNodes for every sound file that has to be loaded. The approach of loading the sound files in an asynchronous way follows the idea of [40]. Thereafter the Ajax-requests for loading the audio files are performed within BufferLoader objects. There are two paths that are taken on the path between SourceNode and DestinationNode: The background sound only needs the functionality to gain the whole audio volume. For that purpose, only a GainNode was created on that path. As opposed to this, every sound effect needs both the possibility to gain the whole sound level and to spatialize the sound. As a result, a GainNode and a PannerNode were created for every sound effect. A PannerNode works with the information where the Listener object and the sound source are located in relation to each other. For that purpose, the position and orientation of the Panner and the Listener can be configured by simply giving them 3D-coordinates respectively the direction of the normal. The Listener is the equivalent of the DestinationNode that has to be positioned on the AudioContext. To achieve the desired results, the sound sources move on a circular orbit around the static Listener if a rotation-action takes place. The sound that the user hears has a specific sound level and imitates the spatial character that

the application wants to achieve.
How this works in action will be shown in the examples chapter.

4.3.3 Device Orientation API

Next to the Web Audio API, the orientation module of this application plays a significant role for the success. In the beginning it was clear that a compass- or gyroscope-based solution would only make the application available for mobile devices. Although the website can be accessed from each browser (that supports the used APIs), the mobile version is in the foreground. A possible desktop version is discussed in the conclusion. As shown in table 4.5, the browser support for this API can be considered very good. The reason why most browsers are only partially supported is the missing 'compass-needscalibration' event that is fired, when the compass is in need of calibration.

IOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile
9	8	44	10	30
Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	
44	40	11	9.9	

Table 4.5: Can I Use Device Orientation API: supported browsers [9]

It is recommended by the W3C, that the "common sources of information include gyroscopes, compasses and accelerometers" [6]. So due to the API, the high-level JavaScipt does not depend on any raw sensor data at all. Events that are fired by the browser can be received by adding an EventListener to the current browser window (represented by the 'window' object). The EventListener has to watch for 'deviceorientation' events to be able to get the alpha-, beta- and gamma-properties of this event. An alpha-event is equal to a rotation around the Z-axis, a beta-event means a rotation around the X-axis and a gamma-event equals a rotation around the Y-axis. The device is flipped 90° to the left to be in landscape-mode.

In the beginning, it was assumed that the device-orientation can simply be achieved by getting the alpha-property of the current orientation-event. But as this only works for a device laying flat on a table when being rotated, it had to be assumed that both the

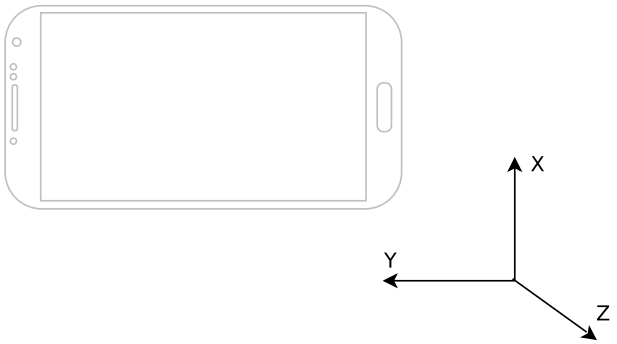


Figure 4.8: Coordinate system of the Device Orientation API flipped by 90° to the left to be in landscape-mode

beta- and gamma-values are not equal zero. Taking into account that the device can also be tilted towards the user or sideways, the following trigonometric formula has to be used to calculate the heading of the device [6]:

$$\Theta = \tan^{-1}\left(\frac{-\cos\alpha \cdot \sin\gamma - \sin\alpha \cdot \sin\beta \cdot \cos\gamma}{-\sin\alpha \cdot \sin\gamma + \cos\alpha \cdot \sin\beta \cdot \cos\gamma}\right) \tag{4.1}$$

For performance reasons, there is a buffer that was implemented to only process the event data if it is above a certain threshold. This may be improved in a further way but is discussed in the improvement paragraph. In the beginning of the implementation-period, the Painter object was used to draw a dot in north direction of the user in order to allow easier debugging concerning the device-rotation. This was dismissed later. In the comparison chapter will be discussed why the compass lost reliability during the survey.

4.3.4 Web Storage API

As the decision came up how to save the data, the first idea was to create a RESTful web service [7] on the server in order to save game results. Due to the use of the Slim framework, the implementation of this would not have been a big deal. But on the one hand it was decided to create a fully client-side application, on the other hand the question raised, which underlying conditions had to be met to save the user data legally. The server-sided implementation of a RESTful interface would also have raised the need for a web interface to retrieve the acquired data that would have to be saved in a database. At that point of time, the game settings like the number of sound effects were saved with the non-persistent SessionStorage object of the WebStorage API [22].

The saved key-value-pairs usually exist for one session. In other words, as long as the browser window is opened.

IOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile
9	8	44	10	30
Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	
44	40	11	9.9	

Table 4.6: Can I Use Web Storage API: supported browsers [12]

The API has an excellent mobile browser support, as seen in table 4.6. Next to the non-persistent SessionStorage, there is a more persistent way to save data in key-value-pairs within the used API. The LocalStorage object allows to save the data in a persistent way on the client's device until the browser's cache is cleared. In extraordinary cases the Session- as well as the LocalStorage objects can be flushed by the user agent itself. This depends on the actual implementation and the state of the user's device. But this should only happen if the browser exceeds the available space and requires all resources.
In the Chrome browser (version 45), that was used on the test device (with operating system Android 5.0.2), the WebStorage API has a fixed space of 5 MB to use. Considering that a single char needs 8 bits, this space would suffice for about 6200 different datasets of the three test-cases that were used in the survey: three times one single sound with background sound, three times two sounds with background sound and three times only two sounds without background sound. It has to be mentioned, that the format of the saved datasets can be adapted and for that purpose could be relieved of many unnecessary chars.

5 Interaction Example Cases

It is intended to show some basic actions which appear very often during the usage of the application: a spawn event, an orientation-change event and an user-interaction event. The most important objects that are involved in these actions should be depicted. The central character of these actions is the ActionHandler object. As the application follows the MVC-pattern, this class is the controller that is responsible for the performed actions. The three examples show the connection that the classes have against each other.

5.1 Spawn Event

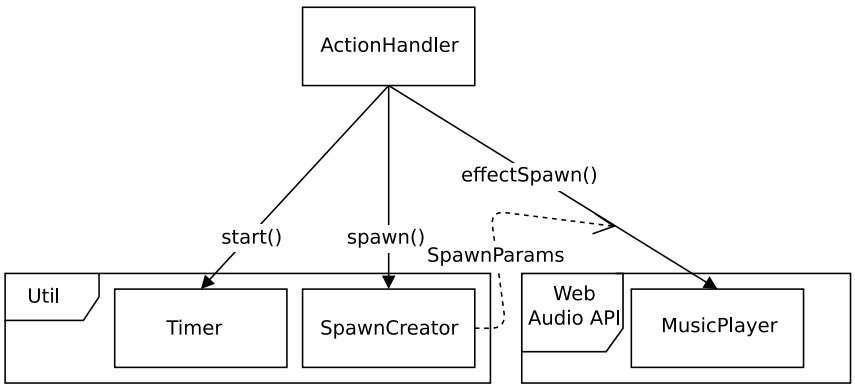


Figure 5.1: Spawn

This action occurs every time a new sound effect should spawn in a spatialized way. This happens for the first time, when the MusicPlayer finished loading the sound files asynchronously and is ready for playback. Every time a game round finished and the user starts a new round, this action takes place again. Due to the ability of stopping and restarting the background music from the in-game settings drawer, all the sounds have to be loaded completely when the MusicPlayer is initialized. As soon as these sound-files are available on the client's device, the ActionHandler is informed about the successful transfer. The ActionHandler now checks, how many rounds remain until the current game is finished.

In case of the first played round, the ActionHandler now has to instantiate the Timer objects and needs to create the random spawn parameters for each sound effect. With the help of this set of parameters, the MusicPlayer creates the corresponding sound nodes and starts with the playback. Additionally, a message which sound to look for is displayed. The spawn parameters contain the spatial position of the sound effect and its index of the sound-buffer- and sound-name-array. As mentioned in the implementation part of this work, the WebAudio API uses 3D-coordinates to position the listener and the sound source. Based on the agreement that the sound effects always have the same elevation, the sounds only move on a circular orbit around the listener.
If it is not the first round, the playback of the current first sound effect as well as the timer for that sound start. Here it has to be mentioned that in the beginning of the development-process the idea was to make the sound effects disappear after a randomly generated time. So, certain difficulty levels could be created. This plan was dismissed at a later point of time.

5.2 Orientation-Change Event

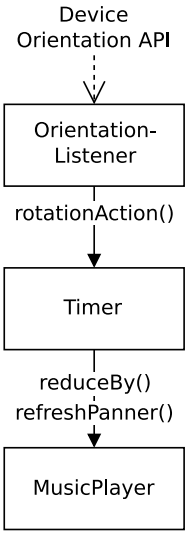


Figure 5.2: Orientation Change

As already mentioned in the implementation part of the Device Orientation API, on start of the main application page an EventListener is attached to the current browser window to listen for 'devicemotion' events. If the device orientation changes around the X-, Y- or Z-axis, such an event is fired. For performance reasons, a buffer was implemented to detect only changes above a certain threshold. First of all, the ActionHandler calculates the difference in direction that was caused by the rotation. As the

effects move along a circular orbit around the Listener, it has to be calculated where the new positions of the sound effects are. These new positions are updated in the spawn parameters of each effect. As the sound sources can also be behind the listener, the sound effects are gained if they are behind the user's field of view (> 90° or < -90°). A simulation by using the orientation of the PannerNode and the AudioContext listener did not bring the desired effect. The sounds did not seem to have changed their positions. As a last step, the MusicPlayer uses the reduction percentage and new PannerNode positions to update these sound nodes. The reduced audio volume effects the GainNode that is connected intermediate between the SourceNode and the PannerNode (Figure 4.7). The new coordinates on the circular orbit around the listener effect the position of the PannerNode. In doing so, the illusion of a spatially positioned sound effect, that stays on the same position even if the user rotates with the device, is created.

5.3 User-Interaction Event

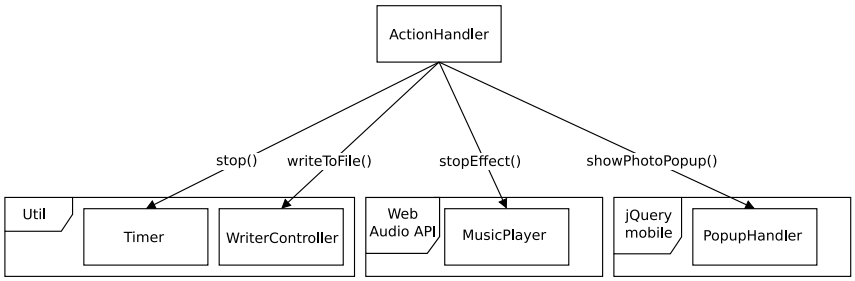


Figure 5.3: User Interaction

This action presupposes the playback of one or more sound effects. This implicates that the spawn event already took place and as a result the timer is running and the OrientationListener listens for rotation events that effect the position of the PannerNodes. The users rotate with their devices, until the sound source they are searching for appears to be right in front of them. The playback of the sound stops as well as the Timer object does. The jQuery mobile popup-widget is used to display the result of the currently searched sound effect. Depending on the number of remaining rounds, that are determined by a RoundCounter object, a decision, which actions take place after the popup is closed, is made. As a first step, it is checked if this sound effect is the last within this round. If it is, an overview-popup of this round is shown. Closing this new popup results in starting a new round or in finishing the game by redirecting to another

page. A RoundCounter object determines, if there are remaining rounds to play. If the sound was not the last within this round, another spawn action takes place. The popup that shows the result for each sound effect displays the difference angle between the orientation of the device and the orientation the sound source really has. As the Timer object stops when this action happens, it is also possible to determine and display the time that elapsed from the spawn to the user-interaction. The majority of the popup surface is next to texts containing the results filled with a rendered image of the virtual photo the user took. Based on the following formula, the angle that a standard photo covers horizontally was determined [1].

$$\alpha_{visual} = 2 \cdot \tan^{-1}(\frac{d}{2 \cdot f}), \tag{5.1}$$

where d is the diagonal and f is the focal distance. According to the focal distance of the used lens, a visual angle of between 62° and 40° can be reached (when using a standard lens with a focal distance of 36-60 mm).
By simplifying the formula with the assumption that $d = f$, the following visual angle is reached:

$$\alpha_{visual\_simplified} = 2 \cdot \tan^{-1}(\frac{1}{2}) \approx 53^{\circ} \tag{5.2}$$

So the image consists of a background image and the image of the corresponding animal that belongs to the sound effect in the foreground. The rendered animal image is only visible within the determined angle of 53° and according to the difference angle shifted to the left or right. The shifting level depends on the absolute value of the difference angle. The overview popup that is visible at the end of each round contains a list of the differences and the needed time for each sound effect that was searched.

6 Comparison

The next paragraphs compare the list of requirements that was established in the requirements chapter with the finally implemented features. According to the requirements, this chapter is segmented into functional and non-functional requirements and their corresponding subsections. The rating is described in table 6.1.

Ratings (from requirements not met at all to fully met)				
[–]	[-]	[o]	[+]	[++]

Table 6.1: Rating Explanation

6.1 Functional Requirements

All the functional requirements could be implemented. The way they meet the requirements will be shown in the following paragraphs.

6.1.1 Game

Requirement	Comparison	Rating
Game-like application with main-menu and mostly auditory signals for the user. Furthermore, time measurement and orientation-differences should be determined. There must be visual feedback as well as a round-based game play.	Requirements fully met.	[++]

Table 6.2: Game Requirements Comparison

It was the requirement to create an application with the structure and the functionality of a game. More precisely, there must be a main-menu where to alter the game-settings,

the game should mostly work with auditory signals and give the users a visual feedback with the reached result if they interact. Thereby, the needed time and the differences between user-orientation and sound effect should be displayed. The game should work in a round-based manner.
The requirements could be fully met.

6.1.2 Orientation Detection

Requirement	Comparison	Rating
Detect real-time device-orientation change events in an accurate and reliable way.	The API is not reliable at all points of time. In all other aspects, fully implemented.	[+]

Table 6.3: Orientation Detection Requirements Comparison

The web application must be able to use the built-in sensors to determine changes in orientation. This should work in real-time and be as accurate as to measure changes correct to one degree. This should be done in a reliable way.
The requirements could almost be met. Through the used Device Orientation API, a real-time orientation change detection could be realized. The final implementation works really accurate and allows it to control other application modules. In the survey, it was found, that the application did not work as expected after a bigger count of played rounds. As a reboot of the browser fixed the problem for some time, it was assumed that the problems occured either because of issues within the hardware or because of issues with this or the Web Audio API.
It could not be determined, where the problems came from. Unfortunately, the 'compassneedscalibration' event is not supported by all browsers. So it could also not be determined if the lack of calibration caused the issues.

6.1.3 Spatial Sound

It must be able for the user to clearly determine the position of each spatialized sound effect. These must be randomly distributed around the user. A change in direction, which the users perform with their devices, must result in the illusion of each sound source still being at the same position. This module must be able to receive controls of

Requirement	Comparison	Rating
Spatialize sound effects and be able to receive and process changes in orientation.	API is not standardized yet and as a result not reliable. Reached all requirements.	[+]

Table 6.4: Spatial Sound Requirements Comparison

the orientation detection module, for instance.
In the survey after a big number of played rounds, a orientation change did not result in a changing auditory position of the effect sound. Like mentioned in the Device Orientation comparison, it could not be determined where the problem came from. Besides, the requirements were fully met.

6.1.4 Time Measurement

Requirement	Comparison	Rating
Measure time from spawn to user-interaction correct to one decimal place.	Reached all requirements.	[++]

Table 6.5: Time Measurement Requirements Comparison

The time measurement starts with the spawn and ends at the point of time, the user interacts. The measured time is correct to 0.1 s.

6.1.5 Saving Results

Requirement	Comparison	Rating
Save data reliable and persistent in the CSV format so that it is available at every point of time.	Possible loss of data if the browser flushes the storage.	[o]

Table 6.6: Saving Results Requirements Comparison

The requirements for the storage of the acquired data were, that the data is saved in a standardized format in a reliable and persistent way. An access must be possible at every point of time.

The data is saved in a quite persistent and reliable way on the user's device. Data loss should only happen in extraordinary cases, when the browser flushes the cache. Else it is accessible until it is manually deleted by the user. As the storage format can be adapted, it can fit every format requirements.

6.2 Non-Functional Requirements

In the broadest sense, the non-functional requirements could be met. The application allows a normal usage with loading times within an acceptable range of time, feedback on user-interactions and real-time changes in the behaviour of the application. The single aspects from the requirements chapter are picked up and compared with the implementation.

6.2.1 Performance

Requirement	Comparison	Rating
Fast-loading and real-time reacting application with modules.	Loading times could partly be improved. Despite, fully met requirements.	[+]

Table 6.7: Performance Requirements Comparison

The whole application as well as the modules must react within an acceptable range of time. Feedback must be given in real-time and the screen is not allowed to freeze at any point of time. In particular the sound-localization and orientation-detection should work in real-time without any perceivable delay.
The loading times could be improved, all other requirements are totally met. Improvement possibilities are named in the conclusion chapter.

6.2.2 Feedback

The users must receive a visual or auditory feedback after each interaction. Especially the visual feedback during the game must contain an image of the animal that was searched for. Additionally, the needed time as well as the difference in orientation between the user's heading and the heading of the sound effect must be visible. An

Requirement	Comparison	Rating
Visual and auditory feedback for user-interactions.	Reached all requirements.	[++]

Table 6.8: Feedback Requirements Comparison

overview of the results must be visible at the end of each round. Rotations must result in changes of the sound.
All requirements could be met.

6.2.3 Accuracy

Requirement	Comparison	Rating
Measure time correct to one decimal place and differences in direction correct to one degree.	Reached all requirements.	[++]

Table 6.9: Accuracy Requirements Comparison

The time measurement from the spawn of an effect until the user-interaction must be accurate to one decimal place. Orientation changes must be measured correct to one degree.
All requirements could be met.

6.2.4 Availability

Requirement	Comparison	Rating
Be available at every point of time for a maximum of devices.	Browser support for some used APIs are not that good. The server is available if mobile device is connected to the internet.	[o]

Table 6.10: Availability Requirements Comparison

The application must be useable at every point of time if the device has a connection to the internet. The support of devices must be as high as possible.

Some of the used APIs support a wide range of browsers, whereas others are in a development state and are not standardized yet. The server is available all the time if the mobile device is connected to the internet.

7 Survey

Within the borders of this thesis, it was also possible to carry out a survey. This survey should depict how the web application, which was realized within this thesis, as well as the applications for the other mobile platforms behave in a survey situation. Additionally, it should be possible to see differences between the single platforms or normal and Tinnitus-diseased participants.
For this purpose, a participant should use all the applications, one after the other. As this is not a clinical study in general that should only draw conclusions about the Tinnitus disease, it was no requirement for the survey participants to be Tinnitus-diseased. Despite, two of the participants were suffering from Tinnitus and one other suffered from a smaller auditory system related disease (multiple ear infections).

7.1 Survey Design

The initial development processes of the different mobile applications brought up that the used interfaces for the orientation detection were highly sensitive for disturbance sources. The test applications for the orientation detection revealed enormous disturbances even if the device lies flat on the table. To reduce these electromagnetic disturbances to a minimum, the survey was performed in the open air. To reduce background noises and improve the spatial hearing, it was necessary wear headphones during the survey.
The actual survey was divided into a questionnaire that had to be completed by the participant and the use of the application on each mobile platform for a defined number of rounds. The mobile platforms were the Android, the iOS, the WindowsPhone and this web platform. The rounds each participant had to perform were structured as follows:

1. 3 $\times$ one effect sound without background sound
2. 3 $\times$ two effect sounds without background sound
3. 3 $\times$ two effect sounds with background sound

The starting platform should be determined randomly to have a more expressive result and to prevent the survey from habituation effects. The survey questionnaire contained questions about the age, gender and if the person already suffered from auditory system related diseases. The technical background of the participants was asked by questions to video game and smartphone experiences. It was also asked, which kind of smartphone the respective person was used to.

7.2 Survey Realization

Due to the recently named disturbance problems, the survey was performed on the rooftop of the building of the University of Ulm. All participants completed the anonymized questionnaire on which they received an identification number that was used for the further evaluation. Afterwards, every participant started with the usage of the first random application. During the survey, all users were standing upright, holding the devices in their hands and wearing headphones. After the settings for each round were made and the sound volume was adjusted, each user had to locate the current effect sound and tap on the screen to save the result. After the nineth round, the user changed to the next platform until each was used.

7.3 Survey Result

It was not directly the intention to draw conclusions about the Tinnitus disease, but rather monitoring the behavior of the applications during a survey and seeing potential differences between the platforms and under the users. So, the results rather concern the angular differences and needed times to find the effect sounds within each user regarding the different platforms.

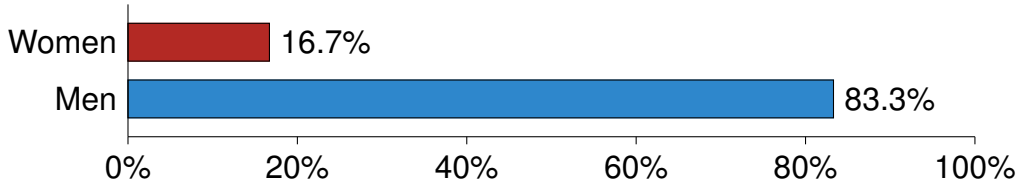


Figure 7.1: The percentage of women and men that took part in the survey.

7.3.1 Paper Questionnaires

Evaluations of the questionnaire brought up that the majority of the participants were male (see chart 7.1) with an average age of 27.8 years. The female average was at 25.5 years. All participants had to rate their experiences concerning the usage of mobile devices and their experiences in video games.

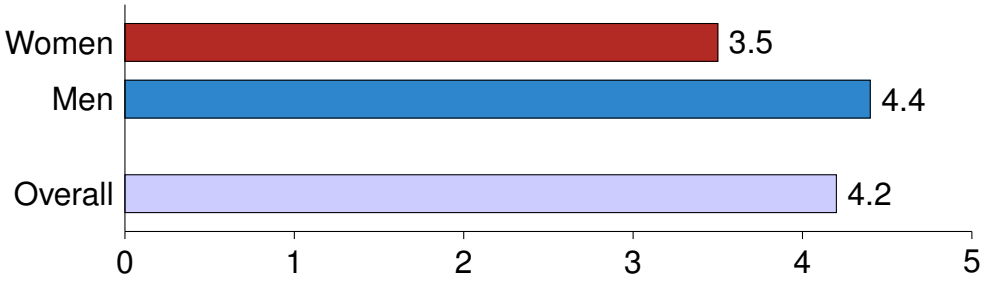


Figure 7.2: The average experience ratings concerning the mobile devices.

As can be seen in charts 7.2 and 7.3, the experiences on mobile devices were slightly higher under men (about 1 skill point). Additionally, men were more used to play video games than women; the average skill level was 1.5 points higher. If men are more affine to technical devices or if psychological aspects influenced the questionnaire, can not be determined here.

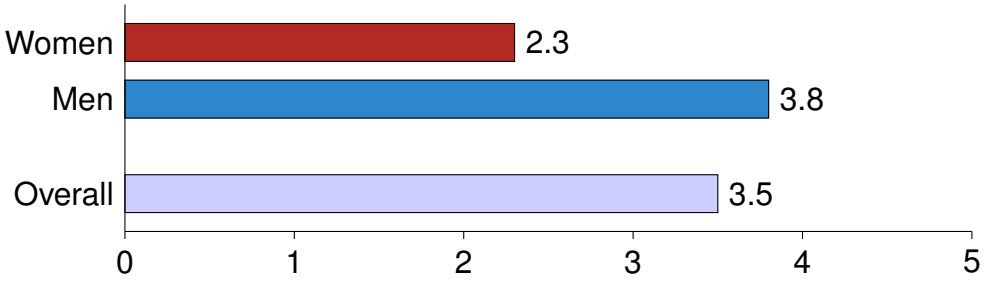
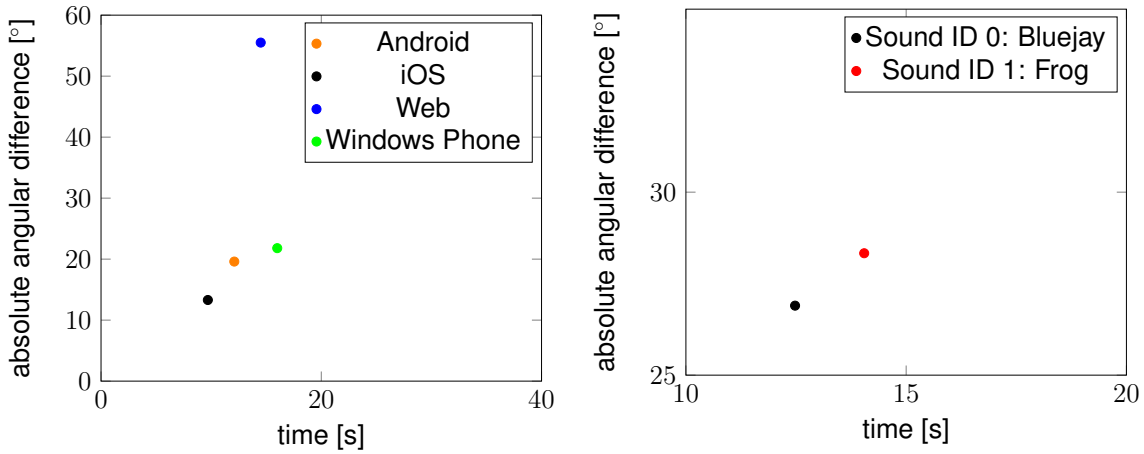


Figure 7.3: The average experience ratings concerning video games.

7.3.2 Platform Evaluations

For statistical analysis SPSS Statistics 21 (IBM) was used to create the following graphics 7.4 and 7.5. The time and the absolute angular difference between the single platforms were compared with ANOVA (Analysis of Variance). ANOVA compares the average values of the dependent variable (here: time, absolute angular difference) within the defined groups (here: platforms). For all statistical tests $p \leq 0.05$ was considered significant. To identify whether there are differences in time and angle in recognizing the target IDs (ID 0 for the bluejay, ID 1 for the frog), a t-test was performed. In graphics 7.4 (a), the average data points of all participants on each single platform are plotted. Graphics 7.5 shows the average values of every single user for every platform, whereas in graphics 7.4 (b) the two effect sounds are compared.



(a) ... all platforms.

(b) ... the frog and bluejay sound effects.

Figure 7.4: The average data points for ...

Due to the ANOVA, it could be observed that there is a significant difference in time ($p \leq 0.001$) and absolute angular difference ($p \leq 0.001$) between the single platforms. To identify these correlations between the time and the absolute angular difference, scatter plots were generated for each platform in graphics 7.5 and 7.4 (a). In the first graphics, each single dot represents a pair value (time on x-axis, absolute angular difference on y-axis) of one individual questionnaire participant whereas in graphics 7.4 (a), each dot represents the average of all participants within a single platform.

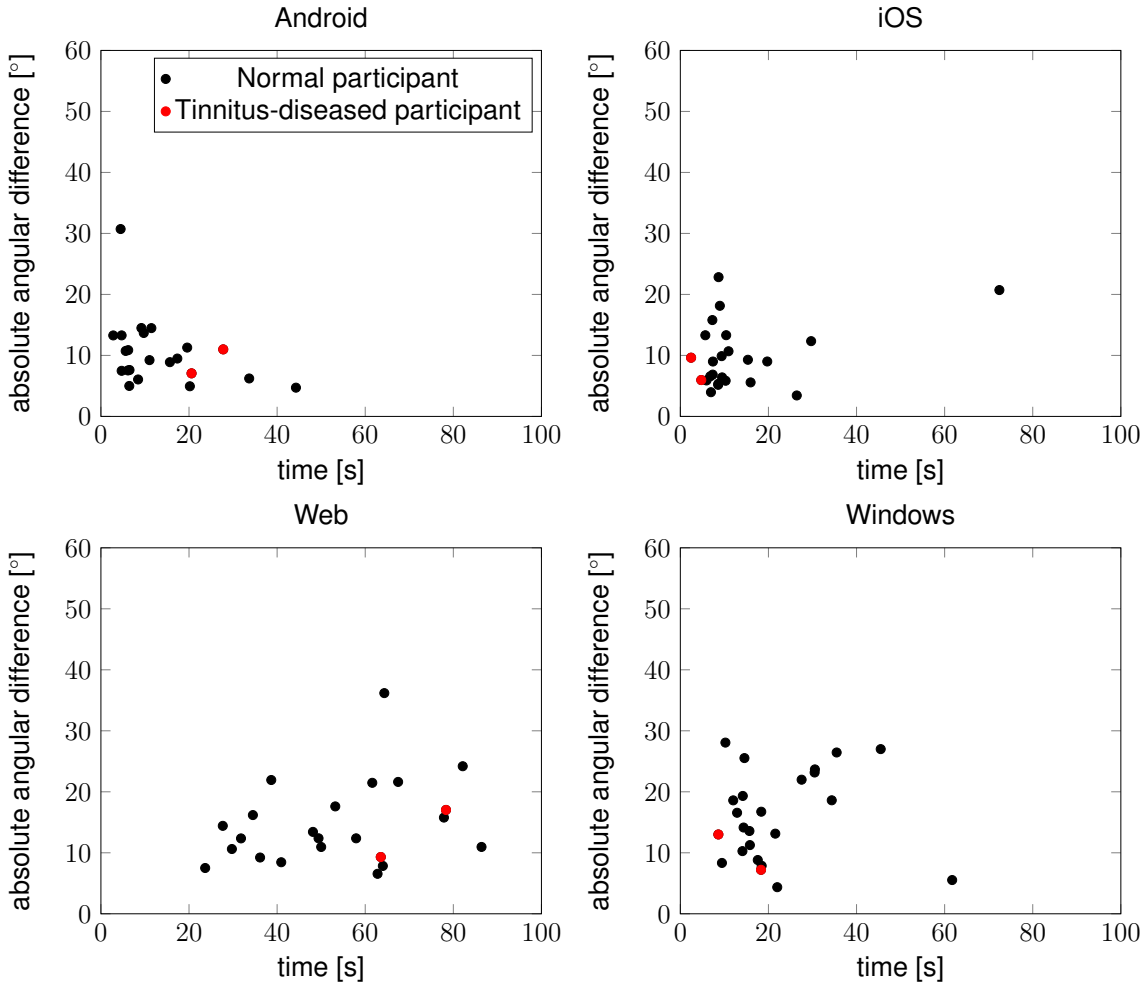


Figure 7.5: The average user results for the platforms Android, iOS, Web and Windows Phone applications.

Two of the 24 participants of this survey were suffering from Tinnitus. They were specially marked by red dots in the plots that represent all of the participants. Imaginary, each plot can be divided into four quadrants. Data points that are located in the lower left square represent persons that localize the sound origin rapidly and only with little angular difference. By contrast, pairs of values that are located in the right upper square represent persons that could not localize the sound origin this fastly and correctly. It can be observed that in the Android and iOS applications most data points concentrate in the lower left part whereas they are more spread in the Web and Windows applications. One could expect that persons suffering from Tinnitus rather might be located in the right upper square compared to healthy individuals. Interestingly, this was not the case for the Android, iOS and Windows applications. In the Web application they are slightly shifted.

Further analysis of the sound effects (see figure 7.4 (b)) showed that the time difference

in localizing the bluejay and frog effect sounds is significant $(p = 0.01)$. By contrast, the angular difference is not significant $(p = 0.536)$.

7.3.3 Conclusion

The results of the survey brought up that the native iOS and Android applications made it easy for the survey participants to locate the sound effects. The majority of the participants is located in the lower left quadrant of the plot. This means that the sound effects were rapidly located with only a small angular difference. In the iOS platform, the average values of the Tinnitus-diseased participants even belonged to the best in the whole data set. In the Android platform, these participants are located in the average of all participants. Slightly worse were the results concerning the Windows application. The worst results were made with the web application. Here, the angular difference was extremely high. These bad results are mainly caused by the not standardized APIs that were used to develop this application. During the survey, the browser had to be restarted for several times because the rotation of the user did not result in changes of the localization of the sound effect.
The two APIs Web Audio and Orientation Detection have to be improved to use this application for further surveys.

8 Conclusion and Future Work

In the scope of this thesis, an application was created that allows it to measure the results from the corresponding safari game. These data sets allow it to draw conclusions about the spatial hearing abilities of the user and might support further research in this direction.
The application could fulfill the requirements of being an accessible web application in the broadest sense. As the development of websites is always a big challenge due to the big variety of user-agents and devices, this application can be considered a success. Problems that appeared during the development process as well as possible improvements will be given in the following paragraphs.
The development of this application lead past the usage of different APIs that emerged after self-produced features did not lead to success. As an example, the Web Audio API can be named. After the self-produced cueing of different audio bands did not bring the desired results, this API was used, although its work-in-progress character. This development could give an insight on how modern browsers give the opportunity to run complex applications - only client-side, the server only supplying the resources. A survey could test the abilities and show weaknesses of this, as well as the other mobile applications. It brought up, that the web solution can not directly be compared to the native implementations but is an alternative that can be used.

8.1 Problems

This section deals with the problems that appeared during the development-process. Most of them came up because of compatibility problems or non-standardized features.

8.1.1 Work-in-progress Character

As can be seen in the implementation part of this work, some of the used APIs like the Web Audio API are not standardized yet and as a result are not supported by all

browsers. Others are not fully implemented in each browser like the Device Orientation API. This work-in-progress character produced problems because one cannot rely on the used APIs.

8.1.2 Storage

The biggest problem within the application storage feature was to determine whether the data should be saved on the user's device or on a server. Because it was not clear how to save the data legally on the server, the Web Storage API was used to save the results of the game rounds directly on the user's device. The problem that appeared through this implementation is possible data loss. This only happens in extraordinary cases, if the user agent runs out of resources and so finally flushes the cache data where the Web Storage API data is saved in. As the data is saved client-side, the user can delete it by either deleting the cache or through an option in the in-game menu.

8.1.3 Device Orientation API

As this feature is not standardized yet, there is no uniform implementation of the API. As mentioned before, the W3C recommends fetching the orientation data from gyroscopes, compasses and accelerometers [6]. As the 'compassneedscalibration' event is not fully implemented in all browsers, issues in the orientation change detection may appear. During the survey, this problem appeared due to the big number of played rounds. A rotation did not result in firing rotation events and so the spatial sound position did not change for the user. The browser had to be closed and reopened again to fix this.

8.1.4 Web Audio API

The application ran on the mobile Chrome browser (version 45) on an Android device (version 5.0.2) during the development-process of this work. This worked very performant and almost without any bigger problems in the smaller scale. During the survey, problems with orientation changes appeared. A orientation did not result in a repositioning of the spatialized sound source. Additionally, the browser support for this API is only given within the modern browser implementations.

8.2 Future Work

After handling the problems within this work, the possible future work will be named. This contains improvements for concrete modules and features or for the application in general. Next to improvements of the already realized application features, it would be more comfortable for the evaluation process if the paper questionnaire that was used in the survey was also an implemented feature. A recommendable approach would be the process-driven data collection that was realized in this paper [38].
The use of external sensors (like an oximeter) would allow it to measure other vital parameters while the application is used. Within the limitations of this paper [37], a framework was realized that allows it to address these external sensors. It has to be mentioned that compatibility problems may occur due to non-existing interfaces but this will be discussed no further.

8.2.1 Storage

To improve this module, it might be possible to create a REST-interface [7] that saves the user data legally on a server. In addition, a web interface would be required to manage the saved data and to retrieve them.
A further implementation might also improve our knowledge about Tinnitus. If this application was distributed to a large group of patients, a server connection would be very helpful. Due to the mass of people that used the application periodically, a large amount of data would be gathered - regarding for example the different life situations or points of time. In other words: Mobile Crowd Sensing would be realized [33].

8.2.2 Device Orientation API

This concerns the OrientationListener that takes the 'orientationchange' events from the window object. The smoothing that is realized through the soften() method calculates whether the change in orientation is above a certain threshold. At the moment, rotations around the Z- and Y-axis from figure 4.8 can not be handled without using the formula from [6]. Maybe there is a more performant way to only detect orientation changes around the X-axis.
In the actual implementation, the application is only available to users with mobile devices. This is mostly caused by the used Device Orientation API. A device without

built-in orientation sensors cannot use the API. A possible solution for this problem might be another GUI for users without a mobile device and users whose device does not support the API. These players would see buttons or be able to use the direction keys on their keyboard to change their direction.

8.2.3 Web Audio API

The support of the used API might only be improved by using another library. The API only supports the latest mobile browsers. This might seem to be a big disadvantage, but there is the question whether devices that use unsupported browsers have the required hardware resources to run this advanced application. This may lead to other performance issues.
The current spatial audio feature might be improved in some points. The first implementation required an in-game menu to change settings like the number of sounds, to toggle the background sound or to alter the sound-level. As these setting options remained in the in-game menu but the main-menu appeared, it is questionable whether these settings are still necessary. These settings resulted in always loading all sound files. The application might be improved by deleting these settings and thus be only loading the required sound files per game.
Another point is the audio volume gain that is also available in the in-game settings. As all devices give the opportunity to alter the audio volume from a flip-switch, it is the questionable whether the GainNodes from 4.7 are necessary. This may save technical resources and improve the performance. To reach this, there has to be found a way to cue the audio volume if it is behind the user. Maybe this can be done by increasing the distance between Listener and SourceNode if the sound is behind the user. The approach of simply altering the orientations of the Panner- and ListenerNodes did not work properly.

8.2.4 Preloading

The jQuery mobile framework provides the functionality to preload HTML content from another subpage of the same domain. Scripts and styles are excluded.
The framework manages the single pages that are located in one or more HTML files within a PageContainer object [15]. This allows preloading of the HTML content of other pages but excludes scripts and styles due to processing problems that may appear.

Loading times might be improved by loading the sound files already in the main-menu. These files caused additional loading times of about one second in a normal DSL wifi network - in slower networks the loading time is accordingly longer. As this is not a practicable solution, maybe a loading screen should be implemented that is visible while loading and afterwards starts a countdown before the game is started.

8.3 Final Statement

The application gives the opportunity to improve our understanding of Tinnitus. Next to the native solutions for the mobile platforms Android, iOS and WindowsPhone, this web application states a good alternative. Although the browser support is not as high as desired, at least older browsers like Opera Mini and BlackBerry browser could be fully supported [11, 9, 12]. As a result, this application could contribute to the ambulant treatment of Tinnitus and helps to understand this disease in a better way.
In which way the assumptions of a coherence between Tinnitus and worse hearing abilities could be supported will be further discussed in the survey.

9 Acknowledgements

I would like to express my special appreciation to my advisor Marc Schickler, who guided me through the whole thesis, always presenting a red thread.
Furthermore, I want to express my gratitude to everyone who supported me throughout the interesting development-process that lead to this terrific result.
Special thanks go to Paul Mohr for introducing me in the fabulous universe of the world wide web. I also want to thank Linda Hofmann for her emotional support and her knowledge she could offer. Last but not least I want to thank the rest of the team that planned, performed and evaluated the survey together with me.

Bibliography

[1] *Camera*. Camera Incorporated, 1916 (Bd. 20)

[2] ADENOT, Paul ; WILSON, Chris ; ROGERS, Chris: *Web Audio API*. http://webaudio.github.io/web-audio-api/, 2015. – Accessed: 09/08/2015

[3] ARAS, Aliyar: *Design und Konzeption einer mobilen Anwendung zur Unterstuetzung tinnitusgeschaedigter Patienten*, University of Ulm, Bachelor thesis, 2014

[4] BLAUERT, Jens: Sound localization in the median plane. In: *Acta Acustica united with Acustica* 22 (1969/1970), Nr. 4

[5] BLAUERT, Jens: *Communication Acoustics*. First. Heidelberg, Germany : Springer, 2005. – pp. 78 - 80

[6] BLOCK, Steve ; POPESCU, Andrei: *DeviceOrientation Event Specification*. http://www.w3.org/TR/orientation-event/, 2015. – Accessed: 08/13/2015

[7] BOOTH, David ; HAAS, Hugo ; MCCABE, Francis ; NEWCOMER, Eric ; CHAMPION, Michael ; FERRIS, Chris ; ORCHARD, David: *Web Services Architecture*. http://www.w3.org/TR/ws-arch/, 2004. – Accessed: 09/21/2015

[8] CANIUSE.COM: *Can I use - browser usage table*. http://caniuse.com/usage-table, 2015. – Accessed: 09/10/2015

[9] CANIUSE.COM: *Can I Use Device Orientation API?* http://caniuse.com/#feat=deviceorientation, 2015. – Accessed: 09/26/2015

[10] CANIUSE.COM: *Can I Use ECMAScript 5?* http://caniuse.com/#feat=es5, 2015. – Accessed: 09/10/2015

[11] CANIUSE.COM: *Can I Use Web Audio API?* http://caniuse.com/#feat=audio-api, 2015. – Accessed: 09/26/2015

[12] CANIUSE.COM: *Can I Use Web Storage API?* http://caniuse.com/#feat=namevalue-storage, 2015. – Accessed: 09/26/2015

[13] FOUNDATION, The jQuery: *jQuery Mobile 1.4*. http://jquerymobile.com/, 2015. – Accessed: 09/10/2015

[14] FOUNDATION, The jQuery: *jQuery Mobile 1.4 Browser Support*. https://jquerymobile.com/browser-support/1.4/, 2015. – Accessed: 09/10/2015

[15] FOUNDATION, The jQuery: *Pagecontainer Widget*. https://api.jquerymobile.com/pagecontainer/, 2015. – Accessed: 09/23/2015

[16] FSCHOLZ ; YAMINI.SONTAKKE ; GPORTIOLI ; X2357 ; BMORRIS22 ; CHRISDAVIDMILLS ; LIANA72 ; JOE11 travis ; PASALOG ; MUAWIYAH ; MSHOUTX ; JSWISHER ; FELIPE901: *What is JavaScript?* https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript, 2015. – Accessed: 09/10/2015

[17] GHITA, Serban ; ILYIN, Nick: *mobiledetect - The lightweight PHP class for detecting mobile devices (including tablets).* http://mobiledetect.net/, 2015. – Accessed: 09/10/2015

[18] GROUP, MEMS I.: *What is MEMS?* http://www.memsindustrygroup.org/?page=WhatIsMEMS,. – Accessed: 10/24/2015

[19] GS.STATCOUNTER.COM: *Stat Counter - Global Stats*. http://gs.statcounter.com/#mobile_browser-ww-monthly-201405-201505-bar, 2015. – Accessed: 09/26/2015

[20] HAZAËL-MASSIEUX, Dominique: *Javascript Web APIs*. http://www.w3.org/standards/webdesign/script, 2014. – Accessed: 09/29/2015

[21] HERRMANN, Jochen ; SCHLEE, Dr. W. ; STAUDINGER, Susanne ; LANGGUTH, PD Dr. B. ; LANDGREBE, PD Dr. M. ; HERRMANN, Jochen ; PRYSS, Rüdiger ; REICHERT, Prof. Dr. M. ; KIPPES, Christian: *Track your Tinnitus*. https://www.trackyourtinnitus.org/de/, . – Accessed: 10/06/2015

[22] HICKSON, Ian: *Web Storage (Second Edition)*. http://www.w3.org/TR/webstorage/, 2015. – Accessed: 09/08/2015

[23] INSTRUMENTS, National: *Understanding Acceleration and Choosing an Accelerometer*. http://www.ni.com/white-paper/3807/en/, 2013. – Accessed: 10/24/2015

[24] ITWISSEN.INFO: *Gyrosensor*. http://www.itwissen.info/definition/lexikon/Gyrosensor-gyro-sensor.html, . – Accessed: 10/24/2015

[25] KREUZER, Peter M. ; VIELSMEIER, Veronika ; LANGGUTH, Berthold: Chronischer Tinnitus – eine interdisziplinäre Herausforderung. In: *Deutsches Aerzteblatt* 16 (2013)

[26] LABACK, Bernhard: *The Psychophysical Bases of Spatial Hearing in Acoustic and Electric Stimulation*. Cumulative habilitation treatise, University of Vienna, 2013. – pp. 21 - 30

[27] LIU, Ming: A Study of Mobile Sensing Using Smartphones. In: *International Journal of Distributed Sensor Networks* 2013 (2012/2013). – Article ID: 272916

[28] LOCKHART, Josh ; SMITH, Andrew ; ALLEN, Rob ; TEAM, Slim F.: *Slim - micro framework for PHP*. http://www.slimframework.com/, 2015. – Accessed: 09/10/2015

[29] LOUI, Ronald P.: *In Praise of Scripting: Real Pragmatism*. http://www.cse.wustl.edu/~loui/praiseieee.html, 1999. – Accessed: 09/10/2015

[30] MARTIN, Robert C.: *Design Principles and Design Patterns*. http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf, 2000. – Accessed: 09/08/2015

[31] NETMARKETSHARE.COM: *Mobile/Tablet Operating System Market Share*. https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1, 2015. – Accessed: 09/26/2015

[32] OSMANI, Addy: *Writing Fast, Memory-Efficient JavaScript*. http://www.smashingmagazine.com/2012/11/writing-fast-memory-efficient-javascript/#, 2012. – Accessed: 09/27/2015

[33] PRYSS, Ruediger ; REICHERT, Manfred ; HERRMANN, Jochen ; LANGGUTH, Berthold ; SCHLEE, Winfried: Mobile Crowd Sensing in Clinical and Psychological Trials: A Case Study. In: *28th IEEE Int'l Symposium on Computer-Based Medical Systems*, IEEE Computer Society Press, June 2015, 23–24

[34] REENSKAUG, Trygve: *The Common Sense of Object Orientated Programming*. http://folk.uio.no/trygver/2008/commonsense.pdf, 2008. – Accessed: 09/08/2015

[35] SCHICKLER, Marc ; PRYSS, Ruediger ; SCHOBEL, Johannes ; REICHERT, Manfred: An Engine Enabling Location-based Mobile Augmented Reality Applications. Version: 2015. http://dbis.eprints.uni-ulm.de/1137/. In: *Web Information Systems and Technologies - 10th International Conference, WEBIST 2014, Barcelona, Spain, April 3-5, 2014, Revised Selected Papers*. Springer, 2015 (LNBIP)

[36] SCHICKLER, Marc ; REICHERT, Manfred ; PRYSS, Rüdiger ; SCHOBEL, Johannes ; SCHLEE, Winfried ; LANGGUTH, Berthold: *Entwicklung mobiler Apps Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health*. First. Heidelberg, Germany : Springer, 2015

[37] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Ruediger ; NIENHAUS, Hans ; REICHERT, Manfred: Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 2013, 509–518

[38] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Ruediger ; REICHERT, Manfred: Process-Driven Data Collection with Smart Mobile Devices. Version: 2015. http://dbis.eprints.uni-ulm.de/1136/. In: *Web Information Systems and Technologies - 10th International Conference, WEBIST 2014, Barcelona, Spain, Revised Selected Papers*. Springer, 2015 (LNBIP)

[39] SHAFRANOVICH, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. https://tools.ietf.org/html/rfc4180, 2005. – Accessed: 10/07/2015

[40] SMUS, Boris: *Developing Game Audio with the Web Audio API*. http://www.html5rocks.com/en/tutorials/webaudio/games/#toc-3d, 2012. – Accessed: 09/17/2015

[41] STEINLECHNER, Peter: *Hoer Spiele - Games ohne Grafik*. http://www.golem.de/news/hoer-spiele-games-ohne-grafik-1501-111884.html, 2015. – Accessed: 06/11/2015

[42] STREPPEL, Michael ; WALGER, Martin ; WEDEL, Hasso von ; GABER, Elisabeth: *Heft 29: Hörstörungen und Tinnitus*. https://www.gbe-bund.de/pdf/Heft29_und_Wertetabellen.pdf, 2006. – pp. 8, 19, Accessed: 06/17/2015

[43] VANIK, Ben: *WebAL*. https://github.com/benvanik/WebAL, 2011. – Accessed: 09/14/2015

Name: Fabian Henkel Student Number: 759825

Statutory Declaration

Hereby I declare that I have authored this thesis with the topic:

"Implementation and evaluation of a mobile web application for auditory stimulation of chronic Tinnitus patients"

independently. I have not used other than the declared resources. I have marked all material which has been quoted either literally or by content from the used sources. Further I declare that I performed all my scientifical work following the principles of good scientific practice after the directive of the current "Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis".

Ulm, November 1, 2015

Fabian Henkel