SONICAKT

- Interactive Music Listening Interfaces on the Web -

Master's Thesis

Karl Chueiri Heding

Aalborg University

Sound and Music Computing

Contents

Preface

The improvements of network technologies, social services and interactive mobile applications for sound and music are influencing the consumer?s musical experience. The experience is becoming increasingly more creative and collaborative. Users have access to a very large number of recorded music, they can assemble their own playlist, share it with others online and discover music based on their listening habits. Besides the various interactive possibilities provided for the experience of recorded music material, there are many music making applications for mobile devices. Algorithms which creatively make use of the device?s various sensors and processing power to create novel ways of manipulating digital and musical sound. Many of these applications besides potentially inspiring professional musicians for discovering mobile ways of producing and performing music, are also very non-musician friendly. This means that users with little to no experience in music are able to easily create their own music by performing simple gestures on their device. Recently the Sound and Music Computing community has seen a growing number of improvements in network technology based applications, facilitating the synchronisation and data exchange between a group of musicians and different applications. The combination of efficient devices for non expert user-friendly applications for multimodal control together with network technologies, give rise to novel possibilities for real-time interaction between musician-musician, audience-audience, audience-musician in live performance situations. The concept of audience participation enables audience members to be active participants with direct influence over visual and musical aspects of the experience.

It has the potential of shaping new formats of live concerts and performances to the extent it challenges the meanings and the conventional approaches, and becomes an entire experience of it's own.
The aim of this thesis is to shed light on some of the issues that concerns the topic of audience participation. To investigate and reflect the musician's point of view on the issues and possibilities to what audience participation proposes to music performances

This research shows the development of interfaces that would allow musicians to have an audience actively engaged and participating in the musical experience of their works outside of concert spaces. The work presented by this paper investigates the possibility of an Interactive Music Listening platform, where musicians can share works that the audience can interact with.

Aalborg University, June 2, 2017

Karl Chueiri Heding

<khedin11@student.aau.dk>

Project Supervisor

Daniel Overholt

Chapter 1

Introduction

1.1 Related Work

Sound and Music Computing is a study that encompasses a large variety of research topics and practices. The first part of the thesis starts by presenting some background work. The first chapter introduces the reader to the various topics of research that influence the work of the thesis. In short the work presented by this paper is concerned with providing users the opportunity for real-time interaction with sound, which in general could mean a variety of things. This chapter will contextualise the topic of the thesis.

1.1.1 Active Listening

A good amount of work and research has been done in the investigation of active listening, term often used in different contexts. In the social and behavioural sciences, it refers to the listener's involvement in hearing for intellectual and emotional messages from what is presented by the speaker. In human-computer interaction and more specifically sound and music computing, active listening and active experience are concerned with enabling listeners to interactively operate on musical content modifying and shaping it in real time while listening.
To better understand this concept, if we consider the time before the widespread

of recording and reproduction technologies, music was an embodied, interactive and social activity. Music was experienced either in concert halls where musicians would perform live or by learning and playing an instrument. Nowadays, music could easily be considered as a passive, non-interactive and non-context-sensitive experience. It is speculated that the greater availability of music enabled by technology lead toward a passive and detached attitude to music listening [North et al., 2004]. However, in contrast to this previous speculation, studies suggest [Krause et al., 2015] that the device people use is related to their degree of engagement. Where more positive responses were associated with digital devices, which in their study is linked to a high degree of choice and user control. With the current state of electronic technologies and all it's potential the need to promote a rediscovery of the interactive essence of music listening is emerging.

Works [Gualtiero and Camurri, 2011] suggest that an effective active experience of music is achieved through the combination of two different design perspectives: content-centric and user-centric.

Content-centric perspectives in the case of music, concerns the characterisation of a song to be modelled as more than it's stereo audio file such as .mp3, .wav or other coding, or any other simple metadata. It considers enabling a number of further data:
(1) real-time or offline features obtained by means of post production signal processing techniques (these could be, for instance: source separation to extract single musical instruments or song sections; spatial rendering to allow the control of 3D audio localisation of music instruments);
(2) data available from the music production process, such as the multi-channel version of the song, including the individual tracks (guitar, bass, drums, vocals);
(3) more metadata regarding semantic, context and usage (basic examples are 'mood' of the song, music genre, "where is best to play this song: 'party' or 'home'") [Wang

et al., 2008b].
With the availability of such data the real time processing and customisation of music content enables the possibilities for a more personalised listening experience.

User-centric perspectives focus on the involvement of the users in the experience. The active listening experience is shaped by the individual and social behaviour of the users involved. Therefore real time techniques for the analysis of expressive, emotional and social processes are useful to understand user behaviour. With this information expressive and joint music listening experiences are made possible.

The research work of Goto [Goto, 2007] uses advanced signal processing techniques on CD recordings. Through different active listening interfaces they investigate a few active listening scenarios: 'Music Playback', 'Music touch-up' and 'Music retrieval and browsing'.

'Music playback' scenarios are exemplified with interfaces that allow the user to: (1) skip, select sections, and navigate different parts of the song; (2) operate on visuals; select a dance sequence of virtual dancers during music playback; (3) read displayed lyrics that are synchronised according to playback (similar to a karaoke machine). The previously mentioned interfaces allows for active listening experiences in which the user does not necessarily operate on the sonic content of the music.
In the case of 'Music touch-up' users would be able to interact with sonic features of a song, for example: (1) An instrument equaliser, in which it would be possible to change the volume of the individual tracks of a recording(guitar, vocals, drums); (2) Manipulate and rearrange in real-time the pattern of the snare and bass drum of a song.
'Music retrieval and browsing' scenarios would allow users to actively discover new music based on musical content similarity. In contrast to the available commercial services where the retrieval of music is based on bibliographic metadata(artist, genre,

etc.) the users have to type or search for.

The above cases of active listening, can be considered as examples of standard GUI's on computers which perform signal processing on audio files. However, active listening scenarios aren't limited to such applications. Many cases have been investigated through installations where users must engage full body movements to activate and manipulate sounds. The orchestra explorer was a virtual orchestra installed on the stage of an auditorium. Visitors were allowed to navigate the physical stage and by doing so, explore the music being played. The installation allowed two modes of interaction. First, the visitors could play the role of the orchestra conductor. Through body movements control the volume of sound from the left and right loudspeakers. The second mode of interaction allowed the users to navigate the stage and explore the individual audio channels (cello, flute, harp, violin, piano, etc.). The manner in which users moved through the space controlled the amount of audio effects applied to the sound. Abrupt movements made the audio effects more perceptible. The orchestra explorer had a few exhibits and different versions of the setting were tested. For instance in one version called Mappe per Affetti Erranti, the full orchestra would only be heard if more participants were collaborating during the experience. Where each participant controlled the expressiveness of the individual instruments in the orchestra.

Active listening scenarios do not always have to be in regard to musical pieces. It can also be experienced in sound art which promote different modes of listening. There are many artists that criticise and challenge the sonic behaviours of everyday life. The authors of Recycled Soundscapes for example, investigated sonic behaviours in urban spaces. Their piece is a good example of an artwork that, promotes active listening experiences of sonic contents which are not necessarily musical. The installation consisted of a set of objects that allowed users to record the surrounding environmental sounds. The sound was played back through speakers. While users

could interact with the objects to affect the resulting sound, shaping and moulding their listening experience of what was heard in the public space.

Many of active listening works aim at allowing the possibility for non-expert users to interact with sound. In many ways is about bringing back the interactive essence of actually playing the music that is heard. A lot of interactive works in the Sound and Music Computing and Artistic communities, are motivated by the idea of connecting people through sonic interaction in everyday life. Establishing novel approaches of making, listening and experiencing music and consequentially new forms and genres of music. One of the challenges is then, to make these works more accessible to the public.

1.1.2 Interactive Sound and Music in the Mobile platform

Recently researches and artists have been exploring the potential of mobile devices to make music. Mobile devices include for example, cellular phones, smart phones, tablets (e.g iPad), portable media players (e.g iPod Touch). Increasingly more popular and nearly indispensable to everyday life, the device's ubiquity, processing power and networking capabilities, are seen as an attractive solution for works involving user interaction with sound in general.

History of sound and music in the mobile platform

Many researchers seem to indicate that, one of the earliest and most relevant examples of works incorporating mobile phones in music performance is Dialtones [Levin, 2001]. According to documentation, Dialtones was a large scale concert performance, premiered at the 2001 Ars Electronica Festival in Austria. The musicians performed their music on stage using mobile phones as instruments and throughout the performance, carefully choreographed dialling and ringing were made to the audience's own mobile phones. This concert required the audience members to register their phone numbers (prior to the event) and by doing so, they were given seat assignments

and ringtones to be downloaded to their devices. A concert with around 200-people makes for a very rich spatialized sonic experience. In 2006, Pocket Gamelan was a project that aimed at enabling non-expert performers to perform microtonal music, as well as exploring the interactive possibilities enabled by Bluetooth network. The phones were implemented to produce sound out of their own speakers. In addition, the devices were placed inside a pouch made of semi transparent fabric attached to a cord. In their presentations, performers swung the chords and as a by-product of these movements audible artefacts such as chorusing and doppler shift were produced [Schiemer and Havryliv, 2006]. Since then, mobile devices have become more efficient.

The appearance of smart phones facilitated the development of mobile music applications. Smart phones offer high computational speed, storage and I/O capabilities which are comparable to that of computers. Researchers reference the Ocarina for iPhone, as an example of mobile music application. It is one of the earliest mobile instrument app that is a virtual realisation of exiting acoustic instruments. It integrates the multiple sensing technologies available in smart phones such as, microphone, multi-touch, accelerometer together with real-time synthesis and interactive graphics. The app incorporates an interesting social dimension that allows users to listen to other users around the world in real-time.[Wang, 2014]

More recently however, the amount of available commercial music making applications is overwhelming. Dominating commercial mobile music applications include but are not limited to sophisticated DAW's (digital audio workstation) and emulations of other instruments (e.g analog synthesizers). With applications such as GarageBand by Apple which allows user to produce complete musical pieces with various virtual instruments such as piano, guitars, drums and bass. Or the Korg iMS-20 that emulates the company's MS-20 synthesizer. Most of these commercial applications however lack the social dimension, or as mentioned earlier the user-centric perspectives. One must turn to the works of researchers in the Sound and Music computing

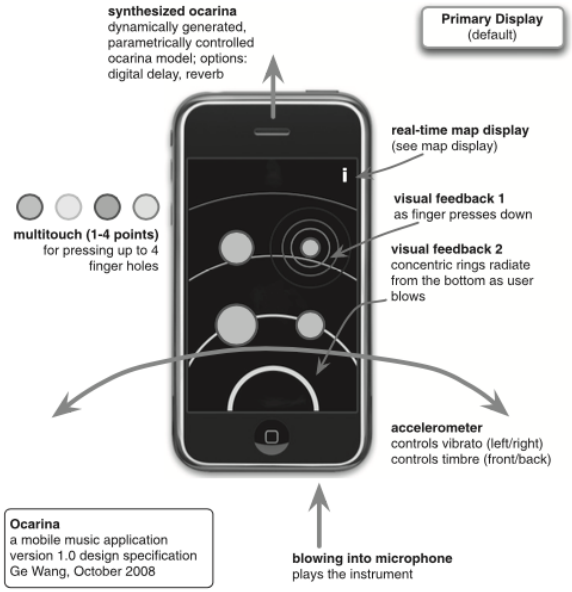


Figure 1.1: Ocarina Design

community to find novel, experimental and interesting ways of interactive and or collaborative music making with mobile devices. The Stanford Mobile Music Phone Orchestra (MoPhO) founded in 2007 at the Centre for Computer Research in Music and Acoustics (CCRMA), with it's first concert in 2008 [Wang et al., 2008a]. Was the first repertoire-based ensemble to use mobile phones as it's primary instrument. The project consisted of different stages of preparation such as crafting the digital instruments for the devices, as well as composing dedicated musical pieces for the ensemble. There are variety of instruments and different pieces that make a repertoire that ranges from scored compositions, structured and free improvisations, to sonic sculptures. One of their first publicly premiered pieces Drone In/Drone Out, explored the mobility and sensor capabilities of phone devices.The phones generated sound through real-time synthesis with the device's accelerometer data mapped to some parameters of the synthesis. Resulting in a user controlled interface. In their debut concert, members of the ensemble were disguised as the audience, with the conductor

alone on stage. As they began playing, the disguised members revealed themselves and surrounded the audience. Creating a specialised audible experience, with the use of sound produced by 12 phones and their built-in speakers [Wang et al., 2008a]. Since the launch of MoPhO other mobile phone orchestras were founded in a number of academic institutions, such as the Helsinki MoPhO and the Michigan MoPhO. More recently the Smartphone Ensemble, explores mobile computer in collaborative musical performance. With a slightly different approach, it investigates the mobile phones social mediation in the context of musical performances. The Smartphone Ensemble created in 2015, works on ways to enhance links between musicians, and musicians and their audience by leveraging the network and mobile capabilities of smart phones. Searching for alternatives to standard musical performance spaces instead of traditional concerts, as their project intend to be urban interventions as the participating group of musicians embark in tours around urban spaces. Using smartphones as instruments, the group designed GUI as instruments in a single app. Which generates a variety of sound synthesis techniques together with audio processing using the MobMuPlat. In their setup each member of the ensemble wears a speaker attached to an arm band. Their debut performance was carried out in a public park, with four performers following a specific trajectory while improvising over different musical ideas. Raising curiosity of nearby spectators which asked for available apps in order to join the experience, indicated the possibility of the audience members as active participants in the experience [Arango and Giraldo, 2016].
With all these successful examples of applications it is important to note, that even though many mobile devices provide networking, multi-sensor and processing capabilities which are attractive solutions for the design of novel instruments and interfaces for musical expression, these devices very often present challenges. One of the challenges is the size of the screen, which limits design and the interface capabilities for controls of musical expressivity and virtuosity. Another challenge is the quality of the built-in speakers of the devices which might not be the best option for musical performances. It is often the case that researches use external speakers or even a

separate sound generation source all together.

The research work of [Michon et al., 2017] addresses some of the issues with mobile devices as performative interfaces, and propose some methodologies towards augmenting mobile instruments, to enhance the capabilities of these devices towards musically expressive instruments. Using 3D printed models designed in a open source 3D Computer Assisted Design (CAD) modeller Mobile3D, that are then attached to the mobile phones. Examples range from attaching mouth pieces that allow for a much cleaner sound to be sensed by the microphone, to resonators that act as passive amplifiers to enhance the sound of the device's built-in speakers. Many other examples can be found in [Michon et al., 2017].

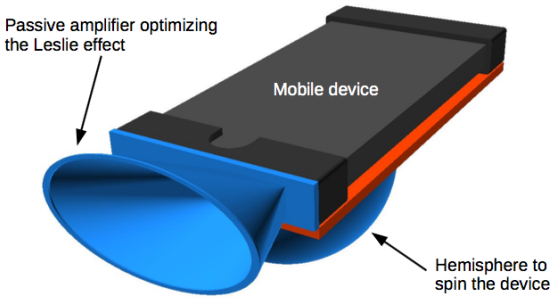


Figure 1.2: Augumented Mobile Device

1.1.3 Audience as active participants in the musical experience

A lot of the works in active listening bring forward the possibility for audience members to be active participants in the concert experience, mediated by the use of technology. This blurs the distinction between musician and audience. If participants are able to influence and or participate in the musical performance, everybody involved in the experience in a sense, becomes a performer.

The concept of audience participation is nothing new and has been practiced for

years. The moment during a concert in which the audience sings along to a chorus or a verse, is a form of an audience being actively engaged in the musical experience. However, many artists have explored the possibilities of the audience participating in the musical experience as performers. One of the earliest examples are seen in the works of Jose Maceda, a Philippine composer and concert pianist with a PhD in ethnomusicology. His explorations began with the piece "Ugnayan (music piece for 20 radio stations)", 1974. The piece is a 20 separate 51-minute original tracks that are recordings of sounds produced by traditional Philippine instruments, each broadcasted over 37 radio stations [Taylor, 2017]. The idea was that people around Metro Manila would go out into the streets with their radio devices and tune in to different radio stations, so that each track is played simultaneously from different sources. This project investigated the possibilities of the performing space, using the entire capital city of the Philippines, Manila as the stage. Maceda then went to explore large scale participation in concerts with works such as Udlot-Udlot (Hesitations, 1975), a music piece for an open-air ritual involving hundreds or thousands of performers. The first performance got 800 players involved, and consists of variations of rhythmic drones played in traditional instruments from the Philippines [Nicolas, 2015].

Research indicate that the aforementioned piece "Ugnayan (music piece for 20 radio stations)" is one of the earliest examples of harnessing electronic devices for the distribution of sound agents across an audience. The rock band The Flaming Lips, in 1996 performed a couple of similar experiments. With the performances of Parking Lot Experiments and Boom Box Experiments, where the singer distributed individual tapes across the audience so that they could play them back on their cars(Parking Lot Experiments) or tape-players (Boom Box Experiments). While the tapes were playing the singer would then conduct the audience to turn up and down the volume of their devices accordingly [Taylor, 2017].

Researchers investigating Audience Participation point out that the aforementioned Dialtones 2001 (A Telesymphony), was the first performance to involve the audi-

ence and leverage mobile phone devices for musical performances. Since then mobile phone devices have become more efficient (release of iPhone in 2007), and novel ways of involving the audience with their own personal devices have been explored by artists and researchers. That brings the emergence of a new genre: Distributed Music [Taylor, 2017].

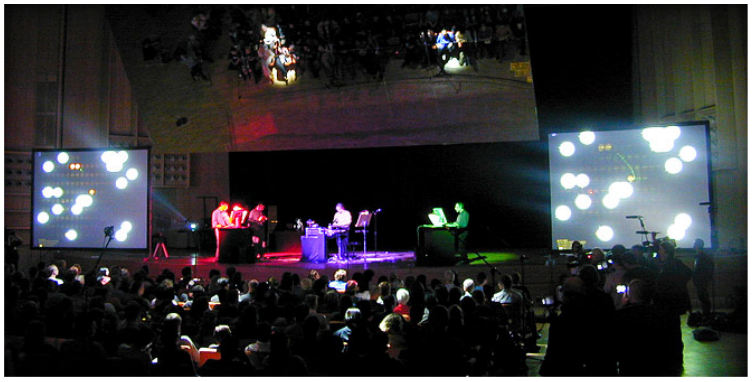


Figure 1.3: Dialtones 2001

With the use of native mobile apps and networked web audio systems, artists and researchers investigated the possibilities of audience participation in concerts. Early notable examples in the academic community are found in the documentations made by the MoPhO in 2010. Through a series of concerts they explored different social mobile computing techniques to get the audience to actively participate in the musical experience, using iPhones and iPads as primary hardware interface. In some of their experiments they used a technique of sampling the audience, which means that the audience generate sounds that become part of the performance. In Madder Libs (2010) [Oh and Wang, 2011], audience members are asked prior to the performance to record video clips of themselves emulating different instruments. The videos are uploaded to a server which the performer then downloads the videos onto an iPad device. During the concert the performer triggers the video snippets with

the use of the iPad touchscreen capabilities. Using the various videos to form a complete musical piece. Converge 2.0 (2010), used the similar approach of pre-sampling. Audience members were asked to record autobiographical moments, which are later used to form a semi-coherent narrative. Using mobile phones through the mediums of text, video and sound, the participants upload their recorded material to a server. These videos are then scored into a complete audio-visual piece. In Orkestra (2010), the sampling of material is done live during the concert. Audience members one by one, perform vocalisation sounds which are recorded using a mobile phone and then uploaded to a central computer. With a live-coding performance by the main performer, the sounds recorded are played back organised in rhythmic patterns through a system of 8-channel surround speakers.

One of the first bands to use smartphones using native apps as collective musical instruments in a large scale concert was the rock band OK GO. Together with National Public Radio, the work Needing/Getting was performed at the event for the radio show "This American Life" in 2012. The audience was asked to download an app to their phones and bring them to concert. The app provided each audience member a set of three buttons corresponding to different notes. The audience was segmented into four different colours, the audience followed a scrolling score onscreen which indicated which button in the app to touch. The band performed as a bell choir. Those who didn't possess the app or a smartphone, were instructed to perform rhythmic stamping of feet and snapping of fingers according to the scrolling score onscreen.

Back to the academic side, Echobo (2012) [Lee, 2012] is a mobile app which the audience uses as an instrument to collaborate with an electronic musician and acoustic musician performing on stage. With a graphical interface the electronic musician is able (in real-time), to send the notes to audience's mobile devices instructing which notes to play. This allows live improvisation of notes, providing more direct communication between musician and audience (contrary to the predefined score of

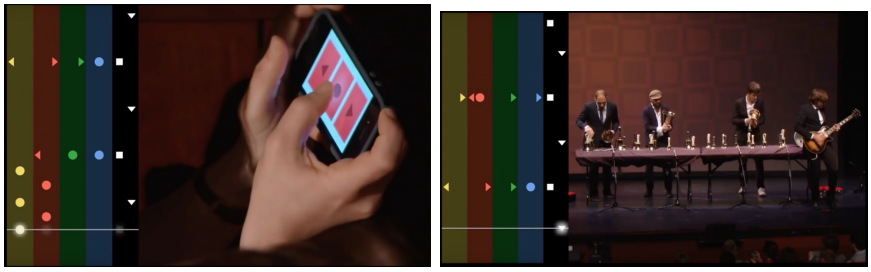


Figure 1.4: OK/GO Needing/Getting 2012

the OK GO Needing/Getting performance). Additionally audience members could send their own pattern of notes to fellow participants, establishing an audience-to-audience communication. The sound produced by the audience's devices serve as background texture to the acoustic musician, in this case a clarinet player. The author of Belzebuth 2014 has an interesting remark. This piece is a 13 minutes performance that included the participation of the audience. The participants download the app through the web, and in this case the audience produces sound through gestures instead of a touch screen interface. The spectators are conducted by the performer, who performs gestures which the audience then reproduce. The author of Belzebuth notes how the active participation of the audience affects how they understand musical composition. That by acting in the performance they would experience sound as material, texture and their influence over the performance. Therefore indicating that viewer's get an insight into the compositional process [Taylor, 2017].

The introduction of the Web Audio API in 2014-2015, brought a number of distributed music performances that uses web-sites that the audience can access to participate. The Web Audio API has made possible the implementation of complex web-based interactive applications. It allows complex processing of sound with capabilities similar to those found on modern desktop audio production applications, such as mixing, processing and filtering. Furthermore, the Web Audio API is designed with possibility to be used in conjunction with other elements on the web platform,

such as 3D graphics rendering API's like WebGL.

IRCAM's CoSiMa (Collaborative Situated Media) group, explores the smartphone's ubiquity, multimodal sensing, audio-visual processing capabilities and web standards, to create collaborative performances. Beginning in 2014, the group created a series of events titled Collective Sound Checks [Schnell et al., 2015a]. It began as workshops where they develop a series of web applications [Schnell et al., 2015b], which allow users to produce sound according to the smartphone's motion. These gadgets can be played individually or collaboratively with a group of players. They created a series of different smartphone applications exploring different techniques, metaphors and sonic material. A noteworthy example is WWRY:R (We Will Rock You: Reloaded). In this application users can collaborate to perform their own versions of the famous song by the rock band Queen. Participants can choose between, drums, vocals, guitars and collaboratively perform gestures holding their smartphone devices that trigger segments of the song. In this case the participants are performers as well as listeners in the experience [Robaszkiewicz and Schnell, 2015]. The CoSiMa group also worked in collaboration with artists to create concerts with audience participation. For one of them the piece Chloé x IRCAM after opening a web page, participants are asked to indicate their position on the map of the concert space. During the concert using four tablets the performer moved sounds over the audience's smartphones, while also letting appear dedicated sound interfaces with instructions for performance on the device's touchscreens.

The performance of the work Crowd in C[loud] [Deusany et al., 2016], has an interesting approach to leveraging the audience's smartphones and cloud service. The participants access a web site, through their mobile devices or notebooks. The performer on stage controls the performance, live coding the programs running in the audience devices. It allows the performer to play tunes and control the chord progression of the music. Another feature that they implement is that each audience member creates a 'sonic profile', and then go on to scrolling through other profiles

Figure 1.5: WWRY:R

created during the performance with the option to like or dislike what they hear (inspired by social dating apps such as Tinder).

Audience participation scenarios involves interaction design and compositional decisions. In the case of sonic interactions, some of these decisions include but are not limited to:

Considering whether the audience participates in musical decisions, or act as passive receivers. If the former is the case, considerations on how their decisions affect the music need to be made. If the sound they produce are the background texture to a performer playing, or if the sounds they make are the main source of the musical experience. If the audience contributes with sonic material to the piece before the performance (e.g by uploading sounds to a dedicated server), or live during the performance.

Which devices the audience use to produce sound, as we've seen the possibilities are not limited to smartphones. How will the audience access these interfaces that produce sound. In the case of a smartphone being is chosen, if the interfaced is accessed through a native app or the web. What sort of gestures is required for

the audience to produce sound: touch (e.g via touchscreen) or hand gestures (e.g shaking, tilting device).

There are a few layers of considerations to be taken when designing a concert, or interfaces for audience participation. Some considerations will be described more in detail in the design process documented in this paper.
Although many of the aforementioned pieces are successful examples of their applications, most of them if not all, are very context-specific. Which from the point of view of this paper is a problem to investigate. For a lot of these works users had to be present on the day of the event to experience it, and produce specific sound events. If the interest is to bring back the essence of a more interactive approach to the experience of music to our everyday lives, it is important to provide more works that do so, in the context of everyday listening. Take advantage of the ubiquity of these personal devices, their mobility with applications that allow users to experience these forms of interaction anywhere and any time in their everyday lives, allowing users to practice and become more sensitive to sonic and musical structures. The motivation of the project presented by this paper is to bring such active, interactive and collaborative music listening experiences to everyday life. To give users the possibility to discover new music and ways of listening as well as interacting with music material that they enjoy.

Chapter 2

Design Strategies

This section describes the design methodologies that are used for the development of three web applications. These web applications all have the same goal of providing users the possibility to interact with musical material in the context of a music listening application. This would give users the possibility to actively control sonic parameters that would influence their listening experience. These applications have the goal of being used by both expert and non-expert musicians. Ultimately the goal is to investigate the possibility of a music listening platform, where musicians would share music that users (musically trained or not) could sonically interact with. However, the term sonic interaction is used under many different contexts for instance, the term may refer to [Wanderley, 2001]:

instrument manipulation, real-time sound synthesis control driven by performance. (performer-instrument interaction).

device manipulation for score level control, could be an interaction where a user can control a rhythmic pattern to previously defined computer generated sequence of sound events.

device manipulation for post production activities, for instance gestural control of digital audio effects.

interactions in multimedia installations, where one or many user's actions are sensed and used as input control for an audio/visual/haptic system.

This section attempts to clarify the sonic interactive aspects of this project, and through examples of methodologies found in literature, define the strategy chosen for the design and implementa-

tion of the interfaces presented by this paper.

2.1 Digital Musical Instrument Model

The application designed falls within the digital musical instrument model. A Digital musical instrument (DMI for short) is a term referring to an instrument that contains two independent modules: **1)***A gestural or performance controller*. Also known as an input device or hardware interface. Is a physical device (computer mouse, MIDI hardware interface with a keyboard, sliders, knobs, or mobile devices), that relies on input gestures from user interaction to provide real-world information. **2)***A sound unit generator* (or sound engine). This engine contains controllable parameters (e.g sound synthesis algorithm, a signal processing algorithm).

These independent modules are connected and related to each other through mapping. 'Mapping' can have different meanings depending on the context in which is used. In this paper 'mapping' refers to the act of taking real-world data as a result of real-time user interaction with an input device, and using this information to control various parameters within a sound engine [Hunt and Wanderley, 2002].

In [Hunt et al., 2000] an interesting discussion of the role of mapping in interactive systems is pointed out, where two main points of view exist:

Mapping is a specific feature of a composition;

Mapping is an integral part of the instrument;

The applications designed for the project presented by this paper, mostly follows the second point of view. Which means that mapping is part of the interactive system, and influences the way a user makes use of it in different contexts. It makes use of explicitly defined mapping strategies to design the different interfaces. Explicitly defined mapping strategies allows the designer to be in control of each of the interface's components by defining the input-to-output relationships.

2.1.1 Mapping Strategies

Three explicit mapping strategies for designing interfaces for musical expression have been proposed by research found in literature [Hunt and Wanderley, 2002]:

one-to-one, one performance parameter controls one synthesis parameter.

one-to-many, one performance parameter influence several synthesis parameters at the same time.

many-to-one, many performance parameters control one synthesis parameter.

many-to-many, a possible combination of the above strategies.

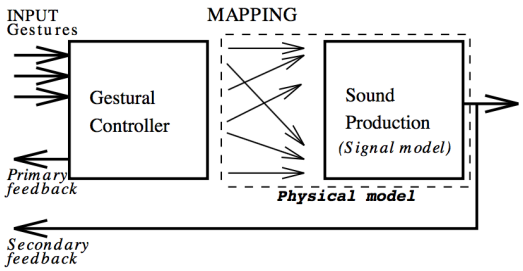


Figure 2.1: DMI Model

2.1.2 Interface Design Guidelines

As has been pointed out by [Miranda and Wanderley, 2006], in order to design a new digital musical instrument or interface for musical expression one typically needs to:

a. decide on the gestures that will be used to control the system.

b. define gesture capture strategies that will translate these movements into electrical signals. This is usually done using a variety of sensors to measure real-world information from the user (hand, arm, lip, or other body movement, velocity of movement, pressure or any other variable of interest).

c. define sound synthesis algorithms that will create the sounds to be played; or, define the music software to be used for control of prerecorded musical processes.

d. map the sensor outputs to the synthesis and music-control inputs. here one can follow mapping strategies guidelines.

e. decide on the feedback modalities (besides sound) available: visual tactile and/or kinaesthetic.
Following these aforementioned guidelines the choices for design can be explained.

a. The interfaces would leverage the web audio API. The goals is for the interface to be accessible to any user, and the Internet is truly ubiquitous as it can be accessed with computers, smartphones etc. The interaction would differ, depending on the device the user chooses to open the web page containing the applications. However, for any case users would use their hands to control the different interfaces.

b. The gesture capture strategies in the case of mobile devices, will leverage the multisensing capabilities of smartphones, such as touchscreen and accelerometer data to translate the movements. The computer will rely on the device's mouse and accelerometer data (depending on the device and browser used to open the application).

c. The sound synthesis and signal processing will be handled by the web audio API, and more specifically as will be seen later in this paper Tone.js. This API is chosen because it's seems to be the most efficient and extensive library currently existing on the web.

d. As will be seen later the application will make use of a combination of one-to-one and one-to-many mappings. Many-to-one are not voluntarily used in the applications, as they are a form of mapping that is more relevant for the design of interfaces with instrument-like behaviour. The goal of this project is not to create a musical instrument but rather a way to interact with musical content.

e. Visual feedback will be provided, as users will interact with visual elements seen on

the device's screen.

2.1.3 Interactive Music Systems

How interactive music systems[Rowe, 1993] respond to user interaction can be distinguished between Transformative, Generative or Sequenced methods.

1. *Transformative* techniques take an existent music signal and apply transformations to produce variants. These variants can be distinguishable from the original signal or not. For transformative algorithms the sound source is a complete musical input or a live input.

2. *Generative* methods use a set of rules to produce the musical output, for instance taking pitch structures from patterns following pre-defined scales according to random distributions, with a set of allowed duration values.

3. *Sequenced* methods use fragments of prerecorded music in response to some real-time input. The aspects of these fragments may vary according to performance, such as rhythmic variations, playback tempo of the sequence, playback speed of the fragments(affecting pitch), etc.

These distinctions, are helpful for defining the context of the applications that will be designed. The envisioned interfaces will be working with pre-recorded musical material as it's sound source. From the methods mentioned above, the Transformative and Sequenced methods can be applied to this project. As it is the case, the Transformative method defines the SNCKT EFFECTS application. The Sequenced approach is useful for defining the SNCKT MOVE interface. The last interface SNCKT COLLAB, it's a little more complicated to define using these aforementioned terms. Perhaps, one could say that it includes all of the above methods. SNCKT COLLAB allows users to trigger fragments of prerecorded music material, as exemplified with the Sequenced method. However, duration, timing patterns and pitch are predefined based on a set of rules. Variations on pitch may occur according to a random distribution, across a set of predefined pitch alterations. Sound events only occur

on specific time instances following a global tempo. These are traits found in the Generative methods. The interface also allows users to apply transformations on the signal using digital audio effects, which is a feature that relates to the Transformative technique. It's important to understand why the interfaces turned out to be as they are. These distinctions make possible the investigation of different ways in which users (musically trained or not) could interact with prerecorded musical material during their listening experience. Moreover, a chosen mapping strategy is one of the most important factors that will define how accessible the interface is to users, in regards to interaction and musical expression.

Chapter 3

Implementation

To create a web interface we begin by creating a .html file. HTML (HyperText Markup Language) documents contain the content that will be displayed on the webpage. It uses "markup" or "tags"(referred to as HTML 'elements') to annotate text, image or other content that is to be displayed on the page. These HTML elements are not displayed on the page, browsers use these "tags" to render the content of the page. These elements look something like <head> <title> <body>, and it include many more different elements. To create a document, developers might use some type of text-editor specialised for web design such as the open source Brackets. However, HTML documents can be written using basic text-editors such as Notepad(PC) and TextEdit(Mac). The example at B.3 creates a blank page with the title "SNCKT EFFECTS", which is the name of one of the interfaces made for this project.

```
1  <!DOCTYPE html>
2
3  <html>
4
5      <head>
6          <title>: SNCKT EFFECTS :</title>
7
8      </head>
9    <body>
10
11          </body>
```

```
12
13 </html>
```

Listing 3.1: Starting a HTML project

HTML tags are written in pairs. The first tag <html> begins and contains the element 'html'. The element ends with a tag holding the same name but with a backslash before the tag name, in this case </html>. The content of each element is placed in between two tags.

Rendering of functionality is handled by other technologies such as JavaScript. It's a programming language environment supported by all modern browsers. The Web Audio API which is used for the interfaces designed in this paper is developed with JavaScript. API stands for 'application programming interface', is a set of routines, protocols and tools used for building software applications. In short an API, specifies how software components should interact. It is related to the software library. A library is an implementation of these specifications. The next step is to specify between the <head></head> tags which libraries the project is going to use. This project makes use of two main different libraries that are designed for sonic interaction purposes: Tone.js [Mann, 2014] and NexusUI.js [Taylor et al., 2014].

Tone.js

Tone.js is a Web Audio framework that is designed to facilitate the creation of interactive music applications in the browser. It was chosen for a number of reasons. First, is a state of the art library based on the Web Audio API. The architecture of Tone.js is designed to be familiar to musicians and audio engineers coming from Digital Audio Workstations. Musicians would be familiar with terms and features such as send and receive buses, master output channel, and global transport which allows different sound modules and events to be synchronised and coordinate along a shared timeline. Second, Tone.js can easily be used with outside libraries and modules. Third, it provides an extensive API documentation, making it even more beginner and musician friendly. For more information about all of Tone.js' features check the document [Mann, 2014].

NexusUI.js

NexusUI is a state of the art set of web-native tools for developing a graphical user interface, which allows for the interaction with time-based objects. It was chosen for this project

because it provides standard audio interaction modules such as buttons, dials, sliders, audio waveform visualisers, graphical keyboard. It also designed to be easy for musicians and designers to implement their own GUI's on the browser with mobile functionality, taking advantage of touchscreen and other sensing data such as accelerometers. Allowing user's not only the ability to send OSC to audio programming environments such as Max, Pd and SuperCollider. But also, combining them with other web libraries that are used for generating sound such as Tone.js.

```
1  <!DOCTYPE html>
2  <html>
3     <head>
4          <title>: SNCKT EFFECTS :</title>
5          <script src="nexusUI.js"></script>
6      <script src="Tone.js"></script>
7          <script src="StartAudioContext.js"></script>
8      </head>
9
10   <body>
11      <script>
12
13      </script>
14   </body>
15 </html>
```

Listing 3.2: Adding .js JavaScript libraries to the project

Additionally, in 3.15 we add StartAudioContext.js, a short file that runs a script that is used for unlocking sound on iOS devices, which by default is blocked[REEEEF]. These extra lines of code are used for referring to the libraries that are to be used in the project. These libraries are .js files that are usually inside the same folder as the .html file. The line <script src="...">> is indicating where those .js files are found. In the case of this project they are located inside the same folder. All the elements which will appear in the web page will be included inside <body></body> tags. Additionally, any line of code that uses JavaScript functionality has to be included within the <script></script> tags.

With the .html document started and with the specification of the libraries used. We can

now start building the interfaces.

3.1 SNCKT EFFECTS

The first interface titled SNCKT EFFECTS is a relatively simple interactive music system. The motivation behind this interface is to allow listeners the possibility to add audio effects, and change the playback speed of the song they're listening to. This will allow for real-time customisation of the user's listening experience. Standard music listening applications (e.g iTunes, Spotify, etc. . .), currently don't support such interactions that operate directly on the sonic content that users listen to. Direct customisation of sonic content found in some modern music listening applications, is often limited to some simple equalisers that enhance or diminish some specified audible frequency content of the sound. Applying sound effects to audio or music signals is a very common practice for musicians and sound engineers (e.g guitar players use effect pedals, sound engineers use analog effect racks, or audio effects found in DAW's, or DJ's with modern turntables). The practice of applying audio effects to music signals is almost indispensable to modern music performance. The interaction with such audio effects units is relatively simple. For instance activating the effect on/off with a switch button, or turning some knobs (e.g potentiometers) that change the values of some of the effect's parameters. The implementation of such features would allow for simple user interaction, with perceptible influence over sonic content allowing a customised listening experience. The sound and musical production would not entirely depend on the user, possibly making it a friendly non-expert user interface.

3.1.1 Digital Audio Effects

In this project we make use of digital audio effects. Effects are implemented using digital signal processors. Effects can be separate modules or built into a keyboard hardware. The processor takes the analog input (referred to as "dry" input), which may be produced by an instrument such as a guitar, keyboard or a previously recorded signal from some medium. The processor samples the signal at an appropriate audio rate 44.1kHz, meaning that it converts a real analog signal into something that can be computed digitally. The audio sampled is

subjected to a digital signal processing (DSP) algorithm, and the result is reconstructed into analog form to be sent on to the next unit, such as a speaker system, a mixer, or even another effects processor. Multiple effects can be applied in any different order and a chain of effects can have a drastic impact in the output audio.

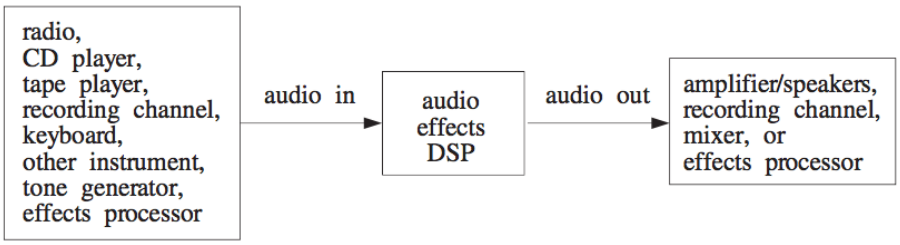


Figure 3.1: Digital Audio Effects Processing [Orfanidis, 2010]

Filtering Effect

First effect implemented for this particular interface is that of filtering. A signal can be seen as a set of partials with different frequency and amplitudes. Filters remove/attenuate audio from the spectrum above or bellow some cut-off frequency. There are many types of filters, a Lowpass filter for instance, selects frequencies up to a specified cut-off frequency fc and attenuates frequencies that are higher than the specified fc. A Highpass filter selects frequencies higher than fc and attenuates all the frequencies below fc. For more detailed information one can find it at [Orfanidis, 2010][Zolzer, 2002].

Using Tone.js we instantiate a high pass filter, using JavaScript as:

```
1 var filter = new Tone.Filter({"type":"highpass","frequency":20}).
    receive("filter").toMaster();
```

Listing 3.3: Highpass FIlter

In 3.3 we define the type of filter we use in our project and specify starting cut-off frequency. So in this case anything below 20Hz is attenuated. The .toMaster(); means that we send the output to the master output (e.g what will be heard through speakers, headphones).

Delay Effect

One of most basic and perhaps the most used of all effects is that of time delay. Also known as an echo filter, is the building block of more complex effects such as reverb. In the listening space, what we hear consists of a direct sound from the sound source as well as the sound waves reflected off the walls and objects in the room. It arrives at our ears with various amount of time delay and different attenuation. The reverberation characteristics of the listening space that we associate with a room, hall, cathedral, are a result of these multiple reflections.

The filter that simulates a single delay line is also called an FIR comb filter, which adds to the input signal $x(n)$ an delayed and attenuated copy of itself $ax(n - D)$.

$$y(n) = x(n) + ax(n - D)$$

A filter that adds infinite number of successive echoes that simulates endless reflections and reverberating nature of a room is called an IIR comb filter.

$$y(n) = ay(n - D) + x(n)$$

Using Tone.js we create a delay effect, using JavaScript with one line of code:

```
1 var delay = new Tone.PingPongDelay({"feedback":0.4,"wet":1}).receive("
    delay").toMaster();
```

Listing 3.4: Delay Effect

The code at 3.4 creates a PingPong feedback delay effect, which means that an echo will be played in one channel and the next on the opposite channel(right and left channels of a stereo system). We specify the amount of feedback and define that the output will be a complete “wet” signal, meaning that the output will be entirely the effected signal.

Chorus Effect

Many other effects are made out of the time delays when combined with some modulation. Chorusing is an imitation of a group of musicians playing the same piece simultaneously, which are more or less synchronised but with small variations on timing and strength. In the Chorus effect a simulation of this effect is accomplished by varying multiple copies of time delays and amplitudes slowly and randomly.

Effect	Delay Range (ms)	Modulation
Resonator	0...20	None
Flanger	0...15	Sinusoidal (≈ 1 Hz)
Chorus	10...25	Random
Slapback	25...50	None
Echo	> 50	None

Figure 3.2: Variations on Delays

In the script we use Tone.js to add a Chorus effect with the following line of code:

```
1 var chorus = new Tone.Chorus({"frequency":4, "delayTime":10,"wet":1,"
    spread":0}).receive("chorus").toMaster();
```

Listing 3.5: Chorus Effect

In 3.5 we create a Chorus effect. We define the frequency of the low-frequency oscillator (LFO) that modulate the delay lines. We then specify the delay time to be set at 10ms. Here we also want an entirely "wet" signal in the output. Spread means that the effect will be played in mono.

The next effect implemented in this application is a reverb. Reverb simulates the reverberation of many reflections of a sound that happens in a room. It differs from a delay or echo effect in that the later implies a distinct delayed version of the sound. A reverb on the other hand, arrives at such a small period of time that the reflection is not perceived as a copy of the original sound. The reflections are generally a little weaker, as it travels and the sound energy gets absorbed by the walls and objects inside the room. However the effect of these series of reflections are still audible.

```
1 var reverb = new Tone.Freeverb({"wet":1}).receive("reverb").toMaster()
    ;
```

Listing 3.6: Reverb Effect

Sound Source

Next we define the sound source. For these applications we investigate user interaction with musical content. Here a 'music player' will be implemented that reads .mp3 files. The sound source will be a selection of existing songs that users can choose from. Additionally, it will

allow users to choose any song or sonic material from .mp3 files that they can themselves provide.

For the player, the imitation of effects provided by playback modes of analog tape recorders is included. These effects are monitoring of playback speed. During a faster playback speed the duration the audio file is shortened, but it also results on an audible increase in pitch of the sound. While a slower playback speed increases the duration of the audio file, the pitch of the sound is lowered. Additionally, in tape recorders users are have the possibility to play the tape backwards, this feature is also included in this application. Tone.js has an object that have all these features already implemented, called Tone.Player.

```
1 var player = new Tone.Player({
2             "url" :  "sounds2/funtonpige.mp3",
3       "loop" : true,
4     }).connect(filter);
```

Listing 3.7: Song Player

In 3.7 we specify the URL for the .mp3 file, which in this case located inside a folder named 'sounds'. We define that the song will loop once it's over, this is accomplished by setting "loop" to be "true". In the end instead of sending the output directly to the 'master out', the output of this player is connected to the 'filter' 3.3 effect that was implemented earlier.

User Interaction

Users will interact with the application using the HTML elements provided by the NexusUI.js library. In the web page users can use the computer's mouse if the application is open with a computer. If the application is open in a mobile phone, users can interact with the elements through the device's touchscreen capabilities. For this interface users will have access to toggle switches, buttons and dials, that will provide similar functionality found in guitar effect pedals, and effect racks. Each of these NuxusUI elements will be mapped to different parameters of the various effects included on the application. This interface makes extensive use of the one-to-one approach for mapping [**?**]. However, there is one instance that resets all the parameters and that can be loosly considered a one-to-many mapping.

The block diagram shows the audio chain of the audio engine of this interface. As can be seen, Tone.Player has it's output connected to the Filter. The filter then effects the sound and sends the result to all the remaining effects: Delay, Chorus and Reverb. The pink circular lines indicate the various sound effect parameters users can control.

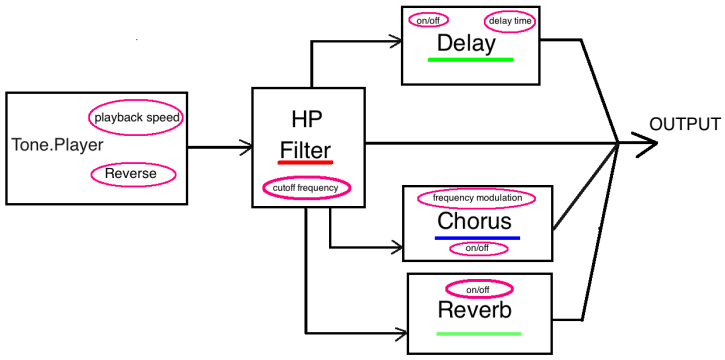


Figure 3.3: Block Diagram of SNCKT EFFECTS Audio Chain

NexusUI objects are created as HTML elements. Providing specifications such as 'nx' (which NexusUI object to create), 'id'(the object's id to be accessed in JavaScript functions), the object's dimensions with 'width' and 'height', and 'label' (name that will appear with the object in the webpage). It is also possible to specify the object's output value range with 'min' and 'max'.

```
1    <canvas nx="toggle" id='delay' label="delay" width="60" height="50
         "></canvas>
2    <canvas nx="dial" id="dial2" width="60" height="60" min="0.1" max=
         "2"></canvas>
3    <canvas nx="button" id='pitch' label="pitch" width="60" height="50
         "></canvas>
```

Listing 3.8: NexusUI Objects

Functions are assigned to these NexusUI objects with JavaScript. The code in 3.9 shows how to use a widget's output values to control the parameters of the sound processing modules. The example shows how the 'dial2' output is mapped to the 'delay time' (dTime) parameter of the delay effect.

```
1  nx.onload = function(){
2
3      delay.on('*',function(data){
4                  if(data.value===1){delaySend.gain.value=0}else{
                       delaySend.gain.value=-100}
5              });
6
7                  dial2.on('*', function(data) {
8                  dTime.setValueAtTime(data.value);
9              });
10
11             }
```

Listing 3.9: mapping to sound processing parameters

In 3.9 'delay.on' refers to the output values of the toggle widget. Here it is mapped to the 'gain' amount of sound that is sent to the delay. The result of this mapping is perceived as an 'on/off' effect that triggers the effect on and off. In 'dial2.on', the output value of 'dial2' is mapped to the delay time (dTime) of the Delay effect created with Tone.js.
A similar mapping approach is implemented for the other user-controlled parameters, such as chorus, reverb, filter, reverse and playback speed. Please refer to Appendix section for the complete code.

In music players users have the possibility to select different songs contained in some kind of song list. Ultimately, the goal would be that users could interact with their favourite songs, and choose songs from their own song lists. However, this project takes the approach of "sonic sketching" [Erkut et al., 2015] and sketches such functionality.

Next in 3.10, a list which will contain the songs that users can choose from is created. This is accomplished by specifying a HTML element called 'selector'. Inside the <selector> element we specify each option available in the list with the <option> tag. First specify the

'value' of the option is defined, in this case a string containing the URL to the .mp3 files. Then, a name for each option is given.

```
    <select id="selector">
  <option value="sounds2/funtonpige.mp3">SkandiMann-Funtonpige</option
      >
  <option value="sounds2/colors.mp3">Kasper Vegas-Colors</option>
  <option value="sounds2/jaruooj.mp3">Alternative Rock</option>
              ...
              ...
              ...
</select>
```

Listing 3.10: Song Selector

A button will be used to trigger the function of actually changing the song. A HTML <button> element is created, and to this button we assign to it a call to the function which is handled by the JavaScript code. Using the object "onclick" we assign the name of the JavaScript function, this will make that when the button is clicked the function will be executed. The "style" is an object inside HTML that is used for defining custom changes in the appearance of the element.

```
<button style="(custom styling...)" onclick="songSelector()">Change</
    button>
```

Listing 3.11: Song Selector Button

The JavaScript function executed on click is as follows 3.13:

```
function songSelector() {
    var x = document.getElementById('selector').value;

    player.load(x); //player.load() refers to a function in Tone.
        Player that loads a different audio file to the player.

}
```

Listing 3.12: Example of songSelector function to change song

The possibility for users to choose their own song to play is also motivated by the fact pointed out in [Krause et al., 2015], that choosing music has a positive affect on the listener.

Further, their findings show that devices that allow for personal input are met with positive consequences.

Even though users don't access their own personal playlists, the interface allows users to post a URL link to load a different .mp3 file. This allows users to choose the sonic content that they will listen to. Indeed, this functionality might not be as convenient for choosing songs as other music players and streaming systems provide (e.g iTunes, Spotify, YouTube, SoundCloud). So other solutions might have to be considered in the future.

```
1 function changeLink(){
2     var userInput = document.getElementById('userInput').value;
3     var lnk = document.getElementById('lnk');
4     lnk = userInput;
5     player.load(lnk);
6 }
```

Listing 3.13: Example of songSelector function to change song

3.2 SNCKT MOVE

This app investigates the possibility of enabling a higher degree of control over sound generation for user interaction with music content. In this case users would have access to various parameters that control sound synthesis. This particular application implements a granular synth that users can play with.

3.2.1 Granular Synthesis

Granular Synthesis is based on a notion about the nature of sound. Quantum physics has shown that sound can be atomically reduced into physical particles, as globules of sonic data [Wiener, 1964]. The notion of the quantum of sound was proposed in 1947 by the physicist Dennis Gabor [Gabor, 1946] where in his theory a granular representation could describe any sound. This evolved into a sound synthesis method where these sound particles are referred to as grains. Gabor's first intent with this method was to reduce the amount of data to convey audio human communication. Xenakis, [Xenakis, 1971] saw potential for this method to be applied in music and in his early works with granular synthesis. He used to manually

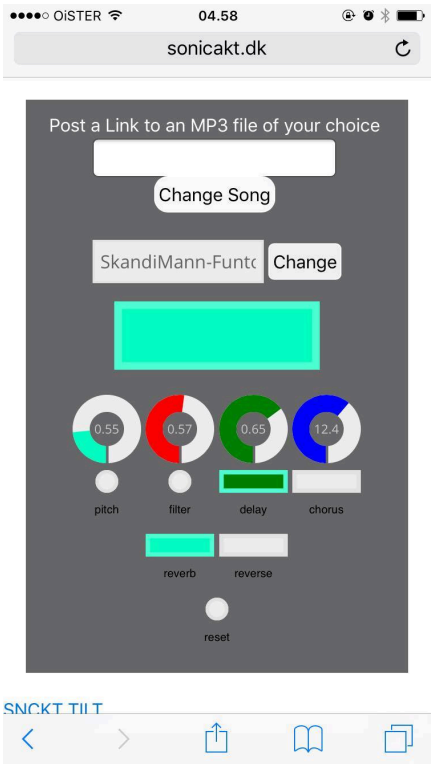


Figure 3.4: SNCKT EFFECTS Interface displayed on a smartphone

slice magnetic tape into small segmented pieces, rearranging and taping these segments together and forming new sonic and musical structures. Curtis Roads, [Roads, 2001][Roads, 1988] started implementing this technique on computers back in 1978, from this Barry Truax [Truax, 1990] started developing techniques to create granular synthesis that could perform in real-time. First realised in 1986, it was then used extensively in his musical compositions. From this point, as computers became more efficient granular synthesis became more accessible to an increasing number of musicians and sound artists.

A grain consists of a small portion of sonic data, in granular synthesis it's duration usually varies from 1 to 100 ms, which approaches the minimum perceivable event for time duration. Multiple grains may be layered on top of each other(this technique results into what is often referred to as clouds), played back at different speeds, phases, amplitude, frequencies. For instance if we take a sampled signal we can take multiple portions of this signal and play those back in different orders and on top of each other.

When $x(n)$ and $y(n)$ are the input and output signals we have

$$g_k(i) = x(i + i_k)w_k(i) \tag{3.1}$$

Where $g_k(i)$ are the grains extracted from the input signal with the window function $w_k(i)$ of length L_k from $i = 0, ..., L_{k-1}$. Here the time instant i_k indicates the point where the segment (grain) is extracted, while L_k determines the amount of the signal extracted. The window $w_k(i)$ waveform ensures fade-in and fade-outs and affects the frequency content of the grains. In this case while long grains tend to maintain the timbre identity of the input signal, short grains acquire a pulse-like quality getting more distinct from the input signal.[Zolzer, 2002]

A synthesis formula could be described as given by

$$y(n) = \sum_k a_k g_k(n - n_k) \tag{3.2}$$

Where a_k is an amplitude coefficient while n_k is the time instant the grain is placed in the Output signal.

Tone.js' API already includes a granular synth. With this synthesis technique implemented, users are allowed to change speed and pitch independently from each other (in contrast to the implementation in SNCKT EFFECTS). For this project, we create a Granular Synth, that is wrapped inside an Amplitude Envelope that gets triggered by a Sequencer.

```
1  var player = new Tone.GrainPlayer({
2              "url" :  "sounds2/fourhy.mp3",
3       "loop" : true,
4           "grainSize":0.01,
5           "overlap":0.05
6     }).toMaster();
7
8     var ampEnv = new Tone.AmplitudeEnvelope({
9   "attack": 0.5,
10  "decay": 0.2,
11  "sustain": 0.2,
12  "release": 0.5,
```

```
                "attackCurve":"sine",
                "releaseCurve":"sine"
}).toMaster();

var loop = new Tone.Sequence(function(time, col){
                 var column = [col];
      for (var i = 0; i < 1; i++){
               if (column[i] === 1){
                   player.connect(ampEnv).start("+0.2",player.toSeconds
                       (Position));
                   ampEnv.triggerAttack("+0.2");
        }
            }
    }, [1], "4n");
```

Listing 3.14: Granular Synth

In 3.14 we create a GrainPlayer (granular synthesis engine) specifying the sample to use, in this case an .mp3 file. Set "loop" to be true and specify starting "grainSize" and "overlap" values. Then we create an Amplitude Envelope, providing values to attack, decay, sustain, and release values. Further we specify the envelope to have a sine like shape. Last in the sound synthesis, we create a Sequencer function, that will trigger and start the granular synth wrapped inside an amplitude envelope.

User Interaction

This app levareges 'tilt motion' of devices to control the synthesis parameters. This tilt motion data is generated with device orientation using accelerometer data. It works on mobile devices and it also works on laptop computers, while for the later such interaction might not be so practical. The values provided by the accelerometer along the x axis go from -x to x, this represents the front to back motion of the device (pitch). For the y axis, the motion is given from -y and y and represents the left to right motion of the device (row).

The NexusUI library provides a widget in HTML element that allows access to tilt motion. It is created using the following code:

```
<canvas nx="tilt" id="tilt1" width="60" height="60"></canvas>
      <canvas nx="tilt" id="tilt2" width="60" height="60"></canvas>
```

```
3       <canvas nx="tilt" id="tilt3" width="60" height="60"></canvas>
4
5               </div>
```

Listing 3.15: Addig Tilt Objects

Each of these tilt objects are mapped to various parameters of the granular synth. Even though these objects use the same data from the accelerometer, they also provide a function to toggle them "active" or "inactive". This becomes a customisable mapping where the user can use accelerometer to control a desired combination of parameters to be affected. This interface uses a one-to-one or one-to-many mapping according to user input.

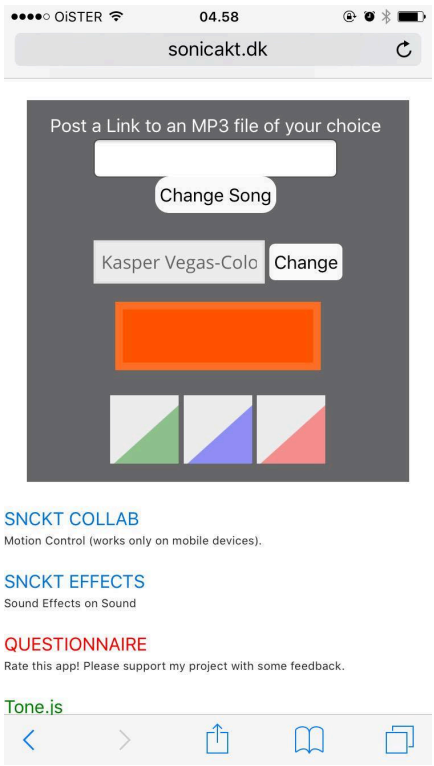


Figure 3.5: SNCKT MOVE Interface displayed on a smartphone

3.3 SNCKT COLLAB

The third application is called SNCKT COLLAB. This interface would allow for multiple users to collaborate together to form their listening experience. In this version, users would choose between individual tracks of a given song. In this case songs are provided as individual tracks that compose the entire piece. A user can select one track at a time, while other users can participate using other tracks, they can collaborate and play the individual tracks together forming a unified musical experience. The approach taken here, follows that of sonic sketching [Erkut et al., 2015]. Here we sketch such functionality. The goal is also to provide a more performance and gestural oriented interaction leveraging the accelerometer data in mobile devices. This application demands interaction for it to produce the musical content that users will listen to. The idea is that for each individual track a user triggers different notes according to their motion holding the device. One other important factor considered is that of synchronisation between users. So a global tempo is set where different types of motions trigger individual sounds at specified time instances.

First we create a NexusUI.js "motion" widget that will give us access to accelerometer data from the user's smartphone. This object provides x, y, z data ranging from -1 to 1.

```
1 <canvas nx="motion" width="200" height="200" label="tap twice to make
    sound"></canvas>
```

Listing 3.16: Adding Motion Object

Then we create samplers that contain the individual sounds and notes for each track. In this case we create a Drum set:

```
1    var kick = new Tone.Sampler("sounds/kick.mp3").toMaster();
2          var hihat = new Tone.Sampler("sounds/hihat.mp3").toMaster();
3          var snare = new Tone.Sampler("sounds/snare.mp3").toMaster();
4          var shaker = new Tone.Sampler("sounds/shaker.mp3").toMaster
               ();
5          var rim = new Tone.Sampler("sounds/rim.mp3").toMaster();
```

Listing 3.17: Drum Set

Then depending on the values of the sensor we set the individual notes that should be triggered with fixed timed instances. Tone.js makes it simple to quantize events. So in the following case, the kick drum is alligned to 4n (quarter note). With the individual notes set

as quantized events, this defines a specific structure that will allow multiple users synchronize with each other.

```
motion1.on('*',function(data){

                        if(data.x > 0.2){kick.triggerAttack(0,"@4n");
                                         kick.player.set({"playbackRate"
                                             :chain.next()});
                                         chain.value = "beginning"};
                        if(data.z > 0.2){snare.triggerAttack(0,"@2n");
                                         snare.player.set({"playbackRate
                                             ":chain.next()});
                                         chain.value = "beginning"};
                        if(data.x < -0.3){hihat.triggerAttack(0,"@16n");
                                          hihat.player.set({"
                                              playbackRate":chain.next()
                                              });
                                         chain.value = "beginning"};
                        if(data.y < -0.2){shaker.triggerAttack(0,"@8n");
                                          shaker.player.set({"
                                              playbackRate":chain.next()
                                              });
                                         chain.value = "beginning"};
                        if(data.y < -0.2){rim.triggerAttack(0,"@6n");
                                          rim.player.set({"playbackRate"
                                              :chain.next()});
                                          chain.value = "beginning"};
          });
```

Listing 3.18: Data mapped to trigger individual sounds

Please refer to the appendix section to see full code and how the mappings are accomplished from a technical point of view.

The web-page containing the interfaces can be acessed through the following link: http://sonicakt.dk/

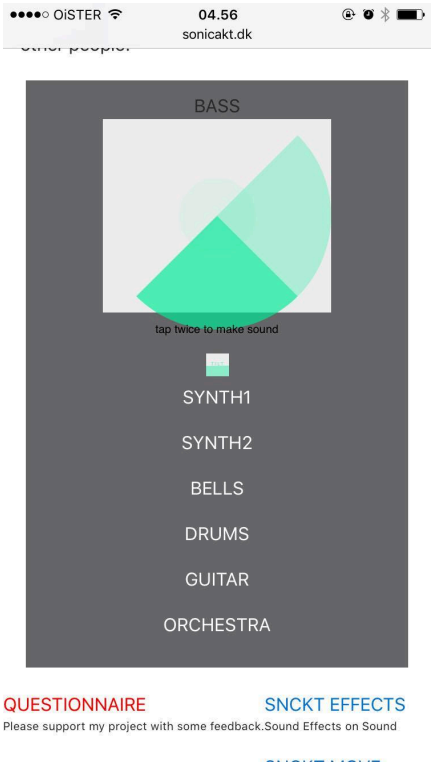


Figure 3.6: SNCKT COLLAB Interface displayed on a smartphone

Chapter 4

Evaluation

To perform evaluation two approaches were taken. A qualitative evaluation similar to Naturalistic Observations, and a second evaluation using a web-based survey with a couple of subjective rating questions to evaluate user's experience quantitatively.

As been pointed out in previous literature [Juslin, 2011], there's a point of view in regards to testing and evaluating people's experience and use of music in everyday life. It proposes that evaluations should move away from the strict lab based experiments and to study user behaviour "in the wild". Indeed there are many disadvantages to this kind of observation, such as defining exact cause of a bahaviour and unpredictable outside variables.

4.1 Naturalistic Observation-based Experiment

In this project, sound and music has a major role in the application and interaction between users. As this application attempts to influence the user's

musical experience. The observations include what the application enable users to experience, how the interface reacts to users performance, how users react to the interface and what is the influence on the environment. In other words, what's evaluated are the multiple affordances of the interface (social, musical, etc), a important term within interactive design evaluations[Tanaka et al., 2012]. For that reason it's important to observe the affordance of the interface in experiments "staged" according to what the application proposes in the musical and social context in regards to listening experiences in everyday life. The approach was to make observations of the participant's performance. In this test I walked around the city centre of Aalborg and asked people to test the application. If people were found in groups and accepted to participate in the experiment, they were asked to immediately test the SNCKT COLLAB interface. The idea was to make observations during their experience and if they comply their interaction would be video recorded. Not many people were willing to be video recorded, but three different groups of participants agreed to be video recorded. Luckily, those groups were of apparent distinct age differences.

Observations included:

access to interaction, if participants intuitively understood how to start playing or if they required some sort of explanation.

their reaction to sound, any indication of behaviour related to the sonic feedback.

attitude and interaction with other participants, any behaviour in relation to interaction between participants.

Groups of people were approached by asking if they were willing to participate in a musical experiment to test a web-app. As the groups complied to the experiments, they were advised to access the web page. This was done by verbally providing the address (URL) to the web page. They were asked to try the interfaces for themselves, in the case of them not seeming to understand how to start playing they were given instructions on how to play.

Few participants understood rather intuitively how to produce sounds with the interface. However there were also cases were the author had to explain the functionality of the applications.

It was observed that many of the participants, demonstrated surprised expressions on their first encounters with each sound. Sometimes it was observed that it was a little disturbance towards some of the sonic elements. As participants had to turn up the volumes of their devices to maximum (due to environmental sounds) in order to hear anything. This seemed to give a distortion (and noisy) characteristic to the sonic elements.

In a successful sonic interaction design, the social affordances in the music would afford joint entrainment, here it refers to synchronisation between participants to an external perceived rhythm, it allows to experience music with others in an intimate way [Krueger, 2010]. It can be observed from the experiments with the interface SNCKT COLLAB as can be seen in the recorded process, that the interface provided this form of social affordance. Participants from all three groups presented some sort of entrainment while interacting with the interface. It can be seen that participants nodded their heads while playing with the interface and affirming to each other some sense of the rhythm

captured. It was also observed that participants had some 'affirmation of enjoyment' as they smile to each other, also selecting different instruments from one and other, attempting to play and synchronise something together.

Figure 4.1: Naturalistic Observations of users with SNCKT

4.2 Survey based experiment

For a survey to gather quantitative data of participant's subjective experience with the interface, the idea was to get users to test the interfaces on 'their own environment'. The approach taken was also similar to that of a naturalistic observation, in that the environment in which test subjects participate is "their own". This approach is inspired by how [North et al., 2004] conduct their experiments. They make an observation that very few studies have investigated

people's musical experiences in everyday circumstances. The use and the degree of which people engage with music, is dependent on the contexts in which they hear it [North et al., 2004]. Their experiments involved participants who owned mobile phones. Participants received one text message per day during a course of 14 days, requiring them to complete a questionnaire regarding their daily listening experiences. More recently, a similar experiment was conducted in [Krause et al., 2015]. For the case of the interface presented by this paper, this seemed as an attractive approach to evaluate how people would interact with the interface in the context of everyday circumstances. Since the project has the goal of providing sonic interactive possibilities with musical material in the context of everyday listening experiences. From the point of view of this paper, this also appeared to be a novel approach worth investigating for testing interfaces for musical expression.

Three different surveys were made to evaluate user experience for each interface. A web-based survey was prepared using Google Forms. It contained seven subjective rating questions using a 5 level Likert-Scale (ranging from 5-Strongly Agree to 1-Strongly Disagree).

The procedure was relatively simple. An instruction was written in text and then shared in social media. The instruction included asking people to test each interface and rate them accordingly.

A total of 14 participants are included in this experiment (9 male, 7 female). Half of the participants reported as being Amateur Musicians, while the rest reported to have little to no experience with playing a musical instrument. The ages were 5 (35-44), 7 (25-34), and 2 (18-24).

The app was easy to use.	3.42
The app was easy to understand.(what each control means, how it affects the sound, etc...).	3.71
The experience was enjoyable.	3.78
The experience was musically satisfying.	3.35
I would consider using this app in the future.	3.28
I would consider using this app if I could play it with songs from my favourite artists.	3.38
I believe this app could make my music listening experiences more enjoyable.	3.5

Figure 4.2: Results of Likert-Scale questions for the Survey

Participants who identify themselves as amateur musicians, were the participants that gave the highest rates for a musically satisfying experience. While level of enjoyment was overall, rated high for both participants with none or little experience playing an instrument and amateur musicians. Amateur musicians also rated high on considering to use the application in the future. While non-musically trained participants had a neutral or low rating on that regard.

From the data gathered an interesting observation can be made. That users with no experience or little experience with playing an instrument generally provided positive ratings to their enjoyment of the experience even though their ratings on a 'musically satisfying experience' and ease of interaction were neutral to low. That could be seen for both their experience during the experiment and their future listening experiences. This is an interesting possible indication (perhaps no surprise) that unexperienced users have a tendency to enjoy interaction while listening to music, even though the interaction isn't necessarily easy or musically satisfying.

Please refer to Appendix section for full data visualisation.

One major downside to the evaluation performed is that since it was based in a

naturalistic observation approach, the experiment suffered from uncontrollable variables. Even though participants were instructed to respond the individual questionnaires that rated each interface, participants either rated their overall experience of the interfaces through only one of the three questionnaires. This had a negative impact towards establishing a precise rating between each of the three interfaces. However, it is possible to identify that users overall rated an enjoyable experience with the interfaces. This indicates that there is apparent interested from both musically trained and non musically trained participants in regards to user interaction with musical content. With that said, more degree of control has to be implemented in further experiments. For instance if the approach to be taken were that of naturalistic observation, it would be important to have participants formally sign up to the experiment as in [North et al., 2004] and [Krause et al., 2015]. Furthermore, from the point of view of this paper it would be very beneficial and practical to conduct controlled lab experiments, to closely observe ease of interaction, as well as other aspects of user interaction with the interface. Perhaps do so even before conducting 'outdoors' experiments.

Chapter 5

Conclusion

The SONICAKT interfaces were designed to investigate 3 different approaches defined by interactive music systems (Transformative, Sequenced, Generative), to explore the possibility of a platform where musicians would share musical content that their audience would interact with. It used research in mobile music making and the web, together with concepts of active listening and audience participation, to propose a novel approach of interactive music listening on the web. This approach is novel in the context of everyday listening where passive listening is much predominant.

There's definite space for improvement, whole new research could be done for the improvement of the systems proposed, to find ways to address the issue of sonic interaction in the context of everyday music listening experiences. Indeed the interfaces implemented and their evaluation during this research might have resulted in works of a very content-centric nature, with few exceptions in the attempts of evaluating musical and social affordances. There is much room for improvement in both implementation and evaluation. This project provides or opens up interdisciplinary challenges and problems to be addressed from the perspectives and methodologies of the various related disciplines, for further

development to form a successful interactive music listening platform. Such platforms of musical interaction aimed at the audience have much potential for engagement from both sides (musician and audience). Where musicians create can entire interactive musical pieces or just allow for users to apply audio effects to their songs. By doing so, it could bring high level of enjoyment for their audience outside the concert space. Ultimately this research provides an alternative to the current state of such active consumption of music, towards an active and perhaps shared participation in musical experiences.

Bibliography

Arango, J. J. and Giraldo, D. M. (2016). The smartphone ensemble. exploring mobile computer mediation in collaborative musical performance.

Deusany, A., Lee, S. W., and Essl, G. (2016). Understanding cloud service in the audience participation music performance of crowd in c[loud].

Erkut, C., Serafin, S., Hoby, M., and Sarde, J. (2015). Product sound design: From, function, and experience.

Gabor, D. (1946). Theory of communication.

Goto, M. (2007). Active music listening interfaces based on signal processing.

Gualtiero, V. and Camurri, A. (2011). A system for embodied social active listening to sound and music content.

Hunt, A. and Wanderley, M. M. (2002). Mapping performer parameters to synthesis engines.

Hunt, A., Wanderley, M. M., and Kirk, R. (2000). Towards a model for instrument mapping in expert musical interaction.

Juslin, P. N. (2011). Music and emotion: Seven questions, seven answers.

Krause, A. E., North, A. C., and Hewitt, L. Y. (2015). Music-listening in everyday life: Devices and choice.

Krueger, J. W. (2010). Doing things with music.

Lee, S. W. (2012). Audience participation using mobile phones as musical instruments.

Levin, G. (2001). ?dialtones: A telesymphony? final report. http://www.flong.com/projects/telesymphony/.

Mann, Y. (2014). Interactive music with tone.js.

Michon, R., Smith, J. O., Wright, M., Chafe, C., Granzow, J., and Wang, G. (2017). Passively augmenting mobile devices towards hybrid musical instrument design.

Miranda, E. R. and Wanderley, M. M. (2006). *New Digital Musical Instruments: Control And Interaction Beyond the Keyboard (Computer Music and Digital Audio Series)*. A-R Editions, Inc.; 1st edition (July 2006).

Nicolas, A. (2015). From ugnayan to udlot-udlot : The music of jose maceda musical ideas in new music in southeast asia.

North, A. C., Hargreaves, D. J., and Hargreaves, J. J. (2004). Uses of music in everyday life.

Oh, J. and Wang, G. (2011). Audience-participation techniques based on social mobile computing.

Orfanidis, S. J. (2010). *INTRODUCTION TO Signal Processing*. previously published by Pearson Education, Inc. Prentice Hall, Inc.

Roads, C. (1988). *Introduction to Granular Synthesis*. Computer Music Journal. MIT Press 12(2):11-13.

Roads, C. (2001). *Microsound*. The MIT Press, 2001 Massachusetts Institute of Technology.

Robaszkiewicz, S. and Schnell, N. (2015). Soundworks a playground for artists and developers to create collaborative mobile web performances.

Rowe, R. (1993). Interactive music systems. https://wp.nyu.edu/robert_rowe/text/interactive-music-systems-1993/chapter-1-interactive-music-systems/.

Schiemer, G. and Havryliv, M. (2006). Pocket gamelan: tuneable trajectories for ying sources in mandala 3 and mandala 4.

Schnell, N., Robaszkiewicz, S., Bevilacqua, F., and Schwarz, D. (2015a). Collective sound checks exploring intertwined sonic and social affordances of mobile web applications.

Schnell, N., Saiz, V., Barkati, K., and Goldszmidt, S. (2015b). Of time engines and masters an api for scheduling and synchronizing the generation and playback of event sequences and media streams for the web audio api.

Tanaka, A., Altavilla, A., and Spowage, N. (2012). Gestural musical affordances.

Taylor, B. (2017). A history of the audience as a speaker array.

Taylor, B., Allison, J., Conlin, W., Oh, Y., and Holmes, D. (2014). Simplified expressive mobile development with nexusui, nexusup and nexusdrop.

Truax, B. (1990). Composing with real-time granular sound.

Wanderley, M. M. (2001). Gestural control of music.

Wang, G. (2014). Ocarina: Designing the iphone's magic flute.

Wang, G., Essl, G., and Penttinen, H. (2008a). Do mobile phones dream of electric orchestras?

Wang, J., Deng, H., Yan, Q., and Wang, J. (2008b). A collaborative model of low-level and high-level descriptors for semantics- based music information retrieval.

Wiener, N. (1964). Spatio-temporal continuity, quantum theory and music.

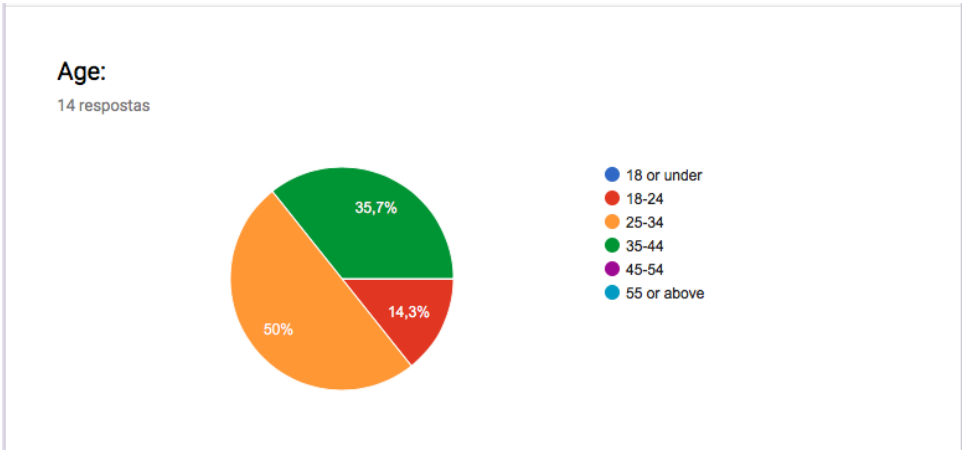
Xenakis, I. (1971). *Fromalized Music, Thought and Mathematics in Composition*. Bloomington: Indiana University Press.

Zolzer, U. (2002). *DAFX: Digital Audio Effects*. 2002 John Wiley and Sons, Ltd.

Appendix A

Appendix A

A.1 Survey Evaluation



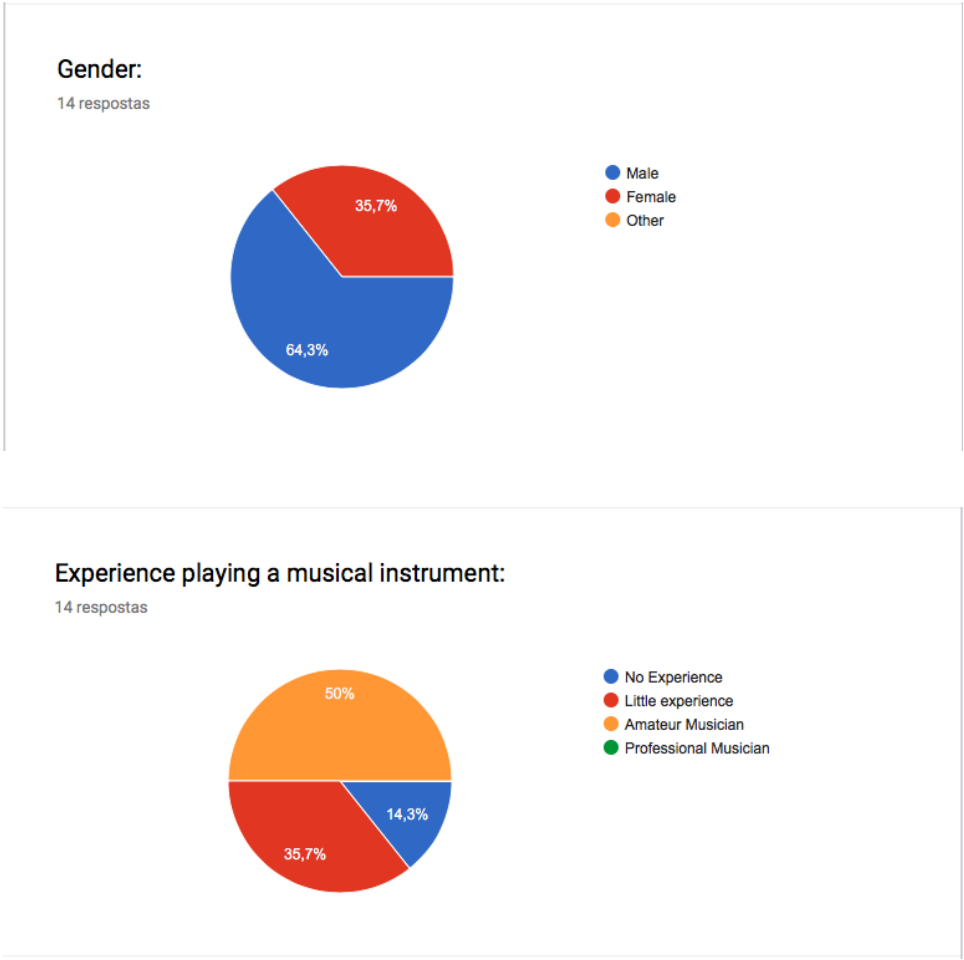
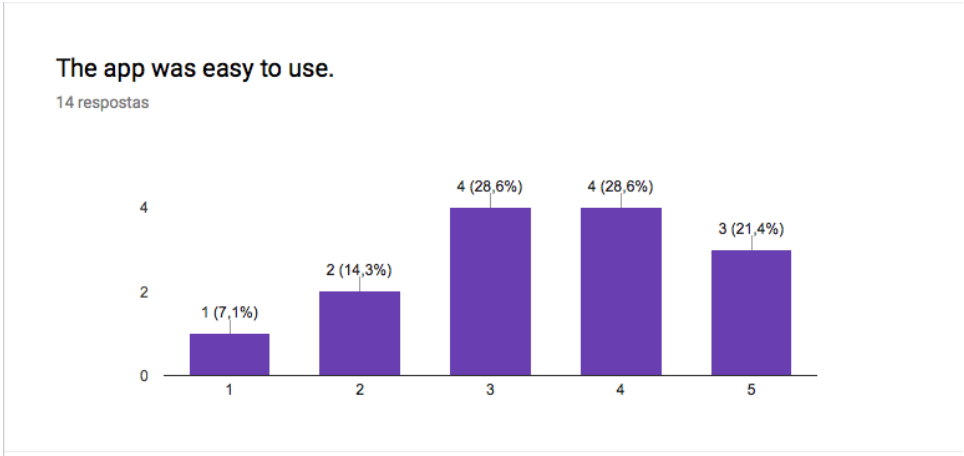
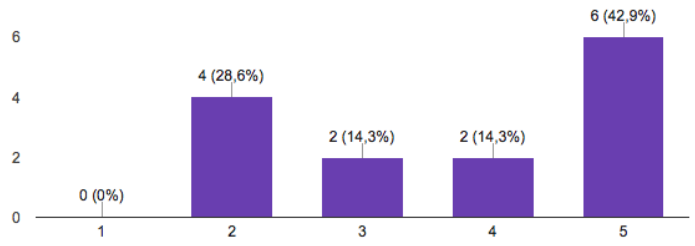


Figure A.1: Musical Training



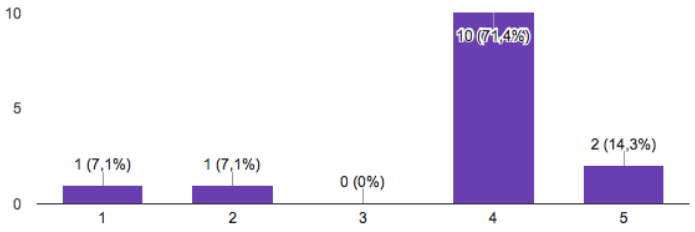
The app was easy to understand.(what each control means, how it affects the sound, etc...)

14 respostas



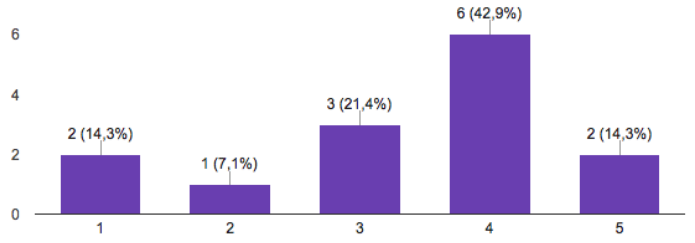
The experience was enjoyable.

14 respostas



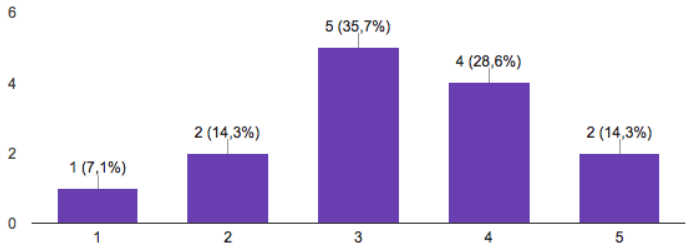
The experience was musically satisfying.

14 respostas



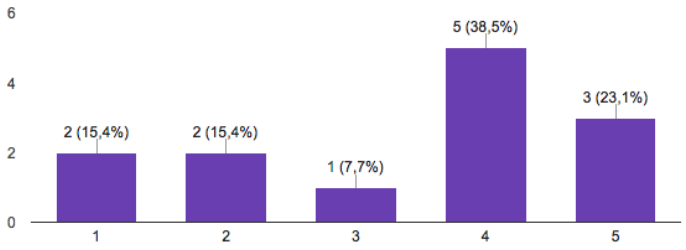
I would consider using this app in the future.

14 respostas



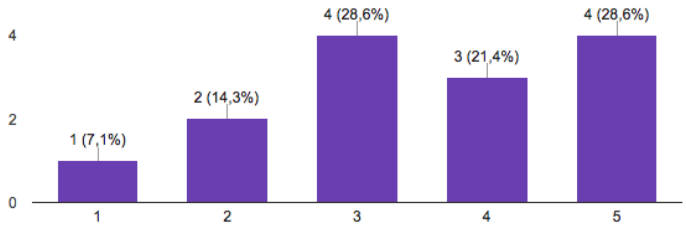
I would consider using this app if I could play it with songs from my favourite artists.

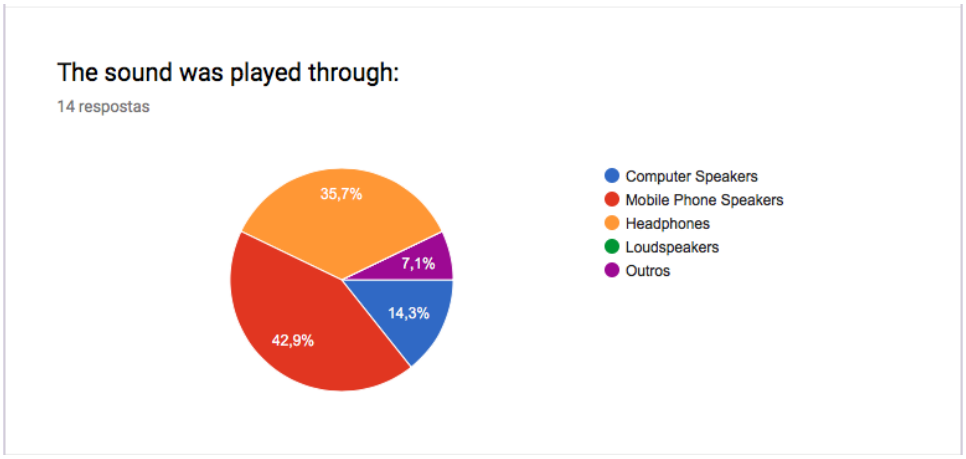
13 respostas

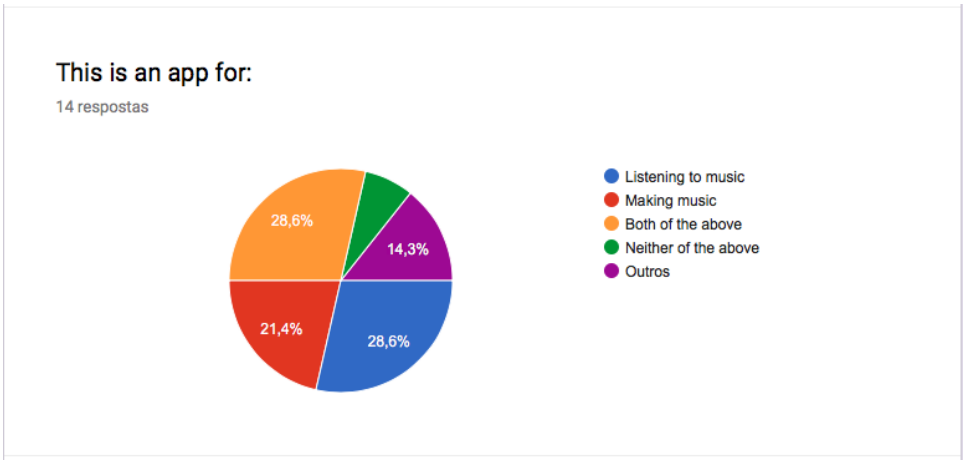


I believe this app could make my music listening experiences more enjoyable.

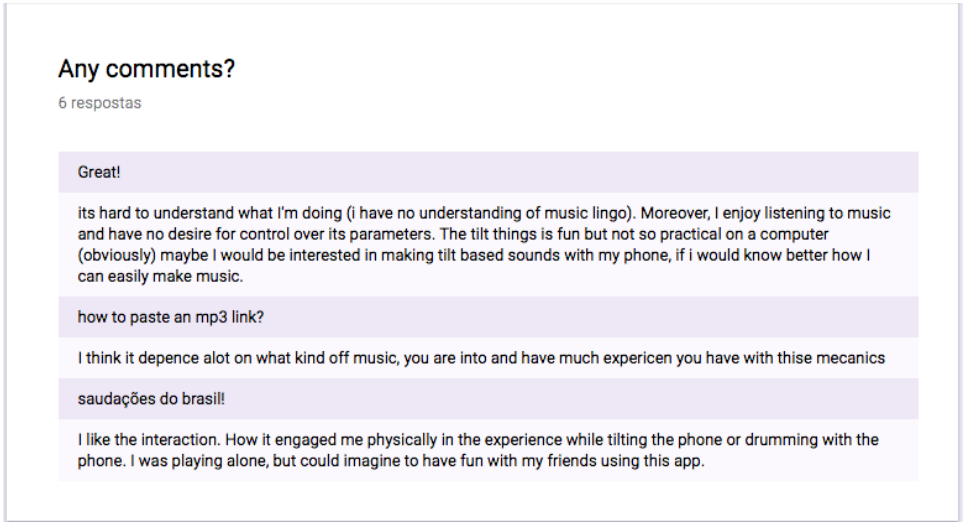
14 respostas







Age:	Gender:	Experience play	Did you open t	The app	The app	The sound was	The ex	The exp	This is an app	I would c	I would c	I belie	Any comments?
35-44	Female	Amateur Musici	It didn't work! (	1	2	Mobile Phone :	1	1	Both of the abc	4	5	2	how to paste an m
25-34	Male	No Experience	Computer	2	2	Computer Spe	2	1	Making music	1	1	1	its hard to understa
35-44	Male	Amateur Musici	Computer	2	5	Headphones	4	4	Making music	3	1	3	
18-24	Female	Little experienc	Mobile Phone	3	4	Mobile Phone :	4	2	Neither of the a	2		2	
25-34	Female	Little experienc	Computer	3	2	Computer Spe	4	4	Both of the abc	2	3	3	
18-24	Male	Little experienc	Ipad	3	3	Ipad speakers	4	3	Listening to mu	3	2	3	I think it depence a
25-34	Male	Amateur Musici	Computer	3	5	Headphones	4	4	Both of the abc	5	4	5	saudações do bras
25-34	Male	Little experienc	Mobile Phone	4	4	Mobile Phone :	4	3	Listening to mu	3	4	4	
35-44	Female	Amateur Musici	Mobile Phone	4	2	Mobile Phone :	5	4	Listening to mu	3	5	5	
25-34	Male	Amateur Musici	Mobile Phone	4	5	Mobile Phone :	4	4	Making music	4	4	5	
35-44	Male	Little experienc	Mobile Phone	4	3	Headphones	4	3	Both of the abc	3	2	4	I like the interactio
25-34	Female	No Experience	Mobile Phone	5	5	Mobile Phone :	5	5	Listening to mu	5	5	5	Great!
35-44	Male	Amateur Musici	Computer	5	5	Headphones	4	4	the app can be	4	4	4	
25-34	Male	Amateur Musici	Mobile Phone	5	5	Headphones	4	5	Manipulation n	4	4	3	



Any comments?

6 respostas

Great!

its hard to understand what I'm doing (i have no understanding of music lingo). Moreover, I enjoy listening to music and have no desire for control over its parameters. The tilt things is fun but not so practical on a computer (obviously) maybe I would be interested in making tilt based sounds with my phone, if i would know better how I can easily make music.

how to paste an mp3 link?

I think it depence alot on what kind off music, you are into and have much expericen you have with thise mecanics

saudações do brasil!

I like the interaction. How it engaged me physically in the experience while tilting the phone or drumming with the phone. I was playing alone, but could imagine to have fun with my friends using this app.

Appendix B

Appendix B

B.1 SNCKT EFFECTS CODE

```
<!DOCTYPE html>

<html>

    <head>
        <title>: SNCKT EFFECTS :</title>
        <meta name="viewport" content="width=device-width, initial-
            scale=1, maximum-scale=2, user-scalable=yes">
        <link href="bootstrap.css" rel="stylesheet"/>
        <script src="nexusUI.js"></script>
    <script src="Tone.js"></script>
        <script src="StartAudioContext.js"></script>
      <script src="jquery.min.js"></script>
    </head>
  <body>
              <script>
              ///////TONE.JS
            StartAudioContext(Tone.context);

              var filter = new Tone.Filter({"type":"highpass","
```

```
                frequency":20}).receive("filter").toMaster();
20
21          var reverb = new Tone.Freeverb({"dampening":6000,"wet":1}).
                receive("reverb").toMaster();
22
23              var chorus = new Tone.Chorus(4, 10, 0.5,{"wet":1,"spread
                    ":0}).receive("chorus").toMaster();
24
25              var delay = new Tone.PingPongDelay({"feedback":0.4,"wet"
                    :1}).receive("delay").toMaster();
26
27
28                var reverbSend = filter.send("reverb", -Infinity);
29
30                var chorusSend = filter.send("chorus", -Infinity);
31
32                var delaySend = filter.send("delay",-Infinity);
33
34                var player = new Tone.Player({
35             "url" :  "sounds2/funtonpige.mp3",  //"https://dl.
                    dropboxusercontent.com/s/ks75d36nu5iqhrr/twohy.wav",
36       "loop" : true,
37     }).connect(filter);
38
39              var frequency = new Tone.Signal(0.5);
40               var dTime = new Tone.Signal(0.5);
41               var rSize = new Tone.Signal(0.5);
42                // var dTime = Tone.ScaleExp(0.2,1);
43
44                var sFQ = new Tone.ScaleExp(20,4000);
45                frequency.chain(sFQ);
46                dTime.connect(delay.delayTime);
47                rSize.connect(chorus.frequency);
48              //  rSize.connect(phaser.frequency);
49                // rSize.connect(reverb.roomSize);
50                sFQ.connect(filter.frequency);
```

```
51
52      </script>
53
54      <script>
55           nx.onload = function() {
56
57          Tone.Buffer.on("progress", function(){
58                start.erase();
59                });
60                Tone.Buffer.on("load", function(){
61                start.init(); start.draw();
62                });
63
64
65                nx.colorize("accent", "#00FFC0");
66                filter.colors.accent="red";
67                delay.colors.accent="green";
68                chorus.colors.accent="blue";
69                filter.set({"value":0});
70                delay.set({"value":0});
71                chorus.set({"value":0});
72                dial1.colors.accent="red";
73                dial1.set({"value":0.1});
74                dial1.draw();
75                dial2.colors.accent="green";
76                dial2.set({"value":0.5});
77                dial2.draw();
78                dial3.colors.accent="blue";
79                dial3.set({"value":0.5});
80                dial3.draw();
81                dial4.set({"value":1});
82                dial4.draw();
83
84            start.on('*',function(data){
85               if(data.value===1){player.start("+0.2")}
86              else{player.stop("+0.2")}
```

```
87                 });
88
89              dial4.on('*',function(data){
90              player.set({"playbackRate":data.value});
91              });
92              pitch.on('*',function(data){
93                player.set({"playbackRate":1});
94                dial4.set({"value":1});
95              });
96
97              filter.on('*',function(data){
98               frequency.setValueAtTime(0.1);
99                dial1.set({"value":0.1});
100             });
101         dial1.on('*', function(data) {
102                     frequency.setValueAtTime(data.value);
103         });
104
105             delay.on('*',function(data){
106               if(data.value===1){delaySend.gain.value=0}else{
                      delaySend.gain.value=-100}
107             });
108                 dial2.on('*', function(data) {
109               dTime.setValueAtTime(data.value);
110             });
111
112             chorus.on('*',function(data){
113               if(data.value===1){chorusSend.gain.value=0}else{
                      chorusSend.gain.value=-100}
114             });
115             dial3.on('*', function(data) {
116                     rSize.setValueAtTime(data.value);
117             });
118
119             reverb.on('*',function(data){
120              if(data.value===1){reverbSend.gain.value=-6}else{
```

```
                reverbSend.gain.value=-100}
121             });
122
123             reverse.on('*',function(data){
124              if(data.value===1){player.set({"reverse":true})}else{
                    player.set({"reverse":false})}
125             });
126
127             reset.on('*',function(data){
128               player.set({"playbackRate":1});
129               dial4.set({"value":1});
130               frequency.setValueAtTime(0.1);
131               dial1.set({"value":0.1});
132               dial1.draw();
133               chorusSend.gain.value=-100;
134               delaySend.gain.value=-100;
135               filter.set({"value":0});
136               delay.set({"value":0});
137               chorus.set({"value":0});
138               dTime.setValueAtTime(0.1);
139               dial2.set({"value":0.1});
140               dial2.draw();
141               dial3.set({"value":0.5});
142               dial3.draw();
143               rSize.setValueAtTime(0.5);
144             });
145
146           }
147     </script>
148
149     <script>
150           ////////////// LINK CHANGER /////////////////
151
152 function changeLink(){
153     var userInput = document.getElementById('userInput').value;
154     var lnk = document.getElementById('lnk');
```

```
155     lnk = userInput;
156     //lnk.innerHTML = lnk.href;
157     start.set({"value":0});
158     player.stop("+0.2");
159     player.load(lnk);
160 }
161
162 function songSelector() {
163     var x = document.getElementById('selector').value;
164     start.set({"value":0});
165     player.stop("+0.2");
166     player.load(x);
167     player.set({"reverse":false});
168     reverse.set({"value":0});
169
170 }
171
172   </script>
173
174         <div class="container">
175         <div id="tester" style="clear:both"></div>
176             <div class="row">
177                         <div class="col-sm-4" style="width:450px;
                                background-color:#686769;padding:10px;margin
                                :20px ;text-align:center">
178          <a id=lnk style="color:white">Post a Link to an MP3 file of
                your choice</a> <br>
179         <input type='text' id='userInput'/>
180 <input type='button' style="border-radius:12px;border:none;padding:3px
    ;background-color:#f9f9f9;border:1px solid #ffffff;" onclick='
    changeLink()' value='Change Song'/>
181                             <br><br>
182
183             <select id="selector">
184   <option value="sounds2/funtonpige.mp3">SkandiMann-Funtonpige</option
        >
```

```
185   <option value="sounds2/colors.mp3">Kasper Vegas-Colors</option>
186   <option value="sounds2/jaruooj.mp3">Alternative Rock</option>
187               <option value="sounds2/onehy.mp3">song4</option>
188               <option value="sounds2/fourhy.mp3">song5</option>
189 </select>
190
191  <button style="border-radius:5px;border:none;padding:3px;background-
        color:#f0f0f0;border:1px solid #ffffff;" onclick="songSelector()">
        Change</button>
192                         <div><p> </p></div>
193       <div>
194       <canvas nx="toggle" id='start' width="180" height="60"></canvas>
195       </div>
196                        <div><p> </p></div>
197
198       <canvas nx="dial" id="dial4" width="60" height="60" min="0.1"
            max="2"></canvas>
199       <canvas nx="dial" id="dial1" width="60" height="60" min="0.1"
            max="1"></canvas>
200       <canvas nx="dial" id="dial2" width="60" height="60" min="0.01"
            max="1"></canvas>
201       <canvas nx="dial" id="dial3" width="60" height="60" min="0.5"
            max="20"></canvas>
202
203                         <div>
204                           <canvas nx="button" id='pitch' label="pitch"
                                width="60" height="50"></canvas>
205                           <canvas nx="button" id='filter' label="
                               filter" width="60" height="50"></canvas>
206                           <canvas nx="toggle" id='delay' label="delay"
                                width="60" height="50"></canvas>
207                           <canvas nx="toggle" id='chorus' label="
                               chorus" width="60" height="50"></canvas>
208                         </div>
209
210                         <div>
```

```
211                          <canvas nx="toggle" id='reverb' label="
                                 reverb" width="60" height="50"></canvas>
212                          <canvas nx="toggle" id='reverse' label="
                                 reverse" width="60" height="50"></canvas>
213                      </div>
214
215                      <div>
216                        <canvas nx="button" id='reset' label="reset"
                                 width="60" height="50"></canvas>
217                      </div>
218
219              </div>
                        <div>
220              <div><a href="http://sonicakt.dk/move.html">SNCKT TILT</
                     a><p style="font-size:10px">Tilt Control</p></div>
221              <div><a href="http://sonicakt.dk/motion.html">SNCKT
                     COLLAB</a><p style="font-size:10px">Motion Control (
                     works only on mobile devices)</p></div>
222                </div>
223
224              <div>
225                <a href="https://goo.gl/forms/atXsSRWTVJIyiOM93" style
                     ="color:red">QUESTIONNAIRE</a><p style="font-size
                     :10px">Rate this app! Please support my project
                     with some feedback.</p>
226              </div>
227               <div>
228                <a href="Tonejs.github.io" style="color:green">Tone.js
                     </a><p style="font-size:10px">For those interested
                     in web-development this app was made with Tone.js
                     and NexusUI.js.</p>
229                <div> <a href="http://www.nexusosc.com/" style="color:
                     green">NexusUI.js</a><p style="font-size:10px"></p>
                     </div>
230              </div>
231        </div>
232
```

```
233       </div>
234
235          </body>
236
237 </html>
```

Listing B.1: SNCKT EFFECTS FULL CODE

B.2 SNCKT MOVE CODE

```
1  <!DOCTYPE html>
2
3  <html>
4
5      <head>
6          <title>: SNCKT MOVE :</title>
7          <meta name="viewport" content="width=device-width, initial-
               scale=1, maximum-scale=2, user-scalable=yes">
8          <link href="bootstrap.css" rel="stylesheet"/>
9          <script src="nexusUI.js"></script>
10     <script src="Tone.js"></script>
11         <script src="StartAudioContext.js"></script>
12     </head>
13
14     <body>
15          <script>
16              ///////TONE.JS
17             StartAudioContext(Tone.context);
18
19               var ampEnv = new Tone.AmplitudeEnvelope({
20   "attack": 0.5,
21   "decay": 0.2,
22   "sustain": 0.2,
23   "release": 0.5,
24                 "attackCurve":"sine",
25                 "releaseCurve":"sine"
```

```
26 }).toMaster();
27
28     var player = new Tone.GrainPlayer({
29             "url" :  "sounds2/fourhy.mp3",  //"https://dl.
                   dropboxusercontent.com/s/ks75d36nu5iqhrr/twohy.wav",
30       "loop" : true,
31           "grainSize":0.01,
32           "overlap":0.05
33     }).toMaster();
34       //position on the player
35              var Position
36
37              var loop = new Tone.Sequence(function(time, col){
38                  var column = [col];
39       for (var i = 0; i < 1; i++){
40                if (column[i] === 1){
41                    player.connect(ampEnv).start("+0.2",player.toSeconds
                          (Position));
42                    ampEnv.triggerAttack("+0.2");
43          }
44              }
45     }, [1], "4n");
46
47          Tone.Transport.start("+0.1");
48       </script>
49
50       <script>
51            nx.onload = function() {
52
53                tilt1.colors.accent="green";
54                tilt2.colors.accent="blue";
55                tilt3.colors.accent="red";
56                tilt1.text="";
57                tilt2.text="";
58                tilt3.text="";
59
```

```
60              start.on('*',function(data){
61                 if(data.value===1){loop.start("+0.2");var d=document.
                       getElementById('tester');d.innerHTML="Tilt your
                       device (sides/up/down) to affect the sound.(Mobile
                       or Google Chrome only!)";start.label=" ";start.draw
                       ();}
62                else{loop.stop();player.stop()}
63                  });
64
65          tilt1.on('*', function(data) {
66                       var scaledy = nx.scale(data.y,-1,1,0.01,0.5);
67                       player.set({"grainSize":scaledy});
68                      var scaledx = nx.scale(data.x,-1,1,0,player.
                           buffer.duration);
69                  Position=scaledx;
70          });
71
72              tilt2.on('*', function(data) {
73                       var scaledy = nx.scale(data.y,-1,1,0.1,2);
74                       var scaledx = nx.scale(data.x,-1,1,20,200);
75                       var scaledxp = nx.prune(scaledx,0);
76                       Tone.Transport.bpm.value=scaledxp;
77                       player.set({"playbackRate":scaledy});
78              });
79
80              tilt3.on('*', function(data) {
81                       var scaledy = nx.scale(data.y,-1,1,-500,500);
82                       player.set({"detune":scaledy});
83                       var scaledx = nx.scale(data.x,-1,1,0.1,0.8);
84                       player.set({"overlap":scaledx});
85              });
86
87           }
88      </script>
89
90      <script>
```

```
91              ////////////// LINK CHANGER /////////////////
92
93  function changeLink(){
94      var userInput = document.getElementById('userInput').value;
95      var lnk = document.getElementById('lnk');
96      lnk = userInput;
97      //lnk.innerHTML = lnk.href;
98      player.buffer.load(lnk);
99  }
100
101 function songSelector() {
102     var x = document.getElementById('selector').value;
103     player.buffer.load(x);
104 }
105
106   </script>
107
108       <div class="container">
109         <div id="tester" style="clear:both"></div>
110             <div class="row">
111                         <div class="col-sm-4" style="width:450px;
                                background-color:#686769;padding:10px;margin
                                :20px ;text-align:center">
112           <a id=lnk style="color:white">Post a Link to an MP3 file of
                 your choice</a> <br>
113         <input type='text' id='userInput'/>
114 <input type='button' style="border-radius:12px;border:none;padding:3px
    ;background-color:#f9f9f9;border:1px solid #ffffff;" onclick='
    changeLink()' value='Change Song'/>
115                               <br><br>
116
117             <select id="selector">
118   <option value="sounds2/fourhy.mp3">song1</option>
119   <option value="sounds2/funtonpige.mp3">SkandiMann-Funtonpige</option
        >
120   <option value="sounds2/jaruooj.mp3">Alternative Rock</option>
```

```
121             <option value="sounds2/colors.mp3">Kasper Vegas-Colors</
                    option>
122             <option value="sounds2/onehy.mp3">song5</option>
123 </select>
124
125  <button style="border-radius:5px;border:none;padding:3px;background-
     color:#f9f9f9;border:1px solid #ffffff;" onclick="songSelector()">
     Change</button>
126                       <div><p> </p></div>
127       <div>
128       <canvas nx="toggle" id='start' width="180" height="60"></canvas>
129       </div>
130                      <div><p> </p></div>
131       <canvas nx="tilt" id="tilt1" width="60" height="60"></canvas>
132       <canvas nx="tilt" id="tilt2" width="60" height="60"></canvas>
133       <canvas nx="tilt" id="tilt3" width="60" height="60"></canvas>
134
135             </div>
136                <div>
137             <div><a href="http://sonicakt.dk/motion.html">SNCKT
                    COLLAB</a><p style="font-size:10px">Motion Control (
                    works only on mobile devices).</p></div>
138             <div><a href="http://sonicakt.dk">SNCKT EFFECTS</a><p
                    style="font-size:10px">Sound Effects on Sound</p></
                    div>
139               </div>
140
141             <div>
142               <a href="https://goo.gl/forms/gJZvPKZcsuXT3YPL2" style
                      ="color:red">QUESTIONNAIRE</a><p style="font-size
                      :10px">Rate this app! Please support my project
                      with some feedback.</p>
143             </div>
144             <div>
145               <a href="Tonejs.github.io" style="color:green">Tone.js
                      </a><p style="font-size:10px">For those interested
```

```
                    in web-development this app was made with Tone.js
                    and NexusUI.js.</p>
146                 <div> <a href="http://www.nexusosc.com/" style="color:
                    green">NexusUI.js</a><p style="font-size:10px"></p>
                    </div>
147               </div>
148         </div>
149       </div>
150
151          </body>
152
153 </html>
```

Listing B.2: SNCKT MOVE FULL CODE

B.3 SNCKT COLLAB CODE

```
1  <!doctype html>
2      <html>
3    <head>
4       <title>: SNCKT COLLAB DRUMS:</title>
5          <meta name="viewport" content="width=device-width, initial-
               scale=1, maximum-scale=2, user-scalable=yes">
6          <link href="bootstrap.css" rel="stylesheet"/>
7          <script src="nexusUI.js"></script>
8      <script src="Tone.js"></script>
9          <script src="StartAudioContext.js"></script>
10     </head>
11
12       <body>
13     <script>
14
15           StartAudioContext(Tone.context);
16
17           var kick = new Tone.Sampler("sounds/kick.mp3").toMaster();
18           var hihat = new Tone.Sampler("sounds/hihat.mp3").toMaster();
```

```
19          var snare = new Tone.Sampler("sounds/snare.mp3").toMaster();
20          var shaker = new Tone.Sampler("sounds/shaker.mp3").toMaster
                ();
21          var rim = new Tone.Sampler("sounds/rim.mp3").toMaster();
22
23          var ampEnv = new Tone.AmplitudeEnvelope({
24    "attack": 0.1,
25    "decay": 0.2,
26    "sustain": 1,
27    "release": 0.1
28  }).toMaster();
29
30
31          var chain = new Tone.CtrlMarkov({
32    "beginning" : [{"value":"0.5","probability" : 0.2},
33                   {"value":"1.5","probability" : 0.3},
34                   {"value":"1","probability" : 0.4},
35                   {"value":"2","probability" : 0.1},
36      ]
37        });
38           chain.value = "beginning";
39
40          var kickSend = kick.send("reverb", -Infinity);
41          var hihatSend = hihat.send("reverb", -Infinity);
42          var snareSend = snare.send("reverb", -Infinity);
43          var shakerSend = shaker.send("reverb", -Infinity);
44          var rimSend = rim.send("reverb", -Infinity);
45
46          var reverb = new Tone.Freeverb({"dampening":6000,"wet":1}).
                receive("reverb").toMaster();
47
48          Tone.Transport.start();
49
50        nx.onload = function() {
51
52          nx.colorize("accent", "#1ee");
```

```
53          nx.colorize("border", "#bbb");
54          nx.colorize("fill", "#eee");
55
56          motion1.on('*',function(data){
57
58                      if(data.x > 0.2){kick.triggerAttack(0,"@4n");
59                                          kick.player.set({"playbackRate"
                                                :chain.next()});
60                                          chain.value = "beginning"};
61                      if(data.z > 0.2){snare.triggerAttack(0,"@2n");
62                                          snare.player.set({"playbackRate
                                                ":chain.next()});
63                                          chain.value = "beginning"};
64                      if(data.x < -0.3){hihat.triggerAttack(0,"@16n");
65                                           hihat.player.set({"
                                                playbackRate":chain.next()
                                                });
66                                          chain.value = "beginning"};
67                      if(data.y < -0.2){shaker.triggerAttack(0,"@8n");
68                                           shaker.player.set({"
                                                playbackRate":chain.next()
                                                });
69                                          chain.value = "beginning"};
70                      if(data.y < -0.2){rim.triggerAttack(0,"@6n");
71                                           rim.player.set({"playbackRate"
                                                :chain.next()});
72                                           chain.value = "beginning"};
73          });
74
75              tilt1.on('*',function(data){
76              var   scaledx = nx.scale(data.x,-1,1,0,1);
77                if(scaledx<0.1){kick.triggerRelease();hihat.
                    triggerRelease();snare.triggerRelease();
78                                  shaker.triggerRelease();rim.
                                        triggerRelease()};
79
```

```
80                var scaledy = nx.scale(data.y,-1,1,-30,0);
81                kickSend.gain.value=scaledy;
82                hihatSend.gain.value=scaledy;
83                snareSend.gain.value=scaledy;
84                shakerSend.gain.value=scaledy;
85                rimSend.gain.value=scaledy;
86      });
87
88    }
89
90     </script>
91
92       <div class="container">
93         <div id="tester" style="clear:both"></div>
94           MOBILE DEVICES ONLY! Shake your device. Try the different
                instruments and play with other people.
95             <div class="row">
96                       <div class="col-sm-4" style="width:450px;
                             background-color:#686769;padding:10px;margin
                             :20px ;text-align:center">
97           DRUMS
98                         <div>
99       <canvas nx="motion" width="200" height="200" label="tap twice to
              make sound"></canvas>
100                        </div>
101           <div>
102          <canvas nx="tilt" width="20" height="20"></canvas>
103                        </div>
104
105        <div><a href="http://sonicakt.dk/ana.html"style="color:white">
               SYNTH1</a><p style="font-size:10px"></p></div>
106        <div><a href="http://sonicakt.dk/ele.html"style="color:white">
               SYNTH2</a><p style="font-size:10px"></p></div>
107        <div><a href="http://sonicakt.dk/col.html"style="color:white">
               BELLS</a><p style="font-size:10px"></p></div>
108        <div><a href="http://sonicakt.dk/bass.html"style="color:white"
```

```
            >BASS</a><p style="font-size:10px"></p></div>
109         <div><a href="http://sonicakt.dk/guit.html"style="color:white"
            >GUITAR</a><p style="font-size:10px"></p></div>
110         <div><a href="http://sonicakt.dk/motion2.html"style="color:
            white">ORCHESTRA</a><p style="font-size:10px"></p></div>
111
112                         <div>
113                 <a href="https://goo.gl/forms/4ccxzJKgox5XEwV52" style
                    ="color:red">QUESTIONNAIRE</a><p style="font-size
                    :10px">Please support my project with some feedback
                    .</p>
114             </div>
115
116             </div>
117             <div>
118             <div><a href="http://sonicakt.dk">SNCKT EFFECTS</a><p
                    style="font-size:10px">Sound Effects on Sound</p></
                    div>
119             <div><a href="http://sonicakt.dk/move.html">SNCKT MOVE</
                    a><p style="font-size:10px">Tilt control</p></div>
120             </div>
121             </div>
122         </div>
123   </body>
124
125 </html>
```

Listing B.3: SNCKT COLLAB FULL CODE