Step into the Soundtrack

Shijie Wang
Georgia Institute of Technology
840 McMillan St NW
Atlanta, GA 30318
(+1)470-985-1378
swang464@gatech.edu

ABSTRACT

This project seeks to build a locative algorithmic music platform based on iOS called *SoundTrack.* It is not only an innovative tool for composers to compose ever-evolving, interactive music pieces with the aid of algorithmic composition technique, but also a platform for novice users to feel, understand and appreciate such music pieces in an intuitive and fun way. Concepts brought from related fields such as Musicology, Algorithmic Composition, Interactive Music, Music Information Retrieval, Audio Synthesis and Digital Signal Processing are integrated in this project.

Keywords

Algorithmic Composition, Locative Music, Interactive Music, Mobile App, iOS

1. INTRODUCTION

Algorithmic composition is a music composition technique with a notion of using algorithms to create music. The musical algorithmic functions can be used to produce whole compositional structures or smaller parts of a composition that are then manually integrated, and create guidelines, context, or situations for human music making[14]. Nowadays, even a mobile device such as iPhone can provide enough power for algorithmic composition. Besides, most of today's mobile devices come with numerous sensors such as cameras, microphones, gyro-scope, accelerometer, which signifies a great potential for interactive music. All the facts suggest that the gate has finally been wide opened for composers to explore their novel ideas and extend their visions on composition with these powerful computational devices. However, there is still one big issue in the way, which is the requirement of programming skill, even the visual programming languages like *Max/MSP*[9] and *Pure Data*[21] are still not transparent enough for composers to use, especially who have no programming background. Therefore, an intuitive, easy-to-use interface is included in this *SoundTrack* for composers to compose algorithmic composition and interactive music pieces. Besides the sufficient computational power offered by mobile devices, their on-board GPS modules raised another possibility — relate music with locality, which is the other highlight feature of this project.

The most significant outcome that can be expected from this project is that, the content provided by *SoundTrack*, the music, has interactivity and evolvability thank to the aid of algorithmic composition. But how to make sure these features can be well expressed to the listeners, in other words, how to ensure novice listeners and non-programming musicians can understand and appreciate such music pieces in an intuitive way? To address this issue, *SoundTrack* features graphic and animation representations for the evolution process, gives users visual feedbacks each time an evolution takes place in the music. With music, itself accompanied by the visual hints, the process is well expressed to, and sensed by the listener. Meanwhile, since all the musical parameters are represented graphically as well, non-programming musicians can control their composition intuitively. On the other hand, music hunting mechanism is carried out in *SoundTrack*. Music tracks are being pinned on virtual billboards, and these billboards are set up at different, users can only obtain them by going out and finding all the tracks instead of simply download them.

2. Related Work

Constructing a platform for algorithmic music and interactive music with locality for both composers and listeners on mobile devices can be considered as an innovative approach. There are numerous musical programming languages available on computer-based system such as *Max/MSP*[9], Pure Data[21], *SuperCollider*[24] and *cSound*[7]. There are, iOS-oriented programming environments available as well, *UrMUS*[10] is one example. Among them, *SuperCollider* and *cSound* can be grouped into one category, text-based languages. They offer maximum flexibility to users, to use them, programming skills is indispensable. *Max/MSP*, *Pure Data*, and *UrMUS* falls into another category which is visual programming environments. They are easier to learn in comparison to text-based languages, and they also offer great flexibility. These two categories of programming environments can be considered as good tools for algorithmic music composition. However, all of them needs programing knowledges to use. Furthermore, even though some of them can be used on mobile devices such as *Pure Data* and *cSound*, exhausting installation and configuration processes are needed, for example, most of them need users to compile from source code by themselves in order.

Beyond programming environments, there are existing higher-level tools specifically for algorithmic composition. *Amper Music*[3] is a one example. It is a website which lets user create and customize original music by tweaking a series of high-level musical parameters, such as mood, style, instrumentation, tempo, duration, etc. This design makes the system easy to use. Due to the lack of documentation and demonstration, the principle of their music generation is not determined. However, since users have no further control beyond the high-level parameters, thus, if creating two music segments with the same set of parameters, the result — the music it creates sounds analogous. In contrast, *Soundtrack* offers user more flexibility by letting them choose which building blocks to use in the algorithm. High-level controls are available as well with intuitive graphic user interfaces to keep the APP's usability.

Location-based is another highlight feature in this project. Amazing projects which relate music to location had been done by various artists and organizations. Such as *Cities and Memory*[6], a global field recording & sound art work founded by sound artist *Stuart Fowkes*, which enables user to explore locations through their actual sounds; *Nature Sound Map*[20], a websites offers an enjoyable and interactive way of exploring the natural sounds of our planet, created by a group of professional nature recordists; *Urban Remix*[11], a collaborative and locative sound project which enables users to explore and experience the soundscapes of the city in a novel fashion. Among these great projects, *Urban*

Remix is most related to this project. It consists of a mobile phone system and web interface for recording, browsing, and mixing audio. Like the other two, users can upload their recordings to the webserver with certain location information, these recordings will be shown on an interactive map view. What makes *Urban Remix* unique is, these recordings can be remixed or concatenate together to create new soundscape by simply drawing lines on the map. *Soundtrack* also follows a similar user-uploading route, and features a mechanism for user to concatenate pieces together as well. However, there are two aspects which *Soundtrack* differs from others. Firstly, the content of *Soundtrack* is music per se, instead of concrete sound or soundscape. Which leads to the second differentiation — it takes a different perspective on the relation between music and location. In *Soundtrack*, a certain location is not represented by the sound itself, but the local musicians who compos the music, and the users who create new music on top of these compositions.

On the other hand, building such a comprehensive platform is a largescale and complex project which requires an interdisciplinary view, diverse fields can be related such as Computer Networking, Digital Signal Processing, Machine Learning, User Interface and User Experience Design, Data Visualization, etc. Thankfully, there are ongoing researches and studies refer to these related fields. In terms of algorithmic music composition, Eduardo Miranda's book *Composing Music with Computers*[18] is a very good resource and reference. For symbolic music information retrieval, there is a great library available called *Music21*[8]. developed by the music departments of MIT, along with Harvard, Smith, and Mount Holyoke Colleges. *TensorFlow*[1] is a powerful machine learning library developed by Google, and Google even carried out a machine learning composition project called *Magenta*[26] based on it. Compare to the fields discussed above, digital signal processing is a much more mature researching area, in which tons of great works can be found, among them, [17], [2] and [19] are helpful resources.

Apart from the recourses and toolkits referenced above, this project is profoundly inspired by some music works and projects: *Music for Airports*[5], a series of ambient recordings created by Brian Eno in 1978; *Koan*[12], a software for creating generative ambient music developed by Brian Eno with Peter and Tim Cole in 1996; *Bloom*[4], an ambient music generator for iOS platform developed by Brian Eno and Peter Chilvers in 2013; and *Grow* old[15], an ever-evolving EP created by Jason Freeman in 2013.

3. Design and Implementation

The development of *SoundTrack* is divided into two parts: a frontend APP, and a backend server. Since the frontend is based on iOS platform, *Swift*[23] became the preferred programming language since it is considered more modern and intuitive compare to the former iOS programming language, Objective-C, which made it a more efficient choice in terms of development process. On the other hand, the server side is written in *python* on top of *Django Framework*[25] since this combination is effective for rapid development with M-V-C(Model-View-Controller) design pattern, which meets the requirement of this project.

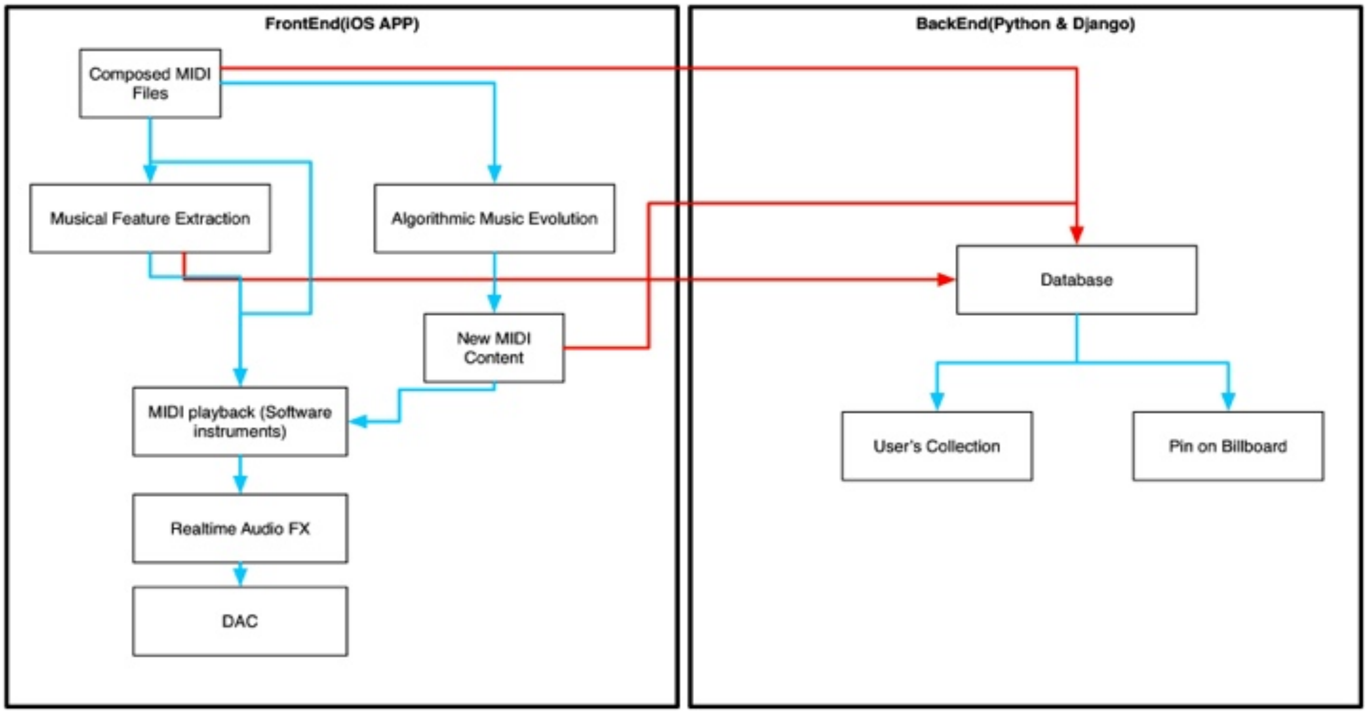

Figure1. Design Diagram

In this project, music is represented in a symbolic way, rather than in raw audio. It makes sense to use MIDI file to store music data since it is a standard music file format being used by various DAWs and most of the musicians and composers are familiar with it. Benefit from using symbolic representation, all the musical parameters are flexible, such as tempo, time signature, key, etc. Which means algorithms can gain access to each of them, and they can even alter the music contents in a note-by-note level, that guarantees maximum flexibility. However, the drawback of using symbolic music representation is also obvious. Firstly, composers may not be able, or find it very difficult to use recorded audio content for their composition which is especially critical when a composer wants to use pre-recorded acoustic instruments in their compositions. Secondly, due to that MIDI relies on software instruments as sound sources to reproduce the actual music piece, there is a chance that the reproduced music may vary from device to device because of different sound sources being used.

There are huge advantages and non-negligible limitations at the mean time of using symbolic music representation, just like every coin has two sides. To reduce the limitation as much as possible, a cross-platform sound source — a software instrument plug-in in VST and AudioUnit format is featured namely *STSynth*[22].

As shown in figure 1, there are different components connected to each other in order to provide different functionalities on both frontend and backend. The connection could be either local connection (shown as blue arrows), or cross-side connection (shown as red arrows). Here is the overview of the complete workflow: Firstly, composer uploaded the MIDI file created on their computer into *Soundtrack*. He/She can listen to it and assign different software instruments to different tracks. Then the system extract musical features from the MIDI file and save the infomations to a file. The features including key, tempo, time signature, pitch lists (per track), note onset timestamp list (per track), note density list(per track), pitch distrubution list (per measure), and track configurations (track name, track type, selected software instruments and effects). After that, the composer can upload the MIDI file to the sever and pin it on the virtual billboards. The retrieved features together with the MIDI file are sent to the database and the database do the management job such as add this file to the composer's collection and the selected billboard for others to collect. After another user collected this MIDI file from certain billboard, this MIDI file goes into his/her own collection and can be retrieved and played back at any time, and this collector can also use this MIDI file as basic building blocks to create new music by using the algorithmic music evolution mechanism. And the newly created music content can also be uploaded and pined on virtual billboards.

In the following paragraphs, the implementation is detailed demonstrated part by part.

3.1 MIDI File Parsing

On iOS, the infrastructure for dealing with MIDI data is Core-MIDI, it was wrapped in *AudioToolbox* framework. There are three types being used in *AudioToolbox* for MIDI file parsing, *MusicSeuqence*, *MusicTrack* and *MusicEventIterator*. A MIDI file is loaded as into a *MusicSequence*, along with all the tracks in the MIDI file represented as *MusicTracks*. Then using *MusicEvent-Iterator* to extract information such as note message and channel message from each *MusicTrack*. The retrieved information then be stored into a custom structure called *MusicBlock*. The *MusicBlock* structure can be altered in real-time and can be parsed back into MIDI file for playback.

3.2 Audio Plugin Development and Hosting

A sound source such as a software instrument is indispensable for MIDI file playback. Software instruments are usually audio plugins, which means the system must be able to host them. The plugin format used on iOS and macOS is called *AudioComponent*. There are numerous software instruments available on macOS, however, due to the limitation of iOS system, only the latest version 3 Audio-Component can be host, which means the existing software instruments on macOS are likely not possible to be directly hosted on iOS without any alteration.

On the other hand, like demonstrated above, in order to make sure the music sounds the same on both computer and the APP, both ends need to use the exact same software instrument. *STSynth* (Figuare 2), a cross platform software instrument is developed.

STSynth is a polyphonic subtractive synthesizer, built upon *JUCE* framework. The synthesizer has 5 generators, sinewave, band-limited saw tooth wave, band-limited pulse wave, triangle wave, and noise. These generators are implemented with the help of *STK* library. The synthesizer also features a multi-purpose filter, a mixer and a LFO component.

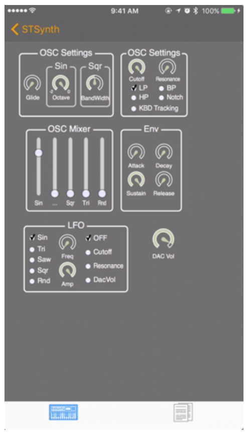

Figure2. STSynth

In terms of hosting it on iOS, firstly, the plugin is compiled as an app extension which is the regulation for hosting it on iOS, it will automatically be placed in the right directory so that system will be aware of its existence. Then the APP can find it by consulting the AVAudioComponentManager class, and host it as an AUAudioUnit class.

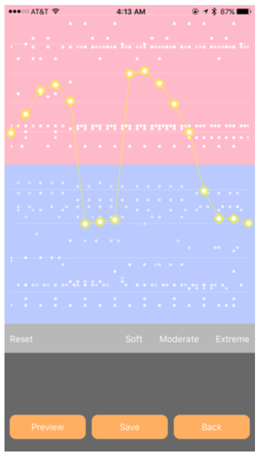

Figure5. Probability Line

3.3 Real-time Audio Routing

For better user experience and user's degree of control, the APP features a real-time audio routing mechanism. This mechanism acts very much like the mixer in a DAW. It enables user to control the pan/volume of each track, and change instrument and audio effects for each track on the fly. This is implemented upon the AVAudioEngine class, this class abstracts each instrument/effect as a node, these nodes get connected to the engine, and the engine's configuration can be changed in real-time.

3.4 Music Evolution

The music evolution mechanism is a process which creates new music content by concatenating two different music clips together. Two methodologies are considered to achieve better result — rule-based algorithm and deep learning.

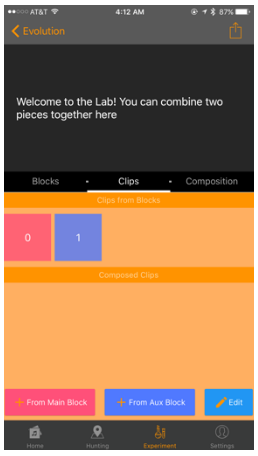
Welcome to the Lab! You can combine two pieces together here

Figure3. Select Clips

Initially, deep learning is the preferred approach since it is known for its exceptional ability to solve complex problem. Taken inspiration form this project[16], a 300-layer LSTM[13] neural network is constructed. Input vector used in the network contains 5 features: the MIDI note number of each note; the pitch class of each note; previous vicinity which representing the context for surrounding notes in the last time step, one octave in each direction; previous context which stands for the pitch class of notes played in the last time step; and beats which is a binary representation of position within one measure. For the output vector, there are two values for each note event: play probability which stands for the probability that this note should be played; and articulate probability, which is the probability of this note being articulated (sustained to the next time step). After implementing the model following this architecture, 33 multi-track pop music pieces was trained in pairs. The result was delightful, firstly, musical features from both input pieces are concatenated together cohesively in the generated music; secondly, the music generated by the model is key-invariant, which means the model treats each note equally, even the input piece pair is not in the same key, the model is able to keep them in the same tonality; Thirdly, it can generate polyphonic contents which means it allows multiple notes to be played simultaneously. However, there is also significant shortages: this network architecture is not able to differentiate notes from different tracks; furthermore, even though the concatenation is cohesive, the generated music is not vivid enough and appears to be lack of musicality, sometimes the model could repeat one single note over and over again till the end; lastly, to the user the model seems like a black box, no further control is available besides choosing two input pieces. Ultimately, deep learning is ruled out by comparing the advantages and drawbacks to the goal of this music evolution process, instead, a well-designed rule-based algorithm is established. Detailed implementation is demonstrated in the following paragraphs.

The whole process can be broken down into four steps. Firstly, the user needs to determine which music block to use as the main block. Then the track configurations in the new piece are set following the main block's track configuration. For example, if the main block has four tracks, a piano track, a guitar track, a bass track and a drum track, then the new piece will keep the same instrumentation with the same software instruments loaded for each track. Secondly, the user will select another music block as "aux block", which is the second music block to be concatenated with the main block. And then, the actual clips being used for evolution process can be selected from the main block and the aux block respectively (shown in figure3). The clips can be in arbitrary range (shown in figure4). After that finally comes the core functionality, drawing the "probability line" between two clips and create a new content as shown in figure4.

The first three steps are straightforward without further explanation. The following demonstration will be focusing on the last step. The "probability line" drawn by the user can be considered as a list of probability of certain notes being chosen at corresponding time stamps. After the line being drawn and the user tapped on the preview button, the evolution process starts. System retrieve the probability list and start iterating the two clips from start to end, each time step is a 16th note. If system finds notes start on certain time step, it determines whether the note should be kept or discarded by looking at which clip do the notes come from and consulting the probability list. Then if the system decides to keep the note, and the notes come from the aux block, system then maps the notes to correct tracks based on finding matched track type in the main block. If the notes come from the main block, then system just simply keeps it without further alteration.

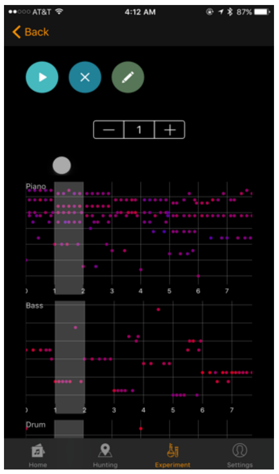

Figure4. Determine the span

4. User Scenario

As mentioned, *Soundtrack* is targeting both musicians and novice listeners. This section intends to discuss some possible user scenarios with both user groups in mind.

4.1 Indie Musicians

Consider an indie musician called James, he makes brilliant music, has fans and followers from all over the world on his social media. However, very few people show up in his live performances. In other words, he is quite famous online, but in real world? Not so much. So how can Soundtrack help? James can simply apply for a virtual billboard near the live house he always plays, and pins some of his singles onto it. That way, other users live in the same community can instantly see the new billboard and get to know James. Since user needs to literally go to the virtual billboard to listen to and collect James's music, there is a good chance that they will not only appreciate his music on the APP, but also stay in the live house and waiting for James's gig at night. After some time, James will gain a faithful fan club in his own neighborhood who will follow his every single show, apparently, his Shanghainese fan won't be able to do that.

4.2 Artists and Celebrities

Think of a famous artist Christina is preparing her annual world tour. She gets a little bit tired of just performing from city to city all around world, she wants to prepare something special for her fans this year. Assuming she is also a *Soundtrack* user, what would she do? She can set up temporary billboards on the venues she will be performing, and making a series of exclusive contents only available during her tour. As people loves surprises, let alone an amazing surprise elaborately prepared by their favorite artist, and, available only in a limited time. It is not hard to imagine that Christina's popularity will hit a new peak.

4.3 Novice Listeners

As demonstrated in the two cases above, novice listeners can get to know great musicians in their community which they never heard of before. They can receive exclusive surprises from their favorite artists. Furthermore, *Soundtrack* enables them to get a taste of making their own music by combining pieces they collected from artists they like, and even upload their creation onto the billboards to let others to appreciate. This exciting experience could motivate them to learn more about music, and even becoming a musician themselves someday.

5. Evaluation

The original interpretation on the relationship between music and location, and the interesting music evolution mechanism are the highlights of this project. How to express these ideas to the users is the most challenging part. Hence, a beta test is performed for user study.

There are 4 participants in the beta test. 3 of them holds a music background, and one has no music background. After their test, they are asked to provide feedbacks regarding the music-evolving system, track hunting mechanism, overall experience and suggestions. According to the feedback collected from test users, all the users with music backgrounds thought that the essence of music-evolving system is well delivered with properly designed and easy-to-use user interfaces; the user with no background comments that the music-evolving system is interesting but a little bit hard to understand without further description or tutorial. And all the testers are enjoying the track hunting *Soundtrack*.

However, the feedback reveals some deficiencies as well. Firstly, *STSynth* is not stable now, it causes crashes sometime and the parameters cannot be properly recalled. Secondly, the user interface for track uploading page is still sketchy, do not cohere with other pages. Thirdly, on-screen instructions are needed since the functionality of this APP is complicated, without it, some operation causes confusion especially in the music evolution process.

From the perspective of the author, extensive improvements can be done in the future. Firstly, there is only two software instruments available at present, *STSynth* and the iOS built-in sampler, which is apparently not enough. In the future, more software instruments will be made. Secondly, the music evolution process can be improved, for example, integrates chord detection and instrument classification to make the contents more musical. Thirdly, the backend server is not powerful enough, lack of functionalities such as genre classification and music/artist recommendation will soon be a bottle neck if this APP goes to production. Lastly, raw audio support will be a great feature to add, which will boost the flexibility of this system.

6. REFERENCES

[1] Abadi, M. et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467 [cs]*. (Mar. 2016).

[2] Adamson, C. and Avila, K. 2012. *Learning Core Audio: A Hands-On Guide to Audio Programming for Mac and iOS*. Addison-Wesley Professional.

[3] Amper Music: *https://www.ampermusic.com/*. Accessed: 2017-04-24.

[4] Bloom - Generative Music - Creative apps for the iPad, iPhone and iPod touch: *http://www.generativemusic.com/*. Accessed: 2016-09-24.

[5] Brian Eno - Ambient 1 (Music For Airports): *http://www.discogs.com/Brian-Eno-Ambient-1-Music-For-Airports/master/6265*. Accessed: 2016-09-24.

[6] Cities and Memory: global collaborative sound project| Field Recordings, Sound Map, Sound Art: *http://citiesandmemory.com*. Accessed: 2017-04-25.

[7] cSounds.com: *http://www.csounds.com/*. Accessed: 2016-09-19.

[8] Cuthbert, M.S. and Ariza, C. 2010. music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. *Michael Cuthbert*. (Aug. 2010).

[9] Cycling74: *http://www.cycling74.com*. Accessed: 2015-08-31.

[10] Essl, G. 2010. Urmus-An Environment For Mobile Instrument Design And Performance. *ICMC* (2010).

[11] Freeman, J. et al. 2012. Rediscovering the City with UrbanRemix. *Leonardo*. 45, 5 (Jul. 2012), 478–479.

[12] From SSEYO to Intermorphic: *http://intermorphic.com/sseyo/*. Accessed: 2016-09-24.

[13] Hochreiter, S. and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.

[14] Holmes, T. 2015. *Electronic and Experimental Music: Technology, Music, and Culture*. Routledge.

[15] Jason Freeman: Grow Old: *http://distributedmusic.gatech.edu/GrowOld/*. Accessed: 2016-09-24.

[16] Johnson, D.D. 2017. Generating Polyphonic Music Using Tied Parallel Networks. *Computational Intelligence in Music, Sound, Art and Design* (Apr. 2017), 128–143.

[17] Konrad, M. 2011. Analysis of audio synthesis possibilities on mobile devices using the Apple iPhone and iPad. *ResearchGate*. (Mar. 2011).

[18] Miranda, E. 2001. *Composing Music with Computers*. Focal Press.

[19] Musicdsp.org: *http://www.musicdsp.org/*. Accessed: 2016-09-19.

[20] Nature Soundmap - Hear the world like never before: *http://www.naturesoundmap.com/*. Accessed: 2017-04-25.

[21] Pure Data — Pd Community Site: *https://puredata.info/downloads/pure-data*. Accessed: 2016-09-19.

[22] Rexhits/STSynth: *https://github.com/Rexhits/STSynth*. Accessed: 2017-04-25.

[23] Swift - Apple Developer: *https://developer.apple.com/swift/*. Accessed: 2016-09-21.

[24] The SuperCollider Home Page: *http://www.audiosynth.com/*. Accessed: 2016-09-19.

[25] The Web framework for perfectionists with deadlines | Django: *https://www.djangoproject.com/*. Accessed: 2017-04-25.

[26] Welcome to Magenta! *https://magenta.tensorflow.org/welcome-to-magenta*. Accessed: 2016-09-19.