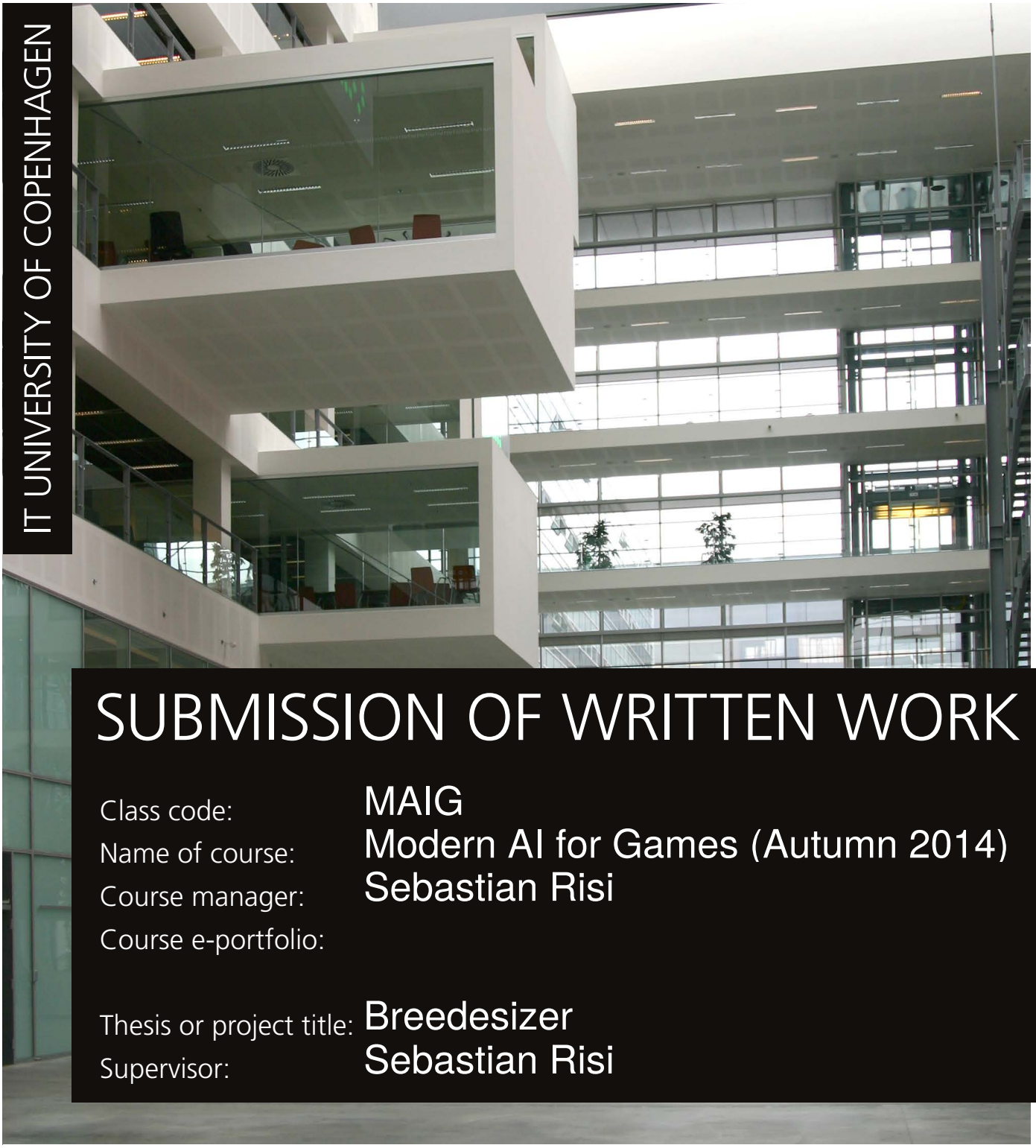
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code: MAIG
Name of course: Modern AI for Games (Autumn 2014)
Course manager: Sebastian Risi
Course e-portfolio:

Thesis or project title: Breedesizer
Supervisor: Sebastian Risi

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Björn Þór Jónsson	03/04-1975	bjrr@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Breedesizer

Project report done in the course *Modern AI for Games*, Autumn 2014

Björn Þór Jónsson - bjrr@itu.dk

***Abstract*— Oscillators emitting audio timbre are the fundamental sources of sound in audio synthesis. The shape of their periodic waveforms is commonly chosen from a canonical set of four types that have played a significant role in analog synthesis techniques, but it is possible to use all kinds of unique shapes. The aim with Breedesizer is to explore the feasibility of utilizing the powerful properties of Compositional pattern-producing networks evolved through NeuroEvolution of Augmenting Topologies (CPPN-NEAT), for encoding structure, in the formation of unique waveforms to use in the construction of oscillators, producing interesting timbres.**

I. Introduction

THE synthesis of sound can be a complex task requiring great expertise in the use of instruments generating sound, synthesizers. Deep understanding of the interaction between different modules that synthesizers are composed of, how various waveforms affect each other, helps in arriving at configurations and settings of the modules to produce the desired sounds. Without an expertise in the use of the various synthesizer instruments and lacking a solid grasp on sound synthesis theory, a novice approaching the task of creating new and interesting sounds to play, will be easily intimidated by the plethora of settings and configuration options. A newcomer to the world of sound synthesis may default to using pre-set configurations, provided by experts, as it may be hard to identify where to start modifying the various settings, guiding the sound exploration in a feasible direction, and so opportunities for discovering interesting and novel sounds may commonly be missed.

Oscillators that produce periodic waveforms are the most common basic building blocks of synthesizers. They act as sources of sound, that are then controlled, combined and modified in various ways, according to the parameters and configuration options afforded by the synthesizer instrument they are part of. A fundamental choice, when designing a sound to be generated by a synthesizer, is what kind of a periodic waveforms should be generated in each oscillator, where they define the perceived timbre. A simple definition of timbre refers to it as "the quality or distinguishing properties of a musical tone *other than its pitch*" [1] and ANSI defines it as "that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar"[2]. The concept of timbre can benefit from further definitions [3, p. 42][4][5] but those quoted here will suffice for the purposes of this report.

Common types of oscillators, built in to synthesizers, generate sine, sawtooth, square and triangle waveforms - a tradition inherited from the construction of analog synthesizers. Those four types of oscillators can be referred to as classic "because of the significant role they played in analog synthesis techniques" [6].

Another type is the digital wavetable oscillator [7], which is "based on periodic reproduction of an arbitrary, single-cycle waveform"[8]. There have been different uses of the word *wavetable* where "academics use the word to describe the sequence of numbers that represent a single cycle of a regular, periodic waveform"[9] whereas *wavetable synthesis* commonly refers to a technique of stepping through a series of periodic waveforms (in a table) for the duration of each note played [10]. In this report we will use the interpretation of a wavetable as a single, arbitrary waveform, playable in a periodic fashion in the same way as the classic oscillators from the analogue heritage.

The project discussed in this report focuses on exploring the feasibility of using the CPPN-NEAT neuroevolution technique, to guide the formation of waveforms in search of interesting timbres. The periodic waves, formed by the method discussed, will be used to construct wavetable oscillators producing timbres that will be evaluated in a process of Interactive evolutionary computation (IEC), where humans select the oscillators producing the most pleasing timbres, as parents for the next generation of wavetable oscillators. Exploration of waveforms generated by this process may reveal interesting timbres that can be used as unique building blocks for the assembly of novel synthesizers.

As a small step towards employing the discussed neuroevolution methods to construct the components of synthesizers in general, rather than limiting their use to the formation of waves in wavetable oscillators, a simple Frequency Modulation (FM) [11] synthesizer will be constructed in connection with each wavetable oscillator under consideration. This will be accomplished by forming two periodic waveforms as outputs from the evolved neural networks, where each will be used to construct its own wavetable oscillator, one acting as a modulator of the timbre produced by the other, acting as the carrier. Offering a human, evaluating fitness, the choice of hearing the unmodulated carrier oscillator or the richer timbre from the modulated FM synthesis of two oscillators, will allow the fitness evaluation of each neural network to be approached from a wider angle, when comparing the two. This simple FM synthesis step can also provide a glimpse into the possibilities of using this technique to arrange more complex synthesizers.

II. Background

This section will begin by surveying previous research into the application of evolutionary computation to synthesize sounds. Then a brief overview of the CPPN-NEAT technique

will be given, followed by a discussion on how waveforms can be represented with CPPNs.

A. Evolutionary computation applied to sound synthesis

Previous research into applying Evolutionary Computation (EC) and Genetic Algorithms (GA) to the configuration and construction of synthesizers, can be roughly categorized by methods that:

- use Genetic Algorithms to navigate the parameter settings of synthesizers
- evolve populations of waveforms
- or evolve the structure of components comprising a complex synthesizer

With evaluations performed with:

- fitness functions automatically matching a known target sound using a given synthesizer
- or users' judgements about the sounds in the system (IEC).

Evolution of synthesizer parameters aims to facilitate an exploratory approach to sound design [12][13], addressing the problem of required intimate knowledge of sound synthesis techniques in order to design sounds that are novel and of musical interest [14][15], using genetic algorithms operating on the parameter settings of a synthesizer [16][3][17][18][19]. Cellular Automata (CA) have also been used to control over time the parameters of a synthesis instrument [20], whereby the emergent behaviour of CA is used to model generative processes for synthesised sound and musical forms [21].

Selective breeding, by a user judging evolved sounds, has been employed to guide the evolutionary processes [14][15][12][13], or attention has been focused on fitness functions used to drive the evolution [16], automatically matching a target sound using either a given synthesizer [3][17][19], evolved populations of sound waves [22][23] or grown sound synthesizers using evolutionary methods, using custom descriptions of sound synthesis algorithms in the form of cyclic topology graphs and acyclic tree graphs with ordered branches [24].

Other novel approaches to sound synthesis include the use of Differential Evolution, which is known as a simple and efficient scheme for global optimization over continuous spaces, for parameter estimation for frequency modulated sound waves [25]; and an investigation into the feasibility of synthesizing sounds with hybrid wetware-silicon devices using in vitro neuronal networks [26].

> [Evolutionary approaches to musical interaction in general] can be separated into two broad categories: interaction at note level and interaction at sound level. Interactions at note level usually produce complete musical pieces or music fragments made up of notes that comply with accepted musical and harmonic rules described in modern music theory. The interaction at sound level is concerned with the manipulation of parameters that define a sound using a particular sound synthesis technique (SST), or with parameters that define a particular deformation on an input stream (sound effects) [27].

The techniques discussed so far, and the approach presented in this report, falls into the category of interaction at sound level. An interesting method of interaction at note level is Functional Scaffolding for Musical Composition (FSMC) that has been used to evolve drum patterns and harmonies through Interactive Evolutionary Computation [28][29][30][31], which has utilization of CPPN networks in common with the project reported here. Other examples of musical composition with GA and IEC are [32] and [33][34].

Rather than investigating ways of searching through the space of all possible input parameters to the settings of synthesizer modules, or exploring ways to guide the composition of synthesizer modules or waveforms to arrive at interesting sounds, the focus in this project is on the feasibility of using interactive neuroevolution, based on CPPN-NEAT, to guide the formation of waveforms in search of interesting timbres.

B. Compositional pattern-producing networks evolved through NeuroEvolution of Augmenting Topologies (CPPN-NEAT)

To evolve the least complex artificial neural networks (ANNs) that can solve a given task, the NeuroEvolution of Augmenting Topologies (NEAT) technique starts with a population of small, simple networks, that are complexified over generations [35]. When mating different network structures, historical markings are used to align compatible genes, to produce meaningful offspring efficiently. One property of NEAT is the protection of structural innovation, "by reducing competition between differing structures and network complexities"[36], but that speciation feature is overridden by the human selection in the IEC process discussed in this report.

Compositional pattern-producing networks (CPPNs) are a variation of ANNs that use a variety of canonical activation functions at each node [37]. "While ANNs are traditionally employed as controllers, CPPNs often function as pattern-generators, and have shown promise in the context of procedurally generated content" [38]. In addition to the sigmoid or Gaussian functions used in ANNs, CPPNs can use periodic functions, such a sine, to produce segmented patterns with repetitions, while symmetric patterns are produced by functions such as the Gaussian, and their use of linear functions can produce patterns with straight lines. The selection of activation functions available to CPPNs can bias their output towards desired types of patterns.

With a slight modification, NEAT can be used to evolve CPPNs, instead of regular ANNs, where the main difference lies in the nodes of CPPN-NEAT including a field for specifying what activation function to use (along with a compatibility distance function supporting the speciation feature, by counting how many activation functions differ between two individuals).

C. Wavetable construction

Waveforms, for building wavetable oscillators, can come from a variety of sources.

Pre-processed and -recorded waveforms form one category of those sources. A recent, notable example of a synthesizer that is based on such sources is *Animoog*, whose "diverse library of timbres is derived from analog waveforms captured from classic Moog oscillators, both vintage and modern, and run through ... high-end outboard and analog signal processors" [39], thus delivering some of the desired qualities of analog hardware synthesizers to the more affordable digital domain.

Some audio synthesis applications offer the ability to draw freely by hand, arbitrary waveforms, in the process of creating wavetable oscillators. Others employ techniques from vector graphics to shape the periodic waveforms, such as Bézier curves, where the *DIN Is Noise* synthesizer [40] is a notable example. Those methods challenge the ability to apply visual imagination to the process of shaping interesting timbres.

CPPN networks, evolved with NEAT, have been demonstrated to offer powerful representational properties, with indirect encoding of complex structures [37]. Picbreeder, an online evolutionary art community in which users can evolve and share images, is a good example of the successful use of CPPN-NEAT to search for spatial patterns [41]. The creative ability demonstrated by Picbreeder, and related projects such as DelphiNEAT and SharpNEAT [37], spawned the idea for Breedesizer, where the aim is to explore how well CPPN-NEAT applies to the search of interesting patterns in the one-dimensional space of sound waves.

There seem to be no previous applications of CPPN-NEAT to create wavetable oscillators. The closest related work the author of this report can find, involving sound waves and CPPNs, is that of evolving soundtrack patterns with CPPN-NEAT [42] and the application of NEAT to evolve ANNs that mimic the effects of particular guitar effect pedals [43][44].

III. Methods

To facilitate the exploration of timbres with CPPN-NEAT neuroevolution, an Interactive evolution computation environment (IEC) was developed that presents populations of phenotype waveforms, where each individual is backed by a genotype composed of one evolved CPPN network. The name Breedesizer has been used for this environment, though other options have been considered, such as Audiovolve.

A. Choice of technology for implementation

The first technology option considered, for development of the timbre IEC environment, was the combination of HTML and JavaScript. Those technologies were considered feasible as they comprise the language of the World Wide Web and implementations based on them are readily accessible in common browsers used to access the web, without requiring any additional setup overhead of software plug-ins. The main inspiration behind this project, Picbreeder, does indeed run in a web browser but relies on a Java Applet plug-in for its central functionality, the breeding of successive generations of pictures, which may act as a hurdle for new interested users willing to try out this novel system. Here the author of this report draws from personal experience, having become very interested in Picbreeder after first hearing about it, balked on the first attempt at trying it out, when the Java Applet loading failed.

Choosing JavaScript as the programming language limited the selection of available ready-made library implementations of CPPN-NEAT. It was viewed as a sensible choice to utilize a tried and tested implementation of CPPN-NEAT in a reusable library instead of implementing this common functionality from scratch.

In fact, only one CPPN-NEAT JavaScript implementation was found, in the form of npm modules [45]. Those modules are part of a proposed infrastructure for turning any evolutionary domain into an online interactive platform [46][47]. As the modules are part of this published infrastructure and are accompanied by suites of unit tests, the correctness of their implementation seemed to have been established.

One significant obstacle to incorporating them to the IEC environment for waveform timbre exploration, Breedesizer, was the complete lack of documentation. As the module code is open source, it was eventually possible to figure out how to instantiate the modules and call their methods. Usage examples would have expedited that process and the initial prototypical implementation of Breedesizer, developed for this project, may serve as such an example.

Another complication became apparent when attempting to use those modules in a web browser environment, as they are assembled as npm packages, designed to be run on the Node.js platform. Several tools exist intended to compile npm modules for use in web browsers. Initially the Browserify [48] project was tried, with partial success, but eventually the Component solution [49] produced a compilation of the modules that ran error free in a browser.

B. CPPN Waveform Representation

The powerful properties of CPPNs for representing structure have already been demonstrated [37][41], and so they can be expected to output structured waveforms that are more interesting than random noise [50].

A fundamental difference between the Breedesizer project and picture evolution projects based on CPPN-NEAT, such as Picbreeder, is that they work with two-dimensional inputs to the CPPN networks, that represent the coordinates of each pixel in the pictures to be generated [51], whereas the waveform timbre evolution project discussed here accepts one-dimensional inputs that represent each sample of the waveform to be generated.

The numbers in a sequence that represents a single cycle of a periodic waveform, are within the range from -1 to 1. To ensure continuity between the ends of the waveforms, the absolute of values y from that range is fed in as the input signal (comparable to the d coordinate, *distance from center*, in [37]). To produce repetition in the outputs from

the CPPNs, $sin(n \times abs(y))$ is fed into another input node, where the integer value n is adjustable in the user interface, to allow different frequencies of repetition (Figure 1). To observe the *Repetition with Variation* property of CPPNs [37, p. 25], the user interface provides the option to remove the application of the absolute value function on the y values passed to the sine wave calculation, possibly resulting in discontinuous waveforms that may produce unpleasant or interesting timbres, subject to evaluation (Figure 2).

> By inputting sine waves, the same coordinate frame repeats over and over again. However, the d coordinate, i.e. distance from center, does not repeat. Therefore, functions higher in the CPPN can utilize both the repeating frame of reference and the absolute reference at the same time. [37, p. 25]

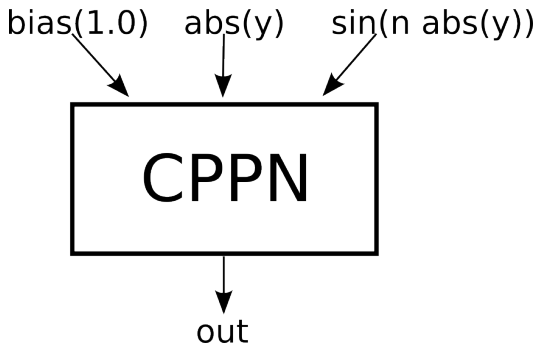


Fig. 1. CPPN Inputs as absolute values of numbers in a sequence that represents a single cycle of a waveform, and periodic inputs as sine waves produced from multiples of the numbers in that same sequence.

To create playable wavetable oscillators from the arbitrary waveforms output by the CPPNs, their signal data needs to be decomposed to its constituent frequencies by a Fourier transform. That results in a table of coefficients in a Fourier series "representing the partials of a periodic waveform" [52]. Using the Web Audio API, those Fourier coefficients are provided as a parameter to the `createPeriodicWave` method [53], which results in a periodic wave, or a wavetable, "representing a waveform containing [the] arbitrary harmonic content" [52]. The bigger the table (or sample size), the closer the approximation by the Fourier transform, and the size is adjustable in the user interface with a default of 1024; an example wavetable size used in [7].

In the current version of Breedesizer, a design decision was made to output two waveforms from each CPPN, to allow for the assembly of a simple FM synthesizer for each evolved CPPN, where one waveform acts as a modulator affecting the oscillation frequency of the carrier oscillator based on the other waveform [54]. More waveforms could easily be added as outputs from the CPPNs and be used as building blocks of other audio synthesis techniques.

The activation functions for each hidden neuron in the CPPNs are chosen from the canonical set of Gaussian, Bipolar sigmoid, sine and linear, each with a .25 probability.

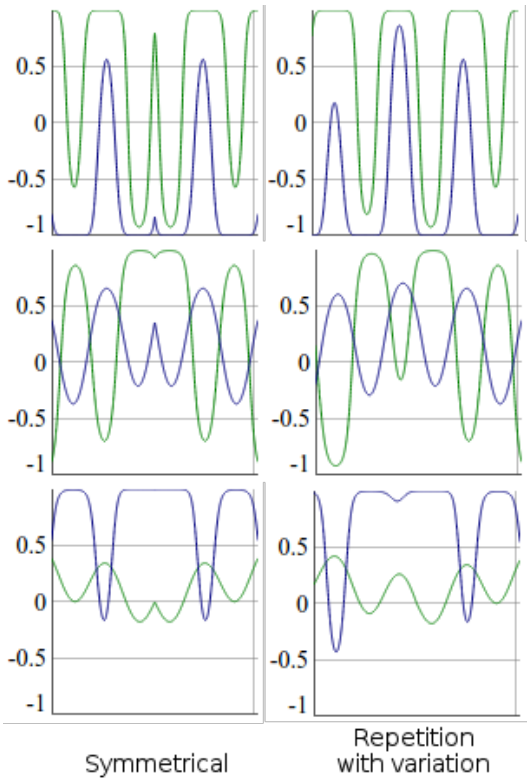


Fig. 2. Waveforms output from the same CPPN, either produced from inputs restricted to absolute values, or with those restrictions lifted from the periodic input node.

C. User Interface design

The Breedesizer user interface presents one population of ten waveform sets at a time. Behind each set is a CPPN network with output nodes producing the waveforms. At the beginning of each IEC session, five seed CPPN genomes are created, which serve as the bases for the first generation. Individuals in the first population have no actual parents, but are instead mutations of some random seed genome. Future generations are based on mating parents selected from the current population and mutating their offspring. A screenshot of the Breedesizer user interface can be seen in Figure 3.

Each waveform presented can be enabled for playback by clicking on it. An on-screen musical keyboard can be clicked with a pointing device to play different notes with the selected oscillator. The keys of a desktop computer's QWERTY keyboard can also be clicked to play different notes.

In an effort to lessen user fatigue, each waveform instantly plays on click one short note (C3 at 130.813 Hz, for the duration of one Quarter Note at 120 BPM, 500 ms). The possibility of continuously playing a selected MIDI song, while auditioning the timbres of the different oscillators, was investigated but the implementation of that feature was not completed for the prototype presented with this report.

Probabilities of mutation are adjustable in the user interface with two sliders, one for the probability of adding a connection and another for adding a node, both set to .13

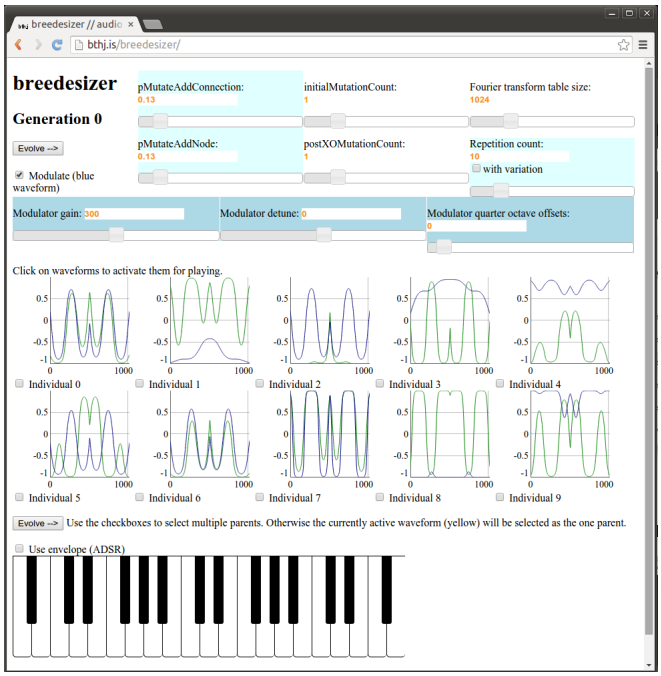


Fig. 3. Breedesizer User Interface.

as a default. Sliders are also present for adjusting how many times to attempt a mutation on each individual, which are set to five times (behind the scenes) for the individuals in the initial population, to offer a variety of waveforms from the start, and then defaulted to a setting of one attempt.

The frequency of the periodic waves input to each CPPN is adjustable with a slider in the user interface, updating the waveform drawings in real time as the slider values are changed, allowing the user to hear different timbre textures from each CPPN. An example of two waveform variations output from one CPPN, resulting from different periodic input frequencies, can be seen in Figure 4.

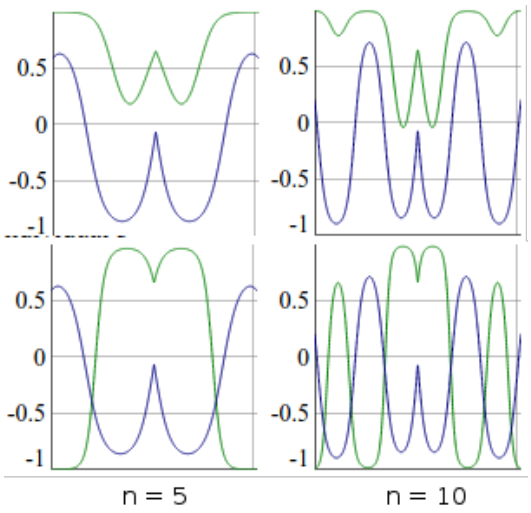


Fig. 4. Resulting waveforms from different frequencies of periodic input to one CPPN.

Three additional sliders are present in the user interface, for adjusting parameters to the FM synthesis configuration. The sliders change the amplitude of the modulator, controlling how much of an effect it has on the carrier wave; another enables the detuning of the modulator wave, where normally it is set to the same frequency as the carrier, and slightly detuning the modulator frequency produces an audible and often interesting phase change between the oscillators where they loop between an in-phase and out-of-phase state; the third slider changes the modulator frequencies by multiples of the carrier frequency, where each slider step corresponds to $carrier frequency/4$.

The presence of those sliders can be considered as contrary to the primary aim with the IEC environment presented here, which is to free the user from the requirement of expert knowledge to be able to discover interesting synthesized sounds, as the adjustment of those parameters does benefit from familiarity with the fundamentals of frequency modulation synthesis [55]. Those knowledge-demanding sliders are present in the user interface nonetheless, as the optional addition of FM synthesis (via a check-box) from the output of each CPPN, is considered an important example of the possible uses for CPPN-NEAT beyond the mere formation of waveforms to base wavetable oscillators on; and the adjustment of those parameters is important for the appreciation of the expressive power that FM synthesis has to offer.

To have played notes start and stop less abruptly and provide a familiar synthesizer characteristic, an ADSR envelope (Attack-Decay-Sustain-Release) can optionally be enabled while playing the musical keyboard.

The implementation of Breedesizer is available, in the file package accompanying this report, which can be run by opening `breedesizer-maig2014/index.html` in a web browser that implements the Web Audio API, such as Google Chrome.

IV. Results

The initial population of waveforms, presented in the Breedesizer interface, already provides an interesting variety of timbres. The miscellaneity of waveforms initially presented can be attributed to the high number of mutations performed on the individuals created from the population seeds, but the interestingness of the timbres can be attributed to the representational properties of CPPNs.

Evolution of timbres through successive generations resulted often times in increased perceived quality, or more pleasing sounds. A little less often the evolution quickly spawned generations of individuals who generated timbres gathering little interest. In the cases where evolution initially resulted in higher ratings of quality, it did not last for long as it deteriorated after a small number of generations, often around the point of fourth generation. Figure 5 shows one lineage of timbres evolved in Breedesizer.

To gather rating data, the author of this report and two other subjects drew on paper an initial line of arbitrary length, to represent the quality of the best individual selected from the first population. Lines were then drawn for the

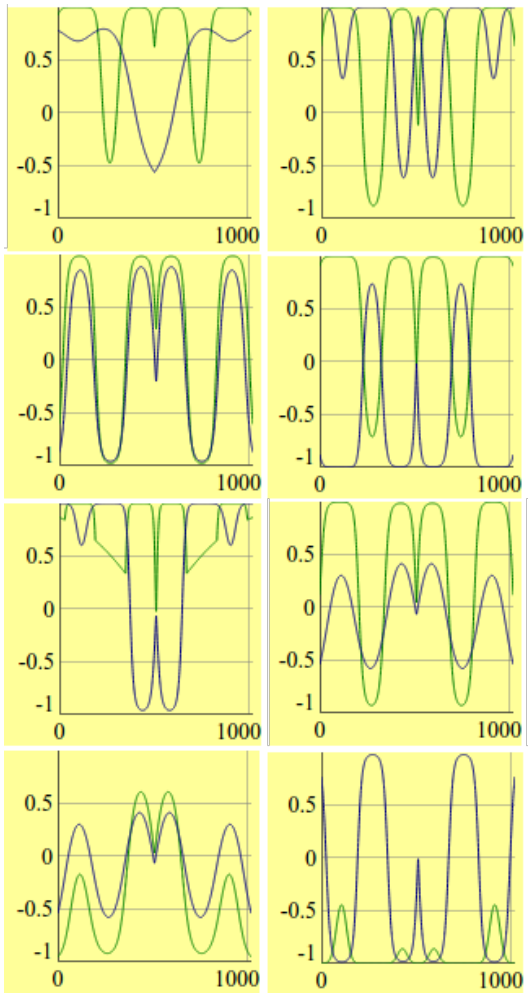


Fig. 5. One lineage of timbre waveforms evolved through successive generations (viewed from left to right, descending). The green curves represent the carrier wave, or timbre as it would be heard without FM synthesis, and the blue curves represent the frequency modulation wave, which is in effect when FM synthesis is enabled. Only one parent is selected for each generation in this evolution example. Generations evolved from the last waveform (bottom right) were monotonic, with most individuals resembling their parent. More varied directions of evolution can be had by selecting multiple parents, as can be seen in the video screen-recordings accompanying this report.

best chosen individuals in later generations, whose lengths were relative to the first line, according their judged quality compared to that form previous generations. This simple process provided instant graphs of the evolution of perceived quality of timbres through the generations, from each IEC session. Quality measurement lines of relative lengths, from three sessions, can be seen in Figure 6.

The relative lengths of lines were translated into rating numbers, in the range from 0 - 5. As it may be difficult for a test subject to estimate what is a good number for the initial rating, without knowing the relative quality of generations to come, an initial default rating of 2 was given to the first population, where ratings for consecutive generations would be relative pluses or minuses from that initial rating. A column chart of average ratings through 16 generations,

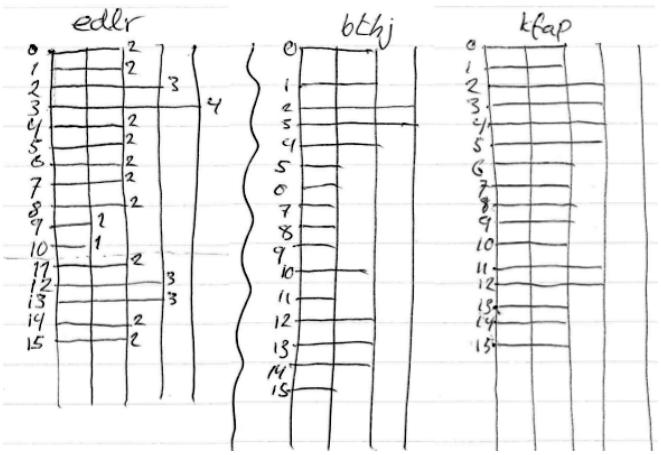


Fig. 6. Lines measuring relative perceived timbre quality, from three test sessions, as an ad hoc way of gathering data through evolved generations in the IEC process.

from seven test sessions performed by the three subjects, can be seen in Figure 7.

Those rating averages show that perceived timbre quality tends to drop after the fourth generation, but they also show a trend of increased quality after the tenth generation. It could be interesting to see whether tests performed over a longer range of generations would demonstrate an oscillating pattern. The results may be somewhat skewed by the occasional interference by the author in the testing process, where he suggested the test subjects might try to adjust the frequency of periodic inputs to the CPPNs (discussed in sections III-B and III-C). In all cases, where the test subjects reduced the input frequency (n) from the default of 10 to a lower number like 5, they perceived the timbres as more pleasing, which resulted higher ratings. Though this interference may have skewed the results, it was interesting in itself to observe that immediate effect, and shows that the settings of parameters to the CPPN inputs play a significant role.

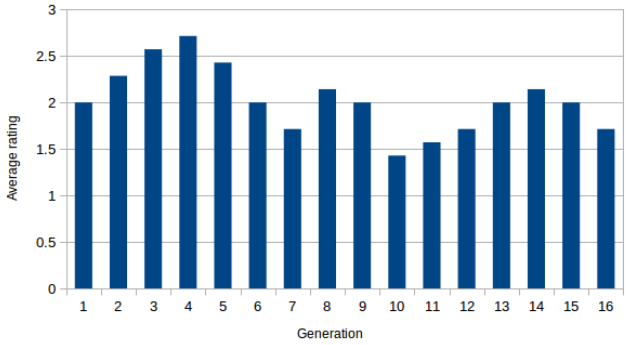


Fig. 7. Average ratings through 16 generations, from seven test session performed by three individuals.

The addition of Frequency Modulation synthesis, to the evolutionary process, gave a richer experience than the one of only listening to single waveform timbres while evaluating the quality of individuals in each population. Enabling ADSR

envelopes on played notes further solidified the feeling of evolving a useful synthesizer. While the singe waveform timbres could all have great potential in a rich synthesizer configuration, it is hard for the untrained user to spot that potential. The synthesis and amplitude envelope features may have helped to appreciate the usefulness of the timbres, and at least they increased user engagement in the process, where they produced fuller and more interesting sounds.

The sliders for adjusting parameters to the FM modulations synthesis added some complexity to the interface and user experience, where diverse sets of sounds could be obtained from each individual, by varying the slider settings. Adjusting the frequency of periodic inputs to the CPPNs, with a slider, also varied the sounds each individual oscillator would produce, as previously discussed. Rather than having those values interactively adjustable, they could more properly be part of the genome and evolve along with the CPPN structures.

Screen-recordings of IEC sessions in Breedesizer are available in the file package supporting this report. They give examples of the different sounds gained from having FM synthesis enabled or disabled, and from enabling or disabling ADSR envelopes, as well as the variety obtained from changing the frequency of periodic inputs. Hopefully they somewhat convey how enjoyable the interactive evolution of timbres can be.

V. Discussion

Using the powerful representational properties of CPPNs to form periodic waves, producing a variety of timbres with wavetable oscillators based on them, has shown to be an interesting process, though limited, where the complexifying ability offered by evolution of the CPPN networks with NEAT often resulted in increasingly pleasant timbres for only a few generations, with the interestingness of the phenotypes deteriorating after a handful of generated populations. Whether that process is more interesting than other methods of wavetable construction, for example free-form drawing or shaping Bézier curves by hand, is subject to evaluation, but at least it can be considered as a viable alternative.

The trend of timbre quality deteriorating after less than ten generations, may be a result of working in one dimension less than previous works evolving two- (or three-) dimensional images, such as Picbreeder. That hypothesis might deserve further investigation. Providing the CPPNs with different sets of activation functions - other than the current set of Gaussian, Bipolar sigmoid, sine and linear - such as the Step and Spike functions [56], could have a significant effect on the timbre evolution, and so might also be worth investigating.

Outputting more waveforms from each CPPN results in a larger search space, that may be more interesting to explore with CPPN-NEAT than the smaller spaces of single periodic waveforms, or two in the case of FM synthesis. The addition of FM synthesis, to the evolutionary process with the prototype discussed in this report, gave an example of how the exploration can become more interesting, when more dimensions are added to the search space. Other audio synthesis techniques are based on the combination of many periodic waveforms, both for timbre and control, and they could provide interesting search spaces for exploration.

Wavetable synthesis, commonly understood as the technique of sweeping through a table containing multiple waveforms [9][8][10], could form one of those search spaces, where the CPPNs would output waveforms for each entry in the wavetable, and one additional waveform to define a curve guiding the sweeps through the series of periodic waveforms in the table, for the duration of each note played.

Arranging waveforms on a two-dimensional grid and using another waveform to guide a point through the Cartesian coordinate plane, cross-fading between the waveform sound sources as it passes them, in a manner similar to Vector synthesis [57] and its evolution implemented by Animoog's Anisotropic Synth Engine [58], could make for another interesting search space.

Granular synthesis, or even Graintable synthesis [59], could constitute yet another search space, where "sounds are broken into tiny grains which are then redistributed and reorganised to form other sounds"[60].

> A grain is a small piece of sonic data. In granular synthesis it will usually have a duration between 10 to 50 ms. The grain can be broken down into smaller components, the envelope and the contents. The envelope is used primarily so that there is no distortion and crunching noises at the beginning and end of the sample. The shape of the envelope though has a significant effect on the grain. The contents of the grain is audio. This can be derived from any source. Sine wave, square wave, audio sample, etc. [60]

Those grains could be produced as outputs from CPPNs.

Whole synthesizer configurations, where oscillator outputs flow through filters and are affected by control signals [61], could be evolved with NEAT, where instead of specifying activation functions, each network node would designate one component of the synthesizer, and the weights on connections between nodes would define the strength of interaction between the components. Such a configuration could obviously be called a *Compsitional sound-synthesis network* (CSSN). The prospect of configuring synthesizers with a CSSN-NEAT could be appealing.

References

[1] D. Wright, *Mathematics and music*. American Mathematical Soc., 2009, vol. 28.

[2] A. S. Association *et al.*, "Acoustical terminology si, 1–1960," *American Standards Association, New York*, 1960.

[3] J. M. McDermott, "Evolutionary computation applied to the control of sound synthesis," 2008.

[4] R. D. Patterson, T. C. Walters, J. J. Monaghan, and E. Gaudrain, "Reviewing the definition of timbre as it pertains to the perception of speech and musical sounds," in *The Neurophysiological Bases of Auditory Perception*. Springer, 2010, pp. 223–233.

[5] Timbre (from wikipedia, the free encyclopedia). [Online]. Available: http://en.wikipedia.org/wiki/Timbre

[6] Classic waveshapes and spectra. [Online]. Available: http://in.music.sc.edu/fs/bain/atmi02/cw&s/index.html

[7] Sound synthesis theory/oscillators and wavetables. [Online]. Available: http://en.wikibooks.org/wiki/Sound_Synthesis_Theory/Oscillators_and_Wavetables#Wavetables

[8] Wavetable synthesis. [Online]. Available: http://en.wikipedia.org/wiki/Wavetable_synthesis

[9] G. Reid. Thor demystified 11: The wavetable oscillator – part 1. [Online]. Available: https://www.propellerheads.se/blog/tutorials/thor-demystified-11-the-wavetable-oscillator-part-1/

[10] S. Howell. Q. can you explain the origins of wavetable ... synthesis? [Online]. Available: http://www.soundonsound.com/sos/feb06/articles/qa0206_1.htm

[11] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Computer Music Journal*, pp. 46–54, 1977.

[12] C. G. Johnson, "Exploring the sound-space of synthesis algorithms using interactive genetic algorithms," in *Proceedings of the AISB'99 Symposium on Musical Creativity*. Society for the Study of Artificial Intelligence and Simulation of Behaviour, 1999, pp. 20–27.

[13] ——, "Exploring sound-space with interactive genetic algorithms," *Leonardo*, vol. 36, no. 1, pp. 51–54, 2003.

[14] J. Mandelis, "Genophone: An evolutionary approach to sound synthesis and performance," *Proceedings ALMMA*, pp. 37–50, 2001.

[15] ——, "Adaptive hyperinstruments: applying evolutionary techniques to sound synthesis and performance," in *Proceedings of the 2002 conference on New interfaces for musical expression*. National University of Singapore, 2002, pp. 1–2.

[16] J. McDermott, N. J. Griffith, and M. O'Neill, "Toward user-directed evolution of sound synthesis parameters," in *Applications of Evolutionary Computing*. Springer, 2005, pp. 517–526.

[17] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis," *Computer Music Journal*, pp. 17–29, 1993.

[18] P. Dahlstedt, "Creating and exploring huge parameter spaces: Interactive evolution as a tool for sound generation," *Proceedings of the 2001 International Computer Music Conference*, pp. 235–242, 2001.

[19] Y. Lai, S.-K. Jeng, D.-T. Liu, and Y.-C. Liu, "Automated optimization of parameters for fm sound synthesis with genetic algorithms," *International Workshop on Computer Music and Audio Technology*, 2006.

[20] J. Serquera and E. R. Miranda, "Evolutionary sound synthesis: rendering spectrograms from cellular automata histograms," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 381–390.

[21] E. R. Miranda, "Evolving cellular automata music: From sound synthesis to composition," in *Proceedings of 2001 Workshop on Artificial Life Models for Musical Applications*, 2001.

[22] J. Manzolli, A. Maia Jr, J. Fornari, and F. Damiani, "The evolutionary sound synthesis method," in *Proceedings of the ninth ACM international conference on Multimedia*. ACM, 2001, pp. 585–587.

[23] M. F. Caetano, J. Manzolli, and F. J. Von Zuben, "Interactive control of evolution applied to sound synthesis." in *FLAIRS Conference*, 2005, pp. 51–56.

[24] R. Garcia, "Growing sound synthesizers using evolutionary methods," in *Proceedings ALMMA 2001: Artificial Life Models for Musical Applications Workshop,(ECAL 2001)*, 2001.

[25] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 3, pp. 526–553, 2009.

[26] E. R. Miranda, L. Bull, F. Gueguen, and I. S. Uroukov, "Computer music meets unconventional computing: towards sound synthesis with in vitro neuronal networks," *Computer Music Journal*, vol. 33, no. 1, pp. 9–18, 2009.

[27] J. Mandelis and P. Husbands, "Don't just play it, grow it!: Breeding sound synthesis and performance mappings," in *Proceedings of the 2004 conference on New interfaces for musical expression*. National University of Singapore, 2004, pp. 47–50.

[28] A. K. Hoover, P. A. Szerlip, and K. O. Stanley, "Interactively evolving harmonies through functional scaffolding," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 387–394.

[29] A. K. Hoover and K. O. Stanley, "Exploiting functional relationships in musical composition," *Connection Science*, vol. 21, no. 2-3, pp. 227–251, 2009.

[30] A. K. Hoover, P. A. Szerlip, and K. O. Stanley, "Generating musical accompaniment through functional scaffolding," in *Proceedings of the Eighth Sound and Music Computing Conference (SMC 2011)*, 2011.

[31] A. K. Hoover, M. P. Rosario, and K. O. Stanley, "Scaffolding for interactively evolving novel drum tracks for existing songs," in *Applications of Evolutionary Computing*. Springer, 2008, pp. 412–422.

[32] N. Tokui and H. Iba, "Music composition with interactive evolutionary computation," *Proceedings of the 3rd international conference on generative art*, vol. 17, no. 2, pp. 215–226, 2000.

[33] P. Dahlstedt, "A mutasynth in parameter space: interactive composition through evolution," *Organised Sound*, vol. 6, no. 02, pp. 121–124, 2001.

[34] ——, *Sounds Unheard of: Evolutionary algorithms as creative tools for the contemporary composer*. Chalmers University of Technology, 2004.

[35] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[36] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Evolving content in the galactic arms race video game," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. IEEE, 2009, pp. 241–248.

[37] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.

[38] S. Risi and J. Togelius, "Neuroevolution in games: State of the art and open challenges," *CoRR*, vol. abs/1410.7326, 2014. [Online]. Available: http://arxiv.org/abs/1410.7326

[39] Animoog (product page). [Online]. Available: http://www.moogmusic.com/products/apps/animoog-0

[40] Din is noise synthesizer. [Online]. Available: http://dinisnoise.org/

[41] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley, "Picbreeder: A case study in collaborative evolutionary exploration of design space," *Evolutionary Computation*, vol. 19, no. 3, pp. 373–403, 2011.

[42] J. Ye and S. Chen. Soundbreeder with multineat. [Online]. Available: http://web.cs.swarthmore.edu/~meeden/cs81/s14/papers/AndyLucas.pdf

[43] S. M. DeBoer and K. O. Stanley, "Systems and methods for inducing effects in a signal," Dec. 3 2013, uS Patent 8,600,068.

[44] S. DeBoer and K. Stanley, "Systems and methods for inducing effects in a signal," Oct. 30 2008, uS Patent App. 11/742,317. [Online]. Available: http://www.google.com/patents/US20080267419

[45] If 'npm' is an acronym, why is it never capitalized? [Online]. Available: https://www.npmjs.org/doc/misc/npm-faq.html#if-npm-is-an-acronym-why-is-it-never-capitalized-

[46] P. Szerlip and K. O. Stanley, "A proposed infrastructure for adding online interaction to any evolutionary domain," *arXiv preprint arXiv:1407.3000*, 2014.

[47] ——, "Steps toward a modular library for turning any evolutionary domain into an online interactive platform," in *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, vol. 14, 2014, pp. 726–733.

[48] Browserify. [Online]. Available: http://browserify.org/

[49] Component. [Online]. Available: https://github.com/componentjs/component

[50] R. Bain. Waveform & timbre. [Online]. Available: http://in.music.sc.edu/fs/bain/atmi02/wt/index.html

[51] J. T. Folsom-Kovarik. Picbreeder: Analyze image dna. [Online]. Available: http://picbreeder.org/analysisDocumentation.pdf

[52] The createperiodicwave method. [Online]. Available: http://www.w3.org/TR/webaudio/#dfn-createPeriodicWave

[53] S. Molines. Using fourier transforms with the web audio api. [Online]. Available: http://www.sitepoint.com/using-fourier-transforms-web-audio-api/

[54] G. Renaudeau. Frequency modulation (fm) with web audio api. [Online]. Available: http://greweb.me/2013/08/FM-audio-api/

[55] Frequency modulation synthesis: Introduction to fm. [Online]. Available: http://rhordijk.home.xs4all.nl/G2Pages/FM.htm

[56] E. Hastings. Cppn/ann activation functions. [Online]. Available: http://www.cs.ucf.edu/~hastings/index.php?content=ann

[57] Vector synthesis, from wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Vector_synthesis

[58] A. Genito. Moog music – animoog v2. [Online]. Available: http://www.ageofaudio.com/en/apps/moog-music-animoog-v2

[59] Granular synthesis. [Online]. Available: http://en.wikipedia.org/wiki/Granular_synthesis

[60] T. Opie. What is granular synthesis? [Online]. Available: http://www.granularsynthesis.com/guide.php
[61] B. Sievers. A young person's guide to the principles of music synthesis. [Online]. Available: http://beausievers.com/synth/synthbasics/