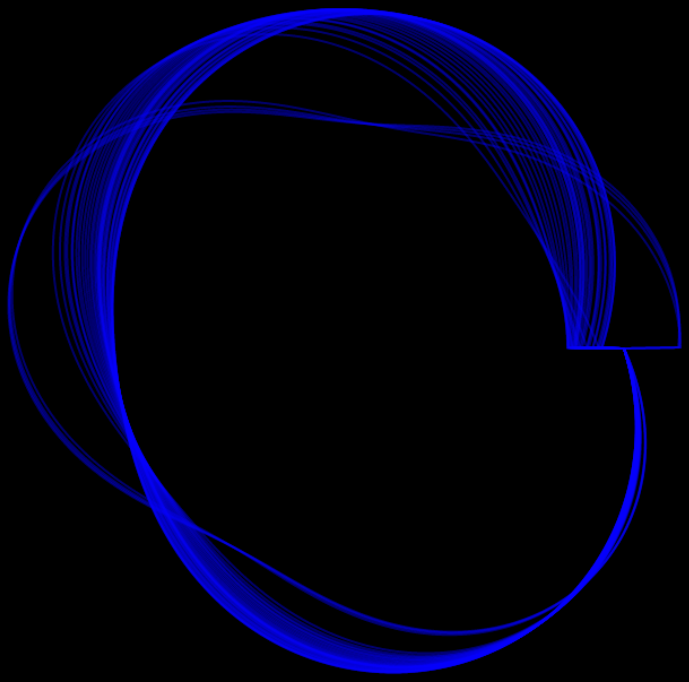
J A M I E S T U A R T

AUDIO PROGRAMMER & SOUND ENGINEER



P O R T F O L I O

2011 - 2015

CONTENTS

1. AUDIO PROGRAMMING PROJECTS

1.1 AI Compressor

An intelligent automated dynamic range compressor audio plugin and standalone analysis application.

1.2 AlgoGranny

An Algorithmic Granular Synthesis Application that transforms inputted audio into a five minute long composition.

1.3 Beat Genetics

An interactive genetic algorithm orientated system that generates and evolves drum beat sequences based upon the subjective aesthetic evaluation of the user.

1.4 Bare Foot Loopy

An Interactive, Intelligent and Responsive Looping and Beat Generation Interface using physical touch pads.

1.5 Modular Synthesizer

A Wavetable/Sampler Modular Synthesizer, with amplitude, ring and frequency modulation. Direct sample level digital signal processing (DSP).

1.6 Framingham Heart Study Sonification

Algorithmic composition generated with data from the Framingham heart study of human features to control sound synthesis, effect units, pitch and rhythmic phrases.

1.7 Flexi Synth

Flexi Synth is a synthesizer, which uses a combination of additive, subtractive and frequency modulation synthesis. Includes a routing matrix to connect LFO's to filters and other destinations and uses an ADSR envelope.

1.1

{ AI COMPRESSOR }

Final year graduation project at the University of Sussex. An intelligent automated dynamic range compressor audio plugin and standalone analysis application.

https://github.com/jamesnapierstuart/19329-AI-Compressor

1.1 AI Compressor

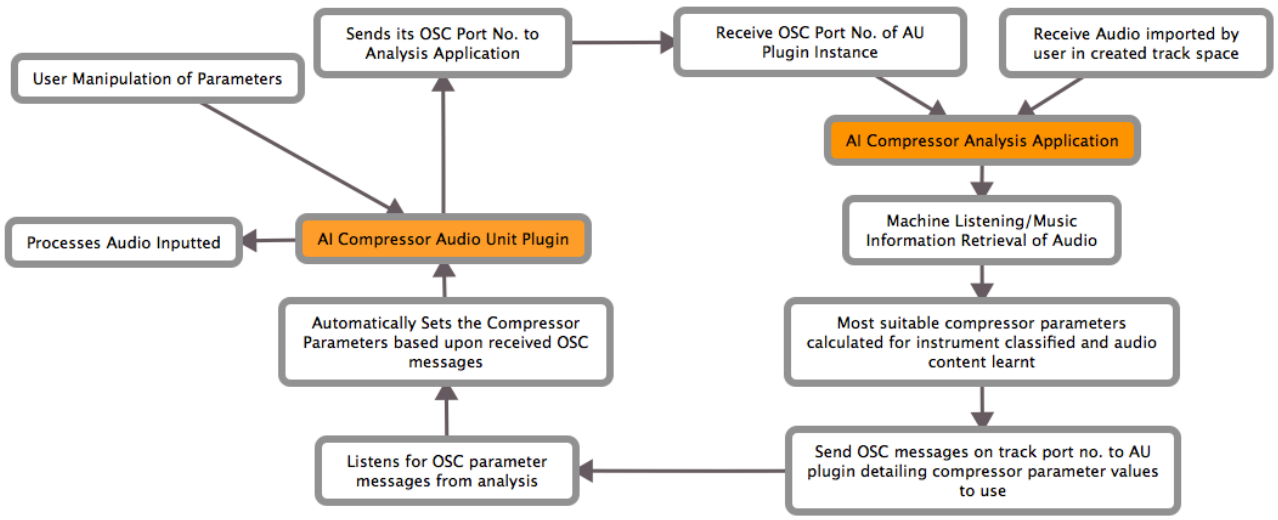
Overview

"Towards Intelligent Autonomous Audio Plugins: Implementing an Intelligent Dynamic Range Compressor Using Machine Learning Approaches".

An intelligent automated dynamic range compressor audio plugin and standalone analysis application. The audio analysis interface for music information retrieval and machine learning computations was written in the SuperCollider Language and the compressor plugin's digital signal processing was built using the C++ AudioUnit Framework. Communication between these two interfaces was handled by Open Sound Control (OSC) protocols.

The intelligent compressor is an AU plugin component that users add to their existing plugin libraries, to be used within their digital audio workstations. It functions in a similar manner to other compressor plugins by performing digital signal processing upon audio in real-time but differs by launching an external analysis application to perform machine listening and music information retrieval from the current audio stem being processed.

From analysing the audio in non real-time the external application calculates the most suitable compression parameter values (threshold, ratio, attack & release and makeup gain) and sends these over OSC to the real-time audio unit plugin, which adjusts its parameters to the received settings automatically.

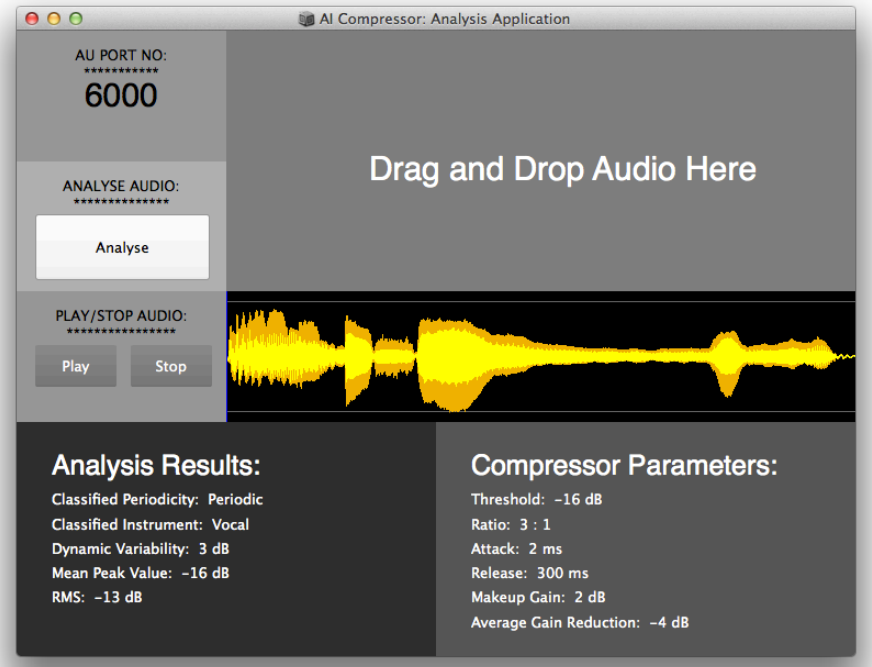


Architecture Overview

1.1 AI Compressor

SuperCollider Analysis Application Architecture

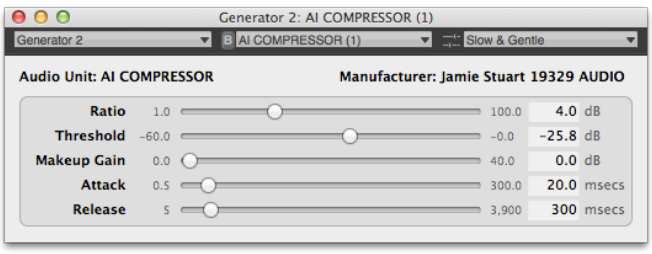
The AI Compressor Analysis Application was written inside SuperCollider for all its functionality. This meant music information extraction and machine learning code could be executed concisely and running the program under a single framework instead of branching out to external ones made it time and computationally efficient. The graphical user interface kept to clear, informative aesthetics and presented information that the target user would be familiar with.



AI Compressor GUI

Audio Unit Compressor Interface

The AudioUnit Framework provided by Apple, enables the developer to create either a ‘Generic View’ or ‘Custom View’. The generic view allowed automatic GUI creation from the parameters I set, however care had to be taken to specify the data types for certain parameters such as decibels and milliseconds otherwise the functionality failed. Using a GUI for the compression implementation fulfilled certain requirements, allowing users to manipulate parameters it also facilitated parameter automation within DAWs.

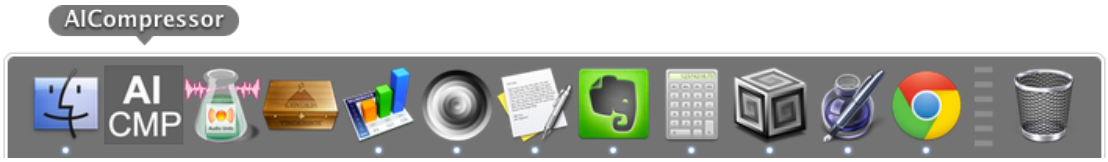


Audio Unit Plugin

1.1 AI Compressor

Compiling to Native Application

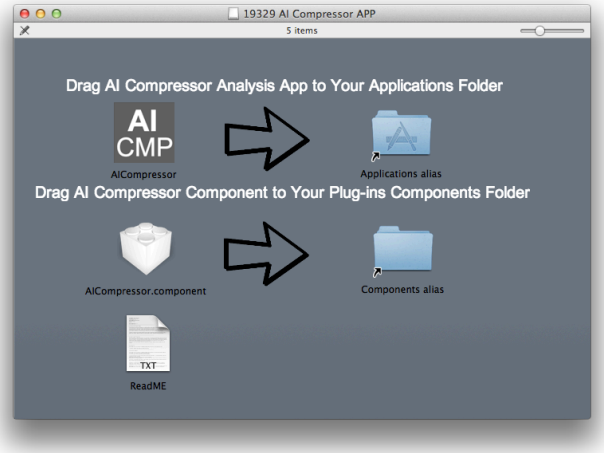
The initial aim of implementing machine learning and music information retrieval processes, was to create a set of SuperCollider extensions to be executed within SuperCollider as a proof of concept. However the build and distribution of the analysis program as a standalone application managed to be realised in addition to the SuperCollider extensions meaning it could be executed as a native application on OSX, accomplishing a future project aim.



Dock App Icon

Packaging up the Audio Plugin & Analysis App as Disk Image

Experience with purchasing music applications and audio plugins gave insight into the distribution of these products. The most common and clean way for users to get hold of audio plugins and apps is through downloading Disk Image (DMG) files from an audio company's website. These files provide the application/component content and alias folders for users to simply drag and drop the new files to the necessary destinations without having to see read me's or manuals beforehand. Disk Image conventions have a background image that has text explaining the action the user must perform and an arrow of where to drag their products to. This although surface level design, was still a step forward for this project, since the intended users are be mix engineers and musicians who are familiar with these paradigms and won't expect anything less than an easy install.



Disk Image

1.1 AI Compressor

Evaluation & Conclusions

The main purpose of this project was to build a proof of concept system that solved the problem areas associated with current generation audio plugins. I believe the AI Compressor system has been a successful implementation that achieved established requirements and with future development could be a system commonly used by engineers. Findings from the evaluation showed improved digital signal processing needs to be implemented and approaches to compression calculations should be based on deeper musical information extraction rather than focusing upon the instrument itself. Initial objectives were to create the system within the SuperCollider program, this was exceeded with the implementation of an AU plugin that can send and receive OSC messages and a standalone analysis application that can run as a native Mac application.

Future Projects

Main developments should focus upon creating a system that retrieves means and standard deviations of peak values, RMS, dynamic variability, transient envelope measures and MFCC data. Different machine learning models should then be taught the sonic meanings of ‘soft’, ‘hard’, ‘medium’ compression types by having experienced mix engineers carry out compression processing upon audio samples. The realised system could then analyse any audio data and apply the most suited parameter settings based upon lower level feature extraction rather than higher instrument classification abstractions and thereby allowing the user to cycle through different compression types that will stay suited to the audio.

1.2

{ ALGOGRANNY }

An Algorithmic Granular Synthesis Application that transforms inputted audio into a five minute long composition.

https://github.com/jamesnapierstuart/19329-AlgoGranny

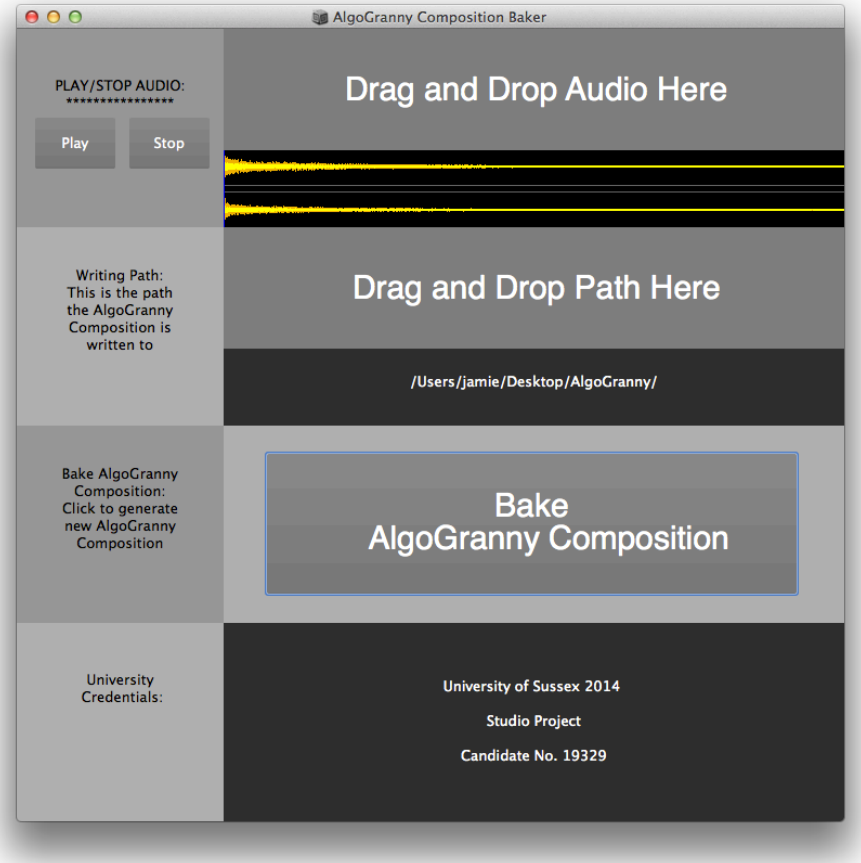
1.2 AlgoGranny

Overview

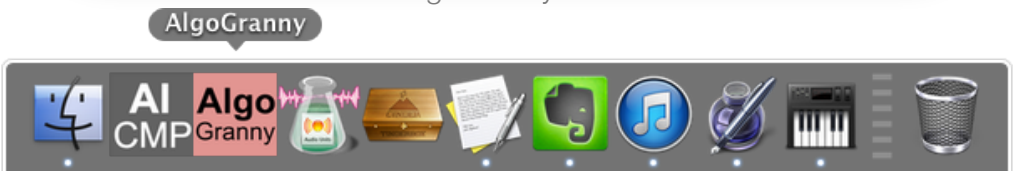
AlgoGranny is an Algorithmic Granular Synthesis Application that transforms inputted audio into a five minute long composition. It breaks down any track or audio sample into grains and rearranges them into a new organised compositional form.

Since the majority of all studio orientated tasks I had undertaken in the past were carried out in higher abstraction environments such as Logic Pro and Pro Tools, I decided to challenge myself by composing in lower levels of sound manipulation. This was the main reason for creating a granular synthesis composition since granular methodologies focus on the most minute instances of sound.

Written within the SuperCollider synthesis language. AlgoGranny aimed to create an interface that could take in any audio input, no matter what the context and create an aesthetically pleasing piece of music using granular approaches.



AlgoGranny GUI



AlgoGranny OSX Native App in the Dock

1.2 AlgoGranny

Compositional Material

Compositional techniques implemented were based upon analysis from listening to the works of Barry Traux, Curtis Roads and other modern granular synthesis pieces that involve visual manipulations. Listening to these works I discovered that there seems to be a focus on two main extreme poles which granular systems can provide with shades of gray in between. First the completely staccato, sporadic and splintering like sounds of hearing the individual grains on their own. This was used as inspiration for section four of my piece where the attack and release durations of envelopes applied to grains restricted their durations and exhibited flickering like sounds.

Secondly the referenced pieces also share musical movements, that see separate, isolated grains morph together to create what are known as ‘clouds’. These clouds can give pitch and timbre coherency between the individual grains and create macro soundscapes. I found myself favoring these cloud like structures and therefore spent most of my time creating opportunities for grains to mesh together to reach what I would label sonic storms. Within section one of the AlgoGrannyCompose.sc I used huge sequences of iterations to create crossovers between individual grain envelopes to create fully blocked sounds that would not seem to have arise form microscopic parts of a sound. These also exhibit ambient and drone like textures that draw the listener in and stretch their minds and ears along with the warping of the sound grains.

The sections, grains and many of the synth parameters were given random seedings to give a sense of movement, especially apparent for panning and the rate of grain playback.

The Microscopic Level

To obtain and manipulate grains of sound, I decided to constrict grain sizes to 690 samples each approx 15ms long. This was achieved by dividing the waveform under study by the amount of times 689 fitted inside the track’s number of frames. Having grain sizes of 690 was initially chosen because I intended to create a composition that was completely formed on a sample by sample level with no help from higher level tools such as SuperCollider UGens or plugin effects. Experiments were carried out on how to loop audio, create white noise and slow or speed up sound purely on their amplitude values. However this proved to be a very hollow and uninspiring paradigm to compose within. Although the challenge was very appealing and I gained an invaluable understanding of DSP’s (digital signal processing) role within all digital musical applications I chose to create this composition with the aid of existing tools instead of programming everything entirely from scratch.

1.2 AlgoGranny

Evaluations & Conclusions

Implementing distortion and clipping as an artistic decision, sounded great whilst the compositions were outputted, however on playback within a DAW the waveform was very skewed and clicks occurred when hoping the mouse through the audio To combat this in future implementations limiters could be placed upon the whole outputted audio. In addition more use of Granular UGens such as Tgrains and Warp could be used for greater effect. Using different test audio examples showed that system didn't break and could still perform well creating an abstract piece.

I believe the aim of creating a system that could create an aesthetically pleasing composition, given any audio file was achieved. However I think the algorithm which leads to this aesthetic outcome is probably too restrictive and it would be good for future development to create wider and deeper algorithmic branches. The design of the application was kept very general using classes to abstract and keep concise code, so changes for future development can be easily made given this generalised approach.

Future Projects

Build a system where the recorded output could be put back into the system or sister system and the original file manipulated becomes the output. Raising implications of file encryption and piracy, topics that would be interesting to explore in a compositional framework.

P.T.O.

1.3

{ BEAT GENETICS }

An interactive genetic algorithm orientated system that generates and evolves drum beat sequences based upon the subjective aesthetic evaluation of the user.

https://github.com/jamesnapierstuart/19329-Beat-Genetics

1.3 Beat Genetics

Overview

Beat Genetics is a genetic orientated system that generates and evolves drum beat sequences based upon the subjective aesthetic evaluation of the user. To realise this project, an interactive genetic algorithm was implemented whose fitness function was determined by the amount of time a user spends listening to a particular beat. The longer a beat is listened to, the greater fitness score it receives.

Beat Genetics GUI

Execution of Concept

The Beat Genetics GUI is divided into six sections, each for every beat in the population. The beat number is listed and so is the generation number underneath, which increments each time the user clicks the next generation button.
To make sure fitness scores didn't interfere with one another, I made sure that only beat could be played at a time. In addition the next generation can only be executed once all beats have been stopped to make sure when the user next listens to the beats, they hear the newly bred children rather than their parents.

1.3 Beat Genetics

Program Overview

1. Generate a random population of beats for all instruments: (Kick, snare, closed hihat, ride cymbal, tom, clap, percussion one, percussion two)

    a. A single drum beat is created using two Pseq arrays that will be imbedded into a Pbind. The first array dictates the level (whether a drum beat is played or not) and the second array assigns the duration of each hit (measured in note lengths).

    b. The array sizes are randomly selected from a range of 1-32. The level array values have a 50% chance of being a 1 or 0 (on or off) and the duration array is populated with values 4/sizeOfArray. Meaning if size = 32 then duration values will be 4/32 = 0.125, which all sum to 4, the time signature that all beats conform to.

    c. Each instrument consists of the amount of beats dictated by the population size. E.g. if the population size is 6 then each instrument will have 6 different beats.

2. The graphical user interface is divided into six sections (the population size) to signify each current generation beat. The user then clicks play or stop upon each beat in turn and the amount of time spent listening to each beat is tracked by the program using the TempoClock. Time listened per beat is stored as a fitness score, the longer the user listens, the higher the fitness score is given.

3. Once the user has listened through each beat and fitness scores have been acquired. The 'Next Generation' button is pressed to create the new set of children beats.

    a. The fitness scores acquired are normalised to give probability values for each beat as a potential parent in the mating pool. Those with higher probabilities will be more likely to become parent candidates in the pool.
    b. Next, two parents are chosen at random from the mating pool to be used as parents for the child beat.
    c. The size of the child level and duration arrays have a 50% chance of being either ParentA's or ParentB's size.
    d. Crossover genes for each level and duration value have a 50% chance from either parent.
    e. Mutation has a 25% chance for each level gene becoming 0 or 1. And 25% mutation chance for each duration gene becoming 4/random(1..32).
    f. Since we choose between array sizes and mutate the duration values the sum of the durations may add up to less than 4 or more than 4, which is not ideal. Therefore a quantization method was used to either contract or expand the arrays to make sure the sum of all durations equals four and the level array size is equal to the duration size.

4. These steps are repeated until we have the complete next generation. Fitness scores are reset and the beat offspring are ready to be listened to and evaluated by the user for another iteration. Time listened per beat is stored as a fitness score, the longer the user listens, the higher the fitness score is given.

1.3 Beat Genetics

Evaluations & Conclusions

An Given the nature of the algorithm, the system can flow in many different ways. Since the fitness function is based upon a user's subjective aesthetic value, the beats can tailor to faster hits (smaller durations), longer hits, minimal beats, dense beats or a deeper variety of patterns.

Within the crossover function when breeding new beat children, the size of each level and duration array is randomly chosen between ParentA or ParentB. I initially programmed the system to always choose the shortest Parent size to avoid creating a quantization method. Implementing this though meant with every generation the beats would become smaller and smaller and eventually reduce to no sound at all. Alternatively I could have always chosen the Parent with the largest array size but this would of had the opposite effect by creating longer and longer length beats. Therefore the quantization function was the best approach to tackle such a problem, because it lets us randomly select the array size each time, meaning the beat length doesn't converge to either very small or very large but constantly varies its length throughout the family beat tree.

A large limitation of the system arises from its multi-layered functionality. Each beat consists of eight different instruments all with different rhythms. If the user enjoys the kick pattern in a specific beat but not the snare, there is no opportunity to just select the kick to be continued in future generations without also regenerating the unfavored snare too. So it performs the genetic algorithm as a blanket that covers the entirety of a single beat, rather than being specific for each instrument. To solve this problem the GUI could allow for different screens to represent different instruments within a beat, however this would mean the user would have to listen through each beat for each instrument = 6*8 = 48, too time consuming.

This was my first attempt at implementing a genetic algorithm within a programming project and I believe that I have achieved this to a good standard. Being able to mutate and cross-breed arrays of varying sizes proved difficult but the quantization method I created to solve this problem was innovative and helped achieve the goal of creating a genetically orientated beat generator. It fulfills the aim of being flexible per user, generating different style beats at the appropriate times.

Future Projects

As discussed within the limitations, having a multi-layered system that plays numerous rhythms at once can create problems for the user especially when they like one instrument but not another. Therefore future developments would see an interactive genetic algorithm that can handle this subjective distinction performing well on a multiple levels without being time inefficient.

P.T.O.

1.4

{ BARE FOOT LOOPY }

An Interactive, Intelligent and Responsive Looping and Beat Generation Interface. Converts live onset detection to drum samples, listens and responds to melodic phrases in the same key/scale and loops user's live input intelligently. All controlled by a physical touch pad interface.

https://github.com/jamesnapierstuart/19329-Barefoot-Loopy

1.4 Bare Foot Loopy

Overview

An interactive/responsive looping and beat generation interface.
Overall functionality; The system has three levels/modes. Mode one, ‘Looping’ takes user’s live input and loops it, intelligently expanding or contracting data to reach whole bars. Mode two, ‘Beat Generation’ takes users’ onset data and creates beats from the rhythms of the onsets. Behind these two modes, the system is listening and responding intuitively in the same key, scale and rhythms as the user has created. Mode Three, ‘Mute Control’ allows users the ability to solo or mute out the three main aspects of this design, live looping, beat generation and automative response.

Execution of Concept

The graphical user interface and all of the computation was written in SuperCollider. The ‘Program Overview’ section details the breakdown of how the system was coded.

For the physical user interface I used the newly released Makey Makey Arduino board that allowed for keyboard input from any analog source that conducted electricity. Custom hardware was built using a cut up yoga pad to make pads, tin foil was placed over them in an outlined square and a smaller square of foil inside to make a gap in the circuit. The user’s foot would be placed over the gap to close the circuit and thus cause electricity to flow and trigger a keyboard press on the Makey Makey board.

The keyboard presses were analysed in SuperCollider and matched with different program functions so that when the user touched the foil pads with their feet, it triggered a response from the musical system.

1.4 Bare Foot Loopy

Program Overview

1. Music Information Retrieval (Responsive System)

   Prior to creating the interface, the retrieval analysis and interpretation of musical data needed to be addressed. Pitch recognition and key detection were implemented into this system and achieved by adapting Nick Collins' OnlineMidi.sc Supercollider code which is based upon key profiles established by Krumhansl and Kessler (K-K profiles)

2. Quantization of audio/conformity to whole bars

   "One Main aim of interaction design is to reduce the negative aspects (e.g. frustration and annoyance) of the user experience while enhancing the positive ones (enjoyment and engagement)". In order to adhere to these principles I needed to make sure musical errors such as timing didn't become too much of a problem for performers. For example, its very common to make small errors in timing and so to make sure this didn't become a really big issue, I implemented my own algorithm to analyse the nearest whole bar to where the user had stopped playing would be. This 'conformity or quantization' means that regardless of the user playing too much or too little to make up a bar the algorithm would always take care of it for them so they would always be aligned in whole bars.

3. Onset Detection and Quantization to Beat Generation (Turning Onsets into Beats)

   Although the quant: 4 method undermined my initial quantization algorithm I had to develop a new, far more complex one for the use of onset detections to be turned into beats. Using the sending of OSC messages at certain rates we can constrict the audio input to be for example at at rate of 1/16th notes or 8hz. However, this means the user has to be absolutely precise with their playing in order to place an onset of a transient directly on the 1/16th count. To resolve this problem I calculated the note durations of sending out OSC messages at a rate of 44100/1024 and determined that 5 values would correspond to a single 1/16th note duration value.

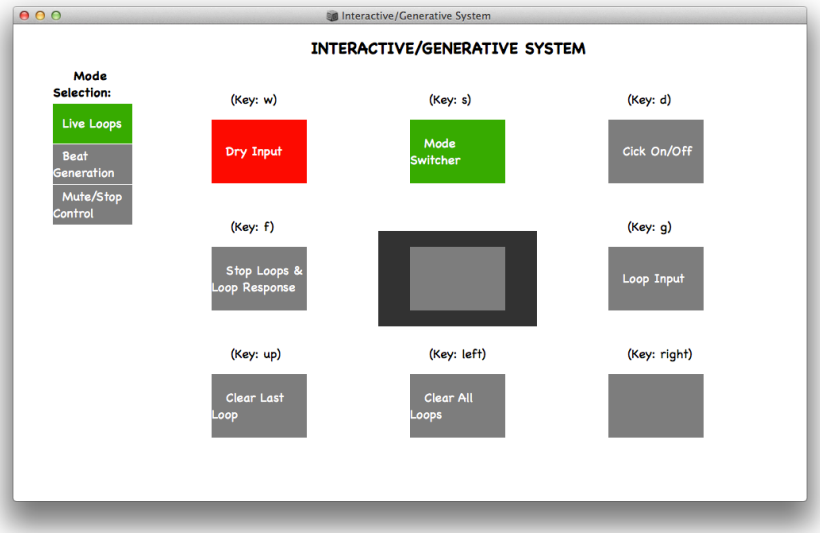
   Therefore we can look for every five values being streamed in from user input, if there is an onset detected in this five, then we shall assign an onset to a 1/16th note duration for use in beat patterns.

1.4 Bare Foot Loopy

Description of the Interface

A 3X3 grid of electronic pads was connected to the computer and Supercollider via a Makey Makey. Figure 1, shows the outboard electronic pads the users step on to perform with. And Figure 2, shows the GUI the user also views simultaneously on the computer. The GUI maps identically to the real world which means when users switch mode they see on the computer which pads have changed their functionality by their label and therefore immediately understand that their actions will cause different effects in different modes.

Interactive Loop Pads



Bare Foot Loopy GUI feedback
display of live physical input

1.4 Bare Foot Loopy

Interacting with the Interface

The user can switch between the modes any point they like with no barriers. The looping mode lets them create multiple sequences of live input that loops in the amount of bars they have been conformed too. The system immediately responds by using the pitch data gathered from these loops and uses it to spit back out.

Whilst in beat generation mode, users make shorter tighter sounds to create more precise onsets for the system to recognize, which it does accurately. Depending on the pad stepped on kick, snare, hihat or tom patterns will be outputted in the rhythm of their onsets. In addition the system will also respond by paying degrees of the key it has detected the user is in. For the kick it plays in sync with the kick rhythm in a low register to mimic that of a bass synth. This following of rhythm in the key is followed by the snare who's partner response synth dots about randomly in the key. The hihat synth counterpart plays in the gaps of the hat hits.

When in the mute/stop control mode the user is able to control all three aspects of this design by muting them whole. This allows for silence, or soloing of certain patterns that the user may wish to focus on. And at any point they can use the dry input pad to improvise over what is being created between them and the computer.

Evaluations & Conclusions

In terms of the breadth of functionality that the system is able to handle I believe I have certainly achieved in creating an interaction and responsive loop and beat generation interface. It lacks for sophistication from the response synths but the onset quantization to beat generation algorithm is definitely a very highly important and enjoyable feature to use.

The interface itself is also very easy and user friendly as it takes the metaphor of a dance mat which users associate with enjoyment. It could certainly do with improving in terms of materials but the fun of using everyday materials was the inspiration so I’m glad the outcome saw this realised.

The onset quantization beat generation algorithm is something I wish to pursue, using natural language as the next input. Where people will be able to say the word of the instrument in time with tempo and that is what will be outputted by the system.

1.5

{ MODULAR SYNTHESIZER }

A Wavetable/Sampler Modular Synthesizer, with amplitude, ring and frequency modulation. Direct sample level digital signal processing (DSP).

https://github.com/jamesnapierstuart/19329-Modular-Synthesizer

1.5 Modular Synthesizer

Overview

Three main aims were set; first to take a new approach to wavetable creation that differs from a drawing input or straight forward pre-calculated table values. Second to implement sample-based wavetable synthesis and third to create a modular system that allows these different synthesis techniques to interact with each other i.e. an environment that allows for ring, amplitude and frequency modulation implementation. Working on the sample level to carry out Digital Signal Processing (DSP).

Execution of Concept

I implemented wavetable synthesis through look-up table waveform calculations and sample based wavetable creations, within a modular environment. Inspired by such programs as MAX/MSP and Hadron I aimed to carry out thee types of wavetable synthesis with modulation synthesis, achieved by creating a graphical environment to patch outputs of modules to inputs of others all within SuperCollider.

Critical to this project was to carry out a direct sample level digital signal processing (DSP) routine.

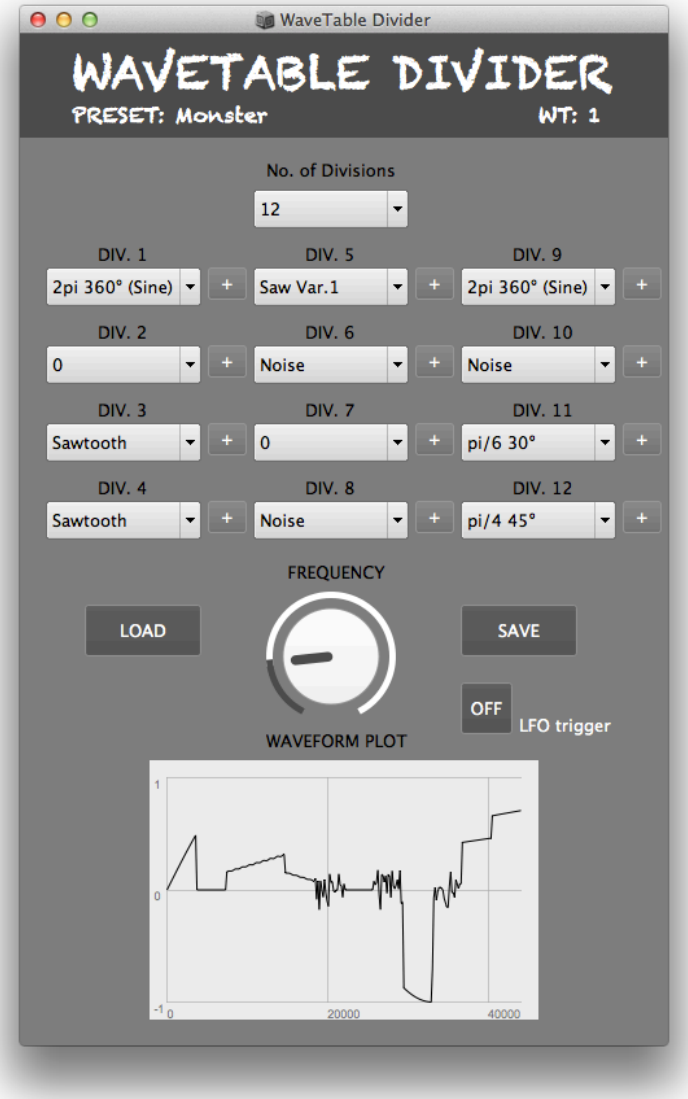
Meaning manipulation of amplitude and phase values without the aid of higher level synthesis manipulation tools such as SuperCollider Ugens. To realise this, wavetable synthesis was chosen, specifically wavetable look-up.

"One method is to let the machine calculate only one cycle of the wave and store samples in its memory" (Miranda 2002) . For my wavetable design I followed Miranda's proposed method, and imposed a law on every equation that the calculated frequency will be 1Hz or one cycle (See MathsEquations class source code). This means when their playback rate is increased, the value of playback rate is equal to cycles per second. E.g. 440 times the rate = 440Hz/440 cycles per second of the waveform period.

1.5 Modular Synthesizer

Wavetable Synthesis

A problem I found when generating wavetables, was that I could only make a waveform of one particular equation at a time. Therefore I proposed an algorithm which allows the user to define the mathematics for sections of the waveform. These ‘divisions’ allow us to split the waveform up into proportionate equal sections, e.g. if divisions = 2: split waveform into halves, = 3: split waveform into thirds. Allowing for a different type of wave per section. It must be noted that amplitude and phase values for every waveform are maintained regardless of number of divisions. I implemented the ability to load and save presets and included an LFO trigger which allows the frequency knob scale to change from 10Hz - 8000Hz to 0Hz - 20Hz.



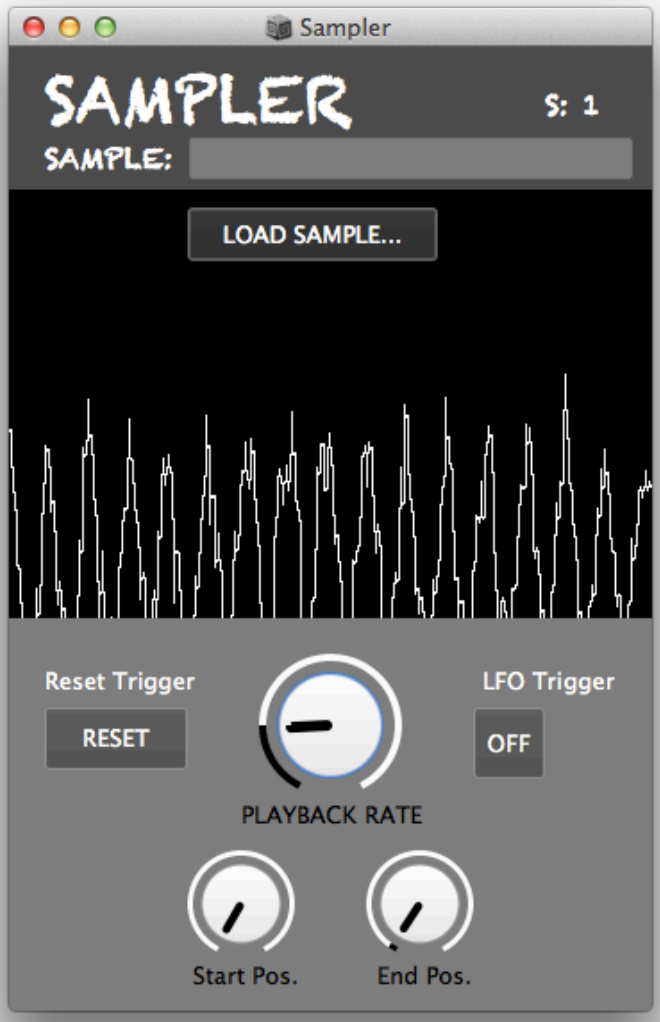
Example of 12 Divisions with the Wavetable Divider

1.5 Modular Synthesizer

Sample-Based Synthesis

The Sampler synth module gives users the ability to create periodic waveforms/wavetables in real-time by adjusting the start and end position of a sound file and looping the section continuously. Thus playback rate changes the frequency of the newly shaped waveform.

A key difference between the frequency rates of the Sampler Synth and the Wavetable divider is the frequency scale mapping ranges from -200 to +200 for the sampler as opposed to +10 - +8000 for the wavetable divider. When the rate is in minus figures the playback pointer instead of skipping forward in the index array, skips backwards. Therefore resulting in reversing the waveform.

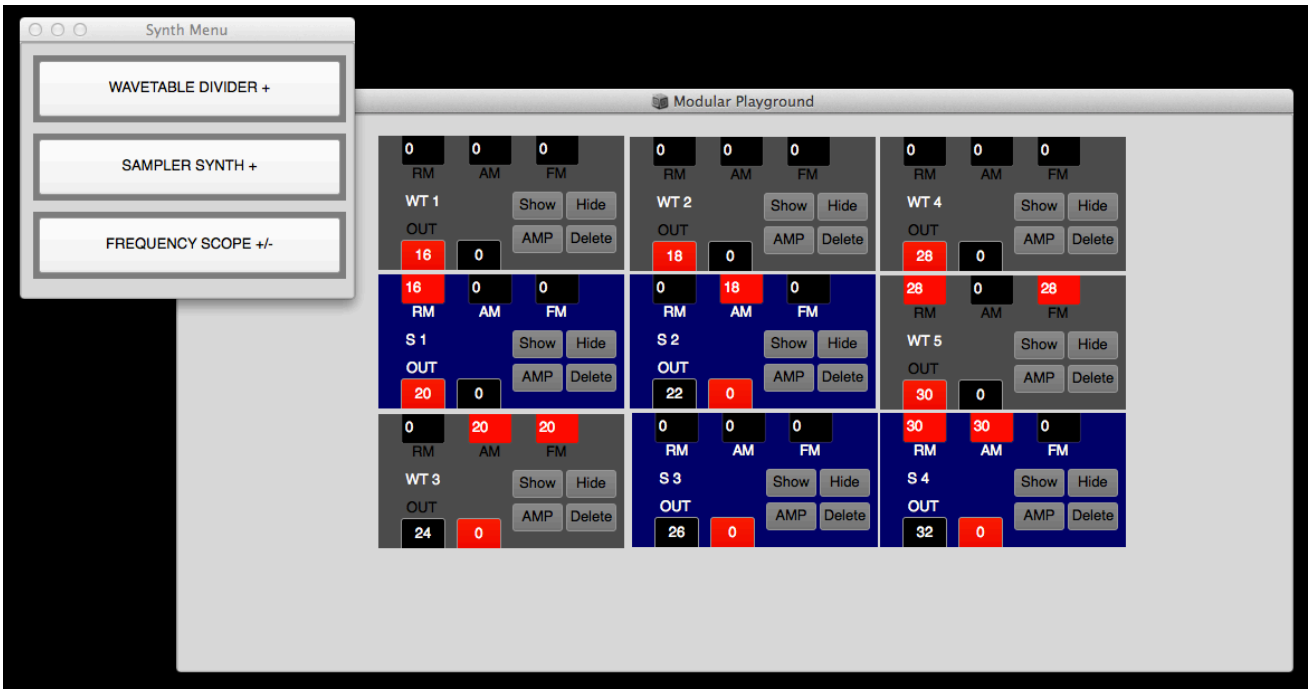


Example of Sampler Synth Module with reverse playback rate

1.5 Modular Synthesizer

Modular Synthesizer Environment

The left window allows the user to add a wavetable divider oscillator synths or a sampler synth (and also display the frequency scope). The right window displays the modular environment that allows the user to connect different synth modules together by typing in the number of their output into one of the three inputs. From left to right the inputs control ring modulation (RM), amplitude modulation (AM) and frequency modulation (FM). Synth modules can be shown or hidden with the show/hide buttons, their amplitudes changed and can be deleted too. In addition the graphic representations of the modules can be moved around where ever on the window and the window will adapt with scrolling functions when the amount of modules begin to grow beyond the size of the screen.



Modular Synthesizer GUI

1.5 Modular Synthesizer

Evaluations & Conclusions

The final result of the implemented synthesis methods above has resulted in a well formatted GUI design to easily interact with modules, enabling the user to connect inputs and outputs with simple text input, which proofed a suitable alternative than the line connection drawings found in existing modular software such as Max/MSP.

A strength of the architectural foundation code, is that it allows for additional modules to be added very simply for further development. Additions such as envelopes, filters and effects units can be written outside the core ModularSynthesizer class and easily implemented inside it just as the Wavetable Divider and Sampler synth have been, using the defined bus incrementation system to take care of new modules.

The overall sonic output possible from this system is vast and diverse. Having the ability of synthesizing sound, sampling sound and the combination of the two with the functionality of modular routing allows for deep sonic possibilities. Harmonic and rhythmic content is achievable as demonstrated in the provided sound examples (see below) and with the further improvements discussed above this wavetable modular synthesizer has potential to become a very powerful sample level synthesis tool.

Future Projects

An interesting path for future development, would be to see the Wavetable Divider implement loading of sound samples (e.g. taking the amplitude and phase values of one period of a sample waveform) and using it as a new maths function. Thus allowing for a combination of mathematical equations and sampled sound wavetables to define the divisions of the 1Hz waveform as seen in the current Wavetable Divider. Following the subject of divisions, it should be made possible for the division locations to be moved into arbitrary positions by the user, instead of having divisions set by proportionate ratios of halves, thirds etc.

P.T.O.

1.6

{ FRAMINGHAM HEART STUDY SONIFICATION }

Algorithmic composition generated with data from the Framingham heart study of human features to control sound synthesis, effect units, pitch and rhythmic phrases.

https://github.com/jamesnapierstuart/19329-Algorithmic-Sonification-Composition

1.6 Framingham Heart Study Sonification

Overview

This original algorithmic composition is a structured sonification of humans and their features based upon the specifications within the 'Hard Coronary Heart Disease (10- year Risk)' from the Framingham Heart Study, which calculates the risk of a human being (male or female) experiencing a heart attack in the next ten years. The premise of the piece was to control created sound synthesis and effects units according to a human's features e.g. a smoker emits a minor sequence of chords or notes whereas a non-smoker emits a major sequence. Thus enabling the listener to recognize the information associated with these sounds and work out the features of a particular instance of a human such as age, gender, smoker status, total cholesterol level etc.

Execution of Concept

The most effective way to create new instances of a 'human' was to use classes with getter and setter methods to allow the user to alter features e.g. increase age, change smoker status etc. The ability to be able to keep the human class flexible was paramount to the success of the composition as this inclusion of methods allows for information to be altered over a periods of time (and enables the Elektrokardiogram sound to signal death of a huma). The sonification strategy employed was 'Discrete point data representation' (6) as we are mapping several dimensions of data to parameters of sound events.

The sound design draws parallels from Bollora's inst1 (fig1.1) whereby all sounds were synthesized as opposed to sample based. Sound synthesis and effects were chosen according to the initial idea of basing the piece within the EDM genre. Therefore FM synths and drum machine emulations are used, with deep flexibility in order to successfully represent different human instances.

In order to illustrate how each human is represented, a key is necessary as a reference to explain which features are conveyed by which sounds. See my interpretation of the Framingham's data tables as a musical key (fig 2.1). The user's initial listening period will begin by studying the key and listening simultaneously, in order to process and store the sonification of specific data in their memory. After this period one can run the piece without having to look at the key and realize from the sounds heard, all the features of the human instance(s).

1.6 Framingham Heart Study Sonification

Structure of Composition

To organize the sonified humans, and to demonstrate their features and risk of heart attack, I chose to keep to a simple, effective structure:

A.
The birth of humans. Firstly half the total amount is heard, consecutively one human after the other, to slowly build their patterns up over time. This allows the listener time to clearly distinguish between different instances of humans and denote their individual qualities based upon the provided musical key. Next the second batch of humans follows the same pattern.

B.
Heart Risk Calculated. Using my calculation based upon the Framingham's heart study point system, the tempo of the human sounds are determined by their likelihood of having a heart attack in the next 10 years. The higher the risk the faster the tempo controlling their patterns.

A1.
Increase age and randomize every feature (except gender) for each human, then run the intro and calculation parts again. This will change the original sounds for each human and those who become too old (80>) die off. Once either all humans have died off or the last one's pattern ends the piece finishes.

1.6 Framingham Heart Study Sonification

Evaluations & Conclusions

My initial thesis behind utilizing the heart risk calculations of males and females was to keep them separate and fuse their sound together using match making algorithms such as the stable marriage problem but it proved a harder notion to fulfill than expected. Problems arose with preference and the ability for the stable marriage algorithm to work correctly with homosexual relationships. I feel very satisfied that my algorithm doesn't have sexual bias and allows for any sound male or female to occur simultaneously.

Keeping to the stereotypical qualities of the EDM genre was hard to fully achieve with every iteration. The instruments heard may sometimes individually conform to this genre but within a sonification context where I allowed cross- rhythms and dysfunctional harmony to occur, containing the piece within the genres bounds proved unsuccessful every time.

Overall I am proud with the success of this algorithmic composition. With the vast amount of rhythmic, melodic and chord progression patterns I feel the piece performs to a very high standard and although may not always sound like a consistent piece of EDM music, achieves the role of portraying the features of humans very well.

P.T.O.

1.7

{ FLEXI SYNTH }

Flexi Synth is a synthesizer, which uses a combination of additive, subtractive and frequency modulation synthesis. Includes a routing matrix to connect LFO's to filters and other destinations and uses an ADSR envelope.

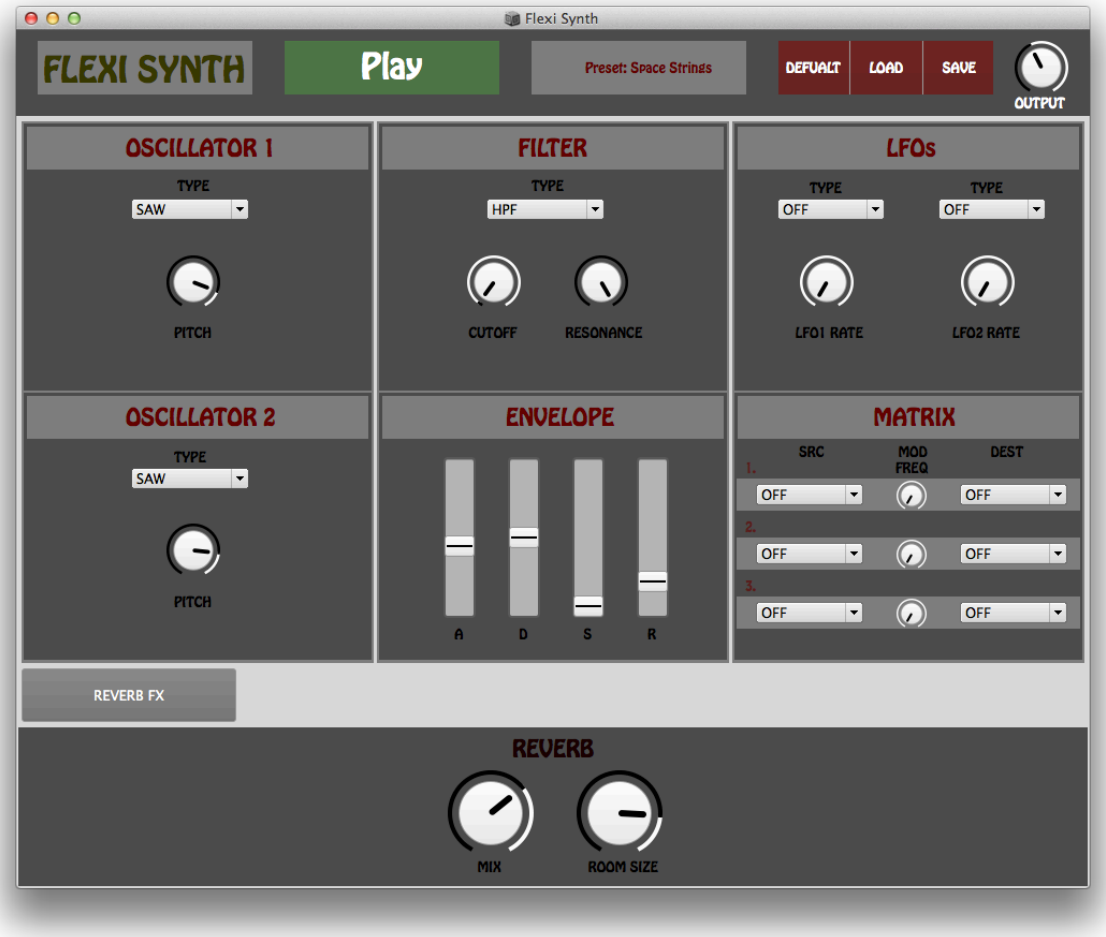
https://github.com/jamesnapierstuart/19329-Flexi-Synth

1.7 Flexi Synth

Overview

‘Flexi Synth’ is a synthesizer, which uses a combination of additive, subtractive and frequency modulation synthesis. Modern day software synthesizers such as Waves ‘Element’ inspired its look and popular analogue synthesizers like Yamaha’s DX7 inspired its functionality.

It Includes two oscillators, a filter, two LFOs, an envelope and a matrix routing control allowing the user to choose a source such as an LFO and plug it into the destination such as pitch of the oscillator. It also features the ability to load and save presets the user creates.



Flexi Synth GUI

1.7 Flexi Synth

Execution of Concept

The main goal of the GUI was to be as clear and simple to use as possible. To achieve this I made sure the main elements such as the oscillators and play/stop button were the most spaced and placed from left to right. All items colour coded and fonts universally mapped.

The main focus I tried to showcase through the GUI was to gradually move progressively more advanced in terms of parameter control when we move across the screen. This allows beginners to easily get involved with creating additive synthesis quickly and gives further advanced users deep control over their sound. It was of prime importance to make this synth as ergonomically efficient as possible. The inclusion of the preset buttons means it's quickly for users to return to factory settings and the ability to write their creations to disk.

Evaluations & Conclusions

The creation of Flexi Synth proved that the proposed aims I set out to achieve could become a reality and I feel that the execution of this synthesizer throughout the creation of its synthesis to its realization in the GUI was an overall success. In terms of effects, there is definitely room for expansion to using delays, chorus by using the space I have laid out already in the GUI.

The matrix modulation proved to be the toughest challenge of all the build. As it takes into account a vast number of conditions that although works within the synth, there are certain possibilities it hasn't accounted for which can throw it off. I hope to improve on this in the future with a deeper controllable synth whereby every parameter available to the user can be controlled and modulated by any other. Pointing in the direction of how a software application such as Native Instruments Massive makes use of drag and drop capabilities to put this theory into practice.

2. SOUND ENGINEERING PROJECTS

2.1 Bloody Knees

Lead sound engineer for punk band Bloody Knees. Recording, mixing & mastering engineer for multiple records.

2.2 Tilapia

Recording, mixing and mastering engineer for indie pop band Tilapia.

2.3 Alex KP

Lead sound engineer for folk artist Alex KP. Recording, mixing & mastering engineer for multiple records.

2.4 BBC The One Show, Warwick Davis Voice Over

Assistant engineer for Zigzag Music Productions. Voice over recording of Warwick Davis for the BBC's 'The One Show'.

2.5 Choral Location Recording

Assistant sound engineer for Zigzag Music Productions. Choral location recording at Gamlingay Baptist church.

2.6 Birdskulls

Recording engineer for Brighton based punk band Birdskulls. Tracking their entire Trickle album.

2.1

BLOODY KNEES

Lead sound engineer for punk band Bloody Knees. Recording, mixing & mastering engineer for multiple records. Releases on Dog Knights Productions, Heart Throb Records, Vinyl Junkie and broadcasts from BBC Radio1.

https://soundcloud.com/bloodyknees

2.1 Bloody Knees

Bloody Knees Latest EP TBA (2015)

Lead sound engineer for their upcoming EP later this year. I acted as the recording, mixing & mastering engineer for the tracks. Providing a comfortable work environment, hospitality, extra equipment and anything else the band required. Meeting deadlines set by the band, their management and their label and working closely with the band in the final stages of mixing to iterate revisions until the best possible sound was achieved.

Release TBA

Bloody Knees Stitches EP (2014)

Lead sound engineer for four of the tracks of the EP. The first two recorded & mixed in London at Lightship 95 studios were later re-mastered by me to fit suitably with the four other tracks. The four I worked on meant acting as the recording, mixing and mastering engineer.

Released by indie label
Dog Knights Productions.
http://www.dogknightsproductions.com/DK/home.html

Press from The NME, Stereogum, ThisFakeDIY, The Guardian and many more.

Broadcasts on BBC Radio1 by Huw Stephens.

Bloody Knees SoundCloud:
https://soundcloud.com/bloodyknees

Bloody Knees Page:
https://www.facebook.com/bloodyknees

Stitches EP Vinyl (Dog Knights Productions Release)

Stitches NME Review 9/10

2.1 Bloody Knees

Bloody Knees ‘Bloody Knees’ EP (2013)

Recording, mixing & mastering engineer on all sessions. Remixed and remastered three tracks from the original LP and recorded three new tracks to accompany them. Using reference tracks and artists that the band liked the sound of to use whilst mixing the EP.

Released by indie label
Heart Throb Records on limited edition tape.
http://heartthrobrecords.bandcamp.com/album/bloodyknees-ep

Released by Japanese label
Vinyl Junkie (JP only).
http://vinyl-junkie.com/label/bloodyknees/

Press from The NME, Noisey (VICE), ThisFakeDIY

Broadcasts on BBC Radio1 by Jen Long & Ally McCrae

Bloody Knees EP (Heart Throb Records Release)

Bloody Knees EP (Vinyl Junkie JP Release)

Bloody Knees ‘Bloody Knees’ LP (2012)

Recording, mixing & mastering engineer for their first entire album. First time recording the band, we split the sessions into two tracking six tracks in the first and three tracks in the second. Experimenting with different recording & mixing techniques to try and capture and mould the sound of the band.

Self released

Broadcasts on BBC Radio1 Introducing

2.2

TILAPIA

Recording, mixing and mastering engineer for indie pop band Tilapia. Including location recording engineer for their latest record.

https://soundcloud.com/tilapiaband

2.2 Tilapia

Tilapia ‘Fluffy Tails’ EP (2015)

Location recording engineer for all sessions. Tracking took place out of the studio and at the band’s chosen location. Working with the band and individual members on their sound and playing to make sure everything was to the best of their ability. Mixing was arranged with a third party so this meant editing and consolidating all audio into stems ready for the mixing engineer to work on with no transition problems.

Release TBA

Tilapia ‘School’ EP (2014)

Mixing and mastering engineer for the EP. Since I wasn’t the recording engineer it meant receiving files from the London studio engineer. For many of the tracks the band asked me to add in extra synth and midi parts and any other sounds/instrumentation I thought appropriate that would lift the tracks.

Released by indie label Disorder Records.

Press from Slate The Disco, Snap South London & more.

Tilapia SoundCloud:
https://soundcloud.com/tilapiaband

School EP

Tilapia ‘Rawshank’ EP (2013)

Mixing and mastering engineer for the EP. Tracking took place with another engineer in London, so this meant I received the stems of the tracks to mix. Communication was vital with the band to achieve the sound they wanted and mix revisions were sent back and forth for feedback and improvements to bring the tracks exactly to where the band wanted.

Self released

Rawshank EP

P.T.O.

2.3

ALEX KP

Lead sound engineer for folk artist Alex KP.
Recording, mixing & mastering engineer for multiple records.

https://soundcloud.com/alexkpmusic

2.3 Alex KP

Alex KP ‘Lady Greyling’ EP (2013)

Recording, mixing and mastering engineer for all sessions for the folk sing & songwriter Alex KP. The instrumentation included voice, acoustic guitar & fiddle. The guitar and voice were tracked simultaneously to accommodate the artist’s performance style.

Self released

Broadcasts on BBC Radio Cambridgeshire

Alex KP Soundcloud:
https://soundcloud.com/alexkpmusic

Alex KP Page:
https://www.facebook.com/akpmusic

Alex KP ‘Buildings’ EP (2012)

Recording, mixing and mastering engineer for the whole EP. This body of work was subject to a very immediate deadline that I had to meet. Therefore in addition to engineering, burning and printing CDs had to be carried out to be ready to sell at the Cambridge Folk Festival a few days later.

Self released

Broadcasts on BBC Radio Cambridgeshire

P.T.O.

2.4

BBC THE ONE SHOW

Assistant engineer for Zigzag Music Productions.
Voice over recording of Warwick Davis for the BBC's
'The One Show'.

2.4 BBC The One Show

Warwick Davis Voice Over, The One Show (2015)

Assistant sound engineer for Zigzag Music Productions for Warwick Davis' voice over for the BBC's The One Show. My role was to keep track of vocal timing. The directors of the show had sent over the script for Warwick to read along with cues and timings for each segment. I had to time each vocal delivery and feedback whether Warwick need to shorten or lengthen phrases, which would later match the visual scenes later on. Also had to take care of hospitality and reading the script over in preparation prior to the recording.

BBC The One Show episode:
http://www.bbc.co.uk/programmes/b052fvdx

Warwick Davis in the vocal booth

P.T.O.

2.5

CHORAL LOCATION RECORDING

Assistant sound engineer for Zigzag Music Productions. Choral location recording at Gamlingay Baptist church.

2.5 Choral Location Recording

Choral Location Recording (2015)

Assistant sound engineer for Zigzag Music Productions for location recording at a Baptist church in Gamlingay. Recording a choral ensemble with keyboard accompaniment. This included loading in gear, setting up the interface and monitoring equipment in a separate room to the choir. Setting up and positioning microphones to best capture the performance. And following along with the score of each piece to offer feedback to the conductor between takes.

The monitoring station we set up in the churches kitchen

The main space where we captured the choir

P.T.O.

2.6

BIRDSKULLS

Lead sound engineer for punk band Birdskulls. Recording engineer for their Trickle LP record. To be released on Dog Knights Productions.

https://soundcloud.com/birdskulls

2.6 Birdskulls

Birdskulls ‘Trickle’ LP (2015)

Recording engineer for the entire album. Tracking all instruments and initial editing for each song. The tracks were then sent off to be mixed by Bob Cooper and therefore all files were edited, consolidated and exported into orderly stems to be accessible by any DAW.

To be released by indie label
Dog Knights Productions.
http://www.dogknightsproductions.com/DK/home.html

Press from The NME, Stereogum, ThisFakeDIY, The Guardian and many more.

Press from The NME, Noisey (VICE), Punktastic.

Trickle LP (Dog Knights Productions Release)

P.T.O.

3. Session Work

3.1 Red Bull Studios London, Fat Lion Hi-Fi

Session bassist for Dub artists/producers Fat Lion Hi-Fi. On ‘Pay your Way’ track for the B.Traits Red Bull Studios London Mixtape.

3.2 Price Production Studios (Ryan Price)

Session guitarist for RnB artist/producer Ryan Price. On ‘By the End feat. Afiya’.

3.3 Imperial

Session guitarist for Hip hop artist/producer Imperial.
On ‘Pencils not Pistols’ EP.

3.1

RED BULL STUDIOS LONDON FAT LION HI-FI

Session bassist for Dub artists/producers Fat Lion Hi-Fi. On ‘Pay your Way’ track for the B.Traits Red Bull Studios London Mixtape.

https://soundcloud.com/fatlionhifi

3.1 Red Bull Studios London, Fat Lion Hi-Fi

Fat Lion Hi-Fi ‘Pay Your Way’ (2015)

Session bassist for dub artists and producers Fat Lion Hi-Fi (formally Flatline). The track was chosen by artist and BBC Radio 1 DJ B.Traits to be featured on the Red Bull Studios Mixtape Vol. 2 2015. I recorded the bass separate from the session so this meant receiving the rough mix and tracking takes with the facilities I personally had. Then making sure all audio stems were aligned and consolidated for the producers to immediately use and mix.

Featured on the B.Traits Red Bull Studios London Mixtape:
http://mixtape.redbullstudios.com/

Fat Lion Hi-Fi ‘Pay Your Way’:
https://soundcloud.com/fatlionhifi/pay-your-way

Fat Lion Hi-Fi website:
http://fatlion.co.uk/

Imperial Website:
http://www.imperialbeats.co.uk/

P.T.O.

3.2

PRICE PRODUCTION STUDIOS

Session guitarist for RnB artist/producer Ryan Price.
On ‘By the End feat. Afiya’.

https://soundcloud.com/price-production-studios

3.2 Price Production Studios

Ryan Price ‘By the End feat. Afiya’ (2012)

Session guitarist for RnB artist/producer Ryan Price. Recording lead guitar solos and chords for different sections of the track ‘By The End’. Working in direct company with the producer improvising phrases together to find which ideas worked best for the track.

Self released

Ryan Price ‘By the End feat. Afiya’:
https://soundcloud.com/price-production-studios/by-the-end-by-ryan-price-ft-afiya-price-production-studios

Price Production Studios Soundcloud:
https://soundcloud.com/price-production-studios

Price Production Studios

P.T.O.

3.3

IMPERIAL

Session guitarist for Hip hop artist/producer Imperial.
On ‘Pencils not Pistols’ EP.

https://soundcloud.com/imperialbeats

3.3 Imperial

Imperial & K.I.N.E.T.I.K ‘Pencils Not Pistols EP’ (2012)

Session guitarist for Hip Hop artist Imperial. Wah Wah scratch guitar on ‘Pencils not Pistols EP’ on the ‘Put your pencils in the air’ track. The work involved session guitar work from a different location, so making sure stems of takes I recorded were all sent back consolidated and aligned with the track he provided.

Released on Illect Records:
http://illect.bandcamp.com/album/pencils-not-pistols

Imperial ‘Put Your Pencils In the Air’:
https://soundcloud.com/imperialbeats/imperial-k-i-n-e-t-i-k-put

Imperial Soundcloud:
https://soundcloud.com/imperialbeats

Imperial Website:
http://www.imperialbeats.co.uk/

Pencils Not Pistols EP

J A M I E S T U A R T

AUDIO PROGRAMMER & SOUND ENGINEER

P O R T F O L I O

2011 - 2015