An Interactive Algorithmic Music System for EDM

Feature Article

Richard Savery

Georgia Institute of Technology (US)

Abstract

Artiin is an interactive computer music system for live performance and composition of EDM. Artiin is able to function as an autonomous musical creator with no human-input, or can generate one or more streams in response to human input: for example creating a ride cymbal pattern influenced by an incoming bass pattern. The system shows how developments in interactive music can be repurposed for a system designed to meet a DJs workflow. This article explores the design of Artiin as well as musical concepts that underpin this work and other systems that influenced its development. Artiin was designed by the author in Max for Live, with multiple custom externals written in Java. While the system is currently used by multiple musicians, it is not publicly available.

Keywords: algorithmic composition; artificial creativity; generative music; computer improviser

Richard Savery is a music technologist, composer and performer focusing on developing software and robots that creatively make music, whether through autonomous musical agents or simple compositional aids that can work with a human.

Introduction

Multiple projects have engaged with the automatic generation of electronic dance music (Dahlstedt 2001; Ulyate and Bianciardi 2001; Collins and McLean 2014). Interactive music systems for concert performance are an established research area in computer music (Drummond 2009). Artiin is an interactive music system created by the author in Los Angeles, USA between 2015 and 2017, with ongoing developments. Artiin was designed for EDM and generates music using a combination of mathematical models, data-driven systems and transformation of input material. Generated material can interact with existing musical material or live input from a human performer. Artiin was designed in Max for Live with externals written in Java. Artiin was created to explore how continuing developments in music generation, interactive music systems and music information retrieval can be applied to an EDM system. This system was designed to be used by DJs and critically to be useful to DJs beyond the author. The author conducted user studies primarily aimed at guiding the development process and engaging potential users.

Artiin can be conceptualized as three separate musical agents performing on drums, chordal synth and bass. Artiin was designed to be versatile for many different uses: it can range from a complete piece with no human-input to generating a supporting chordal accompaniment around a vocal line. Flexibility of use and transparency are key design parameters and Artiin aims to be capable of slotting into the performer's live rig. It can be used for composing new material or generating ideas, and more commonly it is used in live performance to add new material. This article outlines some past systems that have influenced Artiin. It also covers the global concepts behind Artiin before describing the input analysis and musical generation processes. The musical decision-making process and user interface are then described, followed by a user study and potential future work.

Background

Computer composers and improvisers can be used to explore new processes unavailable to traditional human composers. Pearce, Meredith and Wiggins (2002) describe four motivations to devise computer programs to create music: to make new music, as tools for other composers, as a theory of musical style, or to model cognitive processes of music creation. Other authors note that algorithmic music can additionally allow for new musical outcomes (Miranda and Wanderley 2006: 224). These outcomes can occur in the music created by the system and in the human reaction to this input. Artiin aims to create music that wouldn't be created by a human performer alone, using unique methods only available to a computer. In the current work, we explore new musical territory in studio and live electronic dance music, empowered by computer generation.

This work also investigates broader questions about how EDM is generated; to attempt to replicate musical creation many questions must be answered about how musical decisions are made. How can a computer listen to music, how can it create music and how can it decide what to do? Exploring these questions necessarily involves developing a better understanding of how humans compose.

Interactive Music Systems

The present work primarily focuses on developing an interactive computer program that is capable of creating music in real-time with a human. This research builds on a range of interactive computer music systems. Its general design principles are inspired by Voyager (Lewis 2000) and the interactive music systems of Robert Rowe (Rowe 2001, 1993). Interactivity requires that the computer not only reacts to human input but also adds new musical elements. Considering computer systems as interactive, as opposed to only reactive, Dobrian states: "inter- in the word 'interactivity' implies mutual influence between agents that are also in some way autonomous decision makers" (Dobrian 2004: 1). Ideally this software aims to act as an independent interactive musical collaborator, capable of developing its own unique musical concepts, while listening and engaging with a human musician.

Boden defines three ways in which computational creativity can occur: combinational, exploratory and transformational (Boden 2000). Combinational creativity is when familiar ideas are combined in new ways, such as re-arranging audio samples into a new piece. Exploratory involves searching within the bounds of a paradigm and finding new forms of creativity, such David Cope's generation of new music in the style of western classical composers (Cope 1989). Cope generates music through a data-driven approach recombining existing classical works into new pieces. Transformational refers to breaking outside the confines of an existing paradigm and is the hardest to achieve.

Global Concepts and Musical Assumptions

Material is generated through transformative and generative methods. Artiin is built as a Live Algorithm (Blackwell, Bown and Young 2012), defined as an improviser that interacts with musicians. Artiin works primarily within combinational creativity combining existing techniques into a new system. Artiin's approach to algorithmic composition is to view musical creation as an extended series of scaled and quantifiable decisions. These decisions can be made in two categories, firstly as an on (1) or an off (0). Secondly, decisions can be between 0 and 1, such as 0.5. Through a combination of many instances of these decisions it is possible to create a diverse decision-making system. In this way, many cases of simplicity can lead to a deep level of complexity.

These factors then need to be combined with a way to interact with incoming ideas. For this I impose a model of interaction whereby ideas can be ignored, copied or opposite parameters applied. For volume, performing the opposite would mean playing soft when the input is loud. An example of this structure is presented in Figure 1. There are always four decisions made: volume, density (how many notes occur), range and intervals. Artiin decides on levels of interaction for each component individually. It may choose to copy volume levels but ignore density levels, allowing for a much broader range of results.

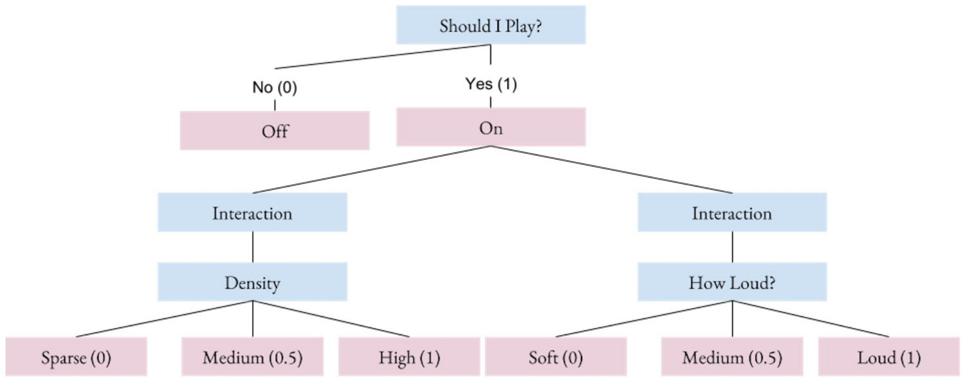


Figure 1. Artiin Decision Making Process

Figure 1 portrays a simplified version of Artiin's decision-making process. Considering Artiin within Roads' (2004) scales of time, these decisions do not reach the micro level (milliseconds) and generally focus on the sound object level. It is my hope that the ideas are reduced to a point where choices are inherently decided in the timbre qualities of virtual instruments. It is possible for the computer to make all decisions at each level, however with transparency and a range of uses a key design goal, the user is able to influence certain parts of the decision-making process.

Input

Artiin can receive input via MIDI or through an audio signal. The user sets which musical information to be processed by Artiin and labels it chordal, bass or drums. Audio is converted into a variation of MIDI. Standard MIDI includes a "note on" message with velocity value for the start of a note and "note off" for the end of the note. With Artiin, audio also includes velocity values during the note to allow the system to register volume changes throughout a note. Once all information is converted to MIDI, it is stored in a database which grows throughout the performance. Artiin also retains information from past performances by the performer.

Audio Analysis

For audio analysis, Artiin uses a custom plugin made in Java (written in collaboration with Jakob Nagel). While there are existing plugins that could complete this task, a custom external was made to prevent Artiin from relying on third party externals. Audio first undergoes preprocessing by removing all signal beneath a volume threshold, which is set before each performance dependent on background noise. This signal is then processed to extract note onsets, pitch and dynamics.

To compute the parameters, the input audio signal is divided into blocks of 1024 samples, each with an overlap of 512 samples. The parameters of notes (pitch rounded to a midi value and onset/offsets), tuning (cents) and dynamics are then extracted for each block. Notes are considered an individual MIDI pitch, while the tuning is the deviation from this pitch. Tuning deviation is measured when a note is bent over a quarter tone.

Pitch tracking is implemented using the auto-correlation function (ACF). The ACF is computed and normalized by dividing through the first value of the ACF. Thus, the maximum value of the normalized ACF equals 1. For each block, the candidate sample indexes are determined to be the sample indexes from the index on, where the autocorrelation at the first instance undercuts a certain threshold of T (Wang and Brown 2006). We determined T to be 0.35 as first few sample indexes generally have very high correlations (in particular, the first sample index will have a correlation of 1, which is the maximum), but would correspond to frequencies out of our target range. For each block b, the index i_b with the highest correlation is extracted out of the candidate indexes. The frequency for each block b is computed by f_s / i_b, where f_s denotes the sampling frequency of the given audio file in Hertz.

$$\text{rms(b)} = \sqrt{\frac{1}{1024}\sum_{n=1}^{1024} b(n)^2}$$

Equation 1

The dynamics of each audio block b are computed (see Eq. 1) where b(n) denotes the value of the n*th* sample of block b and 1024 is the number of samples in block b. For onset and offset detection, a novelty function is used using spectral flux (Bello et al. 2005). Onsets are determined to occur at the timestamp of each block, for which the novelty function surpasses the onset threshold Ton = 0.3. Analogously, offsets are determined to occur at the timestamp of each block, for which the novelty function undercuts the offset threshold. In the case of multiple onsets detected in a row (without offsets detected in between), the missing offsets are added at the same timestamps where the corresponding onsets occur. In the case that multiple offsets are detected in a row (without onsets in between), all of those offsets are discarded, except for the first one.

After the performance parameters (pitch, dynamics, onset and offset for micro-timings) have been computed for each audio block, the actual performance data of the overall audio is aggregated. At this point, the audio is now a text file, containing sampled note values that have a specific starting time and duration, as well as a pitch and volume that may change over time.

Audio Descriptors

With the notes (MIDI value), volumes and tuning extracted, Artiin then creates higher-level descriptors. The high-level descriptors are intervals, range, volume and density. These descriptors were chosen primarily as they are easily understood by the users of the system. Additionally they can be clearly mapped from input to output when appropriate. Signals are first separated into musical phrases, which are determined by a rest length after a sequence of notes. The length required to set the end of a phrase is defined by an inverse relation to the phrase length. That is, at the beginning of the phrase a longer rest is required (3-beats) while the longer the phrase persists, the shorter the rest time required to signal a new phrase. While this is a far from perfect system, it has proven relatively reliable in testing by the author. Through evaluation using the Meertens Tune Collections (Van Kranenburg 2016), the phrase detection system showed an accuracy of 60%, however failure was usually due to doubling the length of a phrase. For the purpose of Artiin, the required element of phrase detection is storing musical motifs that have a logical start and finish. Since phrase detection is a subjective music information retrieval task (Livingstone et al. 2009), a fully accurate system for Artiin is unrealistic.

Intervals is an average of an array containing the distance between each set of notes. Intervals are handled in this way to describe an input's variation between stepwise scalar material and wider movement. Additionally, averaging avoids storing redundant information that is already available to Artiin in the MIDI file. Range is a single value showing the distance between the highest and lowest note in the set time period. Volume is the normalized dynamic level. Density is the number of notes played per measure and is capped at 32. Each of these descriptors are analyzed over the different time lengths of 1 second, 5 seconds and 20 seconds for real-time. These descriptors each have preset levels to range from 0 to 1 (see fig. 2). These descriptors are added together to give a complexity level out of four to each section of material (see fig. 3). Density is weighted three times higher than the other three parameters, due to its higher influence on perceived complexity.

	0.	0.5	1.
Intervals	Unison	Tritone	An octave and above
Range	Unison	Tritone	An octave and above
Volume	-60db	-20db	-5db or above
-Density	1	8	32

Figure 2. High-level descriptors

Intervals	0.5
Range	1
Volume	0.75
Density	0.25 (0.75 weighted)
Complexity	2.5

Figure 3. High-level descriptors sample results

The same metrics are generated for each of the three virtual instrument's outputs. At any stage, any instrument can draw on the volume, or the density and complexity of the other real and virtual instruments in real-time. This allows for interaction between the independent components of Artiin itself.

Storing Music

Artiin stores the audio input and audio descriptors separately. The audio input file records the incoming MIDI or the analyzed audio signal. New ideas are added at the end of each phrase. For the audio signal, stored ideas are an array of numbers containing time in milliseconds, pitch, velocity and tuning distance from MIDI pitch. For incoming MIDI, the velocity remains the same for each sample and tuning is normalized when the MIDI contains no tuning information. The second file contains the high-level descriptors with the start time of each phrase. In this way, Artiin can search for the start point of an idea with a set descriptor or complexity level.

Artiin also maintains a database of all past performances of its own output and the input it has received. From its own output, it only stores ideas that have been sonically realized, not all ideas that have been created. Input monophonic ideas are added to the database that is used for Markov model generation. Input drum patterns are added to the drum generator database. Storing these ideas in this manner allows for a system that can change over time in ways dictated by its users, gradually adding the user's musical language into its own. Users are also given the option to retain what is played within each performance.

Musical Creation

Artiin creates music through three different methods, utilizing transformative and generative processes (Rowe 1993). Generative processes include data-driven methods and mathematical generation. User-created ideas are altered both on real-time input and by changing ideas from the database described in earlier sections. These methods of creation are then chosen from by Artiin to create the final output.

Data-Driven Melody Generation: Markov Models

One of Artiin's key forms of creation is through data-driven systems. This system incorporates a knowledge base drawn from 200 EDM files in MIDI format. This data was compiled from two online sources in 2018: Freaky Loops and WA Productions. Unfortunately, due to licensing restrictions preventing distribution, this dataset cannot be shared. Parts that didn't fit into one of these categories were discarded, such as high-pitched rhythmic ostinatos. Categorization of MIDI files was done through a very simple automatic process separate to Artiin, written in Python. This process separates the MIDI file into distinct channels (consistent in the data sets) and then checks for range, and polyphony. To allow for variation in keys between each piece, the MIDI files were transposed into all twelve keys and therefore represented twelve times in the dataset, without a distinction between major and minor.

A Markov chain is a stochastic process that bases the probability of a new idea on what has occurred before. Markov chains have been used extensively in algorithmic composition

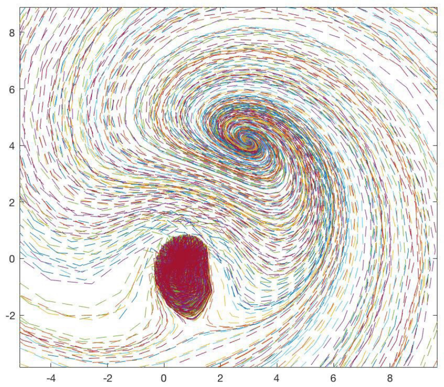
(Ames 1989; Roads 1996). First order Markov chains use only the current state to decide on the upcoming state, while second order chains use the current one and the most recent state. Artiin uses a custom external (coded in Java) for Markov models of rhythmic and harmonic ideas for the bass and chordal synth. It uses a minimum of fourth order chains, trained from the MIDI file dataset.

Data-Driven Chord Progression Generation: Markov Model

Artiin can automatically create a chord progression ranging from four to eight bars long. These progressions all follow the same linear pattern, beginning with a tonic chord. The chords after the tonic are based on a first order Markov model, with set probabilities to move to a different chord. The likelihood of each chord was chosen subjectively based on decisions made by the system creator, aiming for appropriate progressions for EDM.

Mathematical Generation

Many composers have used fractal equations in algorithmic composition (Roads 1996). Studies in perception have shown that fractal contours applied to pitch and loudness can be recognized by listeners (Schmuckler and Gilden 1993). It has also been proposed that the music of Bach can be reverse engineered to represent fractal equations (Hsu and Hsu 1990). This form of generation is useful because it is nonhuman, taking the system into new territory as creative stimulation. To generate in this way, Artiin uses the formulas of the Ikeda Map and the Henon Map and uses the resulting number to generate rhythmic relations, harmonic progressions and influence overall structure. Both the Hénon map and Ikeda Map are two-dimensional maps create a series of x, y coordinates using the equation below. Figure 4 displays a graphical representation of the Ikeda Map (created in MATLAB by the author).



$$\begin{cases} x_{n+1} = 1 + u(x\cos(t) - y\sin(t)) \\ y_{n+1} = u(x\sin(t) + y\cos(t)) \\ t = 0.4 - 6/(1 + x + y) \end{cases}$$

Figure 4. Ikeda Map

These maps are iterative processes and discrete dynamical systems that show chaotic results. A discrete dynamical system moves at chosen time intervals, and continually uses the same formula to move to new locations. New values in a sequence can be chosen whenever required. Chaos in mathematics refers to systems that can show drastically different outcomes despite only slight variations in their initial conditions; the musical effect is to open up many unanticipated trajectories.

Important to Artiin's use is the tendency of chaotic systems to create highly structured almost periodic orbits, with unexpected deviations following logic that is not often easily discernable. The Hénon map and Ikeda map are particularly effective for musical patterns because they display a regular orbit, albeit with chaotic results that contain intermittent outliers. By using an underlying formula such as this, I believe it is possible to create the illusion of an intelligent organized system that has somewhat predictable results, but does show occasional creativity through the chaotic movements, reaching beyond the more predictable patterns.

Balancing between predictability (the regular orbit) and unpredictably (the chaotic results) is a key part of music, assuming pure repetition itself is not the aesthetic basis of the piece. Pierre Boulez describes all Western music as "caught up in a 'dilemma' involving repetition, variation, recognition and the unknown" (Campbell 2010: 154). Iterative processes can be considered a form of musical composition through their internal repeating process that create recognizable patterns. With the addition of chaotic results as the "unknown" these formulas can be used to create musically useful output.

Through experiments substituting the Hénon map with alternate formulas or with random values, it became apparent that this use of an iterative formula does contain musical significance. While my use of the Hénon map does produce defined harmonic and rhythmic characteristics, this sound is never identical in subsequent performances. I would liken its output to that of a performer reinterpreting improvisational guidelines in each performance.

Artiin uses three versions of the Henon and Ikeda maps, with a version of each changing once per four measures, once per measure and once each beat. The starting values for each are randomly assigned each time Artiin is run. By using multiple versions of each map, parameters (such as harmony, rhythm and structure) move at different speeds. This allows for very direct mappings from the variables to musical parameters, without sacrificing musical interest and variety.

Each instrument uses the Hénon map in a slightly different way, but one common characteristic is to use the x and y values to create rhythmic lengths. As a basic variation, each value of x and y could be multiplied by 100 to create a note length (in milliseconds) and once these notes end the next point in the map is created to output the new note lengths. By using this internal structure from the Hénon map across multiple iterations, combined with the input of a live performer, it is possible to create many varied results from the one algorithm.

For example, if the Hénon map output the values of 1 and 0.75 (all numbers are to 4 decimal places), this would be converted to a quarter note and then a dotted eight note. Each unit is considered to be worth one beat; as these maps produce a range of numbers this can create a wealth of rhythmic ideas. Rhythmic ideas can be quantized to any degree by the user (set by the Synco option in the user interface). For EDM, the lowest level of quantization still quantizes the generated material.

Drum Pattern Generation

In addition to mathematical generation, drum parts are also generated using a simple stochastic process based on set levels of syncopation and density. Using these parameters of density and syncopation Artiin places drum hits across a two-bar grid and generates positions to place a drum hit. Higher syncopation levels encourage more placement off the beat while density levels set how many notes are created. This is followed with each note assigned a probability of being hit each time through the cycle, again set by the density and syncopation level. The second level of probability allows continued variance in the beat similar to prior work in the area (Collins 2001). This process is used to create a simple groove that can lock in the more complex interactive ideas. The groove is shown to the user (see fig. 5), and new versions can be created at any time.

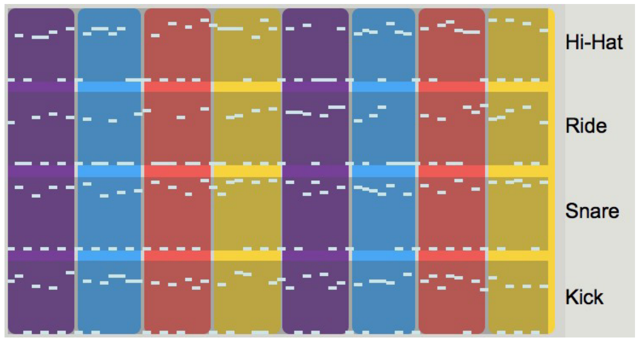


Figure 5. Drum Pattern Generation

Input Transformations

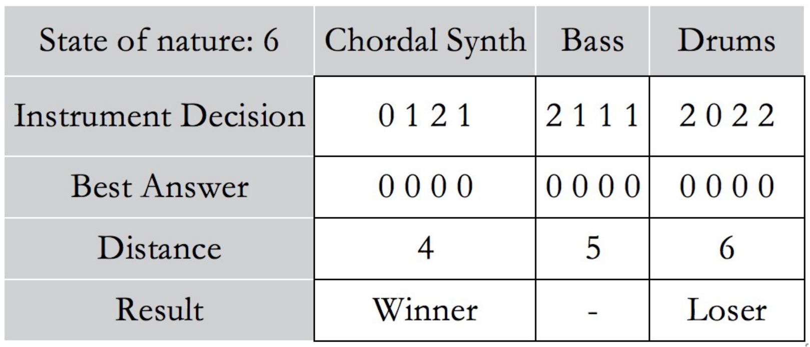
Input material is also transformed and used by Artiin. Artiin is able to repurpose any previous musical information. This is primarily performed by extracting parts of an idea at a set complexity. Once the musical motif is taken, Artiin will play either a fragment of the motif, or a variation of the motif. These ideas can be transformed in many ways, including rhythmically and harmonically. Rhythmic transformations are when the motif is extended or shortened, internally altered (such as with an extra embellishment) or with a start point shifted. Harmonic transformations occur when an entire motif is transposed, or parts of the motif are transposed.

The other important input transformation occurs in real time using incoming rhythmic distances as a base for output. For example, if the distance between the last two notes is a quarter note, a pattern that corresponds to the ratio of this idea is played in real-time. This could be in eighth notes, quarter notes or half notes, relating to half, equal or double the input. Before input transformations are used by the broader system there is a final layer of post-processing that can be enabled. This post-processing can alter the harmonic output by transposing signal notes to fit within a key. It also simplifies rhythmic patterns when appropriate to fit the global parameters.

Musical Decision Making

At this stage, Artiin has an array of algorithmically created musical ideas to draw from. Any of these parameters can be indirectly controlled by the user, as discussed in the user-interfaces section below. As an autonomous decision-making scheme determining the behavior of each component in the absence of human intervention, Artiin plays a basic game amongst each instrument, using ideas inspired from game theory.

Each instrument firstly chooses a random four-number sequence containing any combination of 0, 1 and 2. This list is then compared to a set database of winning ideas for each instrument, that relates to the incoming level of complexity. For example, when the state of nature (set by the complexity level) is 6, the best answer for the bass may be 0 0 0 0 (see fig. 6). If the chordal synth chooses 0 1 2 1, that is then a distance of 4 away from its best answer (0 + 1 + 2 + 1 = 4). The chordal synth is then more likely to choose these answers the next time the complexity level is 6. The goal is not to achieve the best possible answer, but instead to dictate potential musical directions. The game also doesn't necessarily lead to the best answer: in table 3 the chordal synth is now more likely to choose 0 1 2 1 as the best answer, even though it is relatively far from 0 0 0 0.



State of nature: 6	Chordal Synth	Bass	Drums
Instrument Decision	0 1 2 1	2 1 1 1	2 0 2 2
Best Answer	0 0 0 0	0 0 0 0	0 0 0 0
Distance	4	5	6
Result	Winner	-	Loser

Figure 6. Game Based Decision Making

Output

Artiin sends a MIDI out to user selected virtual instruments. These instruments can be from any synthesizer hosted in Ableton including third party plug-ins. The bass part is monophonic and the chordal synth requires a polyphonic instrument. Each individual drum sound (cymbal, snare) can be chosen by the user. Allowing the user control of the sounds is a necessity as it is beyond the scope of Artiin to include its own synthesis methods. With timbre a central component of EDM, it is expected that users will prefer to use their own sounds when creating EDM. This does however mean that Artiin makes no attempt to control lower level timbre parameters.

User Interface

The user interface was designed for ease of use in live performance and composition. With this end goal the generative systems described above are not mapped directly to the user interface, and are instead controlled abstractly. The aim of the interface however is not to restrict control, but instead to simplify decisions by reducing their complexity. As well as chordal synth and bass and drum controls, there is a general control interface. This displays the chord generation that is currently created by Artiin, as well as providing a global "Kill" switch and a global reset option.

Chordal Synth and Bass

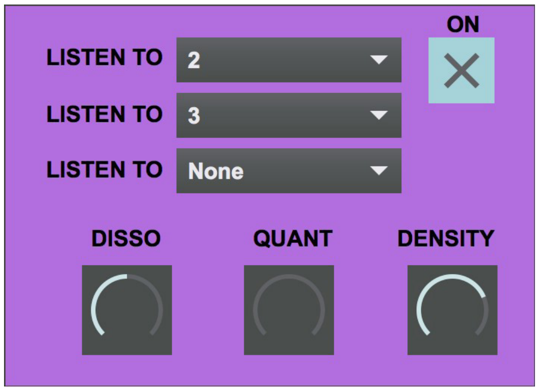

LISTEN TO 2
LISTEN TO 3
LISTEN TO None
ON ✕

Figure 7. Chordal Synth and Bass Interface

The chordal synth and bass have matching interfaces, although they are independently set. The "listen to" options allow the user to choose up to three incoming streams of either audio or MIDI, including other Artiin instruments. While it is possible to bus more streams together, in testing this created an output far abstracted from the incoming streams. "Disso" refers to the dissonance level and impacts how closely Artiin follows incoming harmonic ideas. This also alters the level of post-processing on transformational ideas.

"Quant" refers to the level of quantization. This acts like a regular method of quantization, restricting rhythmic values to a set grid, dependent on the user's chosen level. This method of quantization acts independently of Ableton Live's quantization settings. If this forces notes to overlap, the first note will remain while other notes are dropped. The final parameter density describes how many notes per measure can occur. No selection for any category will allow the computer to decide the parameters.

Drums

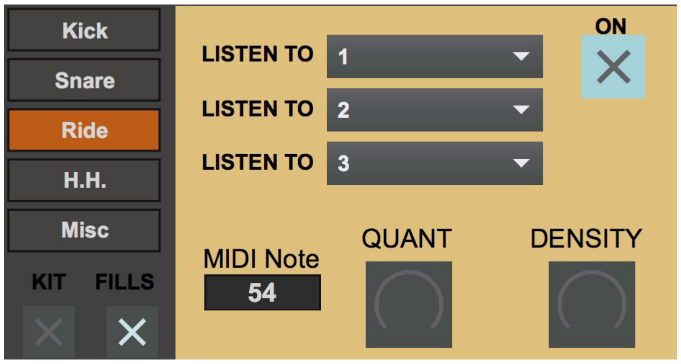


Figure 8. Drum Interface

The user is able to choose a density level and a quantization level independently for the kick drum, snare drum, ride cymbal and hi-hat (see fig. 5). Unlike the chordal and bass interface, however, when no option is chosen while the drums are on, each part reverts to the simple drum generative method. Defaulting to the simple generative drum pattern was chosen as the other drum methods tend to have a distracting influence. There is no single method to control all components of the drums simultaneously as it seemed stylistically ineffective to have the entire drum kit reacting in the same manner. The drum kit is most effective when one element (such as the kick) is responding to the bass while the snare drum is then responding to the kick drum. Allowing each component to react individually creates the sound of multiple percussionists. While such can be used for a novel effect, it is often not stylistically appropriate. The "fills" option allows the drum kit as a whole (not individual components) to create fills around all incoming ideas. The "MIDI note" option allows the user to assign which MIDI note will be triggered.

User Studies

Artiin underwent two separate user studies, the first conducted with an early iteration. The primary goal of the user tests was to identify if the system is usable for DJ practice. The first user study took place in January, 2016 and consisted of an informal discussion among six

DJs, each of whom perform at least once a month. Qualitative responses were recorded in the group discussion, with participants able to lead the discussion. Discussions included an analysis of Artiin and also explored broader ideas for additional software potentially useful in their workflow.

From the first user study three conclusions were reached and used to continue development. Firstly, although this was not a user complaint, the implementation of techniques can always be improved. Overall, the participants thought the musical output worked well with their own inputs and believed the output would be effective in performance. The original intention had been to implement novel developments in interactive music into the system, such as the work performed by Magenta in deep learning (Jacques, 2016). It was originally expected that the majority of work would need to be performed on the algorithmic processes, instead future efforts focused on interface design.

The second conclusion from the first round of user testing was that while tools can be effectively repurposed, the principles behind them must be rethought. The development of an autonomous system was often undesirable for users, who instead prefer the system to accompany and support their work. This understanding led to the final conclusion emerging from this user study: that control over the system is pivotal to the user's enjoyment of the system. The original interface consisted of a single screen that allowed a user to turn each instrument on and off, but no lower level controls.

The second user study was conducted in December, 2016 by which time Artiin was in the form described in this article. The study was again conducted as a discussion, however focused on whether or not the goals of the original discussion had been achieved and other potential improvements for the system. This study took place after adding the lower level controls explained in this article. The same six users unanimously preferred the newer version. It became clear that transparency of control greatly improved their desire to use the system. However, it also became clear that the current scheme did not meet the requirements of all users. Of the six participants, two requested lower level control, one was satisfied with the control scheme, while the remaining three believed the control scheme could be improved further. One suggestion was to develop a modular system where the user can define their own control scheme.

After the second user study, all six musicians were given the option to continue using Artiin with support from its creators. Three participants decided to continue using Artiin and now regularly use it in their work. Their most common use of Artiin is adding single layers at certain times in different pieces. These DJs remain in regular communication and actively contribute to the development of Artiin. The three users who elected to discontinue using the system were asked to explain their decision. While one stated they didn't feel it was a tool useful for their work, the other two mentioned that it did not fit into their current workflow. From initial discussions within the first user study it was found that, on average, each DJ added or removed software and hardware from their setup within six months.

Future Developments

Artiin remains a prototype and has not had extensive user studies beyond the feedback gained from its initial user pool. One key flaw with Artiin is its control of timbre, a key characteristic of EDM. For EDM, MIDI can capture important information on dynamics and pitch, but timbre qualities will need to be addressed in future versions. Using MIDI out does however allow Artiin to work with a user's chosen instrument sounds. Some users deploy Artiin for certain parts of a piece, but then want to transition to composed material.

Artiin currently possesses a limited ability to make longer-term musical structural decisions. The use of game theory for decision-making leads to varied structures, however these ideas are not always effectively connected. This is not usually a problem as when used with live input the structure is determined by the user. Artiin's methods of interaction could be improved through better methods that do not rely on a one-dimensional model (-1,1).

Of significant interest for Artiin is developing further methods of retaining its own musical ideas and interactions from past performances. Users have noticed variation in the output after multiple use, due to changes in the database as the system stores ideas. Artiin's Markov model and drum generation are both altered during the course of a user's interaction with the system, but developing this method further could develop a better user response. Future user research of audience reaction and perception may also be useful before making further changes to the system.

Conclusion

Artiin provides an algorithmic approach to EDM that can be used in many different ways, suitable for a variety of workflows. Artiin demonstrated the application of a range of research in interactive music for EDM. Significantly, it showed that existing research in other genres can be applied very effectively, but new principles for applying this research have to be defined for the genre. In particular, Artiin demonstrated the need for clear control and UI for live performance of EDM.

Acknowledgements

Thanks to Jakob Nagel who was the co-developer of the audio analysis system.

References

Ames, Charles. 1989. "The Markov Process as a Compositional Model: A Survey and Tutorial". *Leonardo* 22(2): 175–87. <https://doi.org/10.2307/1575226>.

Bello, Juan Pablo, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies and Mark B. Sandler. 2005. "A Tutorial on Onset Detection in Music Signals". *IEEE Transactions on Speech and Audio Processing* 13(5): 1035–46. <https://doi.org/10.1109/TSA.2005.851998>.

Blackwell, Tim, Oliver Bown and Michael Young. 2012. "Live Algorithms: Towards Autonomous Computer Improvisers". In *Computers and Creativity* ed. Jon McCormack and Mark d'Inverno, 147–74. Berlin, Heidelberg. <https://doi.org/10.1007/978-3-642-31727-9_6>.

Boden, Margaret A. 2000. "Computer Models of Creativity". *Psychologist* 13(2): 72–76. <https://doi.org/10.1609/aimag.v30i3.2254>.

Campbell, Edward. 2010. *Boulez, Music and Philosophy* (*Music in the Twentieth Century).* Cambridge: Cambridge University Press

Collins, Nick. 2001. "Algorithmic Composition Methods for Breakbeat Science". In *Proceedings of Music Without Walls*. De Montfort University: Music Without Walls

Collins, Nick, and Alex McLean. 2014. "Algorave: A Survey of the History, Aesthetics and Technology of Live Performance of Algorithmic Electronic Dance Music". *Proceedings of the International Conference on New Interfaces for Musical Expression*, 355–58. Goldsmiths: University of London

Cope, David. 1989. "Experiments in Musical Intelligence (EMI): Non-linear Linguistic-based Composition". *Interface* 18(1–2): 117–39. <https://doi.org/10.1080/09298218908570541>.

———. 2005. *Computer Models of Musical Creativity*. Cambridge, Mass.: MIT Press.

Dahlstedt, P. 2001. "Creating and Exploring Huge Parameter Spaces: Interactive Evolution as a Tool for Sound Generation". International Computer Music Conference. Laboratorio Nacional de Música Electroacústica, Havana.

Dobrian, Christopher. 2004. "Strategies for Continuous Pitch and Amplitude Tracking in Realtime Interactive Improvisation Software". Sound and Music Computing Conference. IRCAM, Paris.

Drummond, Jon. 2009. "Understanding Interactive Systems". *Organised Sound* 14(2): 124–33.

Hsu, Kenneth J., and Andreas J. Hsu 1990. "Fractal Geometry of Music". *Proceedings of the National Academy of Sciences of the United States of America Physics* 87: 938–41. <https://doi.org/10.1073/pnas.87.3.938>.

Jaques, Natasha, Shixiang Gu, Richard E. Turner and Douglas Eck. 2016. "Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning". Centre Convencions Internacional Barcelona, Barcelona. Neural Information Processing Systems.

Kranenburg, Peter, Berit Janssen and Anja Volk. Van. 2016. "The Meertens Tune Collections: The Annotated Corpus (MTC-ANN) Versions 1.1 and 2.0.1". Amsterdam: Meertens Institute.

Lewis, George E. 2000. "Too Many Notes: Complexity and Culture in Voyager". *Leonardo Music Journal* 10: 33–39. <https://doi.org/10.1162/096112100570585>.

Livingstone, Steven, Emery Schubert, Janeen Loehr and Caroline Palmer. 2009. "Emotional Arousal and the Automatic Detection of Musical Phrase Boundaries". International Symposium on Performance Science. Univeristy of Otago.

Miranda, Eduardo Reck, and Marcelo Wanderley. 2006. *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*. Middleton: A-R Editions, Inc.

Pearce, Marcus, David Meredith and Geraint Wiggins. 2002. "Motivations and Methodologies for Automation of the Compositional Process". *Musicae Scientiae* 6(2): 119–47. <https://doi.org/10.1177/102986490200600203>.

Roads, Curtis. 1996. “The Computer Music Tutorial”. *Computers & Mathematics with Applications*. Cambridge, Mass: MIT Press.

———. 2004. *Microsound*. Cambridge, Mass: MIT Press.

Rowe, Robert. 1993. *Interactive Music Systems: Machine Listening and Composing*. Cambridge, Mass: MIT Press.

———. 2001. “Machine Musicianship”. Cambridge, Mass: MIT Press.

Schmuckler, Mark A., and David L. Gilden. 1993. “Auditory Perception of Fractal Contours”. *Journal of Experimental Psychology: Human Perception and Performance* 19(3): 641–60. <https://doi.org/10.1037/0096-1523.19.3.641>.

Ulyate, Ryan, and David Bianciardi. 2001. “The Interactive Dance Club: Avoiding Chaos in a Multi-Participant Environment”. (Seattle): CHI’01 Workshop on New Interfaces for Musical Expression (NIME-01). <https://doi.org/10.1162/014892602320582963>.

Wang, DeLiang, and Guy J Brown. 2006. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. WileyIEEE Press. <https://doi.org/10.1109/TNN.2007.913988>.