2017.xCoAx.org

DESIGNING MUSIC WITH MUSEBOTS

ARNE EIGENFELDT
arne_e@sfu.ca

Simon Fraser University
Vancouver, Canada

Lisbon
Computation
Communication
Aesthetics
& X

Abstract

Musebots are pieces of software that autonomously create music, collaboratively with other musebots. Since the development of the musebot protocol, the author has created several generative music and collaborative systems using these musical agents. This paper describes how the desired musical results influenced the design of the musebots themselves. Rather than presenting the latest musebot system——*Moments*——as a system description, the author describes the musical decisions that prompted the design, and re-design, of the musebots themselves.

Keywords

Generative Music
Musical Metacreation
Musical Composition
Musebots
Musical Agents

1. INTRODUCTION

Generative music offers the opportunity for the continual reinterpretation of a musical composition through the design and interaction of complex processes that can be rerun to produce new artworks each time. While generative art has a long history (Galanter 2003), the application of artificial intelligence, evolutionary algorithms, and cognitive science has created a contemporary approach to generative art, known as metacreation (Whitelaw 2004); musical metacreation (MuMe) looks at all aspects of the creative process and their potential for systematic exploration through software (Pasquier et al. 2016).

One useful model borrowed from artificial intelligence is that of agents, specifically multi-agent systems. Agents have been defined as autonomous, social, reactive, and proactive (Wooldridge 1995), similar attributes required of performers in improvisation ensembles. Musebots (Bown et al. 2015) offer a structure for the design of musical agents, allowing for a communal compositional approach (Eigenfeldt et al. 2015) as well as a unified model.

Musebot ensembles, consisting of a variety of agents reflecting the varying musical aesthetics of their creators, have been successfully presented in North America, Europe, and Australia. In these ongoing installations, each ensemble has been limited to five minute performances, after which the ensemble exits and the next ensemble begins. While this limitation can be seen as an opportunity to feature as wide a variety of musebots as possible—as of this writing, there are over seventy-five unique musebots—it raises some questions.

If the musebot ensemble is a proof-of-concept, then it is certainly successful: musebots interact and can self-organise, producing novel output that can be surprising and arguably valued—thus attaining a mark of computational creativity (Boden 2009). However, after listening to any one ensemble for longer than a few minutes, one recognizes musical limitations: the interactions between musebots remain at the musical surface (i.e. harmony, rhythm, density). Moving beyond this duration, the successful outcome is no longer dependent upon interaction, but expands to include structure. It is for this reason that many MuMe practitioners have remained as part of the creative process, whether as musicians interacting with an interactive system, or as operators, triggering global changes when the surface has become too predictable or static (Eigenfeldt et al. 2016). I have used musebots in a variety of artworks in the past two years (Eigenfeldt 2016a, Eigenfeldt 2016b), and have become their main evangelist. Although one of the main desires of the musebot project was to provide a collaborative framework that allowed sharing of ideas and code, I have found the need to move forward independently. In doing so, I hope to not only create interesting and valuable artworks, but, through their documentation, I hope to entice others to join me in the development and extension of the musebot paradigm.

2. MUSICAL AGENTS AND MUSEBOTS

The potential of agent-based musical creation was explored early in the history of computer music (Wulfhorst et al. 2003, Murray-Rust and Small 2006), specifically in their potential for emulating human-performer interaction. The author's own initial investigation into multi-agent systems is described elsewhere (Eigenfeldt 2007). Within MuMe, the established practice of creating autonomous software agents for free improvised performance (Lewis 1999) has involved

idiosyncratic, non-idiomatic systems, created by artist-programmers (Yee-King 2007, Rowe 1992). While musical results from these systems can be appreciated by other composers, sharing research has been difficult, due to the lack of common approaches (Bown et al. 2013).

Musebots are pieces of software that autonomously create music, collaboratively with other musebots. Musebots were designed to alleviate some of these issues, as well as provide a straight forward infrastructure for development (Bown et al. 2015). A defining goal of the musebot project is to establish a creative platform for experimenting with musical autonomy, open to people developing cutting-edge music intelligence, or simply exploring the creative potential of generative processes in music. The musebot protocol is, at its heart, a method of communicating states and intentions, sending networked messages established through a collaborative document via OSC (Wright et al. 1997). A Conductor serves as a running time generator, as well as a hub through which all messages pass. The Conductor also launches individual musebots via curated ensembles.

Musebot ensembles have been presented as continuous installations at a variety of festivals and conferences, the results of which have been described elsewhere (Eigenfeldt et al. 2015). These ensembles have modeled improvisational explorations, albeit with the potential for generative harmonic progressions. Curation of ensembles have consisted of combining musebots based upon their musical function (i.e. a beat musebot, a bass musebot, a pad musebot, a melody musebot, and a harmony generating musebot). A contrasting ensemble might involve combining several noise musebots (see table 1).

Table 1
Musebot types available online (http://musicalmetacreation.org/musebot-test-suite/)

Type	*Number available*
Bass Generators	5
Beat Generators	13
Harmony Generators	5
Keys/Pads Generators	6
Melody Generators	19
Noise/Texture Generators	16

The information shared between musebots have tended to be surface details, such as a current pool of pitches, density of onsets, and volume. Although having virtual performers audibly agree upon such parameters suggests musically successful machine listening, these levels of interaction become mundane surprisingly quickly; the more subtle interactions that occur in human improvisation ensembles, and their development over time, have not yet been successfully modeled within musebot ensembles.

Several musebot developers participated in ProcJam 15 , and devoted a week to exploring the potential for musebots to broadcast their intentions for the immediate future. While the results heightened the perception of machine listening, they were not able to move beyond the generation of musical surface.

2.1. From Self-Organisation to Composition

My own hesitation to more deeply embrace improvisational approaches has to do with my training as a composer; compositional thought suggests a top-down

approach, a pre-performance organisation in which musical structure elicits large scale change. Traditional narrative and dramatic musical forms, based upon tension and release, result in hierarchical structures that are seemingly impossible to negotiate or self-organise. For example, getting agents to agree upon a central section (i.e. a chorus), and deciding how to progress towards that section through an adequate build of tension without a top-down approach, is beyond my comprehension. While I have explored the use of pre-generated structural templates (Eigenfeldt and Pasquier 2013), I found this unnecessarily restrictive outside of the exact model considered (i.e. electronic dance music).

I have proposed an alternative to traditional narrative structures for musical generation (Eigenfeldt 2016b), specifically what Stockhausen called *Moment-form* (1963). Kramer suggests that such non-teleological forms had already been used by composers such as Stravinsky and Debussy (Kramer 1987), while I have described the use of *Moment-form* in ambient electronic music (Eigenfeldt 2016b). *Moment-form* offers several attractive possibilities for generative music, including the notion that individual moments can function as parametric containers. Just as Stockhausen obsessively categorised and organised his material (Smalley 1974), the parameterisation of musical features by applying constraints upon the generative methods can delineate the moments themselves.

3. MOMENTS

Moments is my first generative work that explores *Moment-form* in generative music through the use of musebots. While the work may have future musebots contributed by the MuMe community, thus far all musebots for *Moments* have been my own. *Moments* exists in two separate versions: the original for two Disklavier pianos (hereafter referred to as *M2016*), in which musebots send MIDI data to the mechanical pianos; a second version in which it generates all audio through Ableton Live (hereafter referred to as *M2017*). When describing consistent elements between the two versions, I will refer to it simply as *Moments*. Due to the use of acoustic instruments, *M2016* has been presented as a live concert work; *M2017*, lacking any visual element, has been presented as an audio installation.

3.1. Structural Musebots

As mentioned, musebots have demonstrated the potential to self-organise. However, a decision was made to approach *Moments* compositionally, and generate an entire musical form prior to each performance. This allows for at least one important benefit: a pre-cognition by all agents of the upcoming structure. Knowing a section is, for example, two minutes in duration, allows musebots to plan their activity within that time.

Moments uses structural musebots, which do not actually generate sound, as well as those that generate audio directly, or via MIDI. A ParamBOT generates sections (moments) within a user determined compositional duration. Kramer suggests that proportional relationships between moment durations are integral to their success, because global coherence cannot come from progression between, or order of, moments (Kramer 1978). Adhering to Stockhausen's own preference for golden ratio relationships, durations are generated by continually dividing the requested performance duration using ratios of 2:3 (see figure

1), ensuring that the longest section is less than three minutes, and the shortest is more than fifteen seconds. These durations are then shuffled to avoid predictability, and allow for shorter moments to adjoin longer moments.

In order for moments to be contrasting, each moment consists of varying parameter levels. Sectional values, either stable or dynamic during a section, are generated stochastically for the following parameters: speed (tempo), activity level (number of events), voice density (number of voices per part), complexity, and volume. Pitch is treated differently between the two versions: figure 1 displays the varying pitch range suggestions used in *M2016*, while *M2017* employs a more complex timbral model, discussed later.

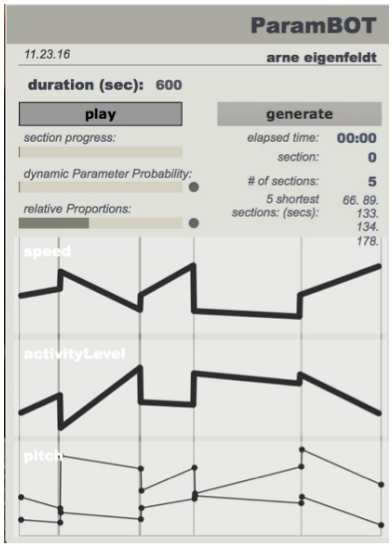

Fig. 1
A portion of the ParamBOT, displaying the request for a ten minute composition's speed, activity level, and pitch range. Vertical lines indicate the five section divisions; horizontal lines indicate parameter change over time.

Because harmony should remain static within a moment, a single pitch class set is generated for the entire composition by a PCsetBOT. Subsets of the larger set are selected for various moments, based upon the section's complexity, as suggested by the ParamBOT. The PCsetBOT broadcasts the pitch class sets for the entire composition, thereby allowing musebots to navigate through sectional divisions.

3.2. Performance Musebots

As with all of my multi-agent systems, individual agents have a unified general design, but operate differently based upon internal variations due to varying attributes; these attributes function very much like personalities (Eigenfeldt and Kapur 2008). In the case of *Moments*, these attributes are:

- Impatience: how long an agent is willing to be inactive;
- Persistence: how long an agent will stay active;
- Vitality: how willing an agent is to add other voices, and how active to be overall;
- Consistency: the amount of variation an agent will attempt, including the potential to play across sections;
- Compliance: how willing an agent is to restrict itself to the ParamBOT's requests;
- Repose: a preference to perform in sparser, or denser, sections.

These internal attributes can be set randomly with each performance, or through an ensemble score, discussed later. While the ParamBOT determines many of the governing parameters for a section, musebots still must decide upon how to interpret these suggestions. Earlier musebot designs consisted of a large number of individual musebots that behaved in a consistent manner, or style. For *Moments*, it was deemed more practical to design a single musebot that could react in different ways depending upon the given musical parameters. Thus, the *M2016* musebots are provided with a variety of performance styles, and they select from these styles based upon their suitability for the current state. For example, a narrow pitch range rules out large intervalic chordal playing styles, while fast speeds rule out rapid repeated notes.

The playing styles for *M2016* are:

- *pointillist*: stochastic pitch, onset, and duration selection within the requested constraints;
- *désordre*: based upon Ligeti's *Piano Etude #1*, a study in polyrhythms and contrasting left / right hand pitch fields;
- *blocky*: five to ten note chords;
- *morse*: a rhythmic motive, favoring a limited number of pitches;
- *arpy*: large intervalic leaps, traveling up and down the keyboard;
- *trills*: alteration of two or three pitches;
- *remembering*: florid melodic shapes in groupings based upon a repeating rhythmic pattern;
- *keith*: rhythmic repetition in left hand of limited pitch sets;
- *herbie*: held chords in left hand, and complex melodic trajectories in right hand;
- *olivier*: "blue green" chords reminiscent of Messiaen's Vocalise from his *Quartet for the End of Time*.

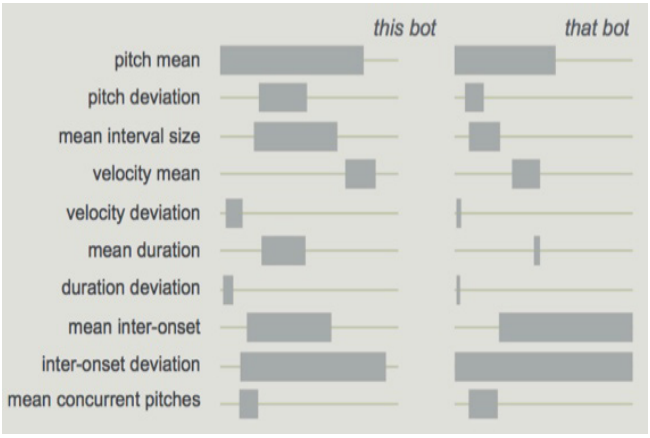

Fig. 2
Musebot self-analysis.

A fundamental aspect of musebots is their ability, and requirement, to communicate their current state through messages. Rather than broadcast their performance style, the *M2016* musebots analyze their own playing, and transmit a feature analysis (see figure 2). The other musebot then attempts to match these features to what it understands about its own playing styles, and has the option of switching its current style. While both musebots share the same stylistic analysis—and thus should be able to translate the current features into actual sty-les—the decision was made to allow for misinterpretation of the other musebot's style based upon its transmitted features.

In other words, what the musebot believes it is doing may not be what the musebot is actually doing. This ambiguity is exploited in *M2016*, in that the musebots will attempt to match one another's playing, based upon their own internal beliefs, creating variation within a given moment.

Given the wide variety of possible states for each moment, as suggested by ParamBOT and interpreted by the two musebots in *M2016*, each moment is usually quite distinct while retaining sectional consistency. Changes between moments display Kramer's notion of discontinuity.

3.3. Translating Musebots

Between July and December 2015, *M2016* was performed three times in concert, each with a ten minute duration. While presenting generative works in concert is always risky—there is never a guarantee that musical agents will produce optimal output within a set duration—the structure of *Moments* enforces variety between a moment's consistency and discontinuity between adjacent section, producing a satisfying balance between moderate predictability (due mainly to the musebot's limited number of playing styles) and surprise (due to the complexity of the overall system). Thus, it seemed natural to take the next step, and eliminate the constraints placed upon timbre, and have the system entirely determine how it sounded.

All of my generative systems, when creating their own audio, have had severe limitations placed upon their timbral output: in most cases, they have been forced to choose from a pre-determined collection of samples or synthesis patches. While one could argue that, given five musebots choosing from synthesis collections of five patches each, the potential combination of timbres is quite large, the main reason for the constraint is to guarantee some level of predictability and musical success. *M2017* would attempt to remove this constraint, and select from over 1,200 sounds available in Ableton Live.

Table 2
Machine learning and possible patch requests in *M2017*.

parameter	*explanation*	*request*
sustain	how much amplitude decay occurs over five seconds?	>, <, =
release	how much amplitude decays between 5 seconds (the release) and 6 secs?	>, <, =
percussive	are the first amplitude slices (@ 100ms or 250ms) the loudest?	boolean
flux	how much spectral change occurs over the duration?	>, <, =
spectrum	how rich is the harmonic spectrum?	>, <, =
harmonic	how strong are the 2nd, 4th, and 8th harmonics?	>, <, =

3.4. Patch Analysis

In order to create generative methods for timbral selection, the system requires some knowledge about the library of available sounds. A machine learning algorithm was created to analyze every synthesizer patch in the Ableton Live library. A script was written to play every patch @ MIDI notes 36, 60, and 84; analysis was done on the amplitude and spectrum over a five second duration.

Space does not permit a detailed description, other than to state that the resulting database allowed for requests to be made (see table 2) in order to provide a sorted list of appropriate patches. For example, a musebot could request a list of sustained patches (sustain = 1.0) with medium release (release > 0.5), a clear harmonic spectrum (harmonic >0.9), fewer harmonics (harmonic < 0.2), and not very much timbral change (flux < 0.2).

3.5. Spectral Models

Due to the homogeneous nature of the piano timbre across its pitch range, *M2016* could exploit a stochastic pitch range request. The aesthetic model of that work——20th century piano music—further allowed for such generation.

M2017 instead uses a model of ambient electronica, requiring pitch selection that incorporates spectrum. Twenty five tracks of ambient electronica, by Christopher Bissonnette, Loscil, and Marsen Jules were analysed using 24 band Bark analysis (Eigenfeldt and Pasquier 2010), once per second. Spectral slices were stored in a database; those that were below the track average (i.e low amplitudes usually found in fade-in and fade-out), as well as those that were similar to existing slices, were discarded. Analysis of 25 tracks resulted in 670 unique spectral slices.

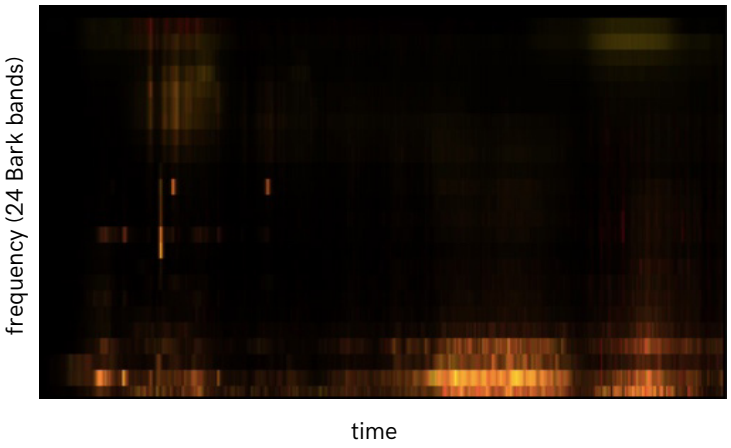

Fig. 3
Bark analysis of Bissonnette's *A Wild Tonic in the Rain*. Brightness indicates higher amplitude for that Bark band.

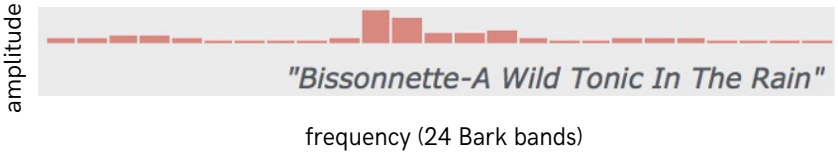

Fig. 4
A spectral slice from Bissonnette's *A Wild Tonic in the Rain*, displaying amplitudes of 24 Bark bands for a one second period.

The ParamBOT selects two slices at random from the spectral slice database, then generates interpolations between them, the number of which depends upon the number of sections for a new composition. These new spectral slices are then shuffled, and are considered the spectral targets for each section. The slices can be considered "real-world" models; the interpolations provide enough variation in timbre between sections, yet enough consistency so as to maintain the perception of a single musical composition.

3.6. Design versus reality: heuristics at play

Using the request system to determine suitable timbres did not, unfortunately, prove to be musically dependable. This may have been due to *what* the machine-learning algorithm was tasked with learning, or it may have been due to the ambiguity of information given to the selection algorithm. For example, asked to provide timbres suitable to be used as drones—sustained, harmonic, with enough timbral flux to remain sonically interesting—the system could suggest very plain timbres (i.e. a bassoon) or unsuitable (i.e. a dance-synth patch with a dramatic vibrato). Other examples included selecting sample-based patches at the extremes of their pitch range that resulted in audible loops, or combing two timbres that would both be considered "foreground", and thus

detracting from one another. These problems could be solved by further refinement of the audio features in the analysis, but it seems that this problem borders upon computational aesthetic judgement: attempting to distinguish *why* certain timbres could, or could not, be used in certain circumstances.

As a composer, rather than a computer scientist, I feel such research is beyond my immediate goals. Other, more "brute-force" methods would include compiling a list of unusable patches, but such a method defeats the purpose of automating the timbre selection. Lastly, attempting to adjust the timbre once a selection is made (i.e. altering the filter cut-off or vibrato) is problematic, as each patch in the Ableton Live database consists of a variety of synthesis methods and algorithms: creating a secondary database of available synthesis parameter settings for each patch does not seem to be an elegant solution (for now).

Instead, two additional audio musebots were created, whose timbre could be more precisely controlled. A resonant sample-based player (KitsilanoBot), with a library of field recordings and soundscapes, was created, in which the individual resonant frequency bands is tuned to match the section's pitches and overall spectrum (described below). An "intelligent" granular synthesis player (GenoaBot) was also coded, and provided with a large library of instrumental samples; sample-selection is based upon the overall timbre of the section, guaranteeing suitable and consistent timbral performance in every frequency range. The Ableton Live synthesis musebots (LondynBots, SienaBots), utilize the request system, and can safely serve as foreground timbres. Similar to the initial musebot installations, ensembles were created so as to provide variety between compositions, including the possibility of specifying individual musebot attributes (see table 3).

Table 3
Audio Musebots in *Moments2017*

Musebot	*Description*	*max. per composition*
KitsilanoBot	Resonant soundscapes/field recordings	2
GenoaBot	Granular synthesis	3
LondynBot	Long tones/drones	2
SienaBot	Short tones/foreground	1

3.6. Design versus reality: heuristics at play

Because all parameter information, including spectrum, is generated and communicated prior to the performance, the audio musebots can negotiate which bark bands they will individually cover for individual sections. This negotiation is preceded by deciding within which section musebots will be active, based upon the section's *activityLevel* (broadcast by ParamBot), balanced by an individual musebot's *vitality*. Musebots broadcast this information as intentions, and other musebots can alter their own plans based upon this shared information.

During the extensive listening to generated output for the earlier *M2016*, it seemed that a musebot's *compliance* attribute could be used to obfuscate the ParamBot's requests; in other words, allowing musebots some flexibility in responding to the overall structural parameters. One example of this in *M2017* can be seen in the "claiming" of Bark bands: if a musebot broadcasts its intention to play in bark band 3 (MIDI notes 55-62), no other musebot would be allowed to use those pitches. However, when it comes to deciding which pitches to play from within its intended bark bands, musebots with low *compliance* attribute can wander outside their limits. This necessitated further communication to the

ensemble: not just what a musebot intends to do, but what it is actually doing. A similar adjustment is required during performance when a new section begins. Due to the complexities of negotiation, it is possible for a section to occur in which no musebots are active. Musebots with high *compliance* attributes are the first to test this case; if they find no other musebots active, they test their own *vitality* attribute — essentially a musebot's energy level — to see if it should play in the section. Because there is no guarantee that a musebot exists with high compliance and high vitality, there remains the possibility for a section's continued silence. Rather than overriding this possibility, this result, however unlikely, is allowed to exist in order to avoid the creation of a collection of heuristic settings that limits the explored space; or, as was suggested to me, differentiating between fixing mistakes and allowing unintended intentions (which may be interpreted as surprise).

4. CONCLUSION AND FUTURE WORK

Musebots offer a flexible method for designing musical agents, but their successful implementation requires an iterative process no different than more traditional modes of artistic creation. Just as one cannot expect to combine a random assortment of human musicians into a functioning musical ensemble, musebots need to be designed, and redesigned, in order to perform adequately within a specific musical environment. Just as musicians may rehearse to achieve their best — and arguably predictable — performance, fine tuning musebots so as to limit their output to best match the aesthetic aims of a generative work remains a necessity.

M2016 demonstrates the potential for moment form as a structural container for generative procedures, and a means for musebots to generate surface features while being constrained by formal elements. *M2017* demonstrates the potential to apply metacreative procedures to timbral organization; however, a great deal remains to be done in this regard. Paramount is an extension of the notion of intention versus actuality: a musebot may claim certain timbral bark bands, for example, but its actual timbre may be much greater. For the musebots to know exactly what they are producing, audio analysis of their own output will be required. Whether this is done by individual musebots, or by a structural musebot "listener" which then transmits current states and required adjustments, remains a research foci.

Acknowledgements. The author wishes to acknowledge the Social Sciences and Humanities Research Council of Canada (SSHRC), the Simon Fraser University Cultural Unit, the School for the Contemporary Arts, and the Generative Media Project at Simon Fraser University.

REFERENCES

Boden, Margaret. *Computer models of creativity.* AI Magazine 30:3, 2009.

Bown, Oliver, Arne Eigenfeldt, Aengus Martin, Benjamin Carey, and Philippe Pasquier. *The musical metacreation weekend: challenges arising from the live presentation of musically metacreative systems*. Proceedings of the Musical Metacreation Workshop, Boston. 2013.

Bown, Oliver, Benjamin Carey, and Arne Eigenfeldt. *Manifesto for a Musebot Ensemble: A platform for live interactive performance between multiple autonomous musical agents.* Proceedings of the International Symposium of Electronic Art, Vancouver, 2015.

Eigenfeldt, Arne. *The Creation of Evolutionary rhythms within a Multi-Agent Networked Drum Ensemble.* Proceedings of the International Computer Music Conference, Copenhagen, 2007.

Eigenfeldt, Arne, and Ajay Kapur. *An Agent-based System for Robotic Musical Performance.* Proceedings of the New Interfaces for Musical Expression Conference, Genoa, 2008.

Eigenfeldt, Arne, and Philippe Pasquier. *Real-time timbral organisation: Selecting samples based upon similarity.* Organised Sound 15:02, 2010.

Eigenfeldt, Arne, and Philippe Pasquier. *Evolving structures for electronic dance music*. Proceedings of the 15th conference on Genetic and evolutionary computation, Amsterdam, 2013.

Eigenfeldt, Arne, Oliver Bown, and Benjamin Carey. *Collaborative Composition with Creative Systems: Reflections on the First Musebot Ensemble*. Proceedings of the International Conference on Computational Creativity, Park City 2015.

Eigenfeldt, Arne, Oliver Bown, Andrew Brown, Toby Gifford. *Flexible Generation of Musical Form: Beyond Mere Generation.* Proceedings of the International Conference on Computational Creativity, Paris, 2016.

Eigenfeldt, Arne. *Musebots at One Year: A Review.* Proceedings of the Fourth International Workshop on Musical Metacreation, Paris, 2016a.

Eigenfeldt, Arne. *Exploring Moment-form in Generative Music.* Proceedings of the Sound and Music Computing Conference, Hamburg, 2016b.

Galanter, Philip. *What is Generative Art? Complexity theory as a context for art theory.* Generative Art Conference, Milan, 2003.

Kramer, Jonathan. *Moment form in twentieth century music.* The Musical Quarterly 64:2, 1978.

Lewis, George. *Interacting with latter-day musical automata.* Contemporary Music Review, 18:3, 1999.

Murray-Rust, David, Alan Smaill, and Michael Edwards. *MAMA: An architecture for interactive musical agents.* Frontiers in Artificial Intelligence and Applications 141:36, 2006.

Pasquier, Philippe, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. *An Introduction to Musical Metacreation.* Computers in Entertainment (CIE) 14:2, 2016.

Rowe, Robert. *Machine composing and listening with Cypher.* Computer Music Journal, 16:1, 1992.

Smalley, Roger. *'Momente': Material for the Listener and Composer: 1.* The Musical Times 115:1571, 1974.

Stockhausen, Karlheinz. *Momentform: Neue Beziehungen zwischen Aufführungsdauer, Werkdauer und Moment.* Texte zur Musik 1, 1963.

Whitelaw, Mitchell. *Metacreation: Art and Artificial Life.* MIT Press, 2004.

Wooldridge, Michael, and Nicholas Jennings. *Intelligent Agents: Theory and Practice.* Knowledge Engineering Review, 10:2, 1995.

Wright, Matthew, and Adrien Freed. *Open Sound Control: A New Protocol for Communicating with Sound Synthesizers.* Proceedings of the International Computer Music Conference, Thessaloniki, 1997.

Wulfhorst, Rodolfo, Lauro Nakayama, and Rosa Maria Vicari. *A multiagent approach for musical interactive systems.* Proceedings of the second international joint conference on Autonomous agents and multiagent systems. ACM, 2003.

Yee-King, Matthew. *An automated music improviser using a genetic algorithm driven synthesis engine.* Applications of Evolutionary Computing, Springer Berlin Heidelberg, 2007.