A Realtime Generative Music System using Autonomous Melody, Harmony, and Rhythm Agents

Arne Eigenfeldt
School for the Contemporary Arts, Simon Fraser University, Burnaby, Canada
www.sfu.ca/~eigenfel
e-mail: arne_e@sfu.ca

Philippe Pasquier
School of Interactive Arts and Technology, Simon Fraser University, Burnaby, Canada
www.sfu.ca/pasquier/
email: pasquier@sfu.ca

Abstract

Kinetic Engine is a realtime generative music system that has been in development since 2005. It has been used as an extended instrument within an improvising ensemble, as a networked performance ensemble, as an interactive installation, and as an independent performance system under the composer's control. The first two versions were solely concerned with polyphonic rhythmic organisation using multi-agents. Version 3 introduced a genetic algorithm for the evolution of a population of rhythms, in realtime, based upon the analysis of music provided. Version 4 explored melodic organisation, again using multi-agents, while the most recent version adds a third order Markov model for harmonic generation.

This paper gives an overview of the different versions of the system. Furthermore, the system's use as a performance instrument, as opposed to an independent installation, will also be discussed, describing the necessary shifts in conception regarding generative algorithms. Finally, an attempt to evaluate the entire system from an artistic, rather than scientific, perspective will be undertaken.

1. Introduction

Kinetic Engine is a interactive generative music system designed and created by a composer, exploring new methods for the generation of musically interesting gestures in realtime. Generative music systems have been used in live performance for decades [1]. The principle organising method for controlling complexity in these systems has been constrained randomness, the limits of which are discussed in detail elsewhere [2, 3]. Simply put, while constrained randomness provides a convenient and adequate solution for generating music gestures in realtime, it cannot come close to the organised complexity of intelligent improvising musicians (the model for most interactive computer music).

Thus, *Kinetic Engine* has been developed, not as a singular approach to employing intelligence in realtime musical organisation, but instead as a series of alternative strategies which explore different aspects, and provide divergent solutions. As new versions appear, older versions continue to exist, fulfilling their specific purposes.

The system has been implemented by the first author, a composer; the second author, a specialist in multi-agent systems and artificial intelligence, has provided much needed advice, direction, and support during the last year.

Section 2 provides some background information, describing the paradigm within which *Kinetic Engine* exists and its basic goals; Section 3 briefly describes the different versions to date; Section 4 gives an artistic evaluation of each system; Section 5 posits some conclusions, while Section 6 suggests our future directions.

2. Background

2.1 Interactive computer music

Realtime computer music, in which the computer makes compositional decisions in performance and reacts to composer/performer interactions, has tended to fall within the domain of improvisatory systems. In an effort to model the "unpredictability" of improvisation, constrained random procedures have been incorporated so that the musical surface - its detail - can be both varied, yet easily controlled [4]. Formal cohesion, or large-scale structural logic offered by recapitulation and restatement, have tended to remain under direct composer/performer control.

2.2 Computational models of musical creativity

Many models exist for generative music systems, including rule-based methods [5], stochastic methods [6], data-driven methods [7], and artificial life models [8, 9, 10]. The use of any given model is perhaps more dependent upon the aesthetic choice of the artist, rather than the success of the model, which in itself may be subjective. Chadabe, for example, views his interactions with his system to be "like sailing a boat on windy seas, interacting with the wind and the waves to keep the boat on course" [11]; within such an paradigm, stochastic generation is entirely viable.

Rowe, however, suggests that "interactive (music) software simulates intelligent behaviour by modelling human hearing, understanding, and response" [12]. Not coincidentally, all these processes are also inherent in musical improvisation. The difficulty in achieving Rowe's precept is in implementing musical understanding, which implies some type of intelligence: the necessitated requirements for this software would include an instantaneous evaluation of the musical environment, an evaluation of the current environment in comparison to the past environment, as well as an evaluation of the evolution toward the desired future environment, which in itself may be constantly changing. One can understand why this integral aspect of spontaneous musical generation has been left to composer/performers, who have themselves spent years acquiring just such skills.

3. Description

Kinetic Engine began as an effort to place more high-level compositional responsibility within software, using aspects of artificial intelligence. Recognising the

complexity of accomplishing this task in a comprehensive manner, the decision was made to limit the exploration in initial versions to rhythmic development and interaction. Other musical aspects - melodic, harmonic, timbral control - have gradually been added.

3.1 Kinetic Engine v.1

Kinetic Engine derives its name from the conception of its first system: software that could generate perpetual high-level musical variations on its own; as such, it was presented as a continuously running installation. This initial version focused on the interrelationship of simple parts that would create a varying rhythmic interplay between four virtual players.

The system was based upon a hierarchical model, with four "dumb" players executing commands sent to them from an intelligent virtual conductor. Each player generated a random rhythmic pattern, of which the number of notes was determined by the conductor based upon an overall density rating that cyclically increased over time. Players performed their patterns, which were in sync to a common tempo and time signature, with their output being sent to a multi-timbral virtual synthesiser, playing percussion samples.

While the notion of repetition is integral to *Kinetic Engine*, continual variation is equally important. As such, each player was required to generate a certain degree of variations on their material, the actual amount of which would have to meet an aggregate level set by the conductor. Each player used a stochastic generator to calculate its own degree of internal variation - which including both pattern and timbral variations - and sent this amount to the conductor. The conductor would accumulate the total amount of variation that was carried out during the previous pattern, and compare it to the variation metre's level. If the actual variation score was more than the required amount, the variation metre would decrease; if the score was less than the required amount, the variation metre would increase.

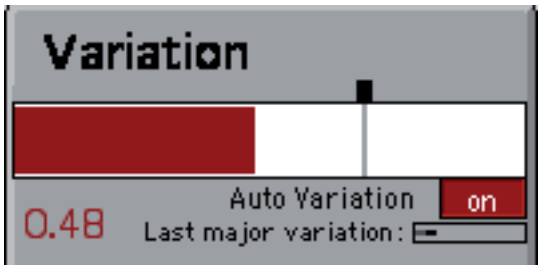

Figure 1. Variation metre, showing the actual variation (red) and required variation (grey line)

The conductor would require more and more variation as time went on, thereby slowly increasing the variation metre; players would thus be required to increase their internal variations. When the individual players could not produce enough variations, the conductor would force a new composition to begin, thus triggering a final, conclusive, variation which would entail a new tempo, time signature, a low density, and instrument changes. Each composition would have a similar overall form of gradual increase in complexity through an accumulation in density and variation.

During a composition, the conductor would monitor the variations of the individual players over the previous few measures, and determine which player had consistently increasing or decreasing variation amounts: these would be displayed in the Evaluation monitor (see Fig. 2, left). If a player was determined to have several successive levels of increasing magnitude, the player would be highlighted by having its volume increased.

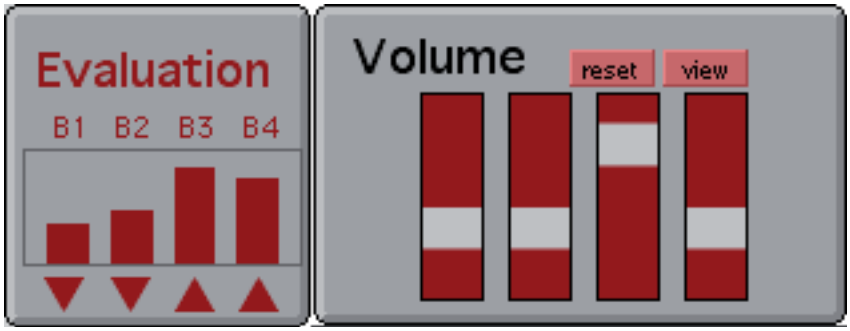

Figure 2. Evaluation of player variations, resulting in a "solo" for player 3

Lastly, specific parameters for each composition (tempo, time signature, length, and ranges for density and variation amounts) would be stored, and a fuzzy logic algorithm was used to construct predicates that ensured subsequent compositions were not "too similar" to previous ones.

Version 1 is the only version that has been retired, since the hierarchical model proved too difficult to expand and update. It's limitations will be discussed in Section 4.

A more detailed, albeit incomplete, description of version 1 is available elsewhere [2].

3.2 Kinetic Engine v.2

Version 2, like its predecessor, is restricted to rhythmic interaction between individual players; it introduces performative control over density, as well as the potential to influence the ensemble through global parameters. Version 2 introduced the notion of intelligent interaction between individual players through the use of autonomous multi-agents. Agents generate their own rhythms in response to a changing environment - primarily based upon overall density - using individual parameters that are considered the agent's personality, as well as their perceived role within the ensemble. Personality attributes include, for example, the agent's amount of social interaction, responsiveness, confidence, mischievousness, and propensity to play on the beat (vs. off the beat) and syncopate.

Once individual rhythms have been generated, agents "listen" to one another, and alter their patterns based upon these relationships. Agents determine when to begin playing (they are autonomous), as well as what to play (they are proactive); they interact with one another (they are social), as well as with their environment (they are reactive).

A great deal of an agent's actions are determined by its perceived role in the

ensemble, which, in turn, is determined by the instrument it has decided to play. Agents know that a shaker is performed differently than a bass drum, for example, and thus will not attempt to create a shaker-like pattern for it. The intelligent method for individual and overall timbral organisation is described elsewhere [13].

Version 2 is a rule-based system whose complex interactions result from multiple probability tables. Most of these tables are pre-determined; however, many are influenced by the agent's personality and the changing environment.

Version 2 is discussed in detail elsewhere [14].

3.3 Kinetic Engine v.3

Version 3 of *Kinetic Engine* attempts to balance spontaneous change with the ability to alter and adapt its rule-set. As such, version 3 offers the potential for both realtime composition through recombinance [7] as well as improvisation through generative means.

Desired tendencies for musical generation can be given to the system through specially composed MIDI files. These files are analysed - prior to performance - for patterns and tendencies (i.e. pattern, repetition, variation, and pitch content); this information is saved as an individual XML file for each input file. During performance, agents read the XML files and produce initial material that is closely related to the analysed music.

A genetic algorithm is used to generate a larger population of rhythms based upon the initial material; individuals from the population are chosen by the agent that best match the current state of the environment. Environmental variables, set by the performer, include density (number of notes per pattern) and complexity (relative amount of syncopation). Agents keep track of those individuals selected for performance: those individuals that have been heard become more likely to be culled prior to the next population generation.

The user can decide when to initiate a new generation, or the software can decide this itself. Successive populations include those individuals from the previous population that were not heard (although, due to a Gaussian selection process, there is a chance that heard individuals might live into the next population), as well as new individuals that are variations of the current population. The amount of variation between generations is constrained; for this reason, the potential for mutation, or more dramatic variation, exists.

A unique level of interaction occurs between agents during performance. Since their populations are evolved in isolation, dynamic rhythmic interaction is much more limited than in previous versions. For this reason, an attempt is made to have agents anticipate and predict other agent behaviours, and interact at the phrase, rather than pattern, level. Agents generate a future event-list, which amounts to their "intentions". The points at which they will switch individuals (patterns), and the amount of change that occurs at these points, is broadcast to the community. Agents will then attempt to alter their own intentions, so that they line up with other agents, in order to create larger cadencial phrase structures.

Version 3 is discussed in detail elsewhere [15].

3.4 Later versions

Version 4 of *Kinetic Engine* is software dedicated to a single composition: *In Equilibrio*, and introduces melodic organisation. Events are generated using a module based entirely upon the multi-agent structure of version 2: these events are then sent to six melodic agents which generate melodic phrases. The pitch and dynamic shapes of these phrases are determined by the individual agent's unique characteristics, similar to version 2's personality attributes.

Each melodic agent attempts to create organic pitch shapes, while at the same time, searching for a harmonic balance between itself and other agents: balance, in this case, is achieved when each agent has its initial phrase point at equidistant intervals from every other. However, this goal is complicated by a constantly changing pitch set, in which each pitch has a differing weight, which exerts its own "pull" on the agent.

Version 5 introduces a single agent dedicated to harmonic generation based upon a modified Markov analysis of a given musical corpus. However, unlike traditional Markov-based generation, a method is employed in which user determined feature vectors (bass line, harmonic tension, harmonic complexity) are defined, and a resultant progression is created that balances user-requested material with coherence with the database. This is an attempt to overcome the perceived weakness of Markov models at handling deeper musical structures [5].

Version 4 is discussed in detail elsewhere [16]; version 5 is discussed in detail elsewhere [17].

4. Artistic Evaluation

The initial goal of *Kinetic Engine*, and reflected in the original version, was for software to make decisions at a level higher than the musical surface. In order to accomplish this, the software had to determine a goal, and monitor the progress towards this goal. In this sense, the first version of *Kinetic Engine* was successful. The musical surface was unpredictable in its detail, due to its use of constrained randomness (through its choice of specific rhythms, timbres, and variations); however, sectional change and evolution was ensured through the gradual increases in both density and variation. The end result was a series of compositions that ranged in duration from 3 to 15 minutes: this variation in duration was due to the fuzzy logic algorithm that determined duration, as well as tempo, time signature, and timbral groupings.

Listeners who spent longer periods with version 1 would have heard several compositions of a contrasting nature; however, the overall formal structure was limited, specifically in its ability to generate unexpected formal constructions. This resulted from a trade-off that ensued from limiting the ranges in the formal scheme; in order to guarantee a certain degree of formal success, limits had to be placed on potential choices. In the short term, this guaranteed a favourable outcome, but

excluded completely novel combinations.

Version 2 explores more evolutionary approaches to form through its multi-agent design: form would develop through the interaction of the agents themselves. This proved to be a very flexible design, as different situations can use different numbers of agents. Direct performance control allows for extremely fast changes to the system in response to a live musical situation; therefore, the composer is once again making structural decisions. This is deemed an acceptable trade-off, since the complexity of the system's output requires very little realtime supervision, and is analogous to a conductor guiding a group of creative musicians (as opposed to controlling a group of dumb software players).

While the rule-set for generating rhythmic patterns is greatly improved from version 1, it remains static. This results in musical generation that, while initially engaging, remains essentially homogeneous over longer periods of time. Similar to its older brother, version 2 is designed in such a way that successful musical output is favoured over completely novel results.

Lastly, when using the system as an intelligent instrument within an improvising ensemble, a situation often arises in which the human musicians request specific output from *Kinetic Engine*, usually in the form of a specific beat. Since the system is entirely generative, this is not possible: any patterns that emerge are the result of the spontaneous reaction to the current environment, and a complex interaction between the agents themselves. As such, version 2 could be considered an improvisational system, without compositional control over its output.

Version 3 was created in an effort to include just such compositional control over pattern generation, while maintaining the complexity of multi-agent interaction. This version was extremely ambitious, particularly since it was coded in MaxMSP, a visual data-flow language not known for its programming structure or ability to handle very large programs. After the initial generation of material based upon analysis, the system was, essentially, generative; however, the use of the generative material was overseen by an algorithm that required a complete understanding of its resources (through an analysis of all generated material) via its requirement to select individuals based upon the immediate, though continually transforming, environment. As such, every agent was spending a great deal of time generating new populations, and then analysing these populations, all in realtime: the system lacked the immediate responsiveness of version 2.

These setbacks could, presumably, be overcome through standard methods of code efficiency testing. However, after over three years of being limited to rhythmic interaction, it was felt that the next version of *Kinetic Engine* should incorporate melodic organisation.

Two compositions were created using version 3 in a studio environment. The first, *Armar*, is a percussion quartet using Cuban music as a corpus. The system was run in realtime, and its output was recorded into a MIDI sequencer. Several "performances" were recorded, the best selected, and then transcribed into a notation program. *Other, Previously*, composed for guitar and cello, used a Javanese ensemble composition, *Ladrang Wilugeng*, as its corpus. *Kinetic Engine*

discriminates 12 different pitches during analysis; as such, it was possible to generate pitch-based output that had less than 12 discrete pitches (which is the case with the gamelan source material).

Melodic generation is more fully explored in version 4, albeit in ways that are not intelligent. As this system is a realtime system under performer control, all structural decisions, including harmonic change, are left to the performer. Melodic agents, while social, reactive, and pro-active, are not autonomous. From an artistic viewpoint, this system is successful, although limited; in other words, it can generate one piece, although many different versions of this piece.

An independent system exists that generates harmonic progressions based upon the analysis of a given musical corpus. Work is underway to combine this system with the performance engine of version 4; however, at this time it is unfinished.

5. Conclusions

Constrained random systems allow for the effortless generation of horizontal gestures; however, these gestures will have little interaction between them. Kinetic Engine *is based on complex interactions between its parts through the use of multi-agents. These agents explore a musical space, which can be either constrained (v.2, v.4) or more open (v.1, v.3).*

A trade-off has existed in every version between a successful artistic result and novel, unexpected outcomes. McCormack [18] discusses this problem in relation to evolutionary art in its search for interesting phenotypes within the constraints of aesthetic selection. One can view generative systems, such as Kinetic Engine*, as systems that make similar aesthetic selections, albeit within parameterised processes, rather than from a population. McCormack goes on to identify the need for artificial creativity to exist within systems, so that the system can not only generate novelty, but recognise when it has done so.*

The limitations placed upon Kinetic Engine*'s output can be considered an aesthetic decision, specifically a rejection of disorganised complexity. Complexity has been, and remains, a goal of artistic creation; interactive generative systems can therefore be seen as complex systems. Weaver [19] suggests that the complexity of a system, whether it is a piece of music or a living organism, is the degree of difficulty in predicting the properties of the system; however, he differentiates between*

disorganised complexity - *which can be analysed and produced using statistical methods - from* organised complexity - *which results from the interaction of its parts, and has the potential for emergent properties.*

Kinetic Engine *has explored organised complexity in its various incarnations; however, a successful, intelligent, and autonomous method of formal control has yet to be found.*

6. Future Directions

Current work includes adding a third software control layer to version 4, subsuming the performer's control in a way that resembles the conductor in version 1. This would return to the installation/non-performative model, but would add a layer of intelligent formal control to the software. Harmonic control would be assigned to the harmonic agent, and additional agents would be designed to monitor overall form and evolution, thus combining a top-down analysis with a bottom-up generation.

Lastly, Kinetic Engine *various versions were engendered by artistic desires. The success of each system was, for the most part, determined by its creator on rather subjective grounds. More extensive, and objective, methods of validation are currently being planned in regards to both* Kinetic Engine *and more general generative systems.*

This research was funded, in part, by a grant from the Canada Council for the Arts, and the Social Sciences and Humanities Research Council of Canada.

6. References

[1] Chadabe, J. (1977) "Some Reflections on the Nature of the Landscape within Which Computer Music Systems are Designed." *Computer Music Journal*, 1(3):5-11

[2] Eigenfeldt, A. (2006) "Kinetic Engine: Toward an Intelligent Improvising Instrument" *Proceedings of the Sound and Music Computing Conference*, Marseille pp. 97-100

[3] Eigenfeldt, A. (2007) "Computer Improvisation or Real-time Composition: A Composer's Search for Intelligent Tools" *Electroacoustic Music Studies Conference 2007*, http://www.ems-network.org/spip.php?rubrique49 Accessed 22 October 2009

[4] Chadabe, J. (1984) “Interactive Composing.” *Computer Music Journal*, 8(1):22-27

[5] Ames, C. (1989) “The Markov Process as a Compositional Model: A Survey and Tutorial” *Leonardo*, 22(2):175-187

[6] Lewis, G. (2000) “Too Many Notes: Computers, Complexity and Culture in Voyager”, *Leonardo Music Journal*, vol. 10, pp. 33-39

[7] Cope, D. (1996) *Experiments in Musical Intelligence*. Middleton, A-R Editions

[8] Miranda, E. (2003) “On the Music of Emergent Behaviour. What can Evolutionary Computation bring to the Musician?” *Leonardo*, 36(1):55-59

[9] Biles, J. (1994) “GenJam: A Genetic Algorithm for Generating Jazz Solos”, *Proceedings of the 1994 International Computer Music Conference*, Aarhus pp.131-137

[10] Blackwell, T., Bentley, P.J. (2002) “Improvised Music with Swarms”, *Proceedings of the Congress On Evolutionary Computation*, Honolulu pp. 1462-1468

[11] Chadabe, J. (2009) *Electronic Music: Unsung Revolutions of the 20th Century*. http://www.percontra.net/6music2.htm Accessed 22 October 2009

[12] Rowe, R. (1993) Interactive Music Systems, Cambridge, Mass., MIT Press

[13] Eigenfeldt, A., Pasquier, P. (2009) “Realtime Selection of Percussion Samples Through Timbral Similarity in Max/MSP”, *Proceedings of the International Computer Music Conference*, Montreal pp. 77-80

[14] Eigenfeldt, A. (2007) “Drum Circle: Intelligent Agents in Max/MSP.” *Proceedings of the International Computer Music Conference*, Copenhagen pp. 9-12

[15] Eigenfeldt, A. (2009) “The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine.” *Applications of Evolutionary Computing, EvoWorkshops 2009 Proceedings*, LNCS 5484 Springer, Berlin pp. 498-507

[16] Eigenfeldt, A. (2009) “Multiagency and Realtime Composition: *In Equilibrio*”, http://cec.concordia.ca/econtact/11_4 Accessed 22 October 2009

[17] Eigenfeldt, A., Pasquier, P. (2010) “Realtime Generation of Harmonic Progressions Using Controlled Markov Selection.” *Proceedings of ICCC-X - Computational Creativity Conference*, Lisbon

[18] McCormack, J. (2005) “Open Problems in Evolutionary Music and Art.” *EvoWorkshops 2005 Proceedings*, LNCS 3449 Springer, Berlin pp. 428-436

[19] Weaver, W. (1948) “Science and Complexity.” *American Scientist,* 36 pp. 536-544