KINETIC ENGINE: TOWARDS AN INTELLIGENT IMPROVISING INSTRUMENT

Arne Eigenfeldt
Simon Fraser University
School for the Contemporary Arts
8888 University Drive.
Burnaby, BC.
CANADA
arne_e@sfu.ca

ABSTRACT

Kinetic Engine is an initial attempt to create an intelligent improvising instrument. An interactive performance system that models a drum ensemble, it is comprised of four multi-agents (players) that collaborate to create complex rhythms. Each agent assumes a role, and makes decisions according to this understanding. Furthermore, the four players are under the coordination of a software conductor that operates as the "eyes and ears" of the ensemble.

1. INTRODUCTION

Recent technological advances, in both software and hardware, have made the development of interactive performance systems increasingly powerful. Such systems, which are often modeled upon improvised music, can be considered as complex instruments, generating multiple gestures that proceed and interact in complex ways. While interactive systems already have a considerable long history [2], these new developments allow for new models of interaction to be developed, particularly in the level of interaction between human and computer. The possibility of an intelligent software instrument is no longer far-fetched, and the first steps towards such a goal are outlined in this paper.

1.1. Context

Twenty years ago, researchers and composers interested in live performance, computer interaction and improvisation were forced to use either homemade hardware, or adapt to the limitations of commercial hardware and MIDI [12]. The result, while creating the potential for a stimulating performance, rarely could compete with tape music in terms of sonic complexity. Therefore, researchers concentrated upon a *complexity of interaction* [9].

Toward the end of the nineties [13], affordable hardware became fast enough to allow for live signal generation and processing. Since that time, composers interested in sonic complexity, formerly limited to studio composition, could also explore live performance; laptop performers are now common at new music concerts, and live signal processing is mainstream. Ironically, the innovation which was the hallmark of early interactive music has been lost, apparently to a fascination with adding processing to prerecorded material, or recording live material and later playing it back processed, a procedure relatively easy to accomplish with programs such as SuperCollider and Max/MSP.

While one could argue that, like tape music thirty years earlier, live computer music has come of age - composers can concentrate on creating good music, rather than making excuses about technological limits - live computer music has, to a large part, become conservative. I propose that we can use the increased power of the technology to further explore the complexity of interaction first attempted in the 1980s, but applying new models which were not possible at that time.

1.2. The Use Of Controlled Random Procedures

Winkler notes the relationship between interactive computer music and existing models of musical improvisation, and the fact that many processes in interactive music tend to model activities found in improvisation, namely listening, interpreting, composing, and performing [12]. This paper will focus upon the latter two situations: live composition and its performance. In such systems, compositional algorithms generate musical material in which musical parameters can be altered by a performer (i.e. the computer operator) or by constrained random procedures, both of which allow for the shaping and influencing of musical output [12]. The use of such random procedures allows music to be unpredictable, yet remain within a defined range that determines a coherent musical style [12]. As Truax suggests, complexity in musical systems is a direct correlation to its unpredictability; in this respect, music that is randomly generated (or influenced) is, by definition, complex [10].

2. THE LIMITS OF RANDOMNESS

The use of constrained random procedures in interactive systems can produce initially interesting music. Through the use of parameter variation, it is easy to create software in which the various musical elements (pitch, rhythm, duration, amplitude, timbre) can be randomly generated within strict limits to produce music that has a certain degree of freshness. The random procedures provide the unpredictability (and thus the complexity), while balanced by limitations (for example, only choosing pitches from predetermined scales) that provide cohesiveness and repetition and thus the avoidance of anarchy.

It can be said that such systems are modeled after certain forms of improvised music, with event-to-event decisions being left to random procedures rather than human "inspiration". However, as is the case with most improvised music, the notion of form requires large-scale structural change that balances this immediacy. Algorithmic music can only appear intelligent when it exhibits such direction and macro-level control. As Lewis suggests, within his *Voyager* program, "tendencies over a long period of time exhibit consistency, (while) moment-to-moment choices can shift unpredictably" [6]. The success of the music created with Lewis' *Voyager* can be credited the looseness of his model: the arrhythmic, atonal music of free jazz affords easier acceptability of many event choices.

The creator of computer improvisation systems can thus choose to not only relinquish control over low level, event-to-event decisions, but many mid-level choices as well, as Lewis does. Contrarily, one can attempt to maintain control via human computer interfaces, as Winkler suggests [12]. In early interactive systems, such control was possible due to the limited number of gestures capable by the technology [2]. While a great deal of research continues in this aspect of live computer performance, this paper outlines the attempts to bring control from within the system itself.

3. THE NEED FOR NEW MODELS

Rowe suggests "interactive software simulates intelligent behavior by modeling human hearing, understanding, and response [9]. In this respect, more consideration must be given to understanding of the totality of the music produced at any given time by the system.

A good deal of research has been undertaken to investigate the application artificial intelligence into music [7, 11]. Other research has focused upon genetic algorithms [5], neural nets [8], and expert systems [3] in order to explore new methods of high-level control. It must be pointed out that the research describing herein comes not from a computer scientist, nor a cognitive scientist; instead, it results from a composer who is searching for new methods of software control in interactive systems. As such, the pursuit of intelligent performance is not abstract, but rather firmly centered within behavior-based AI.

In order to simplify the problem, the initial research described here has been limited to rhythmic systems and the interaction of rhythmic layers. Music that relies upon such qualities has been used as loose models, and observations about their behaviors have been made. At this point, these models have been limited to African drum ensembles (specifically the music of the Ewe in Ghana) and techno, as well as certain elements of pre-1960s jazz.

3.1. Disclaimer on the use of models

The use of one or more models in this research precludes any attempt to replicate the models themselves; no attempt is made to pass a Turing test when comparing its output to the model. Nor is it a musicological study, which attempts to discover new or obscured features and/or defining characteristics within the music, or potential relationships between the models. Furthermore, the analysis of the models is in no way meant to suggest that these methods are used in their creation. Instead, the analysis of any model is purely functional, without concern for the underlying mechanism.

3.2. Modelling Intelligent Behaviour

In user driven interactive systems, in which a performer controls many of the large scale decisions made by the computer, the role of the performer is to provide "intelligence". As Salvatore Martirano suggested, "the computer isn't intelligent, but I am" [Salvatore Martirano, personal communication, 1987]. If we are to surrender high level control to the computer, some limits must be placed upon possibilities.

Although this suggests a move towards determinism, it can also be related to roles that musicians assume within ensembles. As Winkler suggests, the types of limitations placed upon performance choices determine the style of music [12]. Thus, limiting potential choices does not limit the intelligence of a system any more than does the performer who limits his/her choices to those appropriate to a style. Furthermore, such limitations parallel compositional organization in certain improvised music - for example, African drum music – in which variation is assumed on pre-composed patterns to varying degrees. In such music, each performer has a particular role to play within the overall music, defined by their instrumental part.

Similarly, in bebop or related jazz styles, each part (drums, bass, piano, soloist) has a degree of freedom, but must fulfill its role; players are required to work towards a communal goal while listening and responding to one another. The exact pattern played by the drummer, for example, may be unpredictable, but it must be appropriate for the particular instant. Individual choices are limited to varying degrees depending upon the exact style (note choices are more limited in swing music than in bebop, for example), whereas larger formal designs are determined by the song form.

Techno displays a similar stratification within its layers based upon its parts. The bass drum is expected to provide the simplest pulse, while the snare interacts with the bass creating a homogeneous rhythmic layer. Other parts, often given to other percussion sounds and cymbals, exhibit a greater density and complexity. These parts vary more often, and essentially providing foreground material with the greatest degree of change.

In all three models, different levels of consistency are required within each part. Small variations will occur in the background parts, but a large degree of repetition is required as well. Foreground parts display the greatest degree of variation. In the case of the Ewe music, the master drummer can continually change his patterns, dropping out entirely; it is not until every drum changes its pattern that we perceive a large scale formal variation.

Two of these models are forms of ensemble improvisation that balance improvisation with pre-determined musical content, while techno offers a related paradigm. Of particular note in all cases are the different limits placed upon the various parts. In using these models to derive intelligent behavior, several observations and assumptions can be made:

- Each performer understands the directed goal, and its role in achieving it;
- The parts are not progressing independently[1], but are fully aware of their role in relation to others. A certain degree of interaction is expected and required;
- Different parts react differently to the need for dynamic variation.

4. KINETIC ENGINE

In designing Kinetic Engine, the goal was to create a system that involved minimal performer (user) input - fewer actual controls, each of which controlled extremely high level processes. At the same time, the system had to be evolutionary and generative, one that would not rely upon performer input to afford dynamic change. If left alone, it would begin to generate its own musical variations, both at the lower and higher levels. At any point, however, a performer could nudge the system, or push it dramatically.

Kinetic Engine was written in Max/MSP[4].

4.1. Players

The basis of Kinetic Engine is a *Player*, a software agent defined in terms of its behavior. Each of the four Players performs one rhythmical pattern, and has the ability to vary its performance details (panning, dynamics, processing) in a constrained random fashion within predetermined ranges. It also has the higher ability to vary its patterns in a musical fashion.

On its own, the Player has no intelligence, nor does it have an ability to listen or react to the other players – interaction is controlled in a top-down fashion through the use of a virtual conductor. The Player's performance variations are only meant to maintain interest for several measures at a time; as such, the Player can be viewed as a drum machine with random parameter variation. However, higher-level organization gives each Player a particular role within the overall music, considered its *type*. In relation to African drum music, these types can parallel the different drums (the support drums of *sogo*, *kidi*; the double bell *gankogui*; the rattle *axatse*); in techno, these would be the different parts (bass drum, snare, other percussion, cymbals).

4.1.1. Patterns

Earlier versions of the software had each Player performing predetermined patterns; currently, patterns are constructed using knowledge based algorithms. Patterns are dependent upon a Player's type: bass drum patterns, considered type one, are different from cymbal patterns, considered type four. Furthermore, they are generated in relationship to another type - type one and two are paired, type three and four are paired.

The patterns themselves are assembled through the accumulation of individual beats; the combination of different subdivisions are predetermined through weightings which are dependent upon the Player's type. Thus, a generated pattern for a type one Player will have more internal repetition and consistency than a type four Player, and a lower density (fewer syncopated sixteenths, for example, if that is the subdivision).

4.2. Performance Controls: Density and Variation

The two performance controls are limited to overall *density* and *variation amount*. Players react to these global variables in different ways, depending upon their type. For example, type one is usually less dense (has fewer notes) than type four, and type one reacts less often to variation requests than type four.

4.2.1. Density

An initial density is set, and each part calculates its own contribution to this value. When density is changed, each player adjusts its own density by adding or subtracting notes from its pattern, or generating a new pattern entirely. Players of a higher type will react first to the need for increased density, resulting in variations in density occurring mainly in the more foreground layers.

Changes in density do not occur instantaneously; instead, an initial attempt is made by each part to match the new density at the beginning of each measure. The resulting density is compared to the requested density, and an adjustment is made if required - this process can go on for several measures. Due to the independence of each players adjustments, density changes are unpredictable, paralleling Lewis' goal of avoiding "the kind of uniformity where the same kind of input routinely leads to the same result" [6].

Density change is considered a large scale variation – discussed shortly – and can occur in reaction to the need for variation. In such cases, a much lower density is set, and a new target density chosen – over

[1] A comparison can be made to collective goals of heterophonic music as opposed to the independence of polyphony.

several measures (the duration of which is a constrained random variable within a given section) the density slowly increases to this new level. since the Density level is dynamic, each Player attempts to match its own density to this changing variable, resulting in a complex interaction between Players.

4.2.2. Variation

The need for variation is assumed to maintain interest. Small variations may hold the listeners attention for a few measures, but then greater and greater variations are required in order to maintain this interest. Thus, a *variation meter* is used which gradually increases over time. At recurring time intervals – such as every one to eight measures, determined by a variable for the section – this meter is tested, and various variations are initiated by the players. Each variation has a scaled value, depending upon its perceived interest. For example, tuning range, processing range, and delay line range are considered small variations, while pattern variation and timbre/instrument change is considered a medium variation, and density change is considered a large variation. The total amount of variation is summed, and subtracted from the variation meter, causing it to be lowered. Since the actual choice of variation for each player involves complex weightings and random constraints, that summed variation value may be greater, or less, than the variation meter; this results in the meter returning to zero, or close to it.

Because small variations are not considered to maintain interest for very long, the summed variation amount begins to be scaled over time; in other words, it loses its effect, and the variation meter begins to increase. This increase then requires larger variations to occur in order to keep it down; finally, a major variations (density change coupled with new instrument selection) will start the process all over again.

4.3. Sections: large scale changes

The musical result of increasing variations is a gradual increase in complexity, culminating in a major change. The above described pattern generates complex interactions that maintain interest for long periods of time; however, large scale formal change does not occur since there is an overall consistency within a given section.

For this reason, the notion of sections was created. Within a given section, a range of probabilities is predetermined: the group of instrumental sounds; the probability of delays and other processing being added to the Players; the range of densities and the range of time it takes to move between them; how often variation occurs; how quickly the variation meter progresses, and the effectiveness of the variations on keeping it down; and the maximum number of players active at once. Of note is the fact that each of these probabilities is dynamic throughout the section; for example, the section may begin with only two maximum players, but progress to four during the course of the section.

Each section has an approximate duration in seconds. Once the duration is complete, the effectiveness of variations is reduced to zero, forcing the variation meter to increase unabated. The next major variation causes a complete change to the next predetermined section.

4.4. Performance Modes

Kinetic Engine can operate in Performance mode, in which a performer can control the variation and density meters to initiate immediate large scale changes. Once the performance input ceases, the program returns to Autonomous mode. Performance mode also allows for section changes to be initiated, or blocked.

Lastly, Performance mode allows for a “freeze” command, essentially a temporary off switch for the variation meter. This allows the performer to pause the evolution of the system if a particularly interesting rhythmic moment has evolved.

4.5. Future Directions

At present, Kinetic Engine is a top-down hierarchical system. This is more of an aesthetic decision on my part, rather than a functional one, since I am a composer/conductor first, rather than a performer. Perhaps this system represents my ideal musical situation, in which creative performers slavishly follow a conductor’s every whim.

Kinetic Engine could be rewritten so that each Player is an autonomous agent, and decisions are made at the Player level. This would no doubt lead to greater independence, and perhaps more closely model a human improvising ensemble. However, it would also create very different music; for example, it would then fall into the improvised music trap on an inability to quickly and dramatically change musical direction.

One limitation of Kinetic Engine is an immediate understanding of its timbral spectrum. The choice of instruments for each Player is currently based upon a predetermined grouping; i.e. tabla, 4Bit techno, West African drum ensemble, etc. This guarantees a certain consistency, but also limits potential combinations. Greater freedom of instrumental choice could involve an analysis of currently playing instruments using FFTs, thereby determining the existing spectrum, and deciding which available instruments could best complement this spectrum.

Lastly, Kinetic Engine lacks any ability to learn from the music that it creates. In comparing Kinetic Engine to other behavior-based AI programs – such as those involving populations of robots – the difficulty of evaluating its own success is apparent. Most learning-based systems require this evaluation and reinforcement for future applications; in other words, if the system did something well, it should continue to do that in the future. Success, in the case of Kinetic Engine, is the creation of interesting music. Since there is no algorithm

that can make such a determination, human evaluation is required. Such training is possible, and potentially involving the use of neural nets and/or genetic algorithms.

Music generated by Kinetic Engine can be found here:

http://www.sfu.ca/~eigenfel/research.html

5. ACKNOWLEDGEMENTS

I would like to thank Dr. Richard Vaughan of Simon Fraser University's Autonomy Lab and Computing Science Department, for help with concepts related to Artificial Intelligence, and advise on system design in both current and future implementations.

6. REFERENCES

[1] Beyls, P. "Self-Organizing Control Structures using Multiple Cellular Automata", *Proceedings of the 1991 International Computer Music Conference*, (1991), p. 254-257.

[2] Chadabe, J. "Interactive Composing", *Computer Music Journal*, 8 1, MIT Press, Cambridge, 1984.

[3] Cope, David "Facing the music: perspectives on machine-composed music", *Leonardo Music Journal*, 9, (1998), p.79.

[4] Dobrian, C. *MSP, The Documentation*, Cycling74, 1998.

[5] Horner, A and Goldberg, D.E. "Genetic Algorithms and Computer-Assisted Composition", *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[6] Lewis, George E. "Too Many Notes: Computers, Complexity and Culture in Voyager", *Leonardo Music Journal*, 10, (2000), p. 33-39.

[7] Miranda, Eduardo Reck. *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers, Paris, France, 2000.

[8] Mozer, M.C. "Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing", *Connection Science*, 6 2-3, (1994), p.247–280.

[9] Rowe, Robert. *Interactive Music Systems*, Cambridge, Mass., MIT Press, 1992.

[10] Truax, B. (1994) "The inner and outer complexity of music", *Perspectives of New Music*, 32 1, (1994), p. 176-193.

[11] Widmer, G. (1992) "Qualitative Perception Modeling and Intelligent Musical Learning," *Computer Music Journal*, 16 2, (1992), p.51–68.

[12] Winkler, Todd. "Strategies for Interaction: Computer Music, Performance, and Multimedia" *Proceedings of the 1995 Connecticut College Symposium on Arts and Technology*, 1995.

[13] Zicarelli, David. "MSP Learns to Ride a Bike" http://www.cycling74.com/story/2005/12/27/181050/34