THE CREATION OF EVOLUTIONARY RHYTHMS WITHIN A MULTI-AGENT NETWORKED DRUM ENSEMBLE

Arne Eigenfeldt
School for the Contemporary Arts
Simon Fraser University
Burnaby, BC
CANADA

ABSTRACT

This paper presents a multiple-agent architecture that generates polyphonic rhythmic patterns which continuously evolve and develop in a musically intelligent manner. Agent-based software offers a new method for real-time composition that allows for complex interactions between individual voices while requiring very little user interaction or supervision. The system described, *Kinetic Engine* is an environment in which networked computers, using individual software agents, emulate drummers improvising within a percussion ensemble. Player agents assume roles and personalities within the ensemble, and communicate with one another to create complex rhythmic interactions.

1. INTRODUCTION

The promise of agent-based composition in musical real-time interactive systems has already been suggested [17, 13], specifically in their potential for emulating human-performer interaction. Agents have been defined as autonomous, social, reactive, and proactive [16], similar attributes required of performers in improvisation ensembles.

Kinetic Engine [6], created in Max/MSP, arose out of a desire to move away from constrained random choices within real-time interactive software, and utilize more musically intelligent decision-making processes. Agents are used to create complex, polyphonic rhythms that evolve over time, similar to how actual drummers might improvise in response to one another. A conductor agent loosely coordinates the player agents, and manages the high-level performance parameters, specifically *density*: how many notes are being played by all agents.

The software is written by a composer with compositional, rather than research, objectives, the first stage in a long-term investigation of encoding musical knowledge in software. As such, the encoded knowledge is my own; my experience as a composer suggests that I have some knowledge as to what determines interesting music, so I am relying upon that knowledge. No attempt has been made to create a comprehensive compositional system that can reproduce specific styles or genres; the system is rule-based, rather than data-driven, and the rules and logic within *Kinetic Engine* are derived from auto-ethnographic examination.

This paper will describe the implementation of multi-agents in *Kinetic Engine*. Section 2 gives an overview of existing research into multi-agent systems and rhythm generation. Section 3 describes the specific implementation of agents. Section 4 describes how agents activate themselves. Section 5 describes the social behaviour of agents. Section 6 describes how agents learn and evolve. Section 7 offers conclusions and future directions.

2. OVERVIEW OF EXISTING RESEARCH

2.1. Multi-agent systems

Multiple-agent architectures have been used to track beats within acoustic signals [9, 5] in which agents operate in parallel to explore alternative solutions. Agents have also been used in real-time composition: Burtner [3] created a multi-agent, multi-performer system; Dahlstedt and McBurney [4] developed a multi-agent model based upon Dahlstedt's reflections on his own compositional processes; Wulfhurst et.al. [17] created a multi-agent system where software agents employ beat-tracking algorithms to match their pulse to that of human performers.

Many of these systems incorporate improvisatory elements. As already noted, agents seem to suggest the same sorts of specifications required of human improvisers. Benson [1] suggests that there are many shades of improvisation in music, ranging from standard performance – in which musicians fill in certain details which are not specified by the score – to complete melodic and harmonic freedom; as such, the role agents could play in such works is widely varying.

Murray-Rust and Smaill [13] create a theory of Musical Acts, an expansion of Speech Act Theory, to describe the actions of musicians (represented as agents) engaged in improvisatory ensemble playing. However, they are interested in creating a system that will "enable a wider range of people to create music," provide a "new approach to musical composition," and facilitate "the interaction of geographically diverse musicians," none of which are motivating forces behind *Kinetic Engine*.

2.2. Rhythm Generation

The generation of rhythm through software processes has been explored through a variety of methods, including genetic algorithms [10], cellular automata [2], neural networks [11] and multi-agents [8]. Brown [4] suggests that CA provides "a great deal of complexity and interest from quite a simple initial setup"; while this may be the case, he also comments that his generated rhythms "often result in a lack of pulse or metre. While this might be intellectually fascinating, it is only occasionally successful from the perspective of a common aesthetic." He concludes that musical knowledge is required within the rule representation system in order for the system to be *musically* successful.

Gimenes [8] explores a memetic approach that creates stylistic learning methods for rhythm generation. *RGeme* "generates rhythm streams and serves as a tool to observe how different rhythm styles can originate and evolve in an artificial society of software agents." Using an algorithm devised by Martins et.al. [11] for comparing similar rhythms, agents choose rhythmic memes from existing compositions and generate new streams. The highest scoring memes, however, proved to be of questionable rhythmic interest.[1]

Pachet [14] proposes an evolutionary approach for modelling musical rhythm. Agents are given an initial rhythm and a set of transformation rules from a shared rule library; the resulting rhythm is "the result of ongoing play between these co-evolving agents." The agents do not actually communicate, and the rules are extremely simple: i.e. add a random note, remove a random note, move a random note. The system is more of a proof of concept than a performance tool; seemingly, it developed into the much more powerful *Continuator* [15], which is a real-time stylistic analyzer and variation generator.

Finally, Miranda [12] describes an unnamed rhythm generator in which agents produce rhythms that are played back and forth between agents. Successful rhythms (those that are played back correctly) are stored, and unsuccessful ones are eventually deleted, while rhythms that are too close to each other are merged by means of a quantiser mechanism. A repertoire of rhythms eventually emerges, which Miranda suggests is a cultural agreement between agents. This suggests an interesting possibility for evaluating rhythms outside of a database.

3. AGENTS IN KINETIC ENGINE

For a detailed explanation of *Kinetic Engine* and its use in the performance *Drum Circle*, as well as its implementation in Max/MSP, see [7].

Agent-based systems allow for little user interaction or supervision. While this may seem like a limitation to some readers, this forces more higher-level decisions to be made in software. As such, this models interactions between intelligent improvising musicians, with a conductor shaping and influencing the music, rather than specifying what each musician/agent plays. This is, in fact, the model within *Kinetic Engine*.

Kinetic Engine runs as a distributed network, in which each computer operates as a separate agent, or internally within a single computer. In the latter case, the number of agents is limited due to processing requirements. There are two agent classes: a conductor and an indefinite number of players.

3.1. The Conductor Agent

The conductor agent (hereafter simply referred to as "the conductor") has three main functions: firstly, to handle user interaction; secondly, to manage (some) high-level organization; thirdly, to send a global pulse.

Kinetic Engine is essentially a generative system, and user interaction is limited to controlling *density* – the relative number of notes played by all agents. This value can be set directly via a graphic slider or an external controller[2]. The user can also influence the system by scaling agent parameters (see section 3.2).

Metre, tempo, and subdivision are set prior to playing by the conductor; these values remain constant for the duration of a *composition*. The user can force a new composition, which involves new choices for these values. Each of these values is dependent upon previous choices using methods of fuzzy logic; for example, if the first tempo was 120 BPM, the next cannot be 116, 120, or 126 (which would be deemed to be "too close" to be considered new). If the second tempo was "close" to the previous (i.e. 108/112 or 132/138), then the *next* tempo would have to be significantly different.

The conductor also manages the initialization routine, in which agents register and are assigned unique IDs. A more truly evolutionary model eventually could be used, in which agents are created and destroyed during the performance, modeling the notion of musicians entering and leaving the ensemble.

The conductor also sends a global pulse, to which all player agents synchronize.

3.2. The Player Agents

Upon initialization, player agents (hereafter referred to simply as "agents") read a file from disk that determines several important aspects about their behaviour; namely their *type* and their *personality*.

Type can be loosely associated with the type of instrument an agent plays, and the role such an instrument would have within the ensemble. See Table 1 for a description of how type influences behavior.

	Type *Low*	Type *Mid*	Type *High*
Timbre	low frequency: • bass drums	midrange frequency: • most drums	high frequency: • rattles, • shakers, • cymbals
Density	lower than average	average	higher than average
Variation	less often	average	more often

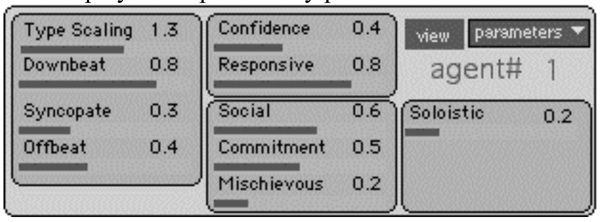
Table 1. Agent *types* and their influence upon agent behaviour.

Agents also assume personality traits. These parameters include *Downbeat* (preference given to notes on the first beat), *Offbeat* (propensity for playing off the

[1] The two highest scoring memes were [11111111] and [01111111], where 1 is a note, and 0 a rest, in a constant rhythm (i.e. one measure of eighth notes).

[2] In the case of *Drum Circle*, this was a data-glove.

beat), *Syncopation* (at the subdivision level), *Confidence* (number of notes with which to enter), *Responsiveness* (how responsive an agent is to global parameter changes), *Social* (how willing an agent is to interact with other agents), *Commitment* (how long an agent will engage in a social interaction), and *Mischievous* (how willing an agent is to upset a stable system). A further personality trait is *Type-scaling*, which allows for agents to be less restricted to their specific types[3]. See figure 1 for a display of all personality parameters.



Type Scaling 1.3
Downbeat 0.8
Syncopate 0.3
Offbeat 0.4
Confidence 0.4
Responsive 0.8
Social 0.6
Commitment 0.5
Mischievous 0.2
Soloistic 0.2

Fig. 1. Personality parameters for a player agent.

4. AGENT ACTIVATION

A performance begins once the conductor starts "beating time" by sending out pulses on each beat. Agents independently decide when to activate themselves by using fuzzy logic to "wait a bit". This is one attempt to accommodate "human time" within the system. It would be possible, for example, to calculate complex interactions between agents within milliseconds; however, such interactions take many measures to evolve between humans. As such, agents independently evaluate, and re-evaluate, their surroundings (in this case, whether to activate or not) every few beats.

Agents determine whether to activate by testing their own responsiveness parameter: less responsive agents will take longer to react to the conductor's demands. When an agent becomes active, it calculates its own density.

4.1. Fuzzy Logic

As well as time, *Kinetic Engine* also attempts to model human approximation of values, specifically to judge success. In the case of density, agents are unaware of the exact global density required (an value between 0.0 and 1.0). Instead, the conductor uses fuzzy logic to rate the global density as "very low", "low", "medium", or "high". Imagine a conductor holding her hand out in front of her - low being closer to the ground – and raising and lowering the hand. Musicians would tend to break the many slight variations of hand position into fewer general positions[4].

Agents know the average number of notes in a pattern based upon this rating, which is scaled by the agent's type and type-scaling parameter. Agents generate individual densities after applying a Gaussian-type curve to this number, and broadcast their density.

The conductor collects all agent densities, and determines whether the accumulated densities are "way too low/high", "too low/high", or "close enough" in comparison to the global density, and broadcasts this success rating.

• if the accumulated density is "way too low", non-active agents can activate themselves and generate new densities (or conversely, active agents can deactivate if the density is "way to high").

• if the accumulated density is "too low", active agents can add notes (or subtract them if the density is "too high").

• if the accumulated density is judged to be "close enough", agent densities are considered stable.

5. SOCIAL BEHAVIOUR

Once all agents have achieved a stable density and have generated rhythmic patterns based upon this density[5], agents can begin social interactions.

Social interaction emulates how musicians within an improvising ensemble listen to one another, make eye contact, then interact by adjusting and altering their own rhythmic pattern in various ways. In order to determine which agent to interact with, agents evaluate[6] other agent's *density spreads* - an agent's density distributed over the number of beats available, given the composition's metre.

An agent generates a *similarity* and *dissimilarity* rating between itself and every other active agent. The highest overall rating will determine the type of interaction: a dissimilarity rating results in rhythmic polyphony (interlocking), while a similarity rating results in rhythmic heterophony (expansion).

Once another agent has been selected for social interaction, the agent attempts to "make eye contact" by messaging that agent. If the other agent does not acknowledge the message (its own social parameter may not be very high), the social bond fails, and the agent will look for other agents with which to interact.

5.1. Interaction types

In polyphonic interaction, agents attempt to "avoid" partner notes, both at the beat and pattern level: both agents attempt to move their notes to where their partner's rests occur.

In heterophonic interaction, agents alter their own density spread to more closely resemble that of their partner, but no attempt to made to match the actual note patterns. See [7] for a detailed discussion of interactions.

[3] For example, low agents will tend to have lower densities than other types, but a low agent with a high type-scaling will have higher than usual densities for its type.

[4] This is not based upon cognitive models, at least not on any other than my own.

[5] For a detailed description of how an agent's density is translated into rhythmic patterns, see [7].

[6] Evaluation methods include comparing density spread averages and weighted means, both of which are fuzzy tests.

6. EVOLUTION OF AGENTS

Agents adapt and evolve their personalities over several performances, and within the performance itself. After each composition (within the performance), agents evaluate their operation in comparison to their personality parameters. For example, an agent that was particularly active (which relates to both the responsiveness and confidence parameters) during one composition, might decide to "take a rest" for the next composition by temporarily lowering these parameters.

Agents also judge their accumulated behaviours over all compositions in a performance in relation to their preferred behaviour (as initially read from disk), and make adjustments in an attempt to "average out" to the latter. At the end of the performance (of several compositions), the user can decide whether to evolve from that performance. Comparing the original parameter with the final accumulated history, an exponential probability curve is generated between the two values, and a new personality parameter – close to the original, but influenced by the past performance – is chosen and written to disk, to be used next performance.

7. CONCLUSION AND FUTURE WORK

This paper presented methods of using multi-agents to create complex polyphonic rhythmic interactions that evolve in unpredictable, yet musically intelligent ways. The software has already been premiered, generating music that can be described as displaying emergent properties.

There are several planned strategies for improving the machine musicianship of *Kinetic Engine*, including the use of a dynamic rule base to avoid a homogeneity of rhythms, the ability for agents to become soloists, the ability to incorporate predefined (scored) ideas, and the ability to interact with human performers.

Example music created by *Kinetic Engine* is available at www.sfu.ca/~eigenfel/research.html.

8. REFERENCES

[1] Benson, B. E. *The Improvisation of Musical Dialogue*, Cambridge University Press, 2003.

[2] Brown, A. "Exploring Rhythmic Automata" *Applications On Evolutionary Computing*, Vol.3449, pp. 551-556, 2005.

[3] Burtner, M. "Perturbation Techniques for Multi-Agent and Multi-Performer Interactive Musical Interfaces" *NIME 2006*, Paris, France, 2006.

[4] Dahlstedt, P., McBurney, P. "Musical agents" *Leonardo*, 39 (5): 469-470, 2006.

[5] Dixon, S. "A lightweight multi-agent musical beat tracking system" in *Pacific Rim International Conference on Artificial Intelligence*, pp. 778–788, 2000.

[6] Eigenfeldt, A. "Kinetic Engine: Toward an Intelligent Improvising Instrument" *Proceedings of the 2006 Sound and Music Computing Conference*, Marseilles, France, 2006.

[7] Eigenfeldt, A. "Drum Circle: Intelligent Agents in Max/MSP", *Proceedings of the International Computer Music Conference,* Copenhagen, Denmark, 2007.

[8] Gimenes, M., Miranda, E.R., Johnson, C. "Towards an intelligent rhythmic generator based on given examples: a memetic approach" *Digital Music Research Network Summer Conference*, 2005.

[9] Goto, M., Muraoka, Y. "Beat Tracking based on Multiple-agent Architecture - A Real-time Beat Tracking System for Audio Signals" *Proceedings of The Second International Conference on Multi-agent Systems*, pp.103–110, 1996.

[10] Horowitz, D. "Generating rhythms with genetic algorithms" *Proceedings of the International Computer Music Conference*, Aarhus Denmark, 1994.

[11] Martins, J., Miranda, E.R. "A Connectionist Architecture for the Evolution of Rhythms" http://cmr.soc.plymouth.ac.uk/publications/Evomusart06_Joao.pdf , 2006.

[12] Miranda, E.R. "On the Music of Emergent Behaviour. What can Evolutionary Computation bring to the Musician?" *Leonardo*, v.6 n.1, 2003.

[13] Murray-Rust, D., Smaill, A. "MAMA: An architecture for interactive musical agents" *Frontiers in Artificial Intelligence and Applications* Volume 141, 2006 ECAI 2006 - 17th European Conference on Artificial Intelligence, 2006.

[14] Pachet, F. "Rhythms as emerging structures" *Proceedings of the 2000 International Computer Music Conference*, Berlin, ICMA, 2000.

[15] Pachet, F. "The Continuator: Musical Interaction With Style" *Journal of New Music Research*, v.32, I.3, pp. 333-341, 2003.

[16] Woolridge, M., Jennings, N. R., "Intelligent agents: theory and practice" *Knowledge Engineering Review*, 10, 2, 115-152, 1995.

[17] Wulfhorst, R.D., Flores, L.V., Flores, L.N., Alvares, L.O., Vicari, R.M. "A multi-agent approach for musical interactive systems" *Proceedings of the second international joint conference on Autonomous agents and multi-agent systems*, pp. 584–591, 2003.