Multi-agency and Realtime Composition: *In Equilibrio*

by Arne Eigenfeldt

Abstract

Live electronics has a history that dates back to as early as the 1920s. Electroacoustic instruments, such as the Theremin, the Ondes Martenot, the Trautonium, and even the Hammond organ, date from this decade. John Cage's use of performative actions on variable speed phonographs in his Imaginary Landscape #1 of 1939 is another early landmark. However, it was the 1970s, and the work of David Behrman, Salvartore Martirano, and Joel Chadabe, that brought forth the possibilities of interactive music when they created systems that could participate in the compositional process by making musical decisions. The composer/performer could then react to these decisions, thereby making the entire process interactive.

The evolution of what Chadabe termed "interactive composition" is "realtime composition" : the ability to deliberately compose material in performance (as opposed to improvisation) with the help of software. This has become possible through the use of methods found in artificial intelligence, one of these being multi-agency. Aspects of multi-agents, and their application in musical performance, will be introduced in this presentation, specifically in the context of the author's own research project, *Kinetic Engine*.

1. Introduction

For the last twenty-odd years, I have been involved in creating software for realtime performance. However, unlike many of those involved in live electroacoustics, my interests are not as a performer who wishes to expand or extend their instrument, but rather as a composer. This may seem a little odd, since live electronics has, for the most part, been concerned with improvisation.

2. Real-time Composition vs. Improvisation

In a paper published in *eContact's* issue on live electronics / Improvisation / Interactivity, I outlined what I consider to be the differences between improvisation and what I call real-time composition. The first point that I make is that I consider composition to be the deliberated ordering of events prior to performance, and it is this deliberation that allows for both the horizontal *and* vertical complexity of Western Art Music. Improvisation, while offering other types of complexities, cannot achieve the same level of horizontal and vertical complexity for long periods of time, simply due to constraints on human memory, and limits on the interaction between players.

Another important point I make is the difference between *organized complexity* vs. *disorganized complexity*, using the definition posited by Weaver in his seminal article "Science and Complexity" from 1948. 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.

Thus, John Cage's aleatoric procedures, or a granular synthesis texture, is extremely complex due to our inability to predict individual events. Weaver considers this to be *disorganized complexity,* which can be understood (as well as generated) by statistical models.

Organized complexity, on the other hand, achieves its complexity through the interaction of its parts, and has the potential to have *emergent properties.* Thus, the type of complexity offered by Western Art Music is organized, whereas random methods create disorganized complexity.

The specific type of music that I am interested in, sometimes called interactive computer music, has very often been concerned with improvisational models, and used constrained random procedures to generate its complexity. My goal is to achieve the organized complexity of deliberated composition, but during performance: thus, realtime composition. This may seem impossible, given the definitions I have just cited, and the physical and mental limitations of humans. I believe, however, through the codification of musical knowledge, that such a system is possible in software.

2.1 Interactive Composing

Joel Chadabe was one of the first composers to purse the notion of live composition through what he considered "interactive composing". This involved "a mutually influential relationship between performer and instrument", a process that he has likened to sailing a boat on a stormy lake, reacting to the immediate environmental changes.

"the instrument is programmed to generate unpredictable information to which the performer reacts during performance. At the same time, the performer at least partially controls the instrument. Since the instrument influences the performer while, at the same time, the performer "influences" the instrument, their relationship is mutually influential and, consequently, interactive."

Chadabe considers this compositional, rather than improvisational, performance, since he does not have complete control over his instrument. Although he considers his system an instrument, it is unlike any other (non-EA instrument) in that it generates multiple musical gestures.

These gestures cannot be directly controlled, and requires what Chadabe calls "flying by wire", an analogy to what pilots of airliners are required to accept: unable to control every aspect on their airplanes, they must rely upon their onboard computer to control non-high-level aspects. Like other practitioners of interactive computer music, Chadabe has had to rely upon constrained random procedures to generate the necessary unpredictability. And such procedures generate disorganized complexity.

3. Artificial Musical Intelligence - *Kinetic Engine*

In order to achieve *organized complexity* in realtime, we require some sense of musical intelligence. This has been my goal since 2005, when I began development of *Kinetic Engine*. *Kinetic Engine* versions 1-3 were concerned exclusively with rhythmic organization, specifically rhythmic polyphony. The current version, 4, is concerned with harmonic organization.

An important aspect of all four versions is the reliance upon multi-agents for complexity (and in this case, *organized complexity*). Intelligent agents are elements of code (programs or patches) that:

- operate without direct user interaction (they are autonomous),

- interact with one another (they are social),

- interact with their environment (they are reactive),

- make decisions as to when they should operate, and what they should do (they are proactive).

In *Kinetic Engine*, musical knowledge is contained within the agents themselves — for example, how to create rhythmic patterns, how to vary them, and, most importantly, how to interact with other agents. A great deal of effort was applied to the interaction of agents, so as to create a rhythmic polyphony that, though created artificially, could have been created by a human (this being one definition of artificial intelligence).

3.1 The virtual percussion ensemble

The musical model I pursued within v.2 is that of an unspecified number of percussionists entering a room full of percussion instruments. What kind of decisions would be necessary to create interesting music (again, as defined by *organized complexity*)?

3.2 Agent roles

When a player picks up an instrument, even if that specific instrument has never been played before, they would most likely be able to make specific judgements about it. For example, small instruments tend to have higher frequencies. Given that information, this should determine how the instrument (and thus the agent) would play. In other words, the agent's *role* within the ensemble. For example, bass drums would play less often (have lower densities), and tend to emphasize downbeats, while shakers would play more often, and tend to syncopate more. Therefore, each agent assumes a role within the ensemble, based upon the instrument they have chosen.

3.3 The Conductor Agent

Once each agent has chosen some instruments, how do they start to play? There are methods of agent negotiation that would allow for a mediated time signature and tempo, but I want to maintain *some* control over the music, so I created a Conductor agent, over whom I have control. Thus, time signature, tempo, and subdivision is set by the Conductor.

Next question - how is the performance to start? Does everybody play all at once, or one at a time? Since the agents are autonomous, they make these decisions; however, they do need a collective goal. Thus, through the Conductor, I provide the first objective: to achieve a cumulative density.

Imagine the conductor with his right hand indicating relative density: low = don't play very much; high = play lots of notes. For a human, "don't play very much" has meaning, but computers require something more discrete. For this reason, throughout *Kinetic Engine*, fuzzy logic is used to approximate human evaluations.

So when the Conductor sends the message to play at a "low" density, each agent interprets this differently, depending upon the role they've assumed within the ensemble - low density shaker is very different than low density bass drum.

3.4 Agent Personalities

But before an agent will generate and play any pattern, some determination must be made as to who starts playing first, and how to actually start playing. In the human ensemble, musicians react differently to the changing musical environment depending upon their specific personalities; therefore, each of the agent's in *Kinetic Engine* have unique *personalities* as well. Thus, in this case, an agent's *responsiveness* parameter would determine how to react to global changes (such as when to start playing), while *confidence* determines number of notes with which to enter.

3.5 Pattern generation

Agents generate actual patterns through a combination of musical rules and personality characteristics. For example, the *Downbeat* parameter determines the propensity to play on the first beat, while *Syncopate* determines how syncopated the player likes to play. These factors then generate probabilities for a pattern generating algorithm, which is then run through knowledge base that checks the pattern against a series of rules in order to avoid certain patterns and encourage others.

3.6 Overall Density - the initial collective objective

The Conductor “listens” to the resulting combination of player patterns, and compares the number of active notes to the requested Density, makes an evaluation, and messages back to the agents whether the cumulative density: for example, “way too low” or simply “too low”. The agents will, once again, react to this message in different ways, with *responsive* agents being the first to react. If the rating is “way too low”, non-active agents can begin playing, whereas if the rating is simply “too low”, active agents can add extra notes to their patterns. This negotiation between Conductor and Players continues, until the cumulative density is considered “close enough” by the conductor.

An important point is that this negotiation, and resulting adjustments, is not made in “computer time”, which would be less than a millisecond; instead, a decision was made to make these adjustments in “human” time: each agent waits “a few beats” before making adjustments, as does the conductor.

3.7 Social Behaviour: Agent Interaction

Once the Conductor has determined that the Density is close enough, the agents can begin social interactions. The agents listen to one another, and attempt to find other agents that have patterns that offer potential for interaction. This tendency is influenced by an agent’s *Social* parameter, or how willing an agent is to interact with other agents.

Imagine certain musicians in our human ensemble happily playing their patterns, grooving away with their eyes closed - not much social interaction happening here - while certain players will be listening to other players.

How do they actually begin interacting? Just as humans will rely upon *eye contact* in order to initiate interactions, *Kinetic Engine*’s agents message one another. The first agent will message the agent it wants to interact with, and wait for a response. If it gets a response in a reasonable amount of time, interaction will occur; however, if the other agent does not respond (perhaps it is not very social, or it is already interacting with another agent), the first agent will look elsewhere for social interaction.

If two agents agree to interact, they begin to adjust their patterns in an effort to create either rhythmic polyphony (interlocking) or heterophonic (similarity). This adjustment is two way, as both agents will adjust their own patterns, which most likely results in further readjustments; again, these adjustments occur “every few beats”, making the process very audible.

3.8 Unpredictability

In most cases, the interaction will eventually stabilize. Two additional personality parameters come into effect at this point: *Commitment* and *Mischievous.* Those agents that have a high commitment rating will happily stay in a social relationship for long periods; those with low ratings will get bored, and attempt to find new partners for interaction. The *Mischievous* parameter determines how willing an agent is to upset a stable system; this can happen in two ways. If the agent is playing, it can suddenly stop playing, causing active players to readjust their patterns, and possibly force non-active players to start playing. Non-active agents can suddenly start playing, again upsetting a stable system, and likewise causing readjustment by both active and non-active agents.

3.9 Influencing the Ensemble

As well as density, I can turn agents on and off, thus also causing the same sort of subtle readjustments. Lastly, I can *influence* the personality of the entire ensemble, increasing, for example, the social parameter of all agents.

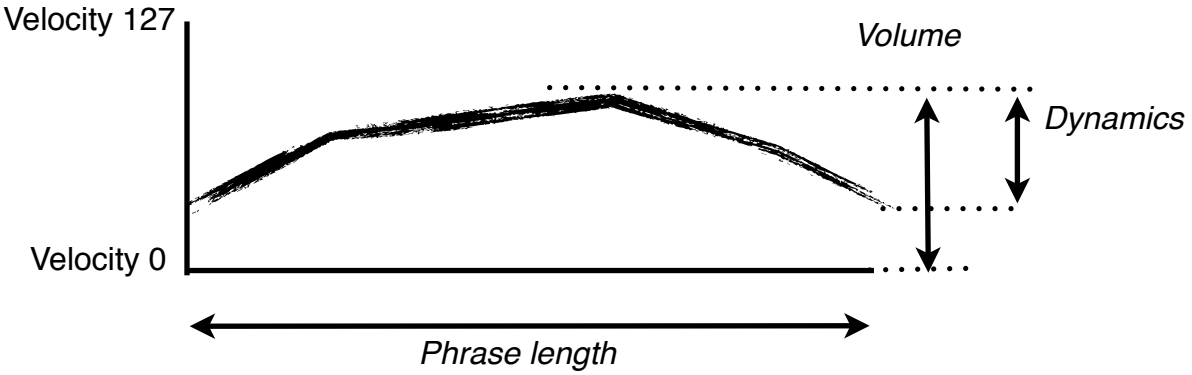
3.10 The End Result

The end result is a complex, evolving rhythmic polyphony over which I have high level control (i.e. top down), but whose *complexity* is determined from within the agent's themselves (i.e. bottom up). The output of *Kinetic Engine* can be direct audio, playing samples, or MIDI. It has been used to control the *MahaDeviBot*, a twelve armed robotic percussionist.

4. Melodic Agents

In *In Equilibrio*, the output of six rhythm agents is sent to six melodic agents. Like the rhythmic agents, the melodic agents have unique parameters that determine how to choose their pitches.

At the start of a new phrase (which is independent from the rhythmic pattern), the melodic agent chooses a starting pitch, based upon its *start pitch* parameter, with the length of the phrase determined by the *phrase* parameter. Each phrase will have a dynamic curve, influenced by the agent's *volume* (the maximum volume of the curve) and the *dynamic* (the "ends" of the phrases' volume).



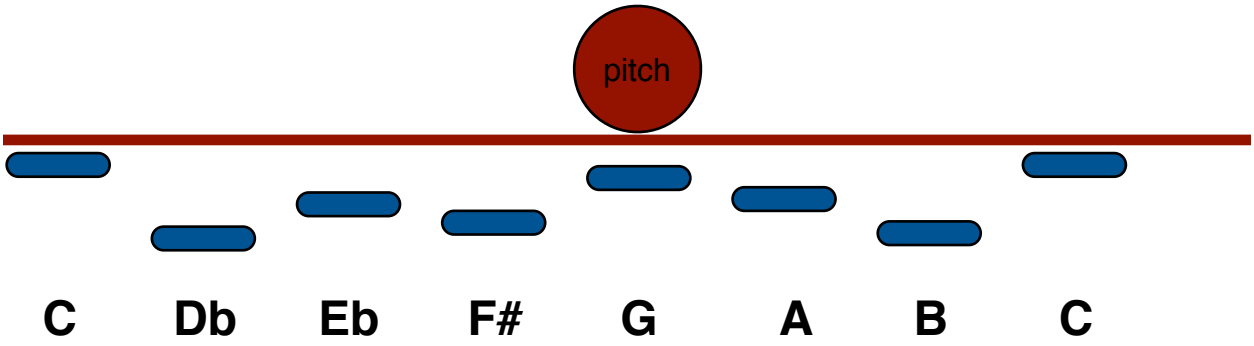
The amount of vertical movement (intervalic distance from a previous pitch) is determined by the agent's *flux* parameter.

Melodic shape: low flux	
Melodic shape: high flux	

Melodic movement is controlled by a series of parameters that defines the melodic tendencies within *In Equilibrio*; in other words, this algorithm is unique to this composition and not a general purpose method. It is, in fact, what I consider to be the defining characteristic of the composition.

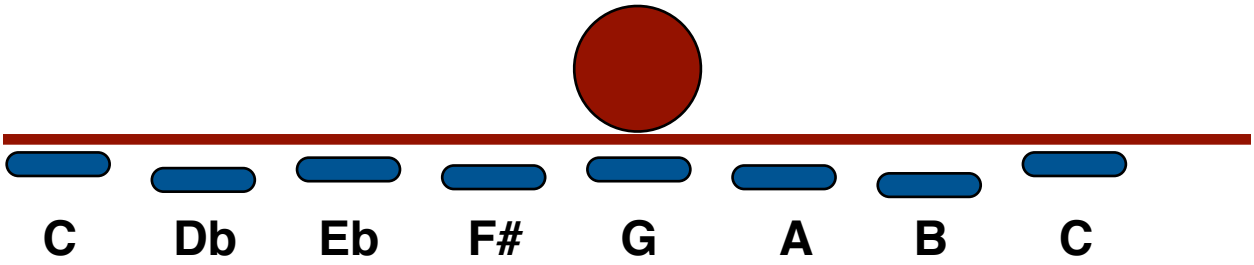
I sometimes think about it as the pitch being a metallic ball rolling around a table, and the pitch-scale being a set of magnets underneath the table of varying proximity to the table - the closeness of each magnet being its *attraction*.

Imagine the pitch ball rolling around, and being attracted to certain locations.



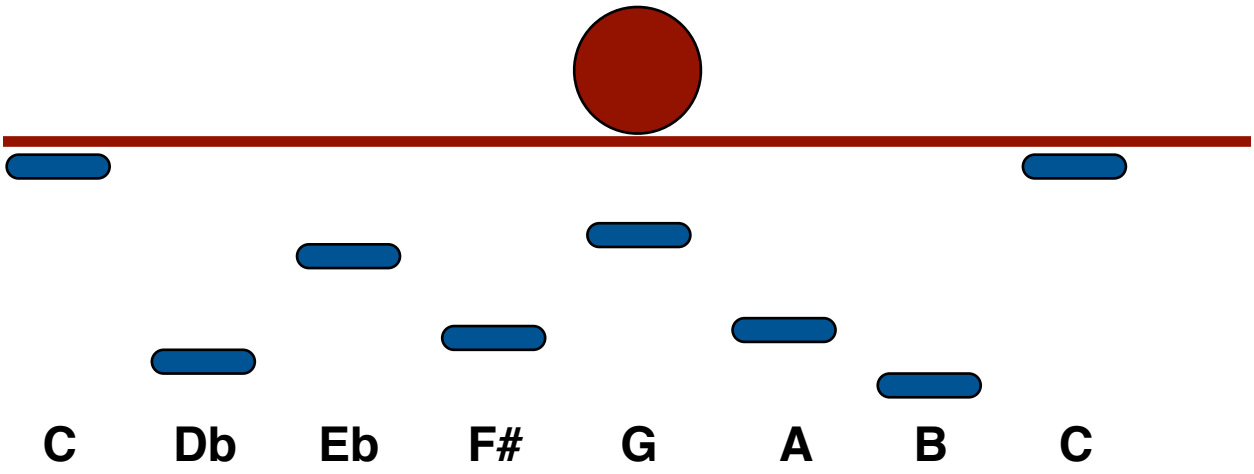
The *overall* attraction can be altered, giving less emphasis to those pitches that were originally close to the top:

i.e. low individual attraction (the 'magnets' are fairly even)

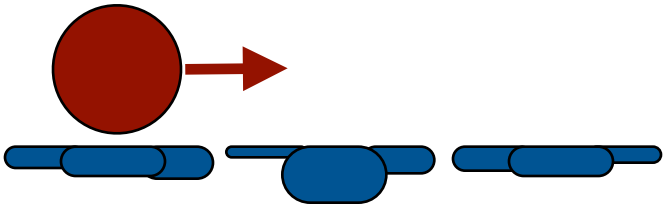


Or by increasing *attraction*, repetition increases, particularly on weighted pitches, since it is harder for the pitch to leave its current location.

i.e.. high individual attraction (the 'magnets' are quite discrete):



Actually a pitch curve is a bit more complicated, as each pitch has a weighting as to whether it remains on the pitch, moves left, or moves right. In this case, the pitch has a greater probability of being pulled to the right...



The scale itself is a continual modal variation of an initial scale. For example, the scale shown (C Db Eb F# G A B C) has an successive semitone relationship of (1 2 3 1 2 2 1). A modal variation of this scale would be to rotate this relationship by one: (2 3 1 2 2 1 1), resulting in a scale of (C D F F# G# A# B C). In performance, I control when a modal change is made. The actual mode selected is made by the software, dependent upon how much change: scales with only one or two note differences between them and the previous scale are considered slight changes, whereas scales with many individual notes differences are considered greater changes (the two examples given above have only three of seven notes in common, and therefore would be considered a significant tonal change).

The result is melodic motion that is somewhat Brownian, influenced by the particular scale during performance. However, the melodic agents are social, and thus there is vertical interaction between melodic gestures. Each agent broadcasts its current pitch. Every few beats, the agent compares its own pitch with that of other agents, and tries to form triads. This further complicates the pitch curve, which is now being influenced by the other active agents.

5. Conclusion

5.1 Is *In Equilibrio* realtime composition?

I spent over a significant amount of this paper explaining the rhythmic control in *In Equilibrio* - this system, *Kinetic Engine v.2*, took almost two years to program. I can safely state that it makes intelligent musical choices, and displays *organized complexity*. The resulting rhythmic interaction exhibits decisions that I certainly could have made. I can let the system run for long periods of time, and it will make interesting musical choices.

The pitch algorithm uses multi-agents to determine melodic choices. While these choices are unpredictable and complex, I will admit that they only display *disorganized complexity.* Without

my direct interaction, pitch gestures become somewhat homogeneous. In order for pitch choice to become truly intelligent, the controls that I exert over the system in performance - scale change, for example - should be made by software, dependent upon the musical environment. In Chadabe's words, I'm "flying by wire", and reacting to the software's musical choices. As such, I don't think I'm there yet.

Portions of the software described, as well as a recording of *In Equilibrio*, can be found on the author's website: http://www.sfu.ca/~eigenfel/research.html

Acknowledgements

The development of *Kinetic Engine* has been funded by a SSHRC Research Creation Grant. The software described has been developed on a Macintosh computer using Cycling 74's MaxMSP software.

References

Benson, Bruce Ellis. The Improvisation of Musical Dialogue. Cambridge University Press (2003)

Chadabe, Joel. "Some Reflections on the Nature of the Landscape within Which Computer Music Systems are Designed." *Computer Music Journal.* 1:3 (1977)

_____. "Interactive composing: an overview." *Computer Music Journal*. 8:1 (1984)

Dahlstedt, Palle, and Peter McBurney. "Musical agents." *Leonardo*. 39:5 (2006)

Eigenfeldt, Arne. "Drum Circle: Intelligent Agents in Max/MSP." *Proceedings of the 2007 International Computer Music Conference.* ICMA (2007)

_____. "Intelligent Real-time Composition", eContact 10.4 http://cec.concordia.ca/econtact/10_4/eigenfeldt_realtime.html (2008)

_____. "Multiagent Modeling of Complex Rhythmic Interactions in Realtime Performance." *Sounds of Artificial Life: Breeding Music with Digital Biology*, A-R Editions (2008)

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

Minsky, Marvin. The Society of Mind. Simon & Schuster, Inc (1986)

Rowe, Robert. Interactive Music Systems. MIT Press (1992)

Weaver, Warren. "Science and Complexity." *American Scientist*. 36:536 (1948).

Wiggins, Geraint, and Alan Smaill. "Musical Knowledge: What can Artificial Intelligence bring to the Musician?" *Readings in Music and Artificial Intelligence* E.R. Miranda, ed. (2000)

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

Woolridge, Michael, and Nicholas R. Jennings. "Intelligent agents: theory and practice." *Knowledge Engineering Review*. 10:2 (1995)

Rodolfo Daniel Wulfhorst, Lauro Nakayama, and Rosa Maria Vicari. "A multiagent approach for musical interactive systems." *AAMAS* (2003)

Biography

ARNE EIGENFELDT is a composer of (mainly) real-time interactive electroacoustic music, although every once in a while, he is forced to push dots around on a page. He also teaches music and technology at Simon Fraser University, where he is an associate professor in the School for the Contemporary Arts.