A FRAMEWORK FOR ECOSYSTEM-BASED GENERATIVE MUSIC

Oliver Bown
Centre for Electronic Media Art, Monash University, Clayton, Australia, 3800
oliver.bown@infotech.monash.edu.au

ABSTRACT

Ecosystem-based generative music is computer-generated music that uses principles borrowed from evolution and ecosystem dynamics. These are different from traditional interactive genetic algorithms in a number of ways. The possibilities of such an approach can be explored using multi-agent systems. I discuss the background, motivations and expectations of ecosystem-based generative music and describe developments in building a software framework aimed at facilitating the design of ecosystemic sonic artworks, with examples of how such a system can be used creatively.

1 INTRODUCTION

A traditional paradigm of generative and evolutionary music is that of the interactive genetic algorithm (IGA), which is based on the notion that an artist can evolve aesthetically pleasing music by using their aesthetic judgement as the 'selective pressure' in an artificial environment. More recently, researchers in generative and evolutionary music have started to broaden their interest to more collective behaviours, such as social learning, cultural dynamics, and niche construction [9], in simulated multi-agent systems [6, 11, 8]. This approach, which I will refer to generally as *ecosystemic* [1] , abandons the goal of exerting direct control over evolving systems through aesthetic selection, requiring a more complex creative interplay between software and artist. For example, McCormack [7] has described modelling processes such as niche construction as a kind of creative *design pattern* with which an artist can design complex generative dynamics in an exploratory manner. One of the major challenges in working with multi-agent systems is that complexity of design rapidly gets in the way of deeper exploratory investigation. Design patterns address this problem by helping us to break down and conceptualise systems, providing general rules that could be applicable to a range of different situations. As such, they reduce the opacity of complex systems.

The framework discussed here (also see [3]) complements this software engineering-inspired approach by offering additional ways to reduce the opacity of complex creative projects, primarily through tools that facilitate the batch processing and analysis of systems in a range of scenarios. Such challenges are already being addressed by frameworks developed for studying multi-agent systems, particularly in the social sciences and in artificial life. But none of these systems have been built to deal with the practical goals and typical methodologies of artists and musicians in mind. Multi-agent modelling experiments are of little creative value if they cannot be successfully transferred to creative domains, or be used creatively. However, they have the potential to form the basis for a new approach to works such as compositions, installations and creative software, due to the variety of behaviour that multi-agent systems exhibit that is not seen in traditional software.

This paper begins by discussing the motivations for pursuing an ecosystemic approach to artificial evolutionary music and art, as compared to existing approaches such as the IGA. I then describe a framework for developing ecosystemic artworks and music, which integrates an experimental multi-agent modelling environment with a real-time music environment. Finally, I demonstrate how this framework can be used to develop ecosystemic sonic artworks: installation works in which audio is used as the basis for an evolutionary environment.

2 BACKGROUND, MOTIVATION AND EXPECTATIONS

Despite moderate successes, the IGA approach to evolutionary art falls somewhat short of its original hopes. In principle, by analogy with natural selection, it promised to produce complex outputs that are both pleasing to, but beyond the understanding of the user, that the user wouldn't have thought of himself, and perhaps couldn't even have imagined. In practice, this goal has proven elusive.

These shortcomings can be discussed in terms of the structure of the *space* that an IGA user navigates in his search. Parameterising a generative system in order to make parts of it evolutionary inherently defines a space – the pa-

[1] Arguably, ecosystem models are a more specific subset of this area. The term is used here to refer more generally to any set of coevolving interdependent elements.

SMC 2009, July 23-25, Porto, Portugal

rameter, or genotype, space of the system. This space may be thought of as a network of points (genotypes) connected by genetic mutations. The IGA user engages in a blind and relatively passive process of evaluating and selecting, which can be represented as a journey through this genotype space. Initially, an important question about the space itself is whether it is structured such that the user can steer the evolving system towards a certain target. If so, this is a potentially powerful design tool. For example, even though the target was pre-specified, it may have other interesting incidental properties.

A more ambitious expectation for IGA software is that it should also guide the user in interesting and surprising directions. This prompts a second question about the design of the genotype space: if interesting and surprising generative artefacts appear on this journey, will they be positioned *en route* to even more interesting and more surprising artefacts? These questions highlight the importance of the parameterisation of the system itself, suggesting that the art of designing such systems lies in designing genotype spaces that lead us to interesting places. Little development has been made in this area, although examples, such as the Neuroevolution of Adaptive Topologies (NEAT) project [12], have pointed to ways in which evolutionary lineages could lead in diverse aesthetic directions. NEAT specifies a procedure for neural nets to increase in topological complexity during their evolution. Here the evolved entities could be argued to have accumulated a user's preferences over time.

The aesthetic selection of computer generated art and music by individual users is also bound for practical reasons to the simple case where genotypes map directly to phenotypes that can be treated as isolated units for evaluation. Evolutionary entities with any kind of developmental process, environmental influence, or interaction between the individuals within the population, are complex enough to necessitate a different approach. One solution to this *fitness bottleneck* [1] is to employ multiple users as aesthetic selectors. Recent research explores this avenue by borrowing from areas such as social networking technology, human computing [2], and grid computing efforts such as *SETI@Home* [3], in order to find ways to distribute the process of selection (*e.g.* Draves' *Electric Sheep* [4]).

The ecosystem approach attempts to rethink evolutionary art by focusing on the design of spaces which embody coevolutionary processes such as niche construction [5]. In IGAs, the passivity of users in their interaction with the system is at odds with the supposed aesthetic influence they have over it. Ecosystem models are primarily non-interactive, but still face many of the same challenges as the IGA approach: designing evolutionary spaces that lead 'naturally' to a diverse array of interesting states.

Nevertheless, sonic ecosystems also have great potential for interactivity in more or less direct ways. In the interactive audio visual installation *Eden*, McCormack [6] used audience presence as a resource for a population of evolving agents, such that agents might evolve to draw the attention of the audience.

In other forms, interaction with sonic ecosystems could correspond more closely to the original goals of the IGA approach. This follows the reasoning that the real world of artistic and musical creativity itself resembles an ecosystem, exhibiting heterogeneity, stability, interdependence and the capacity for the components of the system to influence the entire system's future evolution in unpredictable ways. The value of an artwork or piece of music is strongly tied to its situatedness in a cultural context, which determines its relevance [10]. If culturally determined relevance is a significant factor in determining the evaluation of artefacts, then creative cultural domains can be said to involve a strong degree of feedback. Prior experience affects our evaluation of new music, and new music is discovered and consumed via channels of authority that precede pure content-based evaluation. Evaluation of creative artefacts is constantly shifting and heavily influenced by these factors. Accordingly, in a future inhabited by fully fledged creative computational systems, we would expect them not only to be adapted to the cultural factors affecting value judgement, but also to actively manipulate these factors through cultural interaction. In short, it may be a mistake to assume that systems designed to adapt to our preferences will actually produce the most pleasing results.

This discussion highlights the extensibility of ecosystem models towards more interactive and evaluative forms. However, our present focus is on the more manageable problem of creative ecosystems which are either closed or loosely interactive. In this area, an obstacle to good design is that the behaviour of a multi-agent system is typically complex and requires detailed analysis of its macroscopic properties to be clearly understood. Put differently, an ecosystemic approach is particularly interested in the creative exploration and use of exactly those models that are complex and not immediately obvious, and the methodology presented here is aimed at maximising the potential to explore interesting complex systems. In the sonic domain, this might involve generative sonic works that continue to develop and transform indefinitely, but with consistent structural and aesthetic properties.

The issues of complexity suggests the need for an iterative development process based on the analysis of behaviours in successive versions of a model. Similarly, dis-

[2] The use of human brains for data processing, such as in *reCAPTCHA* [13]

[3] SETI stands for the Search for Extra Terrestrial Intelligence. This group maintains an *ad hoc* processing grid of subscribers' home computers for the brute-force analysis of cosmic radio data [5].

[4] http://electricsheep.org

[5] Coevolution typically refers to the mutual influence between the evolution of two species, but can apply to a multitude of interacting evolving entities, including the set of individuals within a species.

cussing approaches to the relationship between artificial life models and real-world complex systems, di Paolo *et al* [4] propose a three-stage development process of exploration, experimentation and explanation. The exploratory phase sets out to see what is of interest and relevance in a model and to record information about the model. In the experimental phase, hypotheses are tested about the behaviour of the model. In the explanatory phase, the understanding gained from the experimental phase is applied to the natural world. The different goals of creative model building mean that there is no explanatory phase, and a more open-ended flexible approach to exploration and experimentation (which includes de-bugging and learning the relationship between a model's logical behaviour and its aesthetic outcomes). A more detailed study of working practices in creative computing would reveal how this kind of methodology could be developed more explicitly in creative domains.

3 FRAMEWORK DESIGN

In [3], I discuss a number of design requirements for a framework for creative ecosystemic models:

- Analysing system behaviours (e.g., comparing many different parameters)
- Integrating multi-agent and sound components in one development environment
- Facilitating the design of live algorithms [2] using a multi-agent approach
- Creating flexible agents and configurations for multiple contexts
- Probing and editing models interactively
- Facilitating new forms of software extensibility, such as being able to embed models inside other models.

A preliminary design for a framework is introduced in [3], with an example of a sonic ecosystem. Here I discuss how the framework aims to integrate multi-agent modelling principles and a computer music library. The framework also offers libraries for more specific ecosystemic and evolutionary functionality like genetic variation, resource management, and handling dynamically changing populations.

One of the main practical considerations for exploring new types of creative generative and evolutionary methodologies is for models to be easily adapted for multiple target applications, such as batch processing for analysis, interactive exploration and real-time performance. For this reason, and for the goal of achieving useful extensibility, popular object oriented programming languages were seen as providing the most power and flexibility. Java was chosen for the current framework, primarily because of its greater user-friendliness. Extendable general-purpose development environments, such as Eclipse [6], also provide core project management functionality that is desirable for the kinds of complex projects that are likely to be needed.

An audio library, Beads [7], was developed by the author in pure Java with this framework and a number of other computer music applications, including live performance, in mind. This audio library follows the principle of good integration; that is, it is advantageous to work in a single development environment when working experimentally. Popular real-time music environments such as MaxMSP [8], SuperCollider [9] and JSyn [10], use separate formats for high-level and low-level (digital signal processing or DSP) elements, requiring users to skip between different environments. For example, users can write *externals* for MaxMSP, but must compile them in a development environment and then launch them in MaxMSP (at least in the case of native C externals). A complete integrated environment allows developers to add DSP routines straight into their code. With a multi-agent modelling environment and sound environment closely integrated, it is also easy to have modelling processes triggered by audio-rate processes, as well as vice-versa.

The current instantiation of the ecosystems framework focuses on the basic requirements listed above. A hierarchical scheduling mechanism provides the core configurability and addresses the requirement of extensibility by allowing schedulers to be easily chained together. Using an XML configuration file, an arrangement of schedulers and listeners can be set up, with a master scheduler at the root. In the most basic multi-agent scenario, this would consist of one scheduler and a population of agents that are updated by the scheduler. Additional listeners, such as a visualiser or sonifier or tools for collecting simulation data from the population, can be specified in the configuration file or added interactively. Thus different configuration files can specify different use cases such as batch processing and live installation. A scheduler can also take its timing from an audio-rate clock instead of the master simulation scheduler, meaning that audio rate processes and higher level scheduling processes run in the same ratio under different circumstances (this is useful in the case of running real-time systems offline). As well as allowing different configurations, the hierarchical structuring of schedulers also facilitates interesting experiments in the extensibility of existing systems, such as the integration of two populations of agents in a common environment, or the embedding of agents and their environments as subsets of bigger environments.

[6] http://www.eclipse.org
[7] http://www.beadsproject.net
[8] http://www.cycling74.com
[9] http://www.audiosynth.com
[10] http://www.softsynth.com/jsyn

For logging data, a data logger class exists which uses the Java Reflect API to probe agent classes. Thus if agents have the field `size`, the logger can be told to log the value "agent:size". This will log all of the size values for all of the agents at each time step. This syntax works for nested classes and data structures. If agent's have a field `memory` which in turn stores a variable length array of known pitches, then telling the logger to log "agent:memory:pitches" will log all of the pitches for each agent at each time step, and so on. Loggers can be switched on and off at different time steps. The logged data can then be formatted to work with a plotting program such as Gnuplot [11] or Mathematica [12]. No extra code has to be inserted into the agent except 'getter' methods for these fields, which the data logger automatically discovers. For interactive probing of data, another class exists which unpacks the entire configuration of schedulers and agents into a tree view. The elements of the tree view can then be edited manually (assuming the specified fields have 'setter' methods). Alternatively, for any elements that are being visualsed, it is easy to configure the system such that double clicking on that element brings up a similar tree rooted at that element.

Such features are in an early stage of development and the ultimate goal of this project is to develop the system and its associated working methodology to the stage where the framework can be used as a general purpose creative toolkit for sonic ecosystems.

4 USING THE FRAMEWORK

An ecosystemic approach to evolutionary computer music attempts to ground the design of an evolving artificial multi-agent system in an environment that is intrinsically related to the world of sound and music that we are familiar with. As discussed in Section 2, this provides, at least in principle, a kind of *coupling* between the agents in the evolving system and our experience, capable of guiding an artist's development of their work.

An ecosystem model typically consists of an environment containing resources (or also environmental conditions), a population of evolvable agents, and a set of rules defining the relationship between the resources and the agents' dynamics of survival and reproduction. This relationship is not the same as a specific fitness function for individual isolated agents, typical of standard genetic algorithms, since the population is involved in myriad interactions amongst each other (directly, or via manipulation of the environment), from which the exact conditions for survival emerge. This invites a variety of evolved relationships of both co-operative and competitive natures. For example, since parents and their offspring are closely related, their behaviour, locality, and their dependence on resources is similar and can often be in conflict. Such emergent relationships can be understood in terms of the theory of evolution, or more specifically in terms of known artificial life models. Often, models are unpredictable in the sense that the best way to know what they will do is to run them. But it is possible to work out the underlying behaviour and adjust the model accordingly (helped by the framework to the extent that it satisfies the design requirements).

In order to embed the system in a sonic context, the environmental resources should reflect acoustic features of sound that is created or affected by the agents in the population, but may also be mixed with sound coming into the model from outside (there is no reason why this couldn't be done at a more abstract level, such as in a MIDI domain, but sound is preferred as it makes fewer musical assumptions). An example is to take a spectrogram of the sound and to treat the level of each band as the literal quantity of a specific resource. Another is to take the amount of 'space' left in each band (e.g., a max value minus the level of the band) as the quantity of the resource, and rewarding health gains to agents in proportion to the relative level of sound they contributed to that band. In the latter case, the environment becomes less inhabitable the more sound is being made, but agents have to make sound to get fit. This contradiction invites the possibility of an *evolutionarily stable state* or an unstable dynamic process (see [3]).

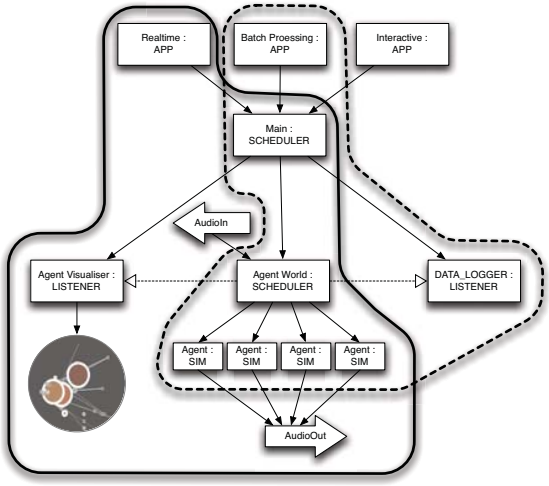

Figure 1. Different model configurations. In the real-time configuration (contained by solid line), the simulation is run with real-time audio and visualisation but no analysis. In the batch processing configuration (contained by dashed line), the simulation is run with non-real-time audio and analysis but no visualisation.

[11] http://www.gnuplot.info/
[12] http://www.wolfram.com/

A basic sonic ecosystem is given by the structure in Figure 1. At the top, three different executable applications are available, each of which sets up a master scheduler and runs it. A configuration file specifying the set-up of the environment and agents can be fed into one of the applications, depending on the required context. The application reads the configuration file and attaches the specified objects to the master scheduler. In the batch processing case, additional configuration data is required (e.g., what parameters to sweep over), but since the applications themselves are only a few lines of code, it is easy to write dedicated applications for specific batch processes. Alternatively, a set of configuration files, one for each parameter set, can be fed into the batch processing application.

There are a number of ways that schedulers, agents and scheduler listeners can be built to reflect the basic design of environment and multi-agent population. In Figure 1, this is achieved by making the environment itself a scheduler (labelled 'agent world'), which sets up the audio context, and then sets itself to be updated from an audio-rate clock, instead of from the main scheduler (it is still *controlled* from the main scheduler). That way we know that environment and agents are always updated at the same ratio with respect to the audio processing. When initialised, the environment generates the initial population of agents from the configuration file, and each agent is attached to the environment's scheduler, which triggers updates in the agents. The scheduler can be configured to perform synchronous or asynchronous updates of the population. After the initial set-up, agents take care of automatically removing themselves from the environment when they die, and adding their offspring to the environment when they reproduce.

In the case of the region enclosed in the dotted line in Figure 1, the population is run in a batch process without real-time audio input and output (determined by the configuration file, but overridden by the batch processing application). A data logger is also attached, which records information about the agents. In the case of the region enclosed in the solid line, the simulation is run in real-time with an additional visualisation unit attached.

As constructed, the creative challenge for such models is to work out how to manipulate aspects of the overall design, such as the resource model, to achieve interesting dynamical behaviour, and also satisfying sonic behaviour, without letting one of these goals eclipse the other. It is assumed that the dynamics of an evolutionary process *could* form the basis for an interesting musical and harmonic structure: one could construct a metaphor in which mass extinction and adaptive radiation act as mechanisms of sudden change, and gradual coevolution leads to complex and harmonic relationships between components. Having glimpsed this possibility, the role of the framework is to facilitate the exploration of the model's behaviour with these combined goals in mind. Since the environment of agents consists largely of other agents, there is fair reason to assume that some variation of such a model exists in which the population continues to evolve over a long period of time, without settling into a stable state, and perhaps even increasing in complexity (this can be see as following a *coevolution* design pattern).

I discuss an example of such a model, in the form of a sonic installation artwork, in [3], which uses the 'available space' resource model discussed above, and demonstrate an example of running a parameter sweep across a number of simulation settings in order to find configurations in which the population of agents divides into two or more species over time. Where such configurations were found, the speciation behaviour is also clear from the audio recording of these runs. In other configurations it was common for separate species to emerge for brief periods, often collapsing back to a single population. The ultimate victory of a single population is a likely outcome of the 'available space' resource model, because agents gain fitness most easily by being more noisy and hogging the audio spectrum: a species that made the noisiest sound would ultimately out-compete less noisy species. This still produced interesting dynamics as the population evolved through various phases of noisiness, often moving through a complex sequence of phases before discovering the noise strategy.

5 SUMMARY

This paper begins by discussing the motivations behind developing new approaches to evolutionary computer music, following an ecosystemic paradigm. Beyond the traditional notion of aesthetic selection, I consider ways to integrate tools from multi-agent evolutionary systems into creative practices. I describe the design of a framework which addresses the design goals established by such an approach. This framework integrates computer music and multi-agent modelling tools into an existing development environment and allows different model configurations to be interactively explored and analysed in a flexible manner. Future work will focus on explicitly formulating a stronger methodology to back this kind of creative exploration, and continuing to develop the framework with this in mind.

6 ACKNOWLEDGEMENTS

I wish to thank Jon McCormack for valuable discussions that contributed to this paper. This research was funded by the Australian Research Council under Discovery Project grant DP0877320.

7 REFERENCES

[1] J. A. Biles, "Autonomous genjam: Eliminating the fitness bottleneck by elim-

inating fitness," 2001, available from http://www.it.rit.edu/˜jab/GenJam.htmljab/GenJam.html.

[2] T. Blackwell and M. Young, "Live algorithms," http://www.timblackwell.com/, 2005.

[3] O. Bown, "Ecosystem models for real-time generative music: A methodology and framework," in *Proceedings of the 2009 International Computer Music Conference (ICMC 2009)*, Montreal, Canada, forthcoming 2009.

[4] E. Di Paolo, J. Noble, and S. Bullock, "Simulation models as opaque thought experiments," in *Articial Life VII: Proceedings of the Seventh International Conference on Articial Life*, M. A.Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, Eds. Cambridge, MA: MIT Press, 2000, pp. 497–506.

[5] E. Korpela, D. Werthimer, J. Kobb, and M. Lebofsky, "Seti@home – massively distributed computing for seti," *Computing in Science and Engineering*, vol. 3, pp. 78–83, 2001.

[6] J. McCormack, "Artificial ecosystems for creative discovery," in *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, D. Thierens *et al.*, Eds. ACM, New York, 2007, pp. 301–307.

[7] J. McCormack and O. Bown, "Life's what you make: Niche construction and evolutionary art," in *Applications of Evolutionary Computing: EvoWorkshops 2009*, 2009.

[8] E. R. Miranda and P. M. Todd, "A-life and musical composition: A brief survey," in *IX Brazilian Symposium on Computer Music: Music as Emeregnt Behaviour*, 2003, pp. 59–65.

[9] F. J. Odling-Smee, "Niche construction, evolution and culture," in *Companion Encyclopedia of Antrhopology: Humanity, Culture and Social Life*, T. Ingold, Ed. Oxford, UK: Routledge, 1994.

[10] V. S. Ramachandran, "The artful brain," Talk given at the 2003 BBC Reith Lectures, available from http://www.bbc.co.uk/radio4/reith2003/lecture3.shtml, 2003.

[11] J. Romero and P. Machado, Eds., *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer-Verlag: Heidelberg, Germany, 2008.

[12] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[13] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "recaptcha: Human-based character recognition via web security measures," *Science*, vol. 321, pp. 1465–1468, 2008.