

INTERACTING WITH MUSEBOTS (THAT DON'T REALLY LISTEN)

Arne Eigenfeldt

School for the Contemporary Arts,
Simon Fraser University

arne_e@sfu.ca

ABSTRACT

tinySounds is a collaborative work for live performer and musebot ensemble. Musebots are autonomous musical agents that interact, via messaging, to create a musical performance with or without human interaction.

1. INTRODUCTION

Generative and interactive systems have a long history within music [1, 2, 3]; more recently, aspects of artificial intelligence have been applied to such systems, creating a contemporary approach known as metacreation [4]. One useful model borrowed from artificial intelligence is that of *agents*, specifically multi-agent systems. Agents have been defined as autonomous, social, reactive and proactive [5], similar attributes required of performers in improvisation ensembles. Musebots [6] offer a structure for the design of *musical agents*, allowing for a communal compositional approach [7] as well as a unified model. An overview of recent musebot ensembles is given elsewhere [8].

2. MUSEBOTS

Musebots are pieces of software that autonomously create music collaboratively with other musebots. They decide how to respond to their environment – and each other – on their own, based upon their internal beliefs, desires, and intentions.

The musebot protocol¹ is, at its heart, a method of communicating *states* and *intentions*, sending networked messages established through a collaborative document via OSC [9]. A *Conductor* serves as a running time generator, as well as a hub through which all messages pass (see Fig.1).

Copyright: © 2019 Arne Eigenfeldt. This is an open-access article distributed under the terms of the [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by/3.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Individual musebots broadcast to the ensemble aspects of their performance; the details of *what* they communicate is left to the designer of the ensemble.

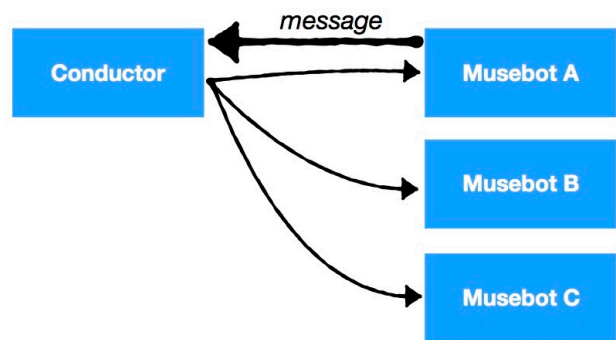


Figure 1. Diagram of messages between musebots and the Conductor. In this case, Musebot A sends a broadcast message to the Conductor, who rebroadcasts it to the ensemble.

3. TINYSOUNDS: FOR VOICE AND MUSEBOT ENSEMBLE

The musebot ensemble in *tinySounds* is a redeployment of an earlier metacreative system, *The Indifference Engine*, which is partially described elsewhere [10]. Live audio is analyzed for features: spectral centroid; spectral flux; loudness; activity level (onset detection); and Bark band spectrum. This information is messaged to the audio musebots and an effectsBot (see Fig.2). This latter musebot adds effects – delay, pitch shift, time stretch, ring modulation, and distortion – autonomously, based upon its interpretation of the analysis messages. For example, it will switch effects when activity is low, and add more processing when flux is high.

The audio musebots – in this case, four instances of *tinySoundBot* – have access to a large corpus of pre-analyzed soundfiles; given a Bark band spectral analysis via the *Conductor*, the audioBots will attempt to find the closest matching recordings from their available database. The audioBots autonomously begin and end playing based upon incoming messages, including activity and flux, as well as reacting to whether other audioBots are active or not.

Audio is generated using a modified version of *CataRT* [11].

¹ <http://tinyurl.com/gngmews>

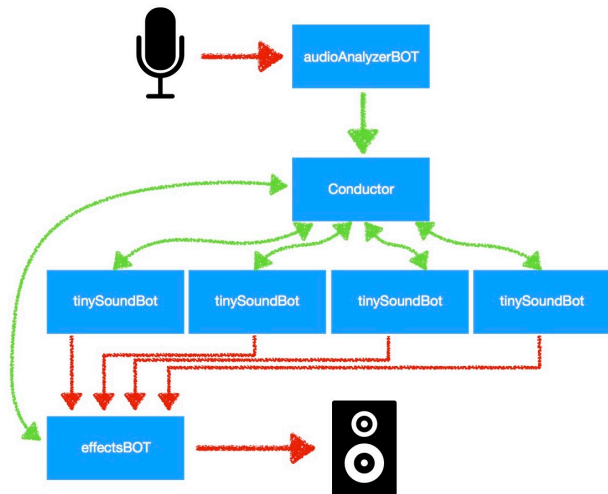


Figure 2. Diagram of musebots in *tinySounds*. Audio paths are in red; musebot messages are in green.

4. PERFORMANCE NOTE

Machine learning algorithms are wonderful for sifting through data and discovering relationships; more challenging is how these algorithms can be used for generation. It isn't that difficult, for example, to train a system to provide similar sounds for a database, given a live sound. But what's the artistic interest in that? Similarly, it isn't that difficult to extract live performance information from an improvising musician – activity level, general frequency range, timbre – so that the system responds likewise. But, again, reactive systems lose interest fairly quickly.

I find it much more interesting when my musebots go off on their own, exploring their own ideas through beliefs they may have formed incorrectly and unintentionally. For that reason, I usually build a lot of ambiguity into my analysis or provide conflicting information. What happens when one musebot is sure of something, while another is absolutely sure of something else? And what if a third musebot just doesn't care?

In *tinySounds*, musebots are trained using a neural net on a corpus that has been hand-tagged for valence and arousal measures, as well as pre-analysed for spectral information. However, the correlation between audio features (what the musebots are listening for) and affect (valence and arousal) isn't direct; in assigning the latter, I may decide that a sound from the corpus is complex and active, but my reasons for doing so may not use the same information as the musebots are provided with. Thus, a musebot may decide that, based upon what it has learned, a live sound is high valence / high arousal, but the listener may perceive it otherwise. This isn't a flaw in the system; it's a feature!

Lastly, my role as overseer in the musebot ensemble allows me to further disrupt how the musebots apply their knowledge. The corpus is organized semantically (i.e. voice sounds, kitchen sounds, transportation sounds, etc.); once a musebot is using a certain subdirectory, it can't easily switch to another. As a result, its choice of related sound, whether affective or timbral, is limited to

what is immediately available to it. If the musebots are frustrated, they haven't mentioned it to me (yet).

Musebots are not straightforward reactive processes; instead, they have their own beliefs (in this case, the incoming analysis data), desires, and intentions. They will happily play on their own, or they may react very closely to the live performance; more often than not, they will offer their own "reinterpretation" of the live performance, with individual reactions to the analysis data.

Acknowledgments

This research was made possible through a Social Sciences and Humanities Research Council of Canada (SSHRC) Insight Grant.

5. REFERENCES

- [1] J. Chadabe, "Interactive music performance system," U.S. Patent No. 4,716,804. 5 Jan. 1988.
- [2] R. Rowe, *Interactive music systems: machine listening and composing*. MIT press, 1992.
- [3] T. Winkler, *Composing Interactive Music: Techniques and Ideas Using Max*. MIT press, 2001.
- [4] M. Whitelaw, *Metacreation: Art and artificial life*. Cambridge MA: The MIT Press, 2004.
- [5] M. Wooldridge and N. Jennings, "Intelligent Agents: Theory and Practice," in *Knowledge Engineering Review* 10/2, 1995.
- [6] O. Bown, B. Carey and A. Eigenfeldt, "Manifesto for a Musebot Ensemble: A Platform for Live Interactive Performance Between Multiple Autonomous Musical Agents," in *Proceedings of the 21st International Symposium of Electronic Arts, Vancouver, 2015*.
- [7] A. Eigenfeldt, O. Bown and B. Carey, "Collaborative Composition with Creative Systems: Reflections on the First Musebot Ensemble," in *Proceedings of the 6th International Conference on Computational Creativity, Park City, 2015*.
- [8] A. Eigenfeldt, "Musebots: Collaborative Composition with Creative Systems," in *eContact! 20.2 TIES 2017: 11th Toronto International Electroacoustic Symposium, Toronto, 2018*.
- [9] M. Wright, "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers," in *Proceedings of the 23rd International Computer Music Conference, Thessaloniki, 1997*.
- [10] A. Eigenfeldt, "Generating Structure – Towards Large-scale Formal Generation," in *Proceedings of the Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, Raleigh, 2014*.
- [11] D. Schwarz, "Corpus-based Concatenative Synthesis," in *IEEE Signal Processing Magazine*, 24(2), 2007.