Generative Music in the Web Audio API

Paul Paroczai
Gold Saucer Studio
1885 Venables Street
paulparoczai@gmail.com

ABSTRACT

While there are many examples of interactive applications being built with the Web Audio API - from new and unique synthesis instruments, to emulators of popular signal processors, to audio engines for games – there are remarkably few that make use of the API's ability to implement systems which generate music with no user input. The work presented with this paper (https://www.paulparoczai.net/#/webaudio/) is a collection of six pieces in which all sounds are synthesized in real time.

1. INTRODUCTION

Algorithmic music has a long history, with examples reaching at least as far back as the musical dice games of the Eighteenth century (Hedges 1978) and extending through numerous compositional practices, including the serialist techniques of composers such as Schoenberg, Webern and Berg in the early 20th century and Steve Reich's experiments with process in the 1960s and 70s. Examples of algorithmic music generated by and distributed on computers include Brian Eno's *Bloom* and *Trope* (2012), Joshue Ott and Morgan Packard's *Thicket* (2014), and Icarus' (Oliver Brown and Sam Britton) *Fake Fish Distribution* (2012). Musical Metacreation is a contemporary exploration of algorithmic music in which computational systems are designed to contribute to the creation of a fully finished artwork (Pasquier et al. 2016).

Given Galanter's definition of generative art as "any art practice where the artist uses a system…which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art" (Galanter 2003) human-interaction does not necessarily need to be excluded from such works. As a result, many systems have included interfaces enabling such interaction, and numerous such systems have been built using the Web Audio API. Few systems have been built which are capable of generating entire and complete musical compositions as the work presented here does.

2. DESCRIPTION

2.1 Description of the Artwork

Each of the six pieces available at the link above uses a unique combination of signal processing and algorithmic compositional techniques to generate a complete piece of music with no user interaction.

While the Web Audio API provides a variety of audio processing nodes, the pieces presented here use only a limited subset of what is available, based on the ways in which nodes behave most reliably across all available browsers and devices. These include the AudioBuffer, AudioBufferSource, BiquadFilter, Delay, DynamicsCompressor, Gain, Oscillator, Panner, PeriodicWave and WaveShaper nodes. Additionally, in order to reduce loading times and memory usage, no pre-existing audio files are used, and all nodes which include buffers or wavetables are filled with custom waveforms. As such, all audio is synthesized in real time within the browser at the moment the "Start" button of a given piece is pressed.

Because of the wide variety of sound-generating parameters used to determine the creation of each piece – from timing data which manages numerous levels of temporal structure, to the timbral effects of synthesis parameters such as filter and oscillator types and frequencies, to buffers used to shape sounds with unique envelopes and wavetables – each program is unique from any other not only in terms of its output, but also in terms of its range of possible outputs.

For example, in *community infinite*, the sequence of pitches played by the synthesizer is completely determined and played in the same way every time, while effects parameters such as delay times and filter envelopes are generated according to systems which allow for the creation of unique timbral qualities between successive runs of the program. Thus, the experience of any given run of the program is similar to that of listening to separate live performances of a notated piece of music, much like hearing the same piece of classical music played by a different orchestra, or even the same orchestra at a different time in a different venue.

On the other hand, pieces like *iudex pastel* have more in common with the experience of watching a performance of improvised music, in which a more-or-less predictable set of timbres, determined by the instruments included in a given ensemble, are deployed in a less predictable manner.

s april 2 (remix) and *s april 2* each use the same collection of synthesizers and compositional processes, but deploy them in different orders and arrangements in much the same way that a "remix" of a recorded song would rearrange the existing elements and ideas of the original. While combining each of these into a single piece would have been possible and fairly easy, I decided to keep them separate so that listeners could reliably choose whichever version they prefer.

apex lime generates a collection of narrow bandpass filters and delays, which are used to process a series of noise impulses, each of which repeats at a different rate. The result is a complex harmonic and rhythmic texture that varies subtly over the course of a run as its individual parts combine in a variety of ways owing to their different temporal proportions.

In *lotus ampersand,* ten algorithmically-generated audio buffers are used to articulate a chord progression in a randomly selected key. The harmonic and rhythmic texture of the resulting piece is

determined both by the contents of each generated buffer and by a series of randomly-generated delays through which each buffer is played.

Every piece presented here thus represents a creation that is unique both in my own approach to its construction and in the details of each instance of its operation. Using the Web Audio API to create these pieces has been an ongoing process of discovery not only in terms of the many incredible sounds and sound-making processes made possible by digital audio synthesis, but also in the distinct workflows and techniques that arise from creating music in this way. A repository of the tools used to make these pieces is available at https://github.com/pparocza/generative-music-web-audio.

2.2 Artist Bio

Paul Paroczai is a freelance audio artist and programmer currently living in Vancouver, BC. Alongside traditional composition, music production, and sound design, he specializes in the creation of generative music software that explores the unique capabilities of computational media.

This work has allowed him to participate in numerous collaborations in which he has designed a wide variety of sound-generating systems, including synthesizers which respond to real-time input from motion-capture data and game controllers, music-generating agents capable of autonomous improvisation, both on their own and within an ensemble, and programs capable of generating multiple complete pieces of music to accompany theatre, film, and video games. His personal practice currently focuses on the creation of generative music in web browsers using the Web Audio API.

His work has been presented in various local and international settings, including Gaudeamus Muziekweek, the Push International Performing Arts Festival, and the International Symposium of Electronic Arts. He received a Bachelor's Degree in Music Composition from the University of California, Berkeley, and a Master's of Fine Arts in Music and Music Technology from Simon Fraser University. Samples of his work can be found at https://www.paulparoczai.net/.

3. REFERENCES

[1] Galanter, P. 2003 What is Generative Art? Complexity theory as a context for art theory. GA, Milan.

[2] Hedges, S. 1978. Dice Music in the Eighteenth Century. Music & Letters 59:2; 180-187.

[3] Pasquier, P., Eigenfeldt, A., Bown, O., & Dubnov, S. 2016. An Introduction to Musical Metacreation. Computers in Entertainment (CIE) 14:2, 2.

Web Audio Conference WAC-2021, July 5–7, 2021, Barcelona, Spain.