Non Audio Signal Processing with the Web Audio API

Christoph Guttandin
Media Codings
Berlin, Germany
chrisguttandin@gmail.com

ABSTRACT

I recently ported a Shazam-like application to create and compare fingerprints of audio files from Python to JavaScript. Although this application handles audio data or data derived from audio data, only a very small part of it actually uses the Web Audio API. In fact I used the Web Audio API only to derive the PCM data of the input file.

My aim was to rewrite the code in a way that it uses the Web Audio API for all the necessary steps to finally compute the fingerprints. Those steps include tasks like filtering, limiting or normalizing the output of the FFT transformations. In Python most of this is done by using the NumPy package or the SciPy library. Both of them are dedicated libraries for scientific and numerical computation. With the Web Audio API those tasks could be executed by using the functionality of the DynamicsCompressorNode, the GainNode, the WaveShaperNode and the IIRFilterNode. All other cases could be handled by an AudioWorklet or a ScriptProcessorNode respectively.

I started by using the OfflineAudioContext to process non audio data and discovered that it can be roughly thought of as the browser's stream implementation which we don't have yet. Besides the advantage of reusing already implemented functionality, it is also potentially faster then any custom implementation and has the benefit of offloading the computation to another thread without adding any additional complexity by setting up a custom WebWorker.

With the new suspend() and resume() methods of the OfflineAudioContext, it is possible to render data as it arrives, although the final size of the rendered AudioBuffer needs to be known before creating the OfflineAudioContext.

I want to summarize my findings and show what's possible in current browser's implementation. A simple example for that could be that almost any numerical data can be processed. Surprisingly an AudioBuffer can happily handle values from -4294967296 to 4294967296, which will be suitable for many types of signals.

I would like to conclude my talk by briefly showing other non audio uses of the Web Audio API. One of those examples could be a package called webaudio-serial-tx published by substack. It sends serial data via the audio output. Another example for using the Web Audio API in a creative way is a project by Daniel Rapp called doppler which is detecting motion by playing frequencies above the audible range and analyzing their response.

WEB LINKS

NumPy package: http://numpy.org http://numpy.org

SciPy library: http://scipy.org/scipylib/

webaudio-serial-tx: https://github.com/substack/webaudio-serial-tx

doppler: https://danielrapp.github.io/doppler

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA.