Music 220D Report (Fall 2012)

Mayank Sanganeria

Advisor: Julius Smith

For my 220D this quarter, I wanted to explore Web Audio and Faust. To this end, I started learning JavaScript and Faust independently. While exploring Web Audio, I also came across the NaCl system built by Google for Chrome and explored that a little bit as well.

# JavaScript

On Myles Borins’ advice, I read Douglas Crockford’s book *JavaScript: The Good Parts*. I had read a few other books earlier but they were all just a reference book for JavaScript and never addressed good style. It was really useful in terms of dealing with JavaScript’s quirks and learning about enclosures and a proper way to do object-oriented programming using JavaScript.

Having done that, I thought that it would be useful working on a JavaScript library to get experience in JS architecture and decided to work on Colin Clark’s Flocking library and contribute a Granulator to the library.

I made a Delay Line and a Granulator ugen for his library that Colin Clark was happy with and will be incorporating into his library. The pull request is currently under code review. I also read an article by Ross Bencina on granular synthesis architecture and I will be incorporating some of these other architectures in all my granular synthesis implementations. The one I am most excited about is having a separate process that triggered grains rather than having grains be continuously deployed as I have been currently.

[GitHub Link](https://github.com/e7mac/Flocking/blob/master/flocking/flocking-ugens.js#L1716-L1827)

[Granular Synthesis Architectures](http://www.rossbencina.com/static/code/granular-synthesis/BencinaAudioAnecdotes310801.pdf)

# Faust

Faust (Function AUdio STream) is an audio signal processing language created at Grame, France. I really enjoyed thinking about building things in block diagram form instead of creating classes for every module. Granular synthesis lends itself more to implementation in code and I thought it would be an interesting exercise to build a granular synthesis unit in Faust. I also didn’t find any granular synthesizers built in Faust online which was encouraging. I especially enjoyed the moment when I no longer had to look up the syntax while writing in Faust and could simply start putting down my ideas. I also tried out Faust in WebKit and found it very exciting that we could have something like that in a future version of the WebAudio standard. Faust is definitely an ideal way to be thinking about DSP code and having it be ubiquitously available on all platforms promises a very bright future for DSP programming.

## Granular Synthesis

The most challenging aspect of getting the granular synthesizer to work in Faust was getting the different random number generators working for each grain line of grains. Since Faust drew into the same stream for each parallel grain, the same random number was used in all the grain lines. I tried a bunch of different approaches that led to some unexpected behavior. Finally I settled on using the multi-channel noiser to get n different white noise generators and used a sequence macro to extract the ith channel from the noiser to use for the ith grain line.

A sample-and-hold was used to randomly select grain positions. Noise was the input to the S&H and it was triggered every time the grain ended, in effect giving a new start position and using a counter to read through the delay line.

Making the number of grains a controllable parameter was done by having the max number of grain lines working in parallel continuously and multiplied with a conditional to silence all the grain lines above the desired number of grain lines.

[GitHub Link - Granular Synthesis](https://github.com/e7mac/faust-code/blob/master/granulator.dsp)

# NaCl

Based on my interest in Web Audio, Julius Smith suggested exploring NaCl as an alternative method to having audio on the web. NaCl allows Chrome to run pre-compiled C++ code straight out of the browser. The NaCl SDK compiles C/C++ code to an \*.nexe file format which can be included on an HTML page and communicates with the HTML file using messages. This allows access to computation that runs at C speed on web pages. The advantage to having this system is that it allows high speed computation for all uses – audio, graphics and anything coded in C/C++. The downside, though, is that it is supported only in Chrome and will not be supported by any browsers in the conceivable future. Even the support in Chrome is experimental currently and requires a flag to be manually set before NaCl applications can be run.

I ran some very simple examples with GUIs on NaCl. The system works really well but the lack of support on other browsers takes away from the excitement of everything it enables. The support on the iOS version of Chrome is also lacking at the moment.