Splatter: Chaotic Additive Synthesis in M4L

David Skaggs
Oberlin Conservatory
dskaggs@oberlin.edu

ABSTRACT

In this project, I set out to create an additive synthesizer where each partial had its own looping randomized envelope. The result was Splatter, a polyphonic Max for Live device with 20 oscillators and scala tuning support, creating sounds simultaneously reminiscent of the “ear-candy” aesthetics of contemporary artists such as Galen Tipton or Andrew Huang, and of the early computer music work of artists like Jean-Claude Risset. This paper will go over the fundamentals of additive synthesis, the current state of additive synthesis, the basics of the instrument’s design and interface, the specifics of how these ideas were implemented in its code, an evaluation of the instrument in its current state, and my plans for further developing the instrument.

1. INTRODUCTION

Additive synthesis relies on combining simple waveforms (typically sine waves) to create complex waveforms. The loIn some additive synthesizers, these oscillators are tuned to a specific ratio and are then mixed at different volumes. The frequency that the ratios are applied to are referred to as the fundamental frequency of the produced complex wave. For instance, to approximate a square wave with sine wave oscillators, one would tune the oscillators to go in odd harmonics, then keep all oscillators mixed evenly. Max for Live is a version of the visual programming language Max/MSP which is fully integrated with the DAW Ableton Live.

2. RELATED WORK

The most famous additive synthesizer is likely the Hammond electric organ, which uses tonewheels spinning at fixed tunings with electromagnetic pickups as oscillators. The player shapes the tone of the organ by mixing between the tonewheels with drawbars. Some software synthesizers, such as Serum or Vital, use wavetable additive synthesis; in this implementation, users can fade between different “frames” of timbres, where every “frame” is a different mix of a large bank of sine wave oscillators.

The work most relevant to this project is Madrona Labs’ plugin Sumu, which uses additive synthesis as a form of spectral resynthesis; “sampled sounds are represented as collections of up to 64 bandwidth-enhanced partials, each with a frequency, volume, and noisiness that can change over time” [1]. You can delay and frequency-modulate each partial separately. Sumu’s interface is much more complicated and modular than Splatter. It’s likely possible to recreate what Splatter does in Sumu, but I was curious if there could be a Max for Live device that was entirely dedicated to it.

3. DESIGN

This sound was originally part of a large drone texture that I made in Max/MSP. I really enjoyed the sound of it, so I was curious if it could be converted into a MIDI-controllable Max for Live device. Going in, I knew that the bare minimum for what I wanted was a bank of sine waves tuned to the harmonic series, where each sine wave had its own looping envelope, with each envelope lasting a randomized amount of time, and each sine wave detuning by a random amount on every envelope. Past this original plan, I’ve added controls for panning, mixing, tuning, and the timbre of the oscillators. The timbral controls let the user choose a base waveform and a “shape” control over the waveform. The base waveforms are a sine wave, a triangle wave, a square wave, and the option to upload a sample. The “shape” control lets the user shape the waveform in various ways (thinning out the shape of the sine wave, morphing the triangle wave into a sawtooth wave, changing the pulse-width of the squarewave, or wave-folding the sample). To keep things simple, this current iteration only uses envelopes with a quick attack (3 milliseconds) and varying decay lengths (between 50 and 3000 milliseconds).

The interface for Splatter assumes which parameters would be actively changed the most during a performance (see Figure 1), leaving the other parameters to be changed in pop-ups that can be found by pressing the “advanced” button in the lower left corner. These controls include a section for changing the tuning, mixing, panning, and timbre of the synthesizer (see Figure 2).

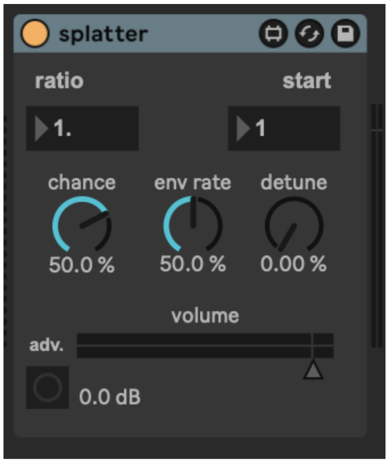

Figure 1. Front-facing controls for Splatter

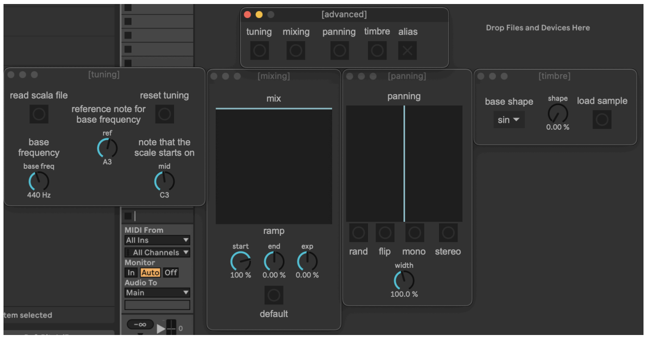

Figure 2. Pop-up advanced controls for Splatter

4. IMPLEMENTATION

The bank of oscillators was handled using Max/MSP’s multichannel wrapper. Originally, the detuning of the oscillators was done at random intervals independent of their envelopes, the ratios of the oscillators was fixed (specifically to the 11th-16th harmonics of the fundamental), and the panning was controlled with random LFOs (see Figure 3). In the Max for Live implementation, the envelopes were further developed so that the rate and chance of the envelopes could be actively and independently changed (see Figure 5). I used [mc.sum~] to combine all of the partials' individual envelopes and combine them into a single envelope with a short decay and release, and full sustain (see Figure 6); this is done so that, when a note is released, all partials' envelopes finish instead of suddenly stopping. Future implementations may make this feature optional in case someone wants such a sudden cutoff, though it could be imitated by automating the output volume or envelope rate. The scala tuning support is done using the "scale" message for the [mtof] object (and an admittedly convoluted subpatcher that converts .scl files into the strange syntax that the "scale" message uses). There's built-in support for [mtof] to read scala files with the "scalename" object, but this only works for scales that are in the user's designated search path. To make it easier to potentially change the amount of oscillators in future versions of the instrument, I used [gen~]'s "mc_channel" variable to make the oscillators' tunings dependent on their designated channels in the multichannel wrapper.

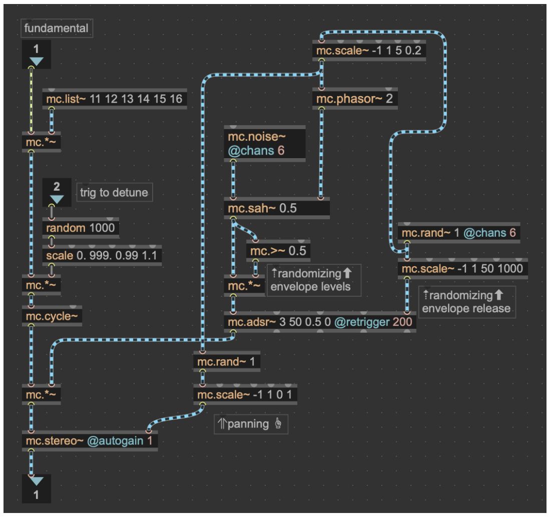

Figure 3. The original implementation of Splatter

Unfortunately Google Docs messed up the formatting for these figures, so here is an abhorrent amount of blank space:

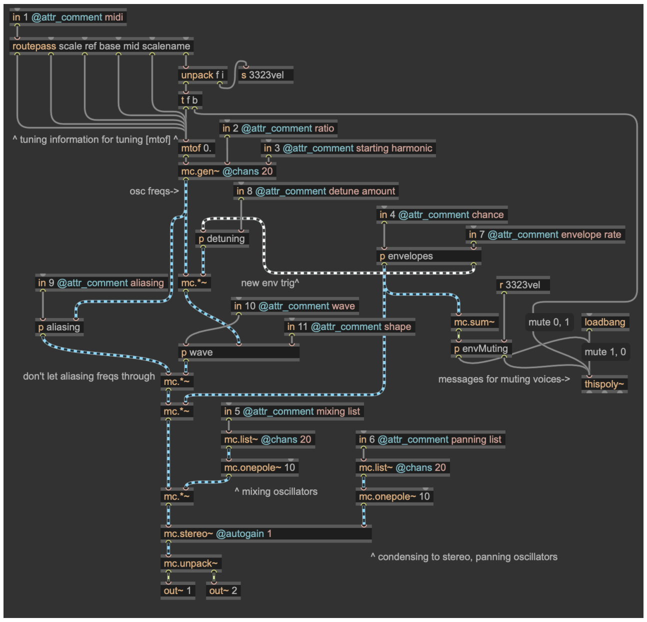
in 1 @attr_comment midi
routepass scale ref base mid scalename
unpack f i
s 3323vel
t f b
in 2 @attr_comment ratio
^ tuning information for tuning [mtof] ^
mtof 0.
in 3 @attr_comment starting harmonic
mc.gen~ @chans 20
osc freqs->
in 8 @attr_comment detune amount
in 4 @attr_comment chance
in 7 @attr_comment envelope rate
p detuning
p envelopes
in 9 @attr_comment aliasing
mc.*~
new env trig^
r 3323vel
in 10 @attr_comment wave
in 11 @attr_comment shape
mc.sum~
mute 0, 1
loadbang
p aliasing
p envMuting
mute 1, 0
p wave
don't let aliasing freqs through
mc.*~
messages for muting voices->
thispoly~
mc.*~
in 5 @attr_comment mixing list
mc.list~ @chans 20
in 6 @attr_comment panning list
mc.onepole~ 10
mc.list~ @chans 20
mc.*~
^ mixing oscillators
mc.onepole~ 10
mc.stereo~ @autogain 1
^ condensing to stereo, panning oscillators
mc.unpack~
out~ 1
out~ 2

Figure 4: The current implementation of Splatter

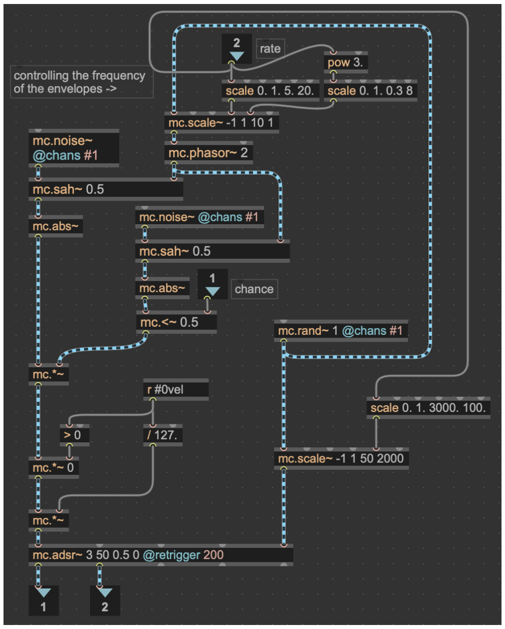

Figure 5. The updated envelopes for Splatter

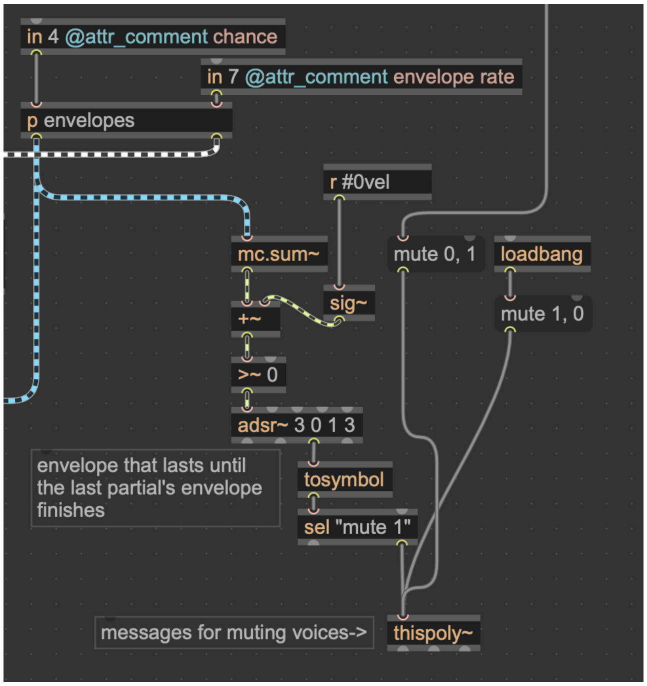
in 4 @attr_comment chance
in 7 @attr_comment envelope rate
p envelopes
r #0vel
mc.sum~
mute 0, 1
loadbang
+~
sig~
mute 1, 0
>~ 0
adsr~ 3 0 1 3
envelope that lasts until the last partial's envelope finishes
tosymbol
sel "mute 1"
messages for muting voices->
thispoly~

Figure 6. Using velocity and envelopes to control voice muting ([p envMuting] from Figure 4, de-encapsulated)

5. EVALUATION

In its current state, Splatter works as a prototype of the instrument that I wanted to make. Many interesting textures can be made, especially when working with samples instead of the more traditional waveforms. When using the mixing options to thin the texture by only allowing a small number of oscillators through, these textures can work very well in a larger drone soundscape, similar to the context of the original implementation of the concept. The ramping function for mixing between the oscillators is also very helpful for creating the illusion of a low-pass or high-pass filter.

6. FUTURE WORK

There are many things I'd like to do with this project in the future. My first goal is to fix a simple bug: the pop-up windows stay visible when the user selects a different track in the Ableton Live set. I'll have to put in the time to get more familiar with the specifics of the Live API. I'll also be working on adding a pop-up menu for editing attack and release rates for each oscillator's individual envelope. I'd also like to add more modulation features for mixing, panning, and timbre; I don't think that this will be too difficult to code. The main goal of this would be an option for random per-oscillator LFOs, the values of which could be scaled and restricted to a specific range. Another feature I'd like to add is a section for per-oscillator modulation sources mappable to any relevant parameter. Something like this might need to be coded in [gen~]'s [codebox] object, which would let me use "if" statements so that it isn't running a large amount of LFOs and envelopes that aren't being used. One feature that I'd really like to add is changing the amount of oscillators in the bank of oscillators. I can't figure out a way to do this since changing the amount of channels in a multichannel object in Max requires the patcher to be reloaded. This works in a standalone instance of Max, but if it's running as a device in Ableton, it stops making sound completely.

I'd also be interested in creating a "spectral mode," where a short audio file could be uploaded and analyzed, then the loudest 20 (or however many) partials could be saved onto a timeline or wavetable; each oscillator could potentially be put on different points in the wavetable at a time. In this mode, the user could also turn the wavetable's control over amplitude and pitch on or off. It's possible that this idea will become its own separate instrument instead of an alternate mode for this instrument.

7. CONCLUSION

In this project, I've created an additive synthesizer built around randomly generated envelopes. While there's still a lot of work that I want to get done on this project, I'm really happy about what I've accomplished so far. I'm excited to keep exploring the kinds of sounds that can come out of systems like this.

8. REFERENCES

1. Jones, Randy. "Sumu." Madrona Labs, n.d. https://madronalabs.com/products/sumu/.

2. Smith, Julius O. "Additive Synthesis." Spectral Audio Signal Processing, 2011. https://ccrma.stanford.edu/~jos/sasp/Additive_Synthesis.html.