Marimbas Errantes Performance and Technical Documentation

Nehemias Alvarado

January 7, 2025

Analysis:
This piece was created with SuperCollider, blending live sound processing and generative composition. The main components consist of custom SynthDefs for instruments such as blip, reverb, and bpfsaw, which generate sound through synthesizers and filters. Each performer will interact with live processing using MIDI or audio inputs, with some sections focused on sound manipulation, dynamic volume changes, and evolving textures.

Live Performance:
Instruments: Synthesized sounds processed in real time, including marimba-like tones and noise-based textures.
Sound Generation: Use the custom SynthDef(bpfsaw) for marimba sounds and other evolving textures.
Effects: Reverb, distortion, and dynamic manipulation with real-time adjustments.

Score: Marimba Loop (Synth: blip) Performers will initiate loops that vary in frequency and duration. These vary in pitch from a 300 Hz base upwards, with random adjustments.

Marimba Variations (Synth: bpfsaw) Dynamic pitch modulation, using Pbind to set randomized note values, pan settings, and filter frequencies.

Sound Layering (Synth: multi) Layer multiple sounds using noise-based effects, producing rich, resonant textures, to be played in sequence.

Technical Documentation (Code Analysis):

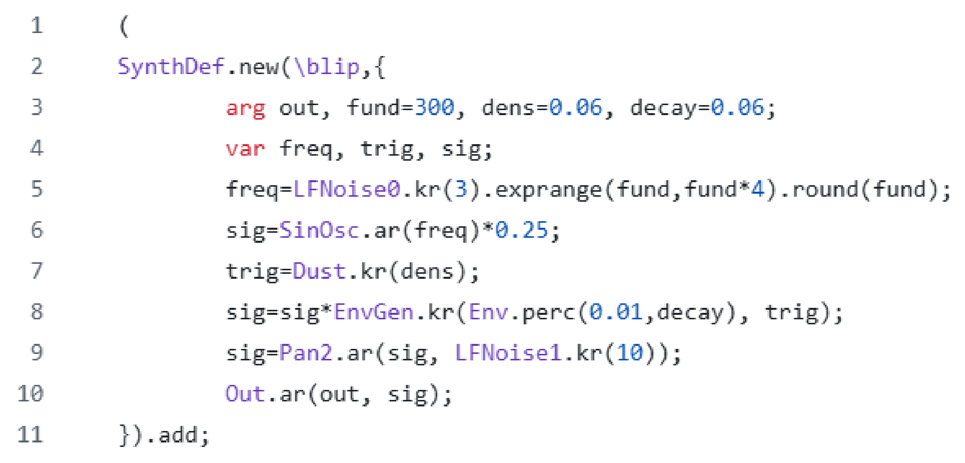
Synth Definitions: Custom SuperCollider SynthDefs for sound generation and processing, including basic tones (sine waves) and more complex effects (filters and delays).
Marimba Sounds: Generated using bpfsaw with modulation of parameters like cfmin, cfmax, freq, rqmin, and rqmax. Multiple layers of marimba-like sounds are played in parallel for a rich, textural effect.

Sound Processing: Real-time effects (e.g., Reverb, FreeVerb) are applied to manipulate the audio dynamically, adding depth and space to the performance. The reverb SynthDef adds atmospheric depth.

Performer Instructions:
Synthesizer Settings: Adjust pitch, timbre, and duration based on the score. Live Manipulation: Performers can adjust the intensity and density of the sound through real-time control of the synthesizer's parameters.
Interactivity: Engage with the surrounding performers and live sound processing to create a dynamic, evolving auditory experience.

Performance Notes: The performance should focus on blending acoustic-like marimba sounds with processed electronic effects, building evolving layers of sound and texture. Engage with both fixed and real-time elements of the piece to add variety and expression to the performance.

SynthDef/ blip:



```
(
SynthDef.new(\blip,{
        arg out, fund=300, dens=0.06, decay=0.06;
        var freq, trig, sig;
        freq=LFNoise0.kr(3).exprange(fund,fund*4).round(fund);
        sig=SinOsc.ar(freq)*0.25;
        trig=Dust.kr(dens);
        sig=sig*EnvGen.kr(Env.perc(0.01,decay), trig);
        sig=Pan2.ar(sig, LFNoise1.kr(10));
        Out.ar(out, sig);
}).add;
```

Figure 1: SynthDef generates a percussive tonal sound.

Description: SynthDef generates a percussive, tonal sound that can be used to create rhythmic elements with evolving frequencies. It combines noise-based control, oscillators, and environmental modulation to create its signature sound. Here's a breakdown of how the function works:

Function Arguments:

out: The output audio bus.
fund: The fundamental frequency of the sound, defaulting to 300 Hz.
dens: Density of the trigger events (density of the sound's "blips"), defaulting to 0.06.

decay: The duration of the envelope's decay phase, defaulting to 0.06.

Frequency Control: freq: The frequency of the oscillator is controlled using LFNoise0.kr(3), which generates low-frequency noise at 3 Hz. This value is then mapped to a range between fund (fundamental frequency) and fund * 4 using .exprange(). This makes the frequency fluctuate randomly within this range, creating a slightly evolving sound. round(fund): Ensures the frequency is rounded to the nearest multiple of the fund.

Oscillator and Signal Generation: sig: A sinusoidal oscillator (SinOsc) is generated at the fluctuating frequency freq. The amplitude of the oscillator is scaled by 0.25 to control the volume of the sound. **Triggering:** trig: The Dust.kr(dens) generates random events at a specified density (dens). This density controls how often the sound is triggered, leading to the ”blip” nature of the sound.

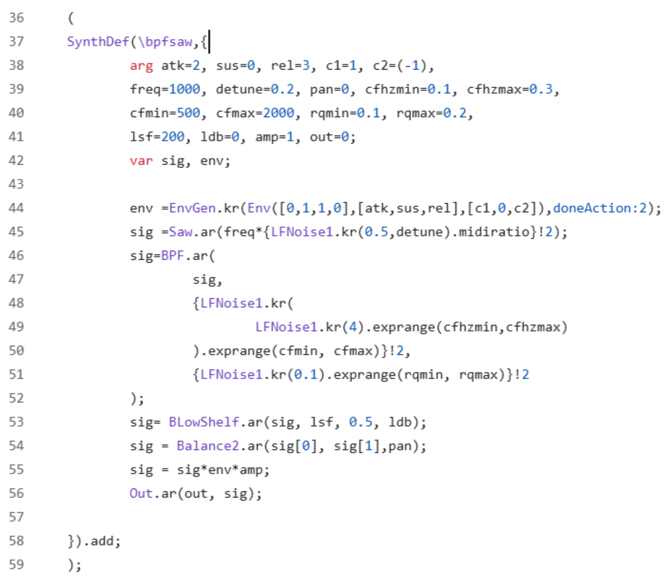
Envelope:

The signal is modulated by an envelope generated with EnvGen.kr(). The envelope is a simple percussive shape (Env.perc(0.01, decay)) with a very fast attack time (0.01 seconds) and a decay time determined by the decay argument. This gives the blip a short, snappy sound.

Panning: sig = Pan2.ar(sig, LFNoise1.kr(10)): The signal is panned using Pan2, where the pan position is controlled by another low-frequency noise (LFNoise1.kr(10)) with a frequency of 10 Hz. This creates a slight panning movement over time.

Output: Finally, the signal is output to the specified out bus using Out.ar(out, sig).

SynthDef/bpfsaw:



```
(
SynthDef(\bpfsaw,{
        arg atk=2, sus=0, rel=3, c1=1, c2=(-1),
        freq=1000, detune=0.2, pan=0, cfhzmin=0.1, cfhzmax=0.3,
        cfmin=500, cfmax=2000, rqmin=0.1, rqmax=0.2,
        lsf=200, ldb=0, amp=1, out=0;
        var sig, env;

        env =EnvGen.kr(Env([0,1,1,0],[atk,sus,rel],[c1,0,c2]),doneAction:2);
        sig =Saw.ar(freq*{LFNoise1.kr(0.5,detune).midiratio}!2);
        sig=BPF.ar(
                sig,
                {LFNoise1.kr(
                        LFNoise1.kr(4).exprange(cfhzmin,cfhzmax)
                ).exprange(cfmin, cfmax)}!2,
                {LFNoise1.kr(0.1).exprange(rqmin, rqmax)}!2
        );
        sig= BLowShelf.ar(sig, lsf, 0.5, ldb);
        sig = Balance2.ar(sig[0], sig[1],pan);
        sig = sig*env*amp;
        Out.ar(out, sig);

}).add;
);
```

Figure 2: SynthDef/bpfsaw , highly flexible and dynamic sound generation model.

Function Definition and Parameters:

atk=2, sus=0, rel=3: These parameters define the attack, sustain, and release times for the amplitude envelope. The attack is set to 2 seconds, sustain is zero (no sustain phase), and release is 3 seconds. This allows for percussive or transient sound shapes, depending on how the envelope is shaped by the other parameters.

c1=1, c2=(-1): These values control the envelope's curve during the attack and release phases. c1 is set to 1 for a linear attack, while c2 is set to -1, providing a decaying release phase.

freq=1000: Defines the central frequency of the sawtooth wave oscillator in Hertz (Hz). By default, it is set to 1000 Hz, but this can be altered in real-time to produce higher or lower pitches.

detune=0.2: This value introduces detuning to the saw wave by applying low-frequency noise (LFNoise1.kr).

The detune parameter controls how much detune is applied to the frequency, resulting in a subtle or strong chorusing effect. The detuning is applied to both channels for stereo modulation.

pan=0: This parameter controls the stereo panning of the output sound, where 0 represents the center, -1 the left channel, and 1 the right channel. The pan value can be modulated to create a more dynamic stereo movement.

cfhzmin=0.1, cfhzmax=0.3: These values control the frequency modulation range of the bandpass filter. The cutoff frequencies of the filter are modulated within this range, allowing for evolving spectral content.

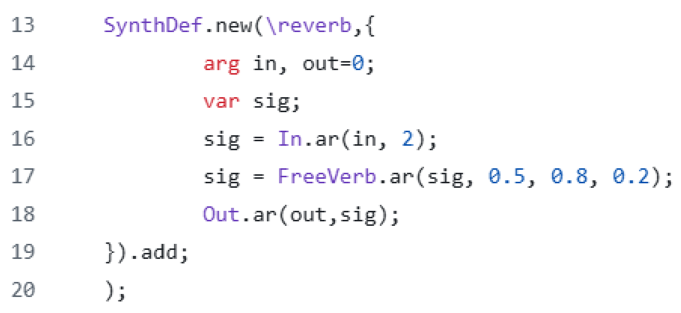
cfmin=500, cfmax=2000: These values set the minimum and maximum cutoff frequencies of the bandpass filter (BPF). The cutoff frequency determines the range of frequencies that pass through the filter, shaping the tonal characteristics of the signal.
rqmin=0.1, rqmax=0.2: These parameters control the resonance (Q factor) of the bandpass filter. Higher values lead to more pronounced peaks at the cutoff frequency, contributing to sharper and more resonant sounds.

lsf=200: The frequency of the low-shelf filter applied after the bandpass filter.

This filter adjusts the tonal balance by cutting or boosting low frequencies below the cutoff frequency defined by lsf.

ldb=0: This value sets the amount of gain applied to the low-shelf filter. A negative value will reduce low frequencies, while a positive value will boost them.

amp=1: This is the overall amplitude scaling factor of the final signal, which determines the loudness of the sound.
out=0: The output bus where the generated sound will be sent. This is typically connected to the default audio output or any other audio bus in Supercollider.



```
SynthDef.new(\reverb,{
        arg in, out=0;
        var sig;
        sig = In.ar(in, 2);
        sig = FreeVerb.ar(sig, 0.5, 0.8, 0.2);
        Out.ar(out,sig);
}).add;
);
```

Figure 3:

Runtime Analysis: To bound the running time

Proof of Correctness using induction:

Claim:

Base Case:

Inductive Case:

-If

-If

We have now exhausted the sample space,