Marimbas Errantes Performance and Technical Documentation

Nehemias Alvarado

January 7, 2025

Analysis:
This electronic work was developed using 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 random note values, pan settings, and filter frequencies.

Sound Layering (Synth: multi) Layer multiple sounds using noise-based effects, producing rich sound 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.

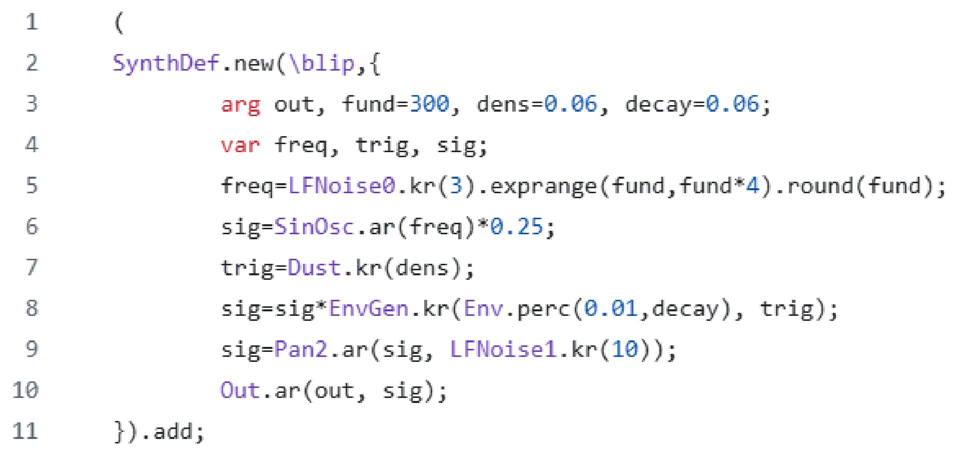
Sound Processing: Real-time effects (e.g., Reverb, FreeVerb) are applied to manipulate the audio dynamically, adding depth and space to the performance.

Performer Instructions:
Live Manipulation: Performers can adjust the intensity and density of the sound through real-time control of the synthesizer's parameters.

Performance Notes: The performer 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 **.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.

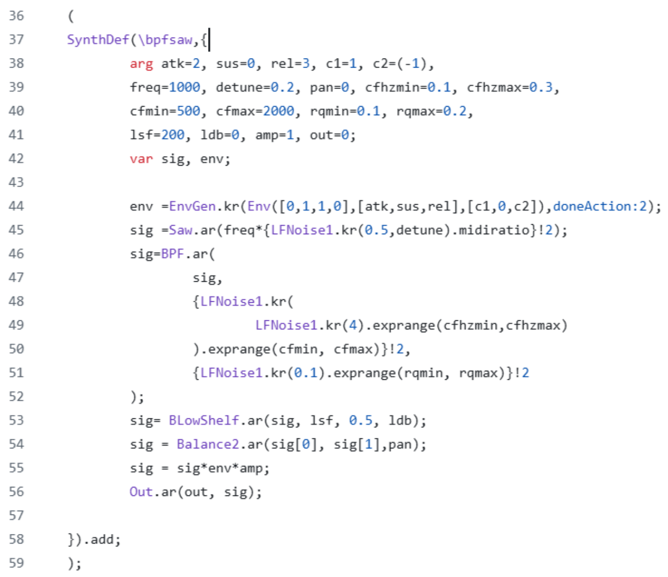
Envelope:

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 , SynthDef is a highly flexible and dynamic sound generation model that combines a saw wave oscillator with an intricate filtering system, creating a rich and evolving sound.

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.

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

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.

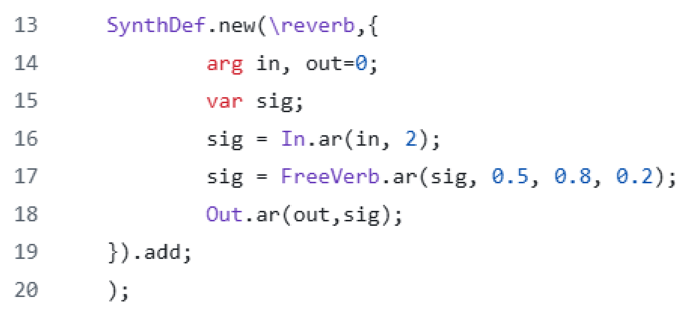
SynthDef/reverb

in: This is the input bus number from which the signal will be read. The in parameter expects a stereo signal (2 channels). out=0: This parameter defines the output bus to which the processed signal will be sent. By default, it is set to 0, which corresponds to the main audio output.

sig: This variable holds the audio signal that is processed throughout the effect chain.

The In.**ar(in, 2) UGen** is used to read the input stereo signal from the specified input bus (in). The 2 specifies that the input is stereo (2 channels). The signal is assigned to the variable sig, which will then be processed through the reverb effect.

Reverb Processing: The **FreeVerb.ar UGen** is applied to the input signal (sig). The **FreeVerb** is a classic reverb effect designed to simulate a reverberant space using four parameters: 0.5: This



```
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: reverb/SynthDef: SynthDef is a simple yet effective reverb effect unit designed for audio processing in Supercollider. It processes an input stereo signal through a reverb effect, using the FreeVerb UGen to simulate the acoustic space's reverberation. The result is an output signal that maintains the spatial depth and richness of the original sound while adding a sense of distance and ambiance.

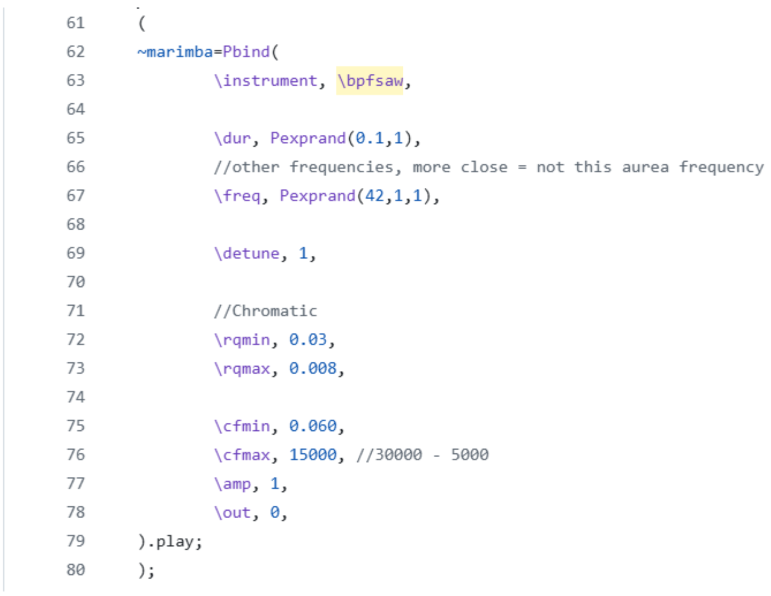
is the **roomSize** parameter, which controls the size of the virtual room. Values closer to 1 simulate a large room, while values closer to 0 simulate a smaller, more enclosed space. 0.8: This is the damping parameter, which affects the high-frequency attenuation over time. A higher value results in more damping, which means that higher frequencies will decay faster. 0.2: This is the dry/wet balance parameter, which determines the mix between the original (dry) signal and the processed (wet) signal. A value of 0.2 means that the wet signal will be less prominent compared to the dry signal. The **FreeVerb UGen** processes the signal, adding reverb and creating a sense of space and **ambience**. Signal Output:

Finally, the processed signal (sig) is sent to the output bus using the Out.ar(out, sig) **UGen**. The out parameter specifies the output bus, and the signal is written to it for further processing or playback.

Description of marimba Pbind:

The marimba pattern is a dynamic and expressive marimba-like sound generator in Supercollider. It uses the **bpfsaw SynthDef** to create a rich, evolving timbre. Key parameters include:

Duration (**dur**): Randomly chosen between 0.1 and 1 second for varied rhythmic patterns. Frequency (**freq**): Random frequencies within the range of 42 Hz to 1 Hz, producing a wide tonal spectrum. **Detune** (**detune**): Resonance (**rqmin** and **rqmax**): Defines the filter resonance range for a sharp or smooth sound texture. Cutoff Frequency (**cfmin** and **cfmax**): A wide range of filter cutoff frequencies (from 60 Hz to 15 kHz), adding brightness and character.
Amplitude (**amp**): Set to 1 for full signal strength. This pattern combines randomness and control to simulate a marimba-like effect with evolving and chromatic tonalities.



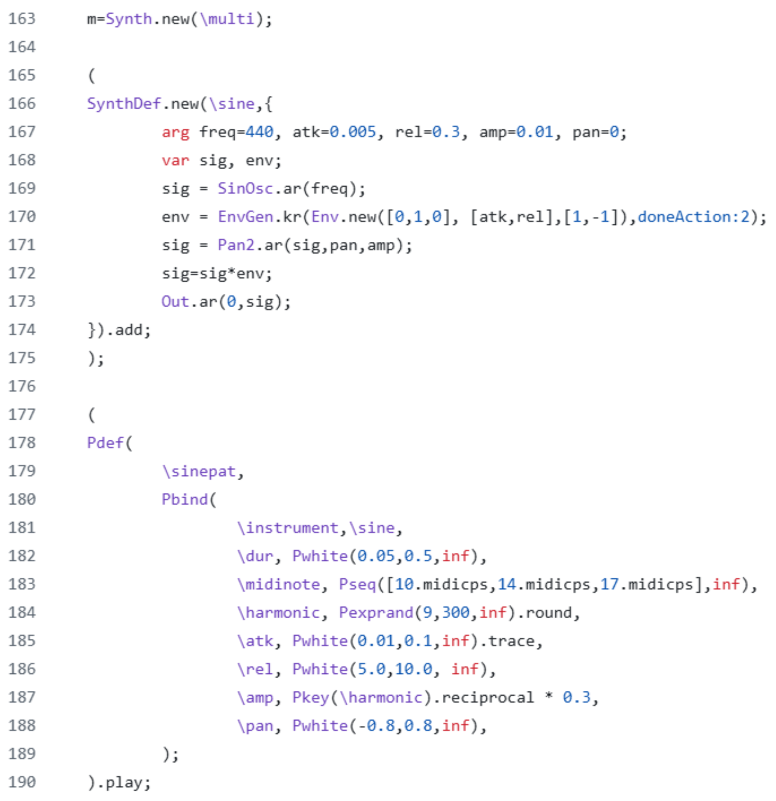
```
(
~marimba=Pbind(
	\instrument, \bpfsaw,

	\dur, Pexprand(0.1,1),
	//other frequencies, more close = not this aurea frequency
	\freq, Pexprand(42,1,1),

	\detune, 1,

	//Chromatic
	\rqmin, 0.03,
	\rqmax, 0.008,

	\cfmin, 0.060,
	\cfmax, 15000, //30000 - 5000
	\amp, 1,
	\out, 0,
).play;
);
```

Figure 4: The marimba pattern is a dynamic and expressive marimba-like sound generator in Supercollider. It uses the **bpfsaw SynthDef** to create a rich, evolving timbre. Key parameters include.

Description of Functions:

Buffer Allocation: A sine wave with harmonic frequencies is stored in buffer b, and an **AIFF** file **(kaqchikel.aiff)** is loaded into buffer c. **Sound Synthesis:** A variety of oscillators **(COsc, SinOsc, Ringz, Decimator, etc.)** are used to create a layered texture, with frequency modulation, noise filtering, and comb filtering. **Effects:** It applies various effects such as **Decimator, LeakDC, AllpassN, Splay,** and Limiter, adding rich movement and distortion to the sound.

Local Effects Processing: A custom **spatialization** technique with **LocalIn** and **LocalOut** processes, including a complex multi-pass effect chain is applied. **Playback:** The final output is mixed, processed through filters, and controlled by an envelope for smooth dynamics. The script is designed for dynamic, rich sound creation, inspired by an experimental approach to sound design.



```
m=Synth.new(\multi);

(
SynthDef.new(\sine,{
        arg freq=440, atk=0.005, rel=0.3, amp=0.01, pan=0;
        var sig, env;
        sig = SinOsc.ar(freq);
        env = EnvGen.kr(Env.new([0,1,0], [atk,rel],[1,-1]),doneAction:2);
        sig = Pan2.ar(sig,pan,amp);
        sig=sig*env;
        Out.ar(0,sig);
}).add;
);

(
Pdef(
        \sinepat,
        Pbind(
                \instrument,\sine,
                \dur, Pwhite(0.05,0.5,inf),
                \midinote, Pseq([10.midicps,14.midicps,17.midicps],inf),
                \harmonic, Pexprand(9,300,inf).round,
                \atk, Pwhite(0.01,0.1,inf).trace,
                \rel, Pwhite(5.0,10.0, inf),
                \amp, Pkey(\harmonic).reciprocal * 0.3,
                \pan, Pwhite(-0.8,0.8,inf),
        );
).play;
```

Figure 5: This Supercollider script generates a complex, evolving soundscape using multiple buffers, oscillators, and effects. The core process involves

Conclusion

This SuperCollider script is a sophisticated blend of traditional synthesis techniques and contemporary sound design principles. The use of oscillators, buffers, and a variety of filters demonstrates a deep understanding of audio signal processing and the creative potential of algorithmic composition. The incorporation of randomness and real-time modulation ensures a dynamic and evolving soundscape, which can be tailored to fit various artistic visions.

By meticulously crafting each SynthDef, the script not only achieves a diverse range of timbres but also showcases the power of SuperCollider as a tool for both experimental and structured music creation. The combination of sound generation and processing, coupled with the thoughtful use of control signals and envelopes, results in a highly expressive and versatile framework.

As an expert synthesis and sound design tool, this documentation reflects the importance of precision and creativity in programming complex audio environments. Future enhancements could include further exploration of spatialization techniques or the integration of external control inputs to expand the interactive capabilities of the system.