In a world where computers are fast enough to produce 44100 samples per second and deceive our ears well enough and we can command a computer to produce any sum of any partials any time, we are given power to virtually create any possible sound. The equivalent of that we can make any possible image by commanding a computer to display some RGB color pixel by pixel in visual production. However even though we got that power, the idea of building from the most infinitesimal level seems overwhelming and also meaningless. We need some kind of brush.
For this random generative algorithm project, I started from considering the fundamental qualities of sequential sound: harmony, dynamics, timbre, and timing. I decided timbre, which I am most interested of, to be the main source of what keeps this sequence of sound “interesting”. So for the ‘brush’, I decided to use the FM synthesis technique, because I found that FM is capable of producing complex timbre with minimal amount of calculation compared to additive synthesis, and potentially of producing infinitely various timbre with just little basic waveforms, which subtractive synthesis is incapable of.
I looked at Native Instrument’s software FM synthesizer ‘FM8’ and Ableton Live’s ‘Operator’ and found out that layering FM is also possible (i.e. making a modulated carrier as a modulator for the next carrier) and makes much more complex timbre. So I implemented an idea of making a matrix of basic oscillators and making them modulate oscillators to their right and their bottom. I also realized that letting each oscillators to have their own envelop that modulates not just their amplitude but also the amount of FM, it makes sound dynamic not only in dynamics but also in timbre.
After setting up the FM matrix(and also a LFO version of that same matrix to add more exciting random modulations to the main FM matrix) and a mixer that will mix sound from each oscillators in the matrix, I let the ultimate carrier(the most right bottom oscillator) frequency, the harmonicity ratio of FMs(I set those up so that the values are pow(2, (some integer -3 ~ 3) + some small random float)), the global multiplier multiplied to all FM amplitudes, and timing of the global metronome object. I chose not to randomize the envelopes throughout time for all oscillators because it already sounded pretty interestingly random and I didn’t want a totally disastrous randomness which is not interesting. I left a preset object so that users can tweak the envelopes and save them, even though no exact same sound will occur twice. Because the matrix mixer gain is randomized, envelope setting only determines rhythm until the next randomization.
Even if sound is not interesting enough at some point, waiting for a few minutes patiently had payoffs. Maybe because randomness is nature-affiliated, playing with this patch was like birdwatching, there were some exciting rare finds.