Evan Magoni
Ian Hattwick and Nicholas Esterer
MUMT 303
4/28/2015

Final Project Report: Generative Music

Goals/Challenges

My initial goals were to have a generative audiovisual piece that used [jit.lcd] and matrix transformations to graphically display MIDI events generated the same patch. I didn't anticipate the level of work that getting even a basically functional and sufficiently modular generative music patch would entail. In retrospect, I'm very glad that I didn't attempt to create a video component also.

My project deviated from my original goals in that my generated set of musical sequences did not end up having as much predefined structure as initially planned. I initially planned to have a larger structure controlling form and repeating elements, but I became bogged down in a lot of the finer implementation details that I hadn't included in my design. As a result, what I have is more like a pattern generator than something that creates a composition. The main thing that I failed to consider implementation-wise was the number of levels of abstraction necessary to make components fully modular—for example, in order to play a changeable sequence of notes with varying rhythm, I first had to index that sequence by sequential numbers and translate those into scale degrees, then feed them through another table. The algorithm itself didn't take very long to figure out, but in some instances implementing the logic did.

Building the synthesis part of the patch did not take very long, but creating the data structures and logic for the polyphonic chord part and even just building a monophonic synth that can play legato MIDI notes properly took more effort than I thought (for example, proper Max implementation of ignoring key-ups if they are not the same note as the last note-on event). In retrospect, in those types of situations, I might have saved some time by referencing preexisting patches.

I mostly overcame the challenges of abstraction using tables and the [zl] object. Tables gave me a way to map sequential numbers to MIDI notes in key, sequencer ticks to steps in rhythm patterns, etc. Another specific problem within the drum sequencer was that [delay] can take metrical time values relative to the tempo of Max's global transport, but not of any other named transports. To get around this, I used [translate], which does have an attribute for calculating time relative to a named transport, to translate metrical values into milliseconds for the delay objects.

While I found writing this patch difficult, I think I am gaining a better understanding of how to implement the basics in Max like loops, iteration, and data structures—there are so many objects that can do similar things, and initially I was going about a very round-about way to do things that are made faster and easier with [uzi] instead of [counter], [z iter] instead of [select] or [t b b …] with a metro, and [route] as a few examples. A few times, I was attempting to do things with logic that become too complex for me to understand, so I just deleted my work and asked myself how I could do it a simpler way. This to my advantage in the example of [p monosynth], an abstraction that acts as a normal monosynth would in that it takes the pitch of the last pressed key and only stops the note when that key sends a note-off, not any of the previously-sounded keys. I realized that I could use [route] as a sort of `if` control structure.

Third-Party Work

I did not reference very much third party work in the building of this project past the design phase. I could have used a better idea of what types of rules are suited to creating i.e. listenable chord progressions. The third party work that I referenced most was Max's own tutorials and object documentation. I was a bit in the dark about using tables for probability distributions (I found it not very clearly documented, so touching on it in class was helpful). I didn't reference anything outside of Max's own documentation.

Improvements

I think I will continue to work on developing generative music. Next time, I'd like for the musical output to have more structural elements/form. Given that my current patch is relatively modular, I think I could extend its current abstractions with multiple patterns to fit inside a larger form-creating structure with larger-scale temporal cues. I would also try to pay attention to and formalize a decent set of rules for chord progressions and melodies—it might be better to start from a very well-established standpoint, like counterpoint, and branch out from there. Alternatively, I could start with a set of commonly used chord progressions and control their probabilities of occurring. After attempting the manual way, I can see why so many people are into Markov Chains for generative music—I suspect they're a very easy way to get a decent-sounding set of rules. I would still prefer to devise my own ruleset. I think I need to experiment a bit more with creating rules and generative music in general, as this was my first foray into it other than our Assignment 2.