FIVE INCITEMENTS FOR ELECTRONIC MUSIC MAKERS

MILLER PUCKETTE
University of California, San Diego
msp@ucsd.edu

To appear in IDEAS SONICAS, Morelia, Mx. Sept. 2020

ABSTRACT

Computers enable the musician to rethink any and all aspects of the way we create and enjoy music. Rather than simply use the computer as a cheaper recreation of pre-digital technologies, to take full advantage of its power we should seek music-making approaches that take specific advantages of what computers offer that the traditional music-making environment doesn't. In this paper I offer five examples of uses of computer technology that came of my own search for new musical ideas that are specifically afforded by computers. These are offered in the spirit of inciting the reader to explore them further.

As far as we can tell, "computer music" was first made in a spirit of fun, not as an attempt to make high art. In an interview with Park (2009), Max Mathews suggests that John Pierce put him up to writing MUSIC, if not exactly as a lark, then not exactly along the lines of Bell Labs's serious endeavors either. Even earlier than that, CSIRAC's musical output was clearly intended as play, not work. Serious computer work, the sort that created output on line printers and punched cards, was generated in the service of science and technology (often warlike), and a bit later, for handling financial ledgers and transactions. It is doubtful that many people in the 1940s and 1950s foresaw that computers would be primarily used today as media devices, replacing land-line telephones, TVs, kitchen timers, and automobile speedometers. It would be particularly urprising to a computer programmer of 1950 to learn that most computers after about 1995 would have audio inputs and outputs.

That there is almost always a computer, and perhaps hundreds of them, involved whenever we make or listen to music implies that all music is now computer music, so that the term "computer music" itself has lost all meaning. This also implies that the heroic period of electronic music, which reached roughly from the time of the Telharmonium until sometime around 2000, is now over and it's time we electronic musicians think of ourselves as normal. It's sad to lose the outsiderness we used to enjoy, but on the other hand it's a comfort to consider that an electronic musician today, unlike, say, Mathews in 1957, can reasonably hope for gainful employment making music.

That heroic period essentially saw the maturation of a platform for music making, including digital preparation of scores, recording, mixing and mastering, audio analysis, processing, and synthesis, and the design of interactive electronic musical instruments. With the one very important exception of recording, this music-making is all done in the same production model as it was in the tenth century. The main thing we seem to have accomplished by digitizing the process is to make it far more flexible and powerful, and more open to participation by people outside the old power structures.

And yet the computer also brings a vast collection of new possibilities for music production, that are novel precisely because they lie outside the performance-space- and recording-studio-centric model. What follows is a description of five ideas, none of them well fleshed out, that serve as examples of what I mean by this. These are all things I've personally tried or am in the process of trying. I mean the following descriptions as incitements: perhaps you, the reader, will figure out how to twist or alter one of them to your own musical ends.

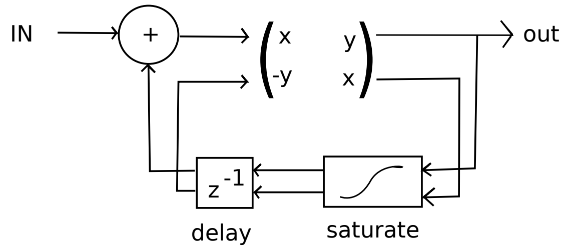

Figure 1. A saturating filter considered as a forced oscillator.

INCITEMENT 1: RELINQUISH CONTROL

Suppose you build an oscillator out of an unstable resonant filter, by pushing its feedback gain to the point of oscillation (and applying a saturation function here or there in the feedback path so that its state stays within set limits.) It being a filter, you can also apply an input signal, perhaps a sinusoid at a frequency different from the resonant frequency. The circuit is as shown in figure 1. To get a resonant frequency of ω radians per sample, we can set x = r cos ω, y = r sin ω, with r, the feedback gain, set to slightly more than one.

This circuit can oscillate either at its own resonant frequency or at the frequency of the input signal, depending on several factors. It can even output more than one possible stable behavior from the same input values depending on the path by which those values have taken previously over time. This is one form of "soft synchronization" of an oscillator, and it was a feature of Buchla's dual oscillator module 258.

Now suppose we make three (or more) of these, and feed their outputs back into their inputs, as in figure 2. Now, in addition to the parameters belonging to the individual oscillators (two apiece), we have to specify a matrix with nine elements, one for every possible feedback path. (If this is an uncomfortably large number of parameters, we could stay within a lower-dimensional subset of the possible parameter combinations).

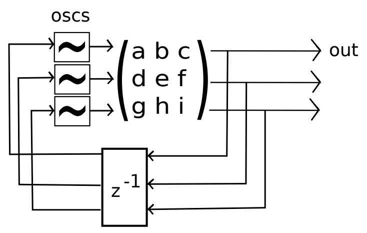

Figure 2.

The result is essentially that the three oscillators are each soft-synchronized to mixtures of all three of their outputs. Depending on the 15 parameters values at any time, the oscillators may eventually stabilize to one or two or three resultant frequencies, or may behave chaotically.

Now consider this as a possible musical instrument, to be played using a physical controller. The result could be similar to what happens when I try to play a violin: we definitely can tell that what I put in is causing the sound output, but on the other hand, imperceptible to the audience is the fact that I actually don't know (and certainly can't control) what kind of sound will come out.

The composer Kerry Hagan and I have used an instrument of this type in a duet we call "Who was that timbre I saw you with?" first presented at NIME 2018 (Blacksburg, Virginia, USA). We map LEAP controller hand skeleton location data (which comes in roughly every 10 msec) to various feedback parameters to make a highly unstable, yet expressive, instrument. A short video excerpt can be found on the web page of our duo Oscillators can be inserted in many points in a classical voltage-controlled synthesis network (either real or in a patchable computer environment), on either an audio or control time-scale (i.e., as waveform or control voltage generators). There is a vast tradition of making semi-autonomous synthesis algorithms that run in real time and are only partially controlled by humans; early examples include the work of Morton Subotnik and Salvatore Martirano. The incitement here is to note that such behaviors can come out of extremely simple networks that are highly pluggable and recombinable.

An earlier version of this idea appears in Puckette (2017) with supporting patches on `msp.ucsd.edu/ideas/icmcl5/` .

INCITEMENT 2: SAYING NO TO RANDOMNESS

When you've been working on a patch that generates a sound palette you're happy with, but when the results coming out of the speaker seem a bit too static, it's a temptation to throw in a pseudo-random number generator or six to create some what Buchla called "uncertainty". John Cage took this a step further, asserting that throws of the I Ching or star charts could be used as streams of information (in the information-theoretic sense) from an unknowable source that might contain mystical properties that would somehow be more authentic than pseudo-randomness. This sort of mysticism hearkens back to pre-enlightenment ideas about music.

One could instead go in the opposite direction and exploit the determinism of the pseudo-random number generation process to imbue the "random" sequences with discernible patterns. If they are too readily discernible the results might become predictable. But there is a spectrum of discernibility on which we can seek points where there is a perceptible signal but not a predictable one. This is what we hear when we listen to very well-crafted classical music: we hear the logic, to the point that we couldn't easily change any individual note to a better one, but nonetheless each new musical moment brings a surprise, even when we've heard the music before.

A pseudo-random process starts with a seed and, at each of a sequence of steps, applies a highly irregular function to the seed to generate a new one. The sequence of seeds probably contains far more bits than we need, so we usually just take a few bits out at each step. The more irregular the function is, the "better" we consider the generator.

So here's a truly bad one: our state is the numbers from 0 to 12, and the function is "add 8 mod 13". Starting with an initial seed of 2, for example, we get the sequence
2,10,5,0,8,3,11,6,1,9,4,12,7,2,after which it repeats.

Despite the obvious inadequacy of this as a random number generator, with a few simple transformations we can make things much more interesting. To start with we could just take 1 for numbers 9 or greater, 0 otherwise: 0, 1,0,0,0,0,1,0,0,1,0,1,0,0,.... Now pick out, say, even numbers, and you have a sort of contrapuntal response: 0,0,1, 0,0,1,1,0,1,1,0,0,1,0,....

For more "bunchiness", if you want it, take the OR of the two binary sequences above, or a 1 if the second sequence has output 1 more recently than the first one has. Now change the three generating numbers 8, 13, 9 and the initial seed 2 and you can sprout a whole patch's worth of not-very-nonuniform sequences that can supply all the parameters you need to fill your parameter space.

For slightly higher complexity, if you find the above too repetitive or guessable, consider that the initial sequence above is just the linear function `x[n]=(8n+2) mod13`, and try a quadratic one of the form `y[n]=(an2+bn+c)` modd. For best results,start out with d a prime number. Incidentally, these are the sequences that control the physical dimensions of Schroeder (1979) diffusers.

Now you'll never have to use a pseudo-random number generator again. And your music might sound less random.

INCITEMENT 3: HIDING PITCHES IN INHARMONIC SPECTRA

I don't know if anyone besides me has tried to make a nice bell sound by writing down an arbitrary collection of frequencies for partials (or using a pseudo-random number generator; see above), but I can tell you that the results will not sound like a nice bell, but perhaps more like a saucepan lid. Most people will agree that saucepan lids don't sound as sweet as well-crafted bells or chimes. But a close look at the spectrum of a real proper bell or two doesn't reward us with an understanding of how to lay out partial frequencies (and decay times, and strike-dependent ranges of likely initial amplitudes) for our own bespoke bell sounds. And more generally, if you want to make a series of inharmonic spectra for purposes other than making a collections of bell sounds, the same question, how to organize the frequencies of the partials, comes up.

Philippe Manoury, in his piece Jupiter for flute and live electronics (1987), came up with an idea: take two generating frequencies f and g, and pretend that we're making an FM tone with modulation frequency f and carrier frequency g. The frequencies of our spectrum are just those that FM would give us: |af + g|, where a ranges, for instance, from -14 to +14, skipping a=0, and taking absolute value to yield nonnegative frequencies. The result contains neither of the generating frequencies f or g (unlike true FM in which the carrier would usually be present). The time-varying amplitudes of the partials are determined in an unrelated way, and the result does not resemble FM except in that its harmonicity or inharmonicity reflects the relationship between the generating frequencies f and g. This class of spectra has two unfortunate characteristics: first, there aren't many different ones; and second, the frequencies are rather uniformly laid out, whereas it might sound more natural to have more irregularly bunched frequencies.

The value g appears many times as the difference between frequencies in the spectrum. Taking note of this, we can consider making other subsets of values |af + bg + ch|, in which a, b, and c are small integers and f, g, and h are three generating frequencies. If we merely allow a, b, c to range freely between fixed limits, the result will again be too regular, and moreover it will also be much more densely clustered than any natural spectrum. But if we then apply a sieve, allowing perhaps 1/6 of the values through, then we would still expect many pairs of remaining frequencies separated by either f, g, or h.

The result is an algorithm Philippe calls "3f", and uses often in his new compositions. The resulting sounds somehow obliquely reflect the three generating frequencies without being perceived as containing them directly. And depending on the ranges of coefficients a, b, c and the type of sieve employed (see previous incitement for some ideas), there is an inexhaustible variety of sounds available.

A realization of the 3F algorithm is available on
`msp.ucsd.edu/ideas/bell-designer/` .

INCITEMENT 4: ANTAGONISTIC MARKOV CHAINS

In some way, for music to function it cannot simply remain at rest. So, for instance, you can't just figure out which of all notes is the sweetest one and play that one over and over. You need, instead, some source of tension or incompleteness to propel the music forward (whatever that means). This is perhaps yet another sense in which randomness has limited musical power–it can bring about perplexity, but not true tension or expectation of resolution.

One idea that might address these concerns is to formulate the process of music composition as a process of optimization, as described in Truchet et al. (2001). Instead of seeing music composition as an attempt to fit into a set of hard constraints (often with either no solutions or lots of them to choose among), we set ourselves the problem of finding a time sequence that optimizes among a suitably thorny collection of soft (that is, breakable) constraints. One way to do this might be by setting Markov chains in conflict with each other.

Markov chains are usually thought of as random processes, but one way to use them deterministically is to consider them as hidden and to use the Viterbi algorithm to find the sequence of Markov chain states that is most likely given a sequence of observations about the states. The observations should contain some information about the states but be incomplete enough to allow for many "solutions" of which the Viterbi algorithm finds us the best one. All we then have to figure out is how to generate the observations.

Now for the incitement: one way to get an interesting sequence of observations could be to set two or more Markov chains the problem of generating the same outputs among them all. There can be probabilistic dependencies between the chains as well as between them and the output sequence. For example, if we are making a tonal melody, one chain could be harmony (I, IV, V, etc) and another one keeping track of our location in the measure. The observable could be pitches, and one rule might be that we're more likely to output a pitch belonging to the current chord if we're on a downbeat (figure 3).

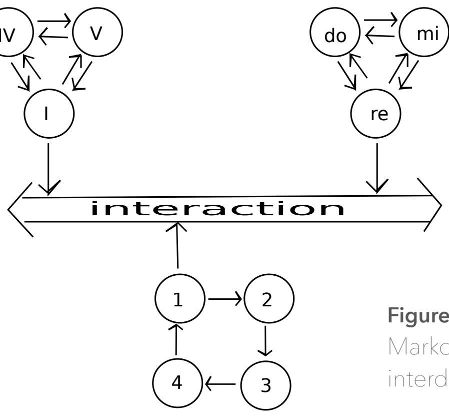

Figure 3. Hidden Markov chains with interdependencies.

We can imagine having chains controlling many aspects of pitch, rhythm, and other less note-oriented musical properties, all negotiating among themselves under the constraint that they have to somehow agree on a sequence of output tokens. We then take whatever "solution" has the highest probability as our output.

I have tried this in some test situations involving up to 5 Markov chains, with various kinds of interdependencies, and managed to output non-repeating sequences up to about 100 tokens long (being a finite-state process, the thing is doomed to repeat sooner or later). My guess is that things will start to get interesting when the size of the state space (the product of the state spaces of all of the chains) reaches a million or so, at which point we should be able to output sequences that take more than a thousand tokens to repeat. (This also happens to be the point at which the computation time might start to become onerous). I plan to keep working on this idea; it's too early to know whether or not it will pan out.

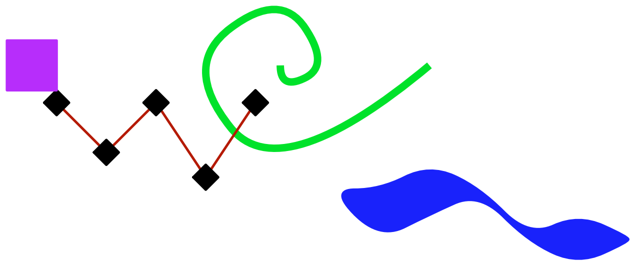

Figure 4. Visualized data structures in Pd. This is from the help window for Pd's "plot" object.

INCITEMENT 5: SCORE AS VISUALIZATION OF DATA

My last incitement is the most ambitious and least certain of yielding useful results. It is an old idea, traceable to the SSSP project of Buxton et al. (1985) via Animal by Lindemann and de Cecco (1991). It's also my original aim in writing Pure Data.

Although more people think of Pd as a Max/MSP clone, that aspect of it is only incidental - the Max/MSP paradigm, somewhat simplified, seemed the most sensible way I could offer real-time interactive audio and other media to what is essentially a different idea. That idea is reflected in the "4.data.structures" examples in the Pd documentation, but is still largely unrealized. It is, essentially, to make an environment for writing scores on a computer that would not be inert documents like classical scores, but instead would be active data structures, capable of updating themselves, communicating with real-time environments during performance, and of being viewed and edited in multiple ways. One example from a help window is shown in figure 4.

The data being visualized can be read and written by Pd objects. This would seem to be a very powerful way to make electronic music scores, but many problems remain. For example, classical musical valuse such as pitches and rhythms are much harder to show and/or edit in this sort of representation than they would be in a common practice notation editor; and, like it or not, these two aspects of a musical sound are often the most salient ones. Second, things tend to collide in this representation since the position of objects is part of their numerical data—to move them out of the way to look underneath would be to change them. Third, a graphical programming environment such as Pd is badly adapted to scouring heaps of data. Reading and writing the data from Pd, via the "pointer" object and its relatives, is an unwieldy and sometimes baffling process.

This idea could use some fresh thinking.

REFERENCES

Buxton, W., et al. 1985. "The evolution of the SSSP score-editing tools." In C. Roads and J. Strawn, eds. Foundations of Computer Music. Cambridge: MIT Press, pp. 376–402.

Lindemann, E., and M. de Cecco. 1991. "ANIMAL—a Rapid Prototyping Environment for Computer Music Systems." Computer Music Journal 15(3):78–100.

Park, T. H. 2009. "An Interview with Max Mathews." Computer Music Journal 33(3):9–22. Puckette, M. 2017. "The Sampling Theorem and its discontents." ICMC Keynote talk,

reprinted in Array Magazine .

Schroeder, M. R. 1979. "Binaural dissimilarity and optimum ceilings for concert halls: More lateral sound diffusion." The Journal of the Acoustical Society of America 65(4):958–963.

Truchet, C., C. Agon, and P. Codognet. 2001. "A Constraint Programming System for Music Composition, Preliminary Results." In Seventh International Conference on Principles and Practice of Constraint Programming (Paphos).