History of Serialism in Computer Music

Serialism has been utilized as a generative music structure for much longer than computers have existed. Early examples of serialized musical structures come from Musical Dice Games by such composers as Mozart, wherein dice results would determine which measure was played from the bank of available measures.

These games would often have a predetermined harmony, formal structure, and meter, meaning that the dice would only determine the figuration of notes.

The earliest example of music generated by a computer is usually attributed to the ILLIAC Suite generated at the University of Illinois in 1957. This system used operations within sets of rules to create music. For instance, the first movement of the suite focuses on generating and focusing on a cantus firmus and the second generates four voices which operate within various rules.

The unified theme of these systems of automated music generation is the heavy utilization of rules. This is a common thread for most music that is at all algorithmically generated. By utilizing a combination of rules and randomly generated elements, one is able to control the "surprise factor" versus the level of cohesion of a piece of music.

Serialism as a Coding Structure

Serialism as a musical structure lends itself exceptionally well to code largely because it provides an easy set of rules within which to create difference. By serializing and listing all elements, one is able to easily assign them to values and events in the piece, which can then be sequenced to create a cohesive piece of music.

The idea of total serialism benefits from a computer music environment largely because of the degree of specificity to which computer generated music can adhere. For most pieces of music, it is necessary to adhere to patterns and structures which appeal to musicians, and are modeled after the way that our brain can interpret music, but serialism both as a compositional approach and as a method of experimentation, is modeled after structures which were not intrinsically developed with the human brain in mind.

Part of the appeal of *Structures* by Pierre Boulez, and other pieces in my paradigm is that music which was random to such an extreme degree was not accessible or available for the reasons that I have mentioned, so when the total serialized sound was created, the sound of "calculated randomness" of every note and dynamic was new. Nowadays, however, it is exceptionally easy to access randomness in music because we have the ability to synthesize sounds and easily create many random values quickly, so that anyone with a basic understanding of coding now has easy access to what before took hours of calculated work.

In a way, total serialism is a system for bypassing human decision-making, which adheres to identifiable patterns, by giving total control to a system. Now, with computers, it is easy to find a system to give total control to.

The Experiment of this Software

The experiment of this software is to determine if there is a real difference between total order/serialism, and randomness. The human brain can record 10 pieces of information when hearing it for the first time as the generally agreed-upon limit of working memory, so to human perception, it follows that the difference between a generated system which consistently utilizes units of size 12, and a system which is generated with infinite randomness would not be particularly distinguishable. I wanted to test this by allowing one to listen to music that would either be completely serialized or completely random without informing the audience of which is which.

With all other factors constant, in this way it would be possible to identify if there really is a serialized sound distinguishable from randomness, and perhaps even what it is.

What I found, however, was that in listening to the generated music, I did not lock in to understanding the underlying patterns of the music, but rather locked on to side effects of the generative mechanisms of which I was aware. For instance, if there was a repeated note, I would know that it likely was not total serialism. In future versions of the software, I could restructure the code to prevent the randomly generated music from allowing repeated notes, which might make the sounds more indistinguishable.

Additionally, because all I was doing was reading the same 12-tone row in 12 transpositions, I focused on intervallic content as an identifier of the music generation. This is closer to locking into the underlying patterns, and I think exemplified an important principle of automated generation, being sort of the idea of Occam's Razor. When presented with a complicated system of generation, my brain immediately locked onto the simplest patterns that I could find in order to understand. I imagine that this idea is important for serialist composers to understand if they want their music to hint at a larger system: the multiplication of factors is not always internally done by the human brain when understanding large systems.

Mechanisms

The software is fairly simple. It is coded in MaxMSP, which is not agreed-upon to be great for large data structures, but is functional for these purposes. Data is processed in the following ways:

- Pitch is stored as MIDI note values. This is especially convenient because note value 0 is a C in MIDI, so note value 12 is also a C, and so on. This means that Normal Order is directly convertible into MIDI with little to no translation.

- Rhythm is stored as a value between 50 milliseconds and 800 milliseconds, meaning that that is the duration between the onset of notes.
- Articulation is stored as a fraction between 0.05 and 3. This is the "percentage" of the rhythm that the note remains active, meaning that the lower that it is, the more staccato, and the higher that it is the more tenuto. This allows for a degree of precision which is not achievable by a human performer.
- Dynamics is stored as velocity values 1-127. This is already the built-in system for MIDI.
- "Registration" is calculated as a whole number 1-7. This value is multiplied by 12 and added to the "Pitch" value. This means that the octave equivalent-pitches are preserved, whereas the register changes separately.

These values are then calculated in 3 ways:

1. Randomly - Wherein each value is calculated completely randomly for each note within the given bounds.
2. Serialized - Wherein a matrix in a system identical to the one used in Pierre Boulez's *Structures* is generated, and values are generated from there (see video).
3. Markov chain - Wherein the values generated by the "serialized" generation are fed into a Markov chain, which is a simple system which calculates proceeding values in terms of probability from an input list. For example, given the list 1 3 4 1 2 1 2, and the value 1, there would be a 33% chance of an output value of 3, and a 66% chance of an output of 2. In this way, there is a smooth system of order and chaos which sets the generative units in between the singular units of chaos coming from total randomness, and the lack of chaos coming from total serialism.

Running the Program:

The program is attached here, and can be run on MaxMSP, which can be downloaded for free to run external files.

Sources:

Lecture by Kemper, Steven on October 29, 2025 at Oberlin Conservatory. "Historical Perspectives on Computational Creativity and Generative AI in Music."

Sandred, Örjan; Laurson, Mikael; Kuuskankare, Mika. "Revisiting the Illiac Suite – a rule based approach to stochastic processes." Accessed 2025. https://www.sandred.com/texts/Revisiting_the_Illiac_Suite.pdf.