REFLECTIONS ON LEARNING LIVE CODING AS A MUSICIAN

May Cheung
LiveCode.NYC
maychives@gmail.com

ABSTRACT

Live-coding music has become a large trend in recent years as programmers and digital artists alike have taken an interest in creating algorithmic music in live settings. Interestingly, a growing number of musicians are becoming live coders and they are becoming more aware of its potential to create live electronic music using a variety of coding languages. As a musician who live codes music, it became apparent to me that my process in creating musical structure, harmonic ideas and melodic ideas were different than live coders who came from a programming background. Using the live code environment Sonic Pi (based on the programming language Ruby, created by Dr. Sam Aaron) as a reference, we argue the parallels and differences between the experience of live coding music versus the experience of performing and playing music. In rare occasions, the dichotomy of the two worlds converge as we present the work of Anne Veinberg, a renown classically-trained pianist who uses a MIDI keyboard as an interface to code through a program created by composer-performer Felipe Ignacio Noriega.

1. INTRODUCTION

After spending many years as a musician from a jazz performance background, I was excited to try a new method of creating music. Introduced to a live coding program called Sonic Pi through friends in January 2018, I have had many opportunities to live code and present in front of audiences: from sizable nightclub crowds to attentive teenagers during their coding summer camp at NYU. After being active in the live code community in New York (livecode.nyc) for almost a year, I noticed that many live coders came from a programming background rather than a professional musician background; many of whom made fantastic, beat-driven, algorithmic and abstract music but perhaps were not coming from a harmonic or melodic approach to their music creation.

Coding music has helped me merge the abstract (ideas) with the concrete (code) and has also helped me become more curious about the possibilities of sounds and synths in music production. Live coding is akin to “live-composing”; lines of code and algorithms are created by live coders that generate music in real time. Coding in real time allows the "coder-composer” to hear the results automatically - sometimes the code crashes or does not work which is noticeable right away. If a musician composed a multi-instrumental piece by hand, which a lot of musicians still do, they would need to wait until rehearsal to play it back to get the full idea of what their piece sounds like. However, if they wrote their piece using a music notation program such as Sibelius or Finale, they could hear what their piece sounds like once they play it back, yet still, certain notes can slip through the cracks and go unnoticed until a band member brings up the error. Or perhaps, a part that a composer wrote for a specific musician may not be feasible to play in real life, depending on the competency of the performer. A hyperbolic example would be: a composer wants a clarinet part to be played from it’s lowest note, D3, to the highest, C6, oscillating in quick succession, without any errors for 32 bars at 190 beats per minute; the musician would nearly faint because humans have a limit since their bodies can only handle so much. The efficiency, accuracy and immediate response that one gets from coding music truly changes the way one goes about creating music in real time.

When comparing the experience of live coding versus using a DAW (Digital Audio Workstation) such as Logic Pro or Ableton Live, a musician would spend more time using a MIDI keyboard or an actual instrument to create music through an audio interface as opposed to using keystrokes to create a line of code that would generate music. From a practical perspective, this also means that the more instruments a musician uses, the more space consumed in the workplace; live coding requires a laptop at its most basic and necessary level but is not limited to speakers, headphones or a computer screen, for example.

I was initially intimidated by the lines of code that would appear on projection screens at performances. Numbers would change suddenly. The syntax looked completely foreign. What is “`sleep`”? Or how about a string of (what appeared to be) random numbers and letters? I recognized *some* words that gave me a sense of relief such as: “synth”, “sample”, “reverb”. Eventually, I gave live coding a chance and dove into Sonic Pi. This new endeavor challenged my way of thinking about language as a whole in terms of structure, nuances (for example: a misplaced comma in a line of code would render it defunct), and general order to carry out the intended result and/or meaning. Coding music also challenged me intellectually - it

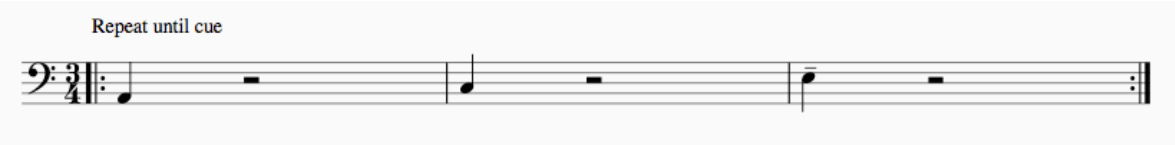
forced me to strategize differently than I would if I were playing an instrument. I soon began to draw parallels and differences between live coding and modern western music having had the experience performing in both realms.

2. PARALLELS

Through my own experiences in live coding and performing music, I discovered many similarities between the two, the first being the act of associating meaning to symbols or syntax and the second being the method in which I prepare myself for a performance, whether it is through a live code or a musical setting. The figures below illustrate the associations I make when coding in Sonic Pi. These "live loops" are like bars of music enclosed with repeat signs:

```
live_loop :letsgo do
  use_synth :dpulse
  play :a3
  sleep 3
  play :c3
  sleep 3
  play :e3
  sleep 3
end
```

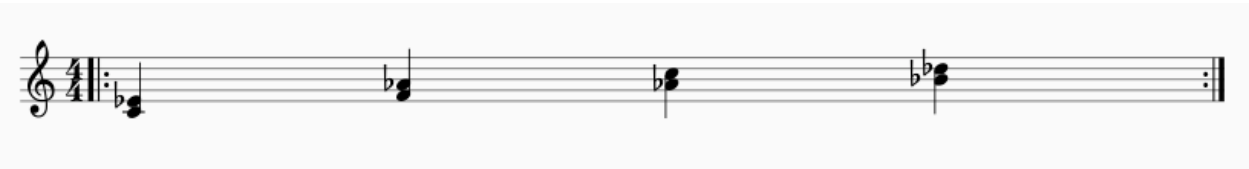
Translates to:



Furthermore, this code:

```
2.times do
  play 63
  play 60
  sleep 1
  play 68
  play 65
  sleep 1
  play 72
  play 68
  sleep 1
  play 73
  play 70
  sleep 1
end
```

Can be visualized in music notation like this:



These translations from code to music, or vice-versa, make the process of live coding easier by association as a musician. More often than not, I hear a rhythm or melody before I code it.

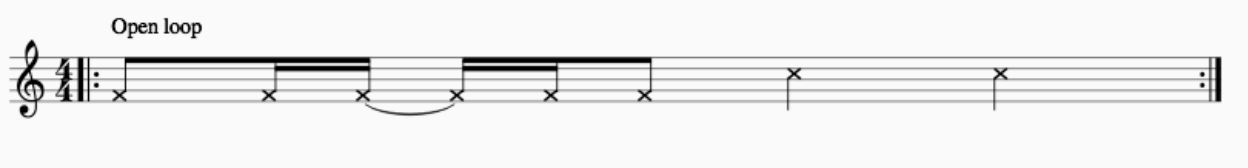
The method I use to prepare for either a live code show or a musical performance are the same. I practice, I memorize and I test. If I am about to do a live code show, I practice executing my whole Sonic Pi set at least five times over several days leading to the performance from memory. Then on the day of the show, I test myself to see if I have retained the process and flow of my set, starting from a blank "buffer". Once I have the code memorized, I take calculated risks and improvise from there. My training as a musician has taught me that the act of practicing is important, especially when it comes to learning new languages: spoken, written, read or coded. I assume that once I attain fluency in Ruby, which is the language that Sonic Pi is based off of, I would not have the need to rely on memory to regurgitate a live set on a blank slate. As Fabia Bertram notes, in her paper, 'Learning Elementary Musical Programming with Extempore: Translating Arvo Pärt's *Fratres* into Live Code Snippets', "the student will finally have a coding performance that can stand on its own while having practiced or improved, or even acquired new programming skills." (Bertram 2014, 3). I must note that not all coders perform this way. Nearly every live coder has their code in order and ready to go before going up on stage. They can change numbers, variables, or comment out sections as they go along during their performance which can prove to be helpful and less stressful.

3. DIFFERENCES

There are levels of differences between coding music and creating music. The most obvious one being that visually, code is expressed abstractly by using the modern Roman alphabet, numbers and keyboard symbols. Written music, however, has its own hierarchal structure by using notes, bar lines, symbols and numbers. In live coding, one must "think in the language" so to speak; we are dealing with syntax after all. In Sonic Pi specifically, there are two ways to code notes: using MIDI note numbers (61 means C#, for example) or by scientific designation (A2, A3 for example). The following excerpt illustrates these visual differences, which result in the same rhythmic pattern:

```
live_loop :clap do
  sample :tabla_ke1
  sleep 0.25
  2.times do
    sample :tabla_ke1
    sleep 0.125
    sample :tabla_ke1
    sleep 0.25
  end
  2.times do
    sample :tabla_tas1
    sleep 0.5
  end
end
```

Versus:



Note also the direction in which this rhythmic passage is read: the code is read top - down (reminiscent of Asian languages) and the music, left-to-right (Western/European). Musical notation has its place in history - widely used and understood, whereas the coding signifies an enormous leap from music notation and creation. As Thor Magnusson points out in his paper 'Code Scores in Live Coding Practice', "in the symbolic

notation for computer music composition and performance, we encounter an important difference in the human and the computer capacity for interpretation: the human can tolerate mistakes and ambiguity in the notation, whereas the computer cannot" (Magnusson 2015, 1). Here, we are presented with the fact that music notation allows for interpretation, whereas coding allows for specification.

The second major difference between live coding and performing live: one can "play" multiple instruments at the same time without physically moving towards each respective instrument on stage. This aspect of live coding serves its practicality and gives rise to the infinite possibilities that one has when coding on stage to a live audience. One can change instruments at the stroke of a few keys. Contrast this practicality to a live music show gig where a truckload of instruments would have to be driven to a venue. Also, from a mixing standpoint, one can also manipulate the reverb ("`with_fx :reverb`") in real time whereas a mixing engineer would be in charge of the overall sound using a soundboard at a live music show.

```
live_loop :ok do
  use_synth :tri
  with_fx :reverb, mix: 0.6, room: 0.7 do
    use_synth_defaults amp: 0.75, attack: 0.4, decay: 0.25, sustain: 0.6, release: 0.2
    play choose([ :a6, :g6, :c6, :e6, :d6, :b6])
    sleep 3
  end
end
```

Another practical aspect I discovered was the ease of coding rhythms in Sonic Pi versus writing them on paper, practicing and executing them. I felt more inspired to discover world rhythms and test them out using code. For example:

```
s = 0.125

live_loop :claveone do
  sample :bd_tek
  sleep 4 * s
end

live_loop :clavetwo do
  sleep 3 * s
  sample :sn_zome
  sleep 3 * s
  sample :sn_zome
  sleep 2 * s
end
```

This is a Reggaeton beat; "`claveone`" is a basic four-on-the-floor bass drum pattern. "`clavetwo`" is the clave pattern that juxtaposes the bass drum with a 3+3+2 rhythm, which is also called a "tresillo" rhythm, stemming from Caribbean roots. Imagine other world beats one could replicate: Hungarian, Gamelan, Indian tabla beats (maybe even coded with drum kit samples) - the list goes on and on.

Live coding requires a different mindset when it comes to structuring musical ideas. This is an example of the process I use for a live code session to garner some ideas: I'd start by laying out live loops with drum beats, then think of a key that I deem suitable for the piece or whatever the moment calls for.
Next, I code a bass line based on the chosen key, then add some chords, followed by a randomized scale or mode that complements the chords. Essentially, I am rehearsing with four instruments in one sitting. Contrast this to my songwriting process: I tune my guitar, I have my phone nearby to capture any ideas and start with simple chords to conjure up any verses that may come to mind. Maybe a fictional story would write itself, or, I might come out of the session with just a few verses and a chorus. Live coding allows for multi-track experimentation whereas songwriting in its most traditionally western definition is usually a one-dimensional experience. Even if we compare a live coding session to using Garageband, for example, we would not be able to randomize notes or pan left to right automatically. Also, coding encourages us to

think of music in a linear way with the use of formal, artificial languages - artificial in that they are constructed by individuals, as opposed to stemming from an organic cultural process that comes as natural languages do (McLean 2011, 17). Through live coding, we are able to explore the infinite possibilities within music composition. For example, one could code a bass drum beat at 200 bpm while a synth plays jazz changes and a randomized melody - all with accuracy and precision for however long one wishes this passage to loop. Tones can also be altered, resulting in microtones that could serve contemporary composers as well as Indian or Balinese or even Baroque musicians (who use A415 as opposed to A440). There is simply no barrier limiting one's musical creativity when one is live coding music.

One of the many aspects of live coding that sets apart from music notation is the possibility for randomization, which allows the computer to "improvise". Jazz musicians, for example, are able to read a chart and improvise upon the changes on the sheet music but are rarely given specific notes to improvise with, since notes are implied in the changes. Take a look at the following example:

```
live_loop :lowend do
  if one_in(6)
    use_synth :pretty_bell
    with_fx :reverb, mix: 0.7, room: 0.8 do
      play :d3
      sleep 0.75
    end
  else
    use_synth :tri
    play choose([ :g3, :b3, :d3, :fs3, :e3, :c2, :a3, :g2]), pan: rrand(-0.5, 1)
    sleep 0.25
  end
end
```

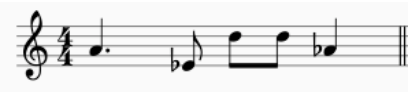
The line "`play choose`" followed by a series of notes states that the line of code will randomly play the notes selected by the coder. This command adds new possibilities to compositions that are less likely to be explored in traditional notated music. The true/false (`if/else`) statements insert a sense of controlled spontaneity rendering it more musical and human-like. Furthermore, the command "`pan: rrand(-0.5, 1)`" states that these notes (played using the synth "`tri`") should be panned from left to right, randomly, within a range (the first "r" in "`rrand`" means "range") between halfway (-0.5) in the left ear all the way to the furthest right (1). This is akin to a DJ, mixing his set live on the dance floor. In the same vein of club settings, getting an audience to move during an Algorave is imperative, since this is the most engagement you will most likely achieve during a set since there is seldom opportunity for human connection.

Artist-audience relationships between live coding and performing music reveal that human connection is lost in Algorave settings. In the twenty-something live code performances that I have attended, there is hardly any introduction, eye contact or human connection between the artist and the audience. Simply, the live code artist gets on stage, plugs an HDMI cable into their laptop, opens it up, finds their code, and presses a button to run their code. After setup and initiation of the code, the audience gazes at the projected code while heavy drum samples or noise blasts away through sub woofers. Since live coding sets last anywhere between 15-40 minutes, the audience is subjected to a longer performance before applauding (the irony is that this is also the case in classical music concerts) which therefore demands a longer attention span from the audience. In a conventional pop or singer-songwriter set, songs are typically 2-5 minutes, maximum. Less attention span is required from the audience and there is more respite between songs for both the performer and audience member. When musicians perform, they are emotionally vulnerable - their thoughts, their words and their music are all laid bare to the ears of the audience. In my opinion, the interaction between artist and audience is essential to a performance as it helps to maintain the audience's attention throughout the show. "How's it going [city]?" or "This next song [explain what the song is about, tell a story related to the song]" are standard phrases used before, after, or between songs. The human connection in conventional performances are more prevalent and therefore, could serve as inspiration to the live code music community even though the medium live coders use are artificial

languages. As TidalCycles creator Alex McLean states “Computers give us privileged access to the digital realm, but we must not lose sight of the analogue, because humans experience and interact with the world as an amalgam of both.” (McLean 2011, 18)

4. CONVERGENCES

Broadly speaking, the “cross-pollination” between musicians and live coders rarely happens. However, after attending my first ICLC conference in Morelia in 2017, I met other musicians who were immersed in the world of live coding, namely, classically-trained pianist Anne Veinberg, who presented her piece CodeKlavier in which she combines live coding with piano performance. In her performances, she uses an 88-key MIDI keyboard as an interface to send commands to a program that coder and composer Ignacio Noriega created. Anne Veinberg, who obtained her Bachelor in Music from the University of Melbourne and Master in music at the Conservatory of Amsterdam, plays the piano in various capacities - from classical performances to live coding music, sending commands to her laptop while playing specific sequences or singular notes. With an acoustic-MIDI piano, specifically a Yamaha Disklavier, Veinberg plays a series of notes that translates into code through a program created by fellow live coder and composer Felipe Ignacio Noriega. For example, if she plays this passage on the piano:



the word “`hello`” would appear quickly on the line of code on the projector screen. The result is astounding, as the audience can see the code being “typed” by a piano - not the conventional way of coding as one would predict.

5. CONCLUSION

The possibilities and sheer accuracy of what live coding brings to the world of music is necessary and important for the evolution of music itself. Live coding as a musician enriches one’s experience of creating music in that the possibilities are infinite and that it helps to further one’s creative potential. A musician’s prior knowledge makes learning live coding easier with the discipline and structural understanding involved in both coding and performing. With this prior knowledge, we find that the structures between code and musical notation are similar as well as the method that comes with preparing for a show. On the contrary, we also find that music notation gives way to interpretation, whereas code acts as specification; the act of live coding requires little if any physical movement between instruments; musical structure and organization of ideas are developed differently; calculated randomization makes the impossible (by musical notation standards), possible. Also, the disparity in artist-audience relationships between live code settings and music performance is apparent as live code performances lack human interactions. Finally, we find the union of the live code and music performance world made possible by musicians such as Anne Veinberg as she combines piano performance with code, furthering the future potential of music technology and music creation.

REFERENCES

Bertram, Fabia. 2014. “Learning Elementary Musical Programming with Extempore: Translating Arvo Pärt’s *Fratres* into Live Code Snippets.” University of Cologne, Germany.

Magnusson, Thor. 2015. “Code Scores In Live Coding Practice.” University of Sussex.

McLean, Christopher Alex. 2011. “Artist-Programmers and Programming Languages for the Arts.” PhD diss., University of London.