EXPLORING LIVE CODING AS PERFORMANCE PRACTICE

Konstantinos VASILAKOS [1]

Abstract

Live coding is the dynamic process of modifying the source code of a running software that generates sound or visuals (Collins, 2011; Collins et al., 2003; Collins & Escrivan Rincón, 2011; Nilson, 2007). It is an emerging laptop performance paradigm that is used in many art scenes and research practices in the digital arts and contemporary musical scenes, such as Computer Music, Electroacoustic performance and the broad landscape of Sonic/Sound Art, as well as in the Algorave community, a live music idiom broadly referring to the creation of dance music using live coding to hack music which interacts with visuals. In the context of Electroacoustic music, it provides an ideal interface for experimentation as it allows one to improvise the structure of algorithms that constitute the sound synthesis engine of a computer-based musical environment, and thus releases the performer from constraints and limitations from fixed interaction possibilities. This not only relates on the interactivity capabilities of a digital system, but also has a great impact on the higher level musical characteristics of a composition, as opposed to other paradigms of live electronic music. This artistic practice, enacted by the technical act of programming, is deployed by the coder/performer as a means to create indeterministic sonic manipulations informed by his/her aesthetic musical decisions during improvisation. Interestingly, live coding also contributes to the question of causality in the performance of electronic music. One can argue that live coding addresses this issue by inviting the audience to engage in the live process via visualizing the alteration of the structure of the code in real time, enhancing the sense of liveness during improvisation. While live coding merely indicates the *techne* of devising sound algorithms "on the fly," it spans many diverse ecosystems in a live electronic music landscape. Live coding also heralds the emergence of dedicated artistic communities, such as the Algorave and Toplap[2] organization, referred as the "home of live coding". Toplap hosts new findings and projects, while a dedicated conference named International Conference on Live Coding is inviting new research internationally, highlighting new tools and practices in the field. The author will report his own research in the field of live coding, reflecting on two personal ongoing projects: the Istanbul Coding Ensemble (ICE), a laptop ensemble working in the field of dynamic programming and improvisation using networked music systems, and an Algorave[3] side project given the alias Chunk No_Reace.

Keywords: Live Coding, Online Music Performance, Live Coding Ensemble

Introduction

Live coding is an emerging performance practice in the field of Computer Music, Electroacoustic Music and the broad Sonic Arts landscape: A real-time performance paradigm of creating algorithms that produce music or visuals live and using improvisation as an impetus for real-time improvisation generating works for clubs, academic events and festivals of electronic music.

1 Dr. at Istanbul Technical University, Advanced Studies in Music Department, konstantinos.vasilakos@gmail.com
ORCID: 0000-0002-3058-0048
Day of Application: 15.04.2021 Acceptance Date: 03.10.2021
2 https://toplap.org
3 https://algorave.com

The performer(s) write algorithms in a programming environment whereby they interact with the produced sound by altering the code on the fly. This live process is usually projected onto a screen to let the audience follow the changes and the development of the code in conjunction with the musical/visual outcome. Live coding is a performance paradigm in Computer Music, but not limited to its genre's characteristics, as it extends to other electronic music styles, such as IDM, Glitch and Electronica and visual arts. Other examples show that live coding is also used for other art forms, such as in choreography (Sicchio, 2014).

In contrast to other contemporary electronic music genres, such as the Acousmatic tradition, live coding shows the audience the moment that the music is being created. Through the projection of the code, which is shown on a screen during the performance, the listener is able to watch the performer's manipulations and to presume how their actions are influencing the overall sound. From this perspective, the audience can understand the presence of the live performer through the code and its development during the musical discourse—something that in other Electroacoustic music practices is absent. (However, one could argue this deliberate neglect of the live presence is an essential characteristic of the Acousmatic tradition).

Kairotic coding seems to ignite an interesting discussion and theoretical interpretation of the concept of improvising with code live. The Greek word *Kairos* (weather or time), in the context of live coding, could be interpreted as the right moment to intervene and take decisions of altering the software at hand while it is running (Cocker, n.d., 2013, 2018).

Some interesting views of live coding include the analogy of it with the process of problem solving ("Behind the Screens: Marije Baalman," 2020; Nilson, 2007). Nilson (2007, p.1) goes further, to argue for an interesting connection with the early days of mathematicians who made their living by challenging other mathematicians in public in order to resolve mathematical problems of the time.

> Some Authors [...] argue that the tournament on cubic equations between the two Italian mathematicians Nicolo Fontana Tartaglia and Antonio Maria Fior about 1539 might be considered an early Live Coding performance (albeit it lasted for several weeks and is thus not directly comparable to today's short-lived performances). (Zmölnig & Eckel, 2007, p. 1)
>
> But the excitement of coding in front of an audience is interesting. I've given presentations to people who work in ICT, and they are quite nervous about doing something like that. It's the fear of making mistakes. But making mistakes is an essential part of coding, of problem solving: it is like taking different viewpoints on a problem and that is part of the process. Process is performance. The risk of failure is what makes performance interesting. ("Behind the Screens: Marije Baalman," 2020)

Live coding can also be seen as similar to the performance of circuit bending inasmuch as it can generate unpredictable sonic results by hacking the original structure of a circuit live. As is often the case, the process may involve pretexts of code with which the performer improvises, and builds something entirely new from its original state.

Live coding is not bound to a specific genre; instead it is used in various styles and communities of live electronic music. Some examples include artists working as solo performers, to name but a few, Alexandra Cardenas, Shelly Knotts, Marije Baalman, Julian Rohrhuber, Alberto de Campo, Alex McLean, Thor Magnusson, Nick Collins, and too many more to list here, and some other artists and creative coders working in the field of visuals, such as Olivia Jack, and Antonio Roberts. A selection of notable coders can be found at this link[4].

Algorave is growing exponentially across many communities, attracting artists who bring new tools and practices to the field. It is based on repetitive beat-based algorithmic processes using live coding with the aim of creating electronic dance music. For some it has been ambitiously anticipated as the future of electronic music, since the music that is created is broadly accessible to the general audience without requiring technical knowledge in programming. The appealing of this movement is also apparent by the numerous articles in some popular and widely read media that highlight Algorave as an emerging practice in the realm of electronic music (Calore, 2019; Luka, 2019; WIRED, 2006). For one thing, Algorave has gained a wide reputation internationally and has attracted many practitioners in the field both from academia and beyond.

A website[5] was devised for a colloquium panel on Algorithms and Live-Coding: Contemporary Practices inviting talks by artists of Istanbul's local scene under the New and Newest Music Festival[6] that was organized by Arter art center, in Istanbul, in February, 2021.

Besides the numerous solo artists of live coding and some examples of live coding communities, such as Algorave, there are also live coding and networked music performance projects (Collins & Escrivan Rincón, 2011; de Campo et al., 2007; Lee & Essl, 2014; Ogborn, 2014; Rohrhuber, de Campo, Wieser, et al., 2007; Wilson et al., 2014). These groups are known as live coding ensembles and are using local networks to distribute various data types in order to communicate with each other during the performance. Examples include PowerBooks Unplugged, Cybernetic Orchestra, and the Birmingham Ensemble for Electroacoustic Research (BEER). An early example of network ensemble is The Hub (Collins et al., 2003, p. 322). One of the members particularly states:

> ... in particular on the piece 'Waxlips', my machine's behavior was controlled by a hunk of code newly auto-generated on my command at the beginning of each section. There was also a certain amount of fooling around with self-modifying code; at the time we were much more interested in finding weird and uncontrollable behavior than in clarity, reliability, maintainability and other such outmoded concepts! (Collins et al., 2003, p. 323)

Live coding also has a recently founded dedicated conference, entitled International Conference on Live Coding[7] (ICLC). Its aims is to bring together the latest and current proceedings elaborating on cutting-edge technologies in live coding.

While these are showcasing coding as a performance strategy as much as a research area, some further investigation shows live coding has been a component in other practices from the Sonic Arts landscape, such as sonification[8], deploying live coding as a means to bridge interdisciplinary fields. Dark Matter created by BEER (Vasilakos et al., 2020).

4 https://en.wikipedia.org/wiki/Live_coding#Notable_live_coders
5 https://konvas.github.io/new-music-liveCoding/
6 https://www.arter.org.tr/en/algorithms-and-live-coding
7 https://iclc.toplap.org
8 Sonification is the use of non-speech data to synthesize sound material (Hermann et al., 2011).

Live coding and the audience (in the post digital era)

As cultural cache goes, effort demonstrates virtuosity in musical performance. A live coder is mostly performing by executing commands via the keyboard, this act is often linked with an "office/email checking" bias. It reasonably makes live coding performances look like an esoteric procedure which is hard to follow, and thus excludes some audiences that find it challenging to understand the specific technicalities of the code. Of course one can just focus on the music and safely ignore the programming aspect. However, the projection of the code shows the real-time process of tweaking the sonic algorithms and thus enabling the connection between performer's actions and the musical result, thereby enacting a causal relationship between the editing of the code and sound generation. The display of the code arguably, exposes liveness and highlights the improvisation with the running software including the decisions and errors throughout the performance.

The importance of the projection of code during the improvisation seems to be part of the main ethos of live coding as a way to enhance audiences' experience (Blackwell, 2015; Burland & McLean, 2016). This is also stated in Toplap's manifesto as a way to sustain the openness and access of the performance tool(s), avoid esotericism, and make apparent the liveness and moment of making (Toplap, 2020). Presenting the inner workings of a coding performance helps to establish a causal relationship between coding and sound, though perhaps not on the same level when comparing with the action of plucking a string of acoustic instrument: to achieve the same dynamics with the sound one would have to interact with the mechanical nature and characteristics of the acoustic instruments, something that in the digital domain that live coding exists is absent.

Instead, the audience is able to follow some changes which are happening in code and observe how this translates to the sound by reading humanly readable snippets and/or keywords in the code. Relating to some languages the level of transparency of the commands is aided by assigning specific parameters and their assigned values, which are understandable without the need to know the programming language, for example, "pitch:2", or "roomsize:10 and revtime" etc. and thus this helps to sustain a logical connection between changing code and the sonic result (demonstrated later). Therefore, if one is repeating an old video recording of a live coding performance this will look detached with the moment of the performance, as the change of the value of the rate of a pitch shift algorithm is replaced it will immediately create a correlation with the sonic outcome, since pitch shifting of sound material has a drastic influence on sound(s) the same as reducing the room size of a reverberating processor, which can be easily understood by even a non-technical audience.

Consequently, even in the case that the audience is not familiar with the language at hand this kind of change will help to relate to this action. That is, however, far from suggesting that the level of tangibility that one has with an acoustic instrument is the same with the code. The idea here is not to reenact this kind of relationship with the sounding object but to be able to interfere with the musical/sonic outcome on a higher level, similar to the interaction affordances offered by the external controller/device (e.g., a MIDI controller that is used for modulating sound parameters of a digital musical system).

Therefore, the visualization of code is something that can help the audience follow the performance and thus highlight other characteristics of the coding, such as liveness and virtuosity, and at the same time enhance openness and accessibility from audience's point of view.

Finally, to put this argument to the test, in a comparison between live coding performances and acoustic instrumentalists' performance, the audience is not expected to know the specifics of what is going on, but they still thoroughly enjoy the performance. In other words:

> A pianist in classical setting makes decisions on details that bring out the structure and the subjective emotional meaning of a piece; the 'text' of the composition itself is usually not touched. Even if the pianist's hands are not seen, an audience can follow and appreciate these aspects quite well. (de Campo et al., 2007)

As in every kind of musical performance, regardless of its genre, the intention of the audience is not to examine the technicalities of what they hear and to analyze, but to enjoy the music. During one of the first Algorave events in Istanbul, a survey about live coding was conducted providing some interesting feedback by the audience (Dağdeviren, 2018). A selection of responses are outlined in verbatim in 0000.

Examples of environments for live coding.

There are many environments for live coding; some of them include the broader and well known environments for sound synthesis and algorithmic composition, such as Pure Data[9], Chuck[10], MaxMSP[11] and SuperCollider[12]. Of these, SC, which is the author's platform of choice, is an open source and state of the art language for algorithmic composition and includes JITLib[13], a live coding library that accommodates various components for live coding and Just In Time programming techniques (Rohrhuber, de Campo, & Wieser, 2007).

Finally, there are some languages for live coding visuals such as Hydra[14] by Olivia Jack and Cyril[15], to name but a few. For an extensive list of programming environments for live coding see this link (https://github.com/toplap/awesome-livecoding#languages).

In addition to commonly used ones, there are also some other derivatives created by artists themselves, which entail rather personal preferences. Most of the times these are built on top of other programs using their sound generators. An example of this is TidalCycles, widely used in the field of Algorave[16]. It embeds Haskell language offering a wide range of patterns as a means to interface with custom-made synthesizers in SC, using Open Sound Control protocol[17] and can run in various IDEs, such as Atom[18], and Emacs[19], as well as in some online editors.

Some thoughts on code as an instrument

There is a common belief that live coding can be seen as similar to the performance with an instrument.

> By bringing the power and expressiveness of the programming language into runtime, an on-the-fly programming system has the potential to fundamentally enhance the real-time interaction between the performer/composer and the systems they create and control. Code becomes a real-time, expressive instrument. (Wang & Cook, 2017, p. 1).

9 https://puredata.info
10 http://chuck.stanford.edu
11 https://cycling74.com
12 https://supercollider.github.io
13 https://doc.sccode.org/Overviews/JITLib.html
14 https://github.com/HydraJS
15 https://cyrilcode.com
16 https://tidalcycles.org/Welcome
17 http://opensoundcontrol.org
18 https://atom.io
19 https://www.gnu.org/software/emacs/

While I find this idea to be generally true, it may be problematic to assume the code as an instrument without examining this concept. When performing with an instrument one learns to familiarize with its physical properties and master its inherent sound capabilities, e.g. the tendency of some instruments to play low registers easier and vice versa; and more idiomatic techniques, such as fingering placement, breathing, as well as extensive bowing techniques. If the performer makes a small mistake while performing with the instrument, for example, slightly misplacing a gesture or playing a wrong note, there is a good chance that it might even go unnoticed (depending on the level of the musical mistake and the audience). In correlation with live coding, a misplaced command or erroneous value inside a running module (unless one can get away with a minor syntax error), this will not be so forgiving and will have a dramatic effect on the musical flow. In some extreme cases it might even cause the crash of the software leading to an interruption of the performance, something that is impossible to go unnoticed. Others argue that this error-prone process is part of the live coding ethos and thus must be embraced under the rubric of live coding. That is not to say of course that the performance with acoustic instruments is less prone to human mistakes, but it is certainly less inoperable than in live coding performances. However some virtuosos and masters of acoustic instruments would probably disagree.

To clarify more the analogy I shall provide the following example. If one runs a loop using a routine[20] with an inappropriate interval time it will most likely freeze and crash the software, which will likely cause the termination of the sound. Similarly, there is also the question of the system used to run the multi-layered synth processes or complex processing of the signal at hand. This depends on the hardware resources of the computer one is using and thus it makes it impossible to predict, but if one pushes their machine to the edge of its abilities it will likely lead to the termination of the performance.

Therefore, while the analogy of the coding environment as an instrument is at a first glance true, in reality many other factors and risks come into play which the coder must become familiarized with and so be able to overcome during the performance.

The code as an interface

Once the coder reaches a certain degree of technical skills – avoiding syntax errors or minimizing the crashes with erroneous commands, and becoming familiarized with the platform's available components – and s/he has climbed the necessary learning curve[21] of the programming platform at hand sufficiently well to build the necessary confidence to go live, then code can be seen as similar to the way of performing with integrated Digital Musical Instruments (DMI), comprising an interface with which to modulate its control parameters. At this point some important distinctions must be made. In the case of the ready-made system, which is already developed, it is detached from the moment of improvisation. This means that the functionality and interaction capabilities of the system are predetermined, due to the decisions taken prior to the moment of the performance. Wang and Cook (2017, p. 1) describe this as an "off-line process of development and preparation, leaving only the finished program to "go live"".

This becomes more apparent when using an Ableton Live patch paired with external hardware controllers. The interaction possibilities are bound to fixed decisions of the software, e.g., access to a set of parameters and predetermined range specifications. Although tempting in terms of the easiness and accessibility that a polished

20 In programming, a routine is used to schedule a repeated function according to given time intervals.
21 Sufficient at least to manage a musically meaningful live performance, if not a masterpiece of coding dexterity and other virtuoso aspects which might be inherited by the classical music tradition but not necessarily adhere with the ethos of live coding.

graphical user interface may have, it will not allow for a deep interference with its inner workings, and thus it will only allow a predetermined[22] array of interaction possibilities; this idea is also supported by Collins (2003, p. 322). Moreover, it will affect the overall temporal progression of the performance and create a rigid time feeling, due to its purpose of sequencing of musical events in a timely manner.

In my attempt to avoid generalization, it is necessary to elaborate further in order not to reach biased and farfetched conclusions. Interactive systems, such as DMIs (Miranda & Wanderley, 2006) have been a fruitful area for experimentation and academic research since the advent of computer-based musical environments for real-time improvisation and generative music. This practice is thoroughly investigated in communities such as the New Instruments for Musical Expression (NIME), and has been at the forefront of many innovative hardware prototypes to modulate sound synthesis parameters as a means to provide an interactive relationship between human agency and computer-based musical systems. However, an extensive elaboration on DMI research is beyond the scope of this article. The author has undertaken extensive studies in this topic and will draw from his experience comparing live coding and physical interaction approaches (Vasilakos, 2016, p. 70).

Live coding and hardware controllers

The practice of using an external controller to enhance gestural expressivity is a well-established norm for computer musicians and laptop performers and tends to be the standard in the field of interactive Electroacoustic and Computer Music[23]. Even though the enormous functionalities of controllers and software may be further developed, and the variety of their control associations to pertain maximum expressivity is endless, yet all these are developed before the performance.

In other words:

> More precisely, the fundamental problem is how to model expression: the entire range of spontaneous expressive behavior a performer/conductor might want to use would have to be programmatically formalized beforehand, in order to map an ‘expression’ recognized in the sensor data to the relevant; expressive performance parameters. If one suddenly has an idea that a specific passage should be realized differently, that must have been prepared, or it will be inaccessible. (de Campo et al., 2007, p. 2)

Interestingly, code can be viewed as a medium, inasmuch as an external hardware device is used to modulate the control inputs of a DMI. Code as interface is not a new concept in the field of live coding, and is demonstrated by other artists, such as PowerBooks Unplugged and BEER (de Campo et al., 2007; Wilson et al., 2014).

In the case of live coding the code can be altered dynamically at the same time of performing allowing one to re-establish the modal relationships between the sound synthesis engines and their controls. This affords a shift towards new interaction possibilities of improvisation which allows one to redesign the performance environment in a recursive way. For example, one may control a set of parameters with the same random values generator; after a while this can be applied into other parameters or even replacing the random stream with a generator that distributes sequential values and so on. Some of the approaches illustrating this concept in SC and JITLib, including some of the author’s standard methods, are outlined and demonstrated as code snippets below (see

22 Using Ableton’s Max for Live integration will of course allow for a greater degree of customization yet it will still be an “off-line process”.
23 I would rather limit the discussion in these, and less to more commercial paradigms such as Electronica/IDM and their aesthetics.

Figure 2 and Figure 3). In addition to these some brief video excerpts of live coding are provided in the following section. However, these were made to illustrate some foundations of live coding instead of suggesting a global way to perform, as the possibilities to combine various techniques and develop a personal style of coding are endless.

Music made using various coding techniques

To provide some musical examples of the code snippets provided in Figures 1 and 2, some brief demonstrations highlighting some commonly used techniques in SC and JITLib respectively were created and outlined as follows. Additional explanation for each step is provided in the videos.

In the following music excerpt[24] coding involves the modification of a source code of a sound node, which consists of an array of sines unit generators and replacing those with saw oscillators (see Fig. 1). This transcends to virtual sonic spaces and at the same time it avoids ad nauseam textural interpretations, a simple yet musically meaningful transition. Some fade time is used in order to ensure smooth changes of the sound. While the performance unfolds a new parameter is introduced, named *freq*. The introduction of new parameters live helps the coder to change the sound output drastically by improvising with a new set of attributes of the node and assigning new values. The new parameters include panning, and frequency multiplication factor for each partial. The multiplication factor is mentioned already is denoted as *freq* and is controlled by a Pxrand pattern. Patterns is a very convenient way to control sound sources in SC. More information about the use of patterns in SC can be found at this link[25]. Pxrand iterates in an array of values without repeating a value until the array is completed. Similarly, the time interval of the iteration process is controlled by a pattern that generates a geometric series named *Pgeom*[26]; it has three parameters: start, growth and the number of values produced. Following this performance method the interactivity is set to a meta level, that is, the control is happening through the alteration of the parameters of the patterns instead of direct connection with the sound parameters. The music output is shifted from drones to granular textures and sonic gestures, constrained by an ADSR envelope which is also controlled using live coded patterns.

Another musical excerpt[27] involves a similar approach but with another type of a sound source, which may be closer to the idea of a synthesizers (see Fig. 2). These can be implemented before the performance and used one by one or all of them at the same time. This is demonstrated by triggering a synth and controlled by patterns in a similar way to the previous example. In this case it can easily create many control streams on top of each other and thus create different layers and separate sonic entities, leading to polyrhythmic and diverse timbres. This approach is a powerful method to improvise and thrust the musical output by combining various types of patterns and modify their contents in real time using fixed synthesis sources.

The last excerpt[28] illustrates how to add effects to the signal to enrich the audio output in a post production level by following similar live coding techniques in SC and JITlib. It shows the implementation of a resonating reverb of SC of which the reverberation time and the room size are modified to create alternative spaces. Instead of

24 https://youtu.be/1EG7PoenePk
25 https://depts.washington.edu/dxscdoc/Help/Tutorials/A-Practical-Guide/PG_01_Introduction.html
26 https://doc.sccode.org/Classes/Pgeom.html
27 https://youtu.be/0wU2icG8V-o
28 https://youtu.be/-8F20v6qPH4

creating the effect from scratch, as this would require a significant amount of time to implement on stage, similar controls can be applied in the effect's parameters as described above.

These approaches help to escape from coding inertia[29] and allow one to shift from textural drones to rhythmical variations, or vice versa, and interactively develop different synthesizers as a means to build a musical arsenal for the live coding improvisation.

This process is broken down into several stages, that is, built, edit, and re-structure of the software, and at the same time it enables the constant exploration and resetting of interaction affordances, for example, introducing more control parameters and/or effect processors. The dynamic relationship yielded from this process opens up an unparalleled interactivity with computer music environments. It offers a higher level of flexibility and a wider palette of interaction affordances which in turn allow for more unpredictable musical discourse. In the case of performing a specific live coding piece, which may rely on specific resources and other programming paraphernalia (e.g., some pieces require additional code to be loaded in order to perform the piece, such as networking interconnection amongst performers) there is a slight chance of interactivity trade offs as the performance strategy is defined in the compositional idea of the piece and dictated by the pre-developed interface or platform that creates the musical piece.

```
Ndef(\updateMe_later, {Splay.ar(SinOsc.ar({ExpRand(120.0, 1220.0)}!8, Rand(0.1, 2.0)*pi, 0.5))});
Ndef(\updateMe_later).fadeTime = 3.0;
Ndef(\updateMe_later, {Splay.ar(LFSaw.ar({ExpRand(120.0, 1220.0) * \freq.kr(1, 0.3)}!8,
Rand(0.1, 2.0)*pi) ** LFNoise1.kr(0.5).range(0.1, 0.25))});
```

Figure 1: Live coding in SuperCollider using JITLib's Ndefs.

Importance of parameter mapping and live coding approaches

One of the key aspects of the development of DMIs is the relationship between the physical input and the way it is linked with the control of the musical parameters of the platform. This important aspect in the design of digital instruments is known as mapping (Hunt et al., 2002; Miranda & Wanderley, 2006).

Besides approaching mapping strategies as merely a technical part of a system, it could be argued that it has a vital importance and a major influence on the musical outcome when comparing it with other interactive laptop performance paradigms (Vasilakos, 2016).

Following similar coding approaches as described already, one can also hardcode the wiring of the mapping and change the control relationships of the gestural inputs to sound dynamically, leaving out the problems of limited[30] mapping strategies which may result in the predictability of the gestural input interpretation (Vasilakos, 2018, p. 66).

29 Coding is a laborious act and error prone process, inevitably it may require some time to develop a specific musical passage during improvisation which might lead to inertia or slowing the results.
30 However, this limitation is far from being a constraint to the musical expressivity. For example, going back to the correlation with acoustic instruments, a clarinet might be a very limited instrument but nobody would suggest that it is devoid of expression or the music that it produces is wholly deterministic.

This dynamic manipulation of the mapping has also been explored by Marije Baalman (Baalman, 2020). The performance's strategy consisted of using embedded sensors of the computer, such as accelerometer and touchpad, which are live mapped during the performance.

The same technique can be extended to any form of control signals, whether a hardware interface is generating them, such as a MIDI controller, or OSC messages in network communication. To describe the process better, I will limit the possibilities to intra-communication using some of SC's patterns in order to control the parameters of pre-made synthesizers, as seen below:

```
Pdef(\a,
Pbind(\instrument, \synth,
\dur, Prand([4/4, 4/8, 4/16, 4/32], inf),
\rate, Prand([0.2, 0.4, 0.6, 0.8, 1, 3], inf),
\buf, ~sounds[0]
);
```

Figure 2: Live coding data streams controlling sound synthesis parameters of a SynthDef using patterns.

Figure 2 shows a way to interact with a synthesizer in real time and interact without interrupting the performance. A *Pdef* in SC is a placeholder that can be modified later without interruption. This approach is extremely handy as one can update the values and all the associations between incoming data and synthesis parameters without disrupting the musical flow. It can also include quantization of many placeholders and thus different layers of sounds which can run simultaneously. The combination of these is as unlimited as the mapping. This includes range specification and pattern design; for example, a pattern may use other patterns for their parameters as illustrated in Figure 3.

```
Pdef(\x, Pbind(\instrument, \foo, \dur, Pgeom(0.05, 1.1, inf), \degree, Pseq([Pseq((1..10), 1),
Pxrand((15..30), 15)]))).play;
```

Figure 3: Patterns controlling parameters of patterns.

Live Coding and the Web

In addition to the merits of live coding expressed thus far, the nature of the script, as a lightweight medium to transmit and/or transfer commands and instructions of both controlling signals and algorithms that make the actual performing software at hand, makes it a very convenient tool for networked performances, that is, distributing various bits of code amongst performers while improvising in live coding in an ensemble. Therefore, it has proven very handy for telematic performances, e.g., remote interactions with coders across the internet. The code can easily broadcast with no heavy lifting resources for back-end support, e.g., dedicated servers, making code ideal for remote interactions across the internet. In addition, one can also perform using online editors[31] which support various languages, such as SC, and TidalCycles and run on web browsers allowing for synchronization of all connected performers across the web.

31 https://flok.clic.cf

Live Coding: personal works

The author has adopted live coding as one of his main tool for experimentation, research, and artistic practice. Analysis of personal projects elaborating in live coding as a means to create original works both as solo and networked/distributed live coding performances is outlined as follows.

Istanbul Coding Ensemble

Istanbul Coding Ensemble (ICE: https://konvas.netlify.app/ice/) is a coding group formed by Sonic Arts postgraduate students[32] at the Center for Advanced Studies in Music (MIAM). It was founded by the author to investigate live coding and networked performance drawing largely from his experience as member of the BEER. Since 2018, the group is contributing to academic events in Istanbul and beyond performing works from the repertoire of various laptop ensembles, such as Dark Matter (2016), NSMRC[33] by BEER, but it has also performed original works, such as Akkaya's MIMIC, a live coding piece using real time sonification of brain waves data. This work also used several hardware to perform, such as, embedded computing boards and several electrodes to render the data while one of the performers is streaming brain waves data throughout the performance. More information can be found at this link[34].

The group has also performed with acoustic instrumentalists, solo performers and orchestras. A collaboration with percussionist Amy Salsgiver (ITU/MIAM) involved machine listening as a means to create real time dialogues between coders and vibraphone; video of the performance found at this link[35]. This was based on BEER's Piano Code (2014) which was premiered by pianist Xenia Pestova. Another iteration of this project is a collaboration with pianist Jerfi Aji based on Kamran Ince's Symphony in Blue, and thus the project was named Symphony in Blue 2.0, presented at the TTI Conference[36], in 2021. A recording of the performance can be seen at this link[37].

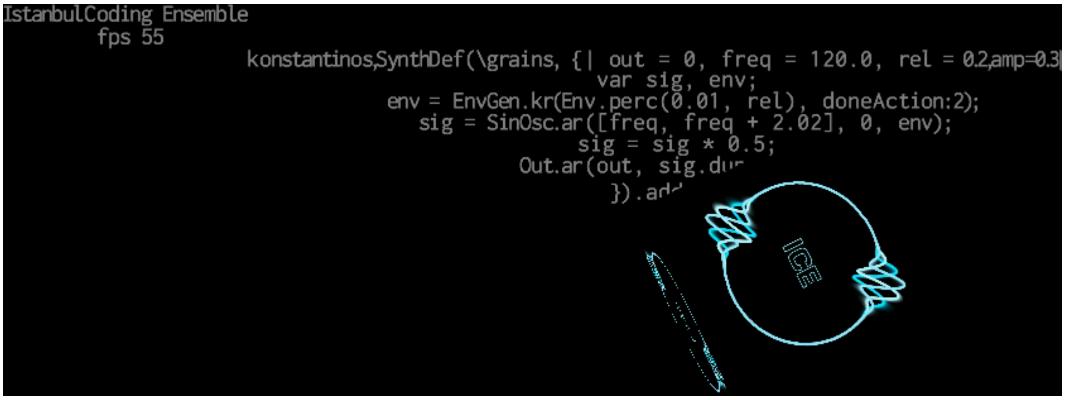
32 Past and current members include, Tuğba Selin Ülker, Gökalp Kanatsız, Anıl Bozkırlı, Selenay Kıray, Bartu Çankaya, Onur Dağdeviren, Serkan Sevilgen, Uğur Can Akkaya, Bidar Kalender, Kerem Ergener and Okan Yasarlar.
33 NSMRC (after Dan Stowell) readapted for live coding ensemble.
34 https://github.com/UCAkkaya/MIMIC-Project
35 https://www.youtube.com/watch?v=fsUI83MMogg&list=PLiCZTYIqSUAb4gVwlbsaXl9bKT-V3xRoG&index=5&t=5s
36 https://nc16653.wixsite.com/ttiprojec0t/conference
37 https://youtu.be/vMXCswvoV2E

Figure 4: From left: Konstantinos Vasilakos, Onur Dagdeviren, Serkan Sevilgen (ICE), and Jerfi Aji (piano) at MIAM Studio. Photo was taken by Cem Onerturk at MIAM Studio, 2021. ITU/Macka, Istanbul.

Software and tools of ICE

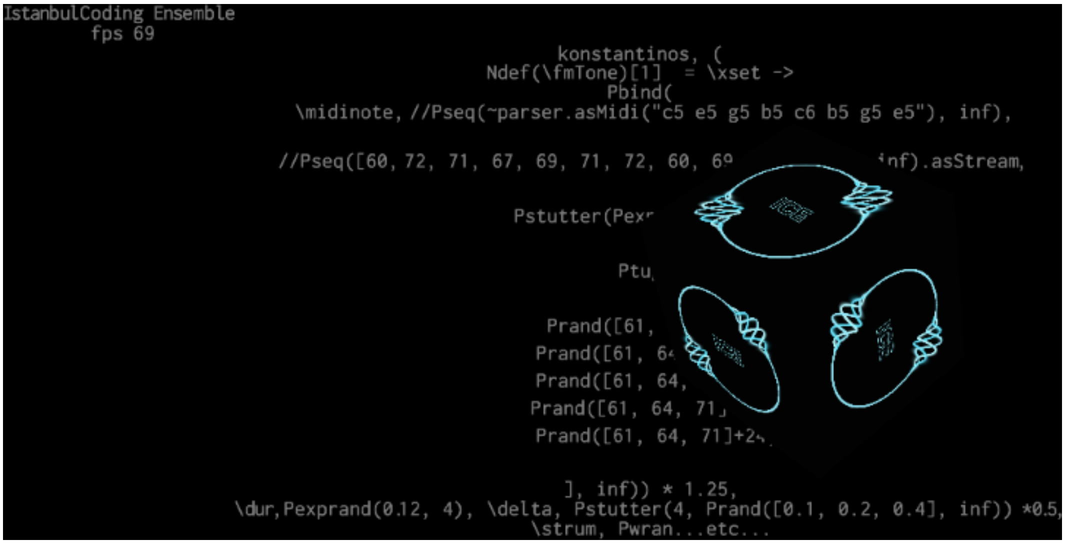
ICE's main programming language for performance is SC. For networking communication the group uses Utopia, a platform developed by Scott Wilson (Wilson et al., 2014), which allows connected performers to share their code to each other and build real-time interactions using a graphic user interface. It allows accessibility to the code that is executed from each coder in real time and it also provides a series of synchronization components and classes for sending and receiving OSC messages.

Lately, the group has adopted a server based display utility, which is used to show all code that is executed by each member during the performance. It is developed as a NodeJS application which is the runtime of JavaScript programming language. A screenshot that was taken during one of the performances can be seen below in Fig 5 and 6.



```
IstanbulCoding Ensemble
        fps 55
                    konstantinos,SynthDef(\grains, {| out = 0, freq = 120.0, rel = 0.2,amp=0.3
                                             var sig, env;
                                env = EnvGen.kr(Env.perc(0.01, rel), doneAction:2);
                                  sig = SinOsc.ar([freq, freq + 2.02], 0, env);
                                            sig = sig * 0.5;
                                        Out.ar(out, sig.du
                                              }).ad
```

Figure 6: Code display excerpt A.



```
IstanbulCoding Ensemble
       fps 69
                                       konstantinos, (
                                  Ndef(\fmTone)[1]  = \xset ->
                                              Pbind(
                        \midinote, //Pseq(~parser.asMidi("c5 e5 g5 b5 c6 b5 g5 e5"), inf),

                       //Pseq([60, 72, 71, 67, 69, 71, 72, 60, 69             inf).asStream,

                                        Pstutter(Pex

                                               Ptu

                                          Prand([61,
                                         Prand([61, 64
                                         Prand([61, 64,
                                        Prand([61, 64, 71
                                         Prand([61, 64, 71]+24

                                              ], inf)) * 1.25,
                 \dur,Pexprand(0.12, 4), \delta, Pstutter(4, Prand([0.1, 0.2, 0.4], inf)) *0.5,
                                      \strum, Pwran...etc...
```

Figure 5: Code display excerpt B.

Chunk No_Reace: Algorave music endeavors

Chunk No_Reace is a solo project of the author performing in Algorave events. Some notable concerts that took part include Istanbul's first Algorave event: Code the Party 1 and 2 at Salt Gallery in 2018, and at Pan's Social House, in 2019 respectively. The events were both organized by Atay İlgün and Caner Bozkur and heralded the inauguration of the Algorave community in Istanbul. To this day, Algorave Istanbul is a vibrant community and well established event of the vibrant electronic music scene in Istanbul and is rapidly growing with members from local universities and artists forming a dynamic and versatile demography of live coders with an anticipation of welcoming new members and audience alike in the following years.

Conclusion

Live coding is a very versatile performance paradigm which spans across many genres in the broad landscape of live electro(nic)acoustic music and other time-based arts. It uses the act of programming while heavily relying on improvisation as impetus for real time composition of audio/visual pieces. By displaying the code during the live coding performances the audience is invited to follow the open-ended construction of programs, including all the associated controlling components that comprise the performance media, which are used to perform sound and visuals at the same time that their structure is changed live. Live coding is embodied in many cutting-edge performance paradigms sparking new aesthetic narratives and sub-cultures of electronic music, such as laptop networked ensembles and Algorave music.

Live coding in performances appeals to both academics and wider groups with no previous technical background, thus making it more accessible, while at the same time providing the field for experimentation with no alienating traits of esotericism for the interested layman. On the musical level, live coding offers the performer a platform to overcome previous constraints (which a fixed system may entail), while exploring the interaction affordances of a computer-based musical environment at the same time as its creation.

Therefore, it provides an interface which allows for drastic variations of sonic outcomes allowing for diverse musical interpretation. In turn, this has a profound impact on the higher-level musical characteristics, including rhythmical and spectral variations of the sonic outcome.

Finally, the author provided information about personal research in a newly formed laptop ensemble, and solo Algorave persona, both involved in the local scene of live coding inviting a new breed of artists and audiences alike.

Acknowledgements

The author would like to thank all faculty and graduate students of MIAM who supported ICE. Special thanks to Scott Wilson for his telematic performance and MIAM Studio at ITU/MIAM for technical backing in Symphony in Blue 2.0. Finally, the author would also like to thank the members and friends of the Istanbul Algorave community.

Appendix A

It is worthy of mentioning that while most of the respondents stated that they had no programming background whatsoever, the projection of the code didn't seem to have a negative impact on the appreciation of the music or cause alienating effects to the overall experience.

I. Sample (P. 4/12)

"Q: To what extent do you engage with the projected code at these events"

"It is like a puzzle or translation for me, b/c I don't speak Coding and I try to make connections between the sound and the code".

"Q: What is your level of experience (if any) with computer programming"

"None"

II. Sample (P. 12/12)

"Q: To what extent do you engage with the projected code at these events"

"I guess I was trying to make sense of the code, although I don't know anything about programming."

"Q: What is the impact of the projected code on your experience of the event"

"First it was the total center. Even more important, than the performer himself. After I got used to it, I lost the overall interest in it and just went on partying."

"Q: What is your level of experience (if any) with computer programming"

"I don't know any computer programming languages."

"Q: What appeals to you most about live coding events"

"The Improvisational aspect and that you see something changing in advance, although you will not know what exactly. Secondly that it was something totally unknown for me before. I was just curious."

References

Baalman, M. 2020. *Code LiveCode Live* [Personal Website]. Marije Baalman. https://marijebaalman.eu/projects/code-livecode-live.html

Blackwell, A., 2015. Patterns of User Experience in Performance Programming. [online] https://doi.org/10.5281/ZENODO.19315.

Behind the Screens: Marije Baalman. 2020, June 30. *Https://Medium.Com*. https://medium.com/behind-the-screens-challenge/behind-the-screens-marije-baalman-559ca3f1696b

Burland, K. and McLean, A., 2016. Understanding live coding events. International Journal of Performance Arts and Digital Media, [online] 12(2), pp.139–151. https://doi.org/10.1080/14794713.2016.1227596.

Calore, M. 2019. DJs of the Future Don't Spin Records—They Write Code. *Wired*. https://www.wired.com/story/algoraves-live-coding-djs/

Cocker, E. n.d.. *Kairos Time: The Performativity of Timing and Timeliness … or; Between Biding One's Time and Knowing When to Act*. 16.

Cocker, E. 2013. Live Notation: – Reflections on a Kairotic Practice. *Performance Research*, *18*(5), 69–76. https://doi.org/10.1080/13528165.2013.828930

Cocker, E. 2018. What now, what next—*Kairotic* coding and the unfolding future seized. *Digital Creativity*, *29*(1), 82–95. https://doi.org/10.1080/14626268.2017.1419978

Collins, N. 2011. Live Coding of Consequence. *Leonardo*, *44*(3), 207–211. https://doi.org/10.1162/LEON_a_00164

Collins, N., & Escrivan Rincón, J. d'. 2011. *The Cambridge companion to electronic music*. Cambridge University Press. http://dx.doi.org/10.1017/CCOL9780521868617

Collins, N., McLEAN, A., Rohrhuber, J., & Ward, A. 2003. Live coding in laptop performance. *Organised Sound*, *8*(3), 321–330. https://doi.org/10.1017/S135577180300030X

Dagdeviren, O. 2018. *Google Form Survey on Live Coding and the audience.* unpublished; A Brief look at Live Coding Events.

de Campo, A., Vacca, A., Hoelzl, H., Ho, E., Rohrhuber, J., & Wieser, R. 2007. *Code As Performance Interface—A Case Study*. 6.

Hermann, T., Hunt, A., & Neuhoff, J. G. (Eds.). 2011. *The sonification handbook*. Logos Verlag.

Hunt, A., Dd, Y., Wanderley, M., & Paradis, M. 2002. The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research*, *32*. https://doi.org/10.1076/jnmr.32.4.429.18853

Lee, S. W., & Essl, G. 2014. *Models and Opportunities for Networked Live Coding*. 5.

Luka. 2019, October 19. *Live Coding and Algorave.* Medium. https://lukabaramishvili.medium.com/live-coding-and-algorave-9eee0ca0217f

Miranda, E. R., & Wanderley, M. M. 2006. *New digital musical instruments: Control and interaction beyond the keyboard*. A-R Editions.

Nilson, C. 2007. Live coding practice. *Proceedings of the 7th International Conference on New Interfaces for Musical Expression - NIME '07*, 112. https://doi.org/10.1145/1279740.1279760

Ogborn, D. 2014. Live Coding in a Scalable, Participatory Laptop Orchestra. *The MIT Press*, *38* (Spring 2014), 15. https://www.jstor.org/stable/24265529

Rohrhuber, J., de Campo, A., & Wieser, R. 2007. N*otes on Language Design for Just In Time Programming*. 4. http://web.cecs.pdx.edu/~dreeder/site/nysc/doc/rohrhuber,etal--jit.pdf

Rohrhuber, J., de Campo, A., Wieser, R., van Kampen, J.-K., Ho, E., & Hölzl, H. 2007. *Purloined_letters*. 7. http://www.wertlos.org/~rohrhuber/articles/Purloined_Letters_and_Distributed_Persons.pdf

Sicchio, K. 2014. Hacking Choreography: Dance and Live Coding. *Computer Music Journal*, *38*(1), 31–39. https://doi.org/10.1162/COMJ_a_00218

Vasilakos, K. 2016. *An evaluation of digital interfaces for music composition and improvisation.* [PhD, Keele University]. http://eprints.keele.ac.uk/1606/1/Vasilakos%20PhD%202016.pdf

Vasilakos, K. 2018. Greap: an Interactive System For Gestural Manipulation of Sonic Material Using a Leap Motion Device. *Porte Akademik Journal of Music and Dance Studies*, *17* (Spring 2018), 15.

Vasilakos, K. n/a, Wilson, S., McCauley, T., Yeung, T. W., Margetson, E., & Khosravi Mardakheh, M. 2020. Sonification of High Energy Physics Data Using Live Coding and Web Based Interfaces. In R. Michon & F. Schroeder (Eds.), *Proceedings of the International Conference on New Interfaces for Musical Expression* (pp. 464–470). Birmingham City University. pdfs/nime2020_paper76.pdf

Wang, G., & Cook, P. R. 2017. 2004: On-the-Fly Programming: Using Code as an Expressive Musical Instrument. In A. R. Jensenius & M. J. Lyons (Eds.), *A NIME Reader* (Vol. 3, pp. 193–210). Springer International Publishing. https://doi.org/10.1007/978-3-319-47214-0_13

Wilson, S., Lorway, N., Coull, R., Vasilakos, K., & Moyers, T. 2014. Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems. *Computer Music Journal*, *38*(1), 54–64. https://doi.org/10.1162/COMJ_a_00229

WIRED. 2006. Real DJs Code Live. *Wired*. https://www.wired.com/2006/07/real-djs-code-live/

Zmölnig, Io., & Eckel, G. 2007. Live coding: An overview. *International Computer Music Conference, ICMC 2007*, 295–298.