Composing and Performing Mixed Electronic Works

Orestis Karamanlis, Dionysis Athinaios

Composing and Performing Mixed Electronic Works

Orestis Karamanlis, Dionysis Athinaios

Keywords Electroacoustic Music, Real-time, Composition, Performance, Mixed, Live Electronics, SuperCollider, CuePlayer, Cues.

Abstract

This paper explores common methodologies and practices for composing and performing mixed works, involving acoustic instruments and real-time electronics using a computer. We proceed to describe a strategy for creating and presenting such works and introduce the CuePlayer, a tool for the SuperCollider programming language which aids in the organisation of processes and musical material in bundles (cues). We discuss the implications from its use with reference to a musical work.

1. Practices in Mixed Music

The following text is written from a composer's perspective and dives straight into the core of the discourse. It assumes awareness of the electroacoustic musical language and a general understanding of the issues surround its creation and presentation to the public. For a historical review on issues specifically relating to real-time computer music see Bevelander (1991), Emmerson (1994, 2000), Risset (1999), Rowe (1999), Stroppa (1999) and Hagan (2016) amongst others.

Put simply, within the realm of *computer music*, we understand *real-time* to involve processes taking place inside the computer which may generate or modify sound without any interruption in the musical flow; and *live-electronics* to involve (possibly human) performers controlling audio parameters in concert. The absence of an audience would not change the nature of those works, but since music has this social dimension of a shared experience it is helpful to think of computer music in a concert situation. We also understand *mixed music* to combine both acoustic sources and electronics, which may be fixed or generated in real-time.

From all the different expressions of electronic music this text explores the combination of acoustic instruments with electronics, both of which may involve different levels of indeterminacy. It is common practice for the electronic sound to range from being fixed on a medium, to being generated/modified in real-time; and for the instrumental part to range from being freely improvised, to being explicitly notated on score. All sorts of combinations of the in-between states of the above are encountered. Still, people working in this genre face a series of challenges when composing and performing new music:

- How will the instrumental and electronic parts be organised?
- How effectively will the acoustic sources integrate with the electronics?
- How can the piece be auditioned and tested prior to the rehearsal?
- What is an efficient method for presenting a mixed work?
- How easily can the piece be brought to different venues?

These questions represent some of the issues faced by composers (and performers) of mixed music, and there are many different answers. It is not unusual for individuals to invent their own solutions depending on the requirements of each piece. We will now consider some specific scenarios according to how *fixed* the acoustic and electronic parts appear to be.

1.1 Fixed Electronic Sound / Fixed Instrumental Part

The approach of having the electronics pre-recorded on a medium and the instrumental part explicitly notated on score immediately solves the problem of organisation from a technical point of view. Considerations on fusion and contrast, cause and effect relationships, growth and expectation are always present but are purely aesthetic. From a strictly technical point of view there is little doubt as to how the piece will be prepared and executed. The electronic sound is most often sequenced within a digital audio workstation and exported in a single or multiple audio files. The instrumental part is conventionally or graphically [1] notated on paper or stored in a digital file. The main worry is how well the electronics will integrate with the instruments and how the two will coordinate during the concert. The latter is usually solved via the use of a prepared click-track (usually fed to the performers via in-ear monitors) and a stop-watch showing the time elapsed since the beginning of the piece. Evaluating how effectively the acoustic sources integrate with the electronics is a more difficult task. This is often tackled with the use of sampler instruments, often controlled via Midi keyboards during the compositional process, which imitate their acoustic cousins.

Works of such format do not pose great difficulties in concert situations and are characteristic of the 1980s when computers and programming languages were not powerful enough to work in real-time. They can be realised in the studio, taking time to fine-tune different aspects of the piece and then be presented without substantial technical obstacles to the audience. The critique usually centers on the fixness of the electroacoustic sound (what is often called the *tape-part*). McNutt (2003) suggests

that ‘for the player, performing with fixed accompaniment is like working with the worst human accompanist imaginable: inconsiderate, inflexible, unresponsive and utterly deaf’. A more flexible approach for the electroacoustics is to divide the entire piece in sections and prepare individual sound-files (cues) to be triggered on specific moments which are marked up on the score, thus allowing for some temporal freedom to the performer between sections. We return to this idea later.

1.2 Fixed Electronic Sound / Improvised Instrumental Part

Some or all of the instrumental part may be improvised, freely, following a series of instructions, or an open score like Cornelius Cardew's (1967) *Treatise*. In this instance the composer is usually uninterested in explicitly controlling the sound-world of the acoustic instrument. The responsibility is transfered to the performer who, upon becoming familiar with certain guidelines, executes passages accompanied by the tape-part, often having freedom within a range of possibilities. Absolute coordination is rarely desired and the performer may simply keep track of the time elapsed with a stop-watch.

But what is it that the composer controls in this format? It is certainly not the timbre, the phrasing, the rhythmic and melodic material of the instrumental part in detail. S/he functions more as a coordinator, providing a framework upon which a collaborative artwork can be built. The critique of such an approach lies, on the one hand, on the fixness of tape, and on the other on this fluidity that exists on fundamental structural aspects of the work.

1.3 Improvised Electronics / Fixed Instrumental Part

In this scenario the instrumental part may be explicitly scored whereas some (or all) of the electronics may be improvised. This usually falls under the umbrella of *live electronics*, involving a laptopist improvising in front of the audience along with the instrumentalist. Certain actions may be scored before-hand but that usually does not change the nature of the electronics, which are generated and manipulated in real-time by a dedicated performer who is usually the composer.

In this scenario the instrumental part could be constructed making use of a sampler instrument as described earlier and the improvised electronics may be trialled and adjusted in the studio. It is often not in the interest of the composer to control in detail every parameter of the electronic sound, but rather to create a system that allows for error and uncertainty during the performance, viewing the laptop more as an instrument as opposed to a play-back engine. The final piece is probably less portable, making it more difficult for the instrumentalist to practice on his/her own without having a feedback mechanism to audition the electronics.

1.4 Improvised Electronics / Improvised Instrumental Part

When improvisers (instrumentalists and electronic musicians) perform on stage, we usually refrain from speaking of composition. If nothing is prepared before-hand but a series of apparatuses, software and processes, then the musical work is built in concert through a dialog between the ensemble's members based on spontaneity and mutual interaction.

1.5 In-Between

Between all the above there is room for many interesting intersections, for works incorporating pre-composed elements and real-time processing, as well as scored and improvised instrumental parts. Adding variability and chance to the electronic sound during the concert is something that many composers embrace, often using algorithmic processes which have been built beforehand. It is in this format that computer music programming languages, like SuperCollider, Pd, Max/MSP, ChucK, etc, have proven to be especially useful. The obvious benefit of using such tools is that they allow significant amount of freedom for the user to accomplish sophisticated tasks which may fall outside of what mainstream audio software offers. Transformations of symbolic musical data are also possible with many computer-aided composition environments, but this is beyond the scope of the present discussion [2].

With the advent of powerful laptops and music programming languages many composers nowadays become programmers, sometimes even developing a substantial background in programming in an effort to realise complex compositional ideas. The programming skills required can be as important as the purely musical and it is not unusual for competent composers to struggle with technology while trying to solve programming issues, devoting time to troubleshoot their software than focus on the musical qualities of the piece. Still, we regard the creation of software accompanying the instrumental score as an integral part of the compositional process. This is even more the case with generative works where a set of algorithms produce ever-changing musical processes; the piece becomes also the process itself and the composer is a listener to the result at the end of the chain. But, to a composer who is more concerned about the actual sound-world, the timbre, the rhythmic and melodic material and their evolution through time (as opposed to the processes that give life to the sounds), specialised programming may appear somewhat frustrating. Adding on top the challenges of structuring, testing and presenting an interactive piece, no-matter how interesting the process may be, it is often far from being a smooth one.

2. Recipe for Composing & Performing

Many computer music programming languages can be regarded as being equivalent, in the sense of being able to accomplish the same tasks. The authors have relied mostly on a text-based dynamic programming language since complex structures can be represented in a clear way using text. Though it is helpful to see the dataflow through a patch and quickly interact with a

graphic user interface of a visual programming language, in the long run it may be difficult to work with large number of objects or more importantly to represent compound sequential processes. We now demonstrate a compositional workflow using SuperCollider which has proven to be fruitful in our own research and which may be adaptable to visual programming languages as well with some effort.

Let us first envisage the composition of a novel work for electronics and one acoustic instrument, a violin for the sake of simplicity. The electronics run in real-time on a laptop, incorporate pre-programmed elements and process a live feed from the acoustic source. The instrumental part is scored, at least partly, via standard notation. In other words, the majority of the work is not freely improvised and requires studio time to structure all material in a musically meaningful way.

A trained composer would not have a major difficulty in imagining how a scored violin part would sound in concert. However, the SuperCollider code would always need to be tested and auditioned to judge how well it works and integrates. In most cases it is a good idea to try to create a decent "performance" of the instrumental part using a virtual sampler instrument within a digital audio workstation. The entire part, or even some key moments, can be sequenced using Midi data, auditioned, transformed and adjusted. A drawback is that the sampler would rarely be able to reproduce extended techniques and phrases as indicated in the score. Hence one idea is to carry out extensive sampling of the instrument in question beforehand according to the initial sketches of the piece; in this way the composer would have at his/her disposal a large bank of custom instrumental gestures to be used while building the piece later on. This practice may be helpful in trying things out and working offline with the soundfiles, but it may also be somewhat problematic in limiting the composer's imagination to the sampled material available. It is up to each individual to judge what constitutes a good middle ground.

At the same time of scoring and sequencing the violin, the electroacoustic part is constructed. Processes which can effectively work non-real-time are prepared, such as certain transformations on pre-recorded material that do no harm being *fixed*. Processes which should run in real-time, such as operations on the live feed from the instrument, are programmed. What we soon end up with is a series of prepared soundfiles, chunks of SuperColider code and a loose approximation of the instrumental part within an audio workstation. How will the electronics be sequenced?

In order to glue things together the old trick of using *cues* has proven to be adequate. The idea is to construct a series of distinct processes (*cues*) within the programming language and mark on the score where exactly they should be executed. Very often composers put a *cue-number* on top of the instrumental part to denote when the respective cue is to be triggered. In our own work each cue is a SuperCollider function which usually generates or manipulates sound. All cues are organised into a list, what we call the *cue-list*, and can be triggered when their function is called. These bundles of code are usually executed sequentially, one after another as the piece unfolds, allowing for a high degree of coordination between the electroacoustic sound and the instrument.

Using cues overcomes the issue of organisation, but how can the composer audition both the electronics and the violin part prior to the rehearsal, to test, adjust and debug? A straightforward solution is to route the audio of the violin sampler instrument to the programming language for real-time processing which can be done internally, via dedicated audio routing software, or on the soundcard, if it allows for flexible signal routing. Furthermore, a mechanism to synchronise the two applications together (audio workstation and programming language) is required so that the piece runs through seamlessly and the cues, resting as code within SuperCollider, are automatically triggered when needed from inside the sequencer.

The way in which the cues will be executed in concert must also be considered. By far the most widely used option is via a foot-pedal from the instrumentalist on stage. The only requirement is a single switch connected to the laptop, triggering each new cue. Cues may also be launched by the laptopist via the keyboard or an external controller, freeing the violinist to focus more on his/her instrument. In any case there should be a system in place to adjust for wrong or forgotten cues. It is recommended that the laptopist oversees the smooth execution of the piece, intervening according to the score when accidental triggers happen or cues are omitted.

For practicing, the instrumentalist is provided with a score coupled with an easy to launch SuperCollider patch (or preferably a standalone application) and a foot-pedal if needed. The code should run with zero calibration from the performer having gone through an extended trial period in the composer's studio. For multichannel works it is suggested that the design of the patch allows for an arbitrary number of output channels to be used depending on the technical specifications of each performance space. Between 2 and 8 would be enough for most cases. This strategy makes the piece more portable to different venues and increases its chances of being performed in the future. It should enable stereo presentation for convenience while being able to adapt painlessly the majority of audio processes in multichannel.

We have described a straightforward workflow for composing and performing. What we have lacked so far is a device to aid in the organisation, scheduling and triggering of musical material and operations within the programming language. Immediately afterwards we present an open source tool for the creation and presentation of mixed music.

3. The CuePlayer

The CuePlayer[3] version 0.2 is an extension for the SuperCollider programming language, freely distributed using the Quarks package manager. There are many features which make it useful for composing in studio and performing on stage electroacoustic works; here we touch upon the very basics.

As its name suggests, it is aimed at "playing cues", assuming that the composer structures the piece by preparing them beforehand. A *cue* is defined as a chunk of code bundled as a function and placed within a list which holds all processes (the *cue-list*). From there we are able to trigger them sequentially or in any order via code, a Graphic User Interface or any device/software which outputs Midi or Open Sound Control data. For example, we could lay out a mock-up of the instrumental part in an audio workstation, create a dedicated Midi-track to send information internally to SuperCollider and trigger the cues from the sequencer via Midi. We could also use an external device to launch cues remotely, use a foot-switch on stage, and so on.

The tool comes bundled with a GUI class which brings up the CuePlayer window. Through the Graphic User Interface the laptopist can monitor input/output buses, use a timer and a metronome, control the levels, adjust cue-numbers and launch cues. Up to 8 input and 48 output buses can be monitored, which should be enough for most pieces, however the inclined user could easily hack the source code to change these. During performance it is possible to project the cue-numbers on a separate monitor for the instrumentalist for better coordination.

Default CuePlayer Window.

A convenient method to organise a piece is to place the individual cues into separate SuperCollider documents and then load them in the cue-list. When modifications in these files happen while composing, live-reloading functionality ensures that the changes are active (when saved) the next time a cue is called, thus allowing for a smooth workflow.

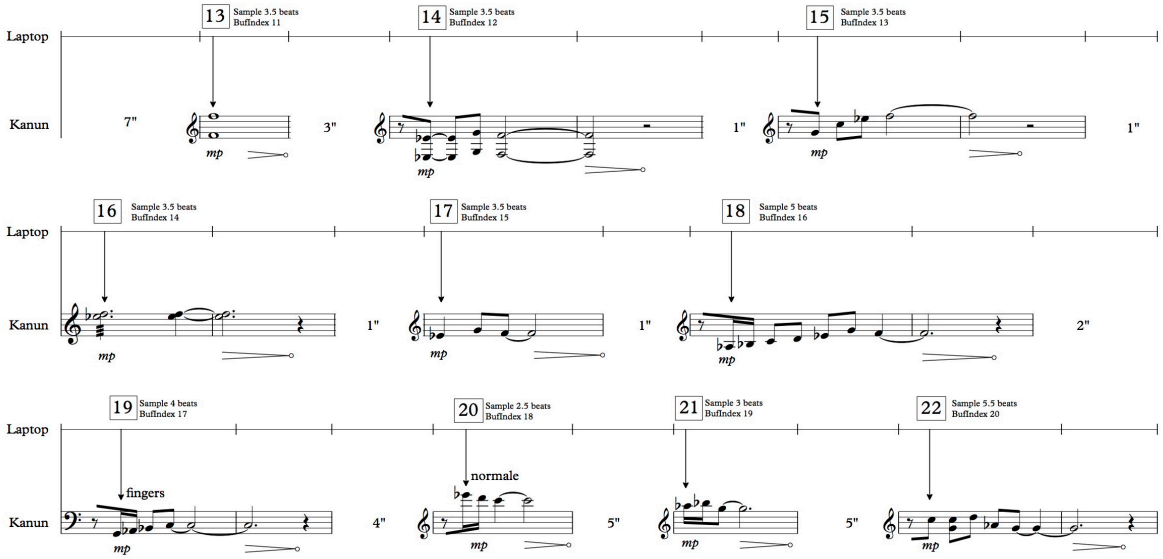
The Timeline class can be used to schedule processes. The composer initially defines an array with *time - function pairs*, where time is a number (beats or seconds) specifying how far in the future the respective function will be called. In this fashion an arbitrary number of processes can be scheduled for execution and quantised based on a given tempo.

4. Application Example

The strategy exemplified in this paper as well as the CuePlayer's design, have been used in interactive works and proved to be useful and stable for composing and performing. In Karamanlis' (2016) *GO*[4], a series of pieces commissioned by Onassis Cultural Centre for traditional instruments (Ney, Kanun) and laptop, the electronics run in SuperCollider and the processes are organised in cues, triggered either via a foot-pedal by the instrumentalists or by the laptopist situated in the centre of the hall. The overall musical structure is fixed, yet any processing takes place in real-time while the music unfolds in front of the audience, involving a significant amount of indeterminacy due to the inherent nature of the algorithms. The electronic sound

makes use of pulse-based material, polyrhythmic structures and expands the timbral pallet of the acoustic instruments. It provides an additional layer of sonic possibilities through the use of technology while viewing the performance space as a compositional parameter by using multiple speakers.

For each piece of the GO project a CuePlayer is responsible for organising the electroacoustic sound, sequencing the events and triggering cues. Following an extensive sampling session for every instrument, a virtual sampler was constructed in each case which helped to create a mock-up of the performance using a sequencer running side by side with SuperCollider as exemplified earlier. The strategy allowed for the pieces to be thoroughly tested in studio in a quadraphonic setup, prepare and structure the material off-line and then be presented in an octaphonic system with minimal adjustments in the code.



Excerpt of *Asynchronous Looping* for Kanun, Ney & Laptop, © 2016, Orestis Karamanlis.

5. Exit

We have looked at a methodology applicable to musical works incorporating real-time electronics and acoustic instruments and introduced the CuePlayer, a tool aimed at mixed music which may also be useful in any scenario where the composer wishes to set up, schedule and trigger bundles of processes. There are different ways to structure interactive works. We do not imply that working with cues is always the most efficient. For many generative works that cannot easily be reduced in sections, thinking in cues may not be the best option. Still, this standard practice, when coupled with a compact cue-playing system with added functionalities, can be very effective when a high degree of coordination is needed. Structuring the electronics using an open-source language may increase the piece's life expectancy, hopefully making it more adaptable to changes as technology advances and reducing the chance of the code, and ultimately the piece itself, becoming obsolete.

Acknowledgments

Many thanks to Ambrose Seddon for his well-thought comments on the initial document. All errors are our own.

References

Agostini, A. & Ghisi, D. 2013. Real-Time Computer-Aided Composition with bach, *Contemporary Music Review*, 32:1, 41-48.

Bevelander, B. 1991. Observations on Live Electronics, *Contemporary Music Review*, 6:1, 151-157.

Emmerson, S. 1994. ‘Live’ versus ‘Real-Time’, *Contemporary Music Review*, 10:2, 95-101.

Emmerson, S. 2000. 'Loosing Touch?': The Human Performer and Electronics, *Music, Electronic Media and Culture*, 194-216, Ashgate.

Hagan, K. 2016. The Intersection of ‘Live’ and ‘Real-time’, *Organised Sound*, 21(2): 138–146.

McNutt, E. 2003. Performing Electroacoustic Music: A Wider View of Interactivity, *Organised Sound*, 8(3): 297-304.

Risset, J.C. 1999. Composing in Real-Time?, *Contemporary Music Review*, 18:3, 31—39.

Rowe, R. 1999. The Aesthetics of Interactive Music Systems, *Contemporary Music Review*, 18:3, 83—87.

Stroppa, M. 1999. Live Electronics or...Live Music? Towards a Critique of Interaction, *Contemporary Music Review*, 18:3, 41—

1. Here, in the case of graphic notation we assume the use of targeted visual symbols which may fall outside the strict boundaries of traditional notation, but without granting absolute interpretive freedom as in §1.2. ↩
2. See for example Agostini and Ghisi's bach library for Max.↩
3. http://fasmatwist.com/opensource, accessed on 12/2016.↩
4. http://orestiskaramanlis.net/go, accessed on 12/2016.↩