Music across music : towards a corpus-based, interactive computer-aided composition

Daniele Ghisi

Music Across Music: Towards a Corpus-Based, Interactive Computer-Aided Composition

A thesis presented by

Daniele Ghisi

Université Pierre et Marie Curie
Université Paris-Sorbonne
Sorbonne Universités
École doctorale Informatique, Télécommunications et Électronique
Institut de Recherche et Coordination Acoustique/Musique
UMR STMS 9912

To obtain the title of Doctor of Philosophy
Specialty: Music Research and Composition

Thesis Supervisor: Carlos Augusto Agon Amado

Defended on December 19th, 2017

Jury:

Reviewers:	Georg Hajdu	- HfMT (Hamburg)
	Pierre Alexandre Tremblay	- University of Huddersfield
Supervisor:	Carlos Augusto Agon Amado	- UPMC (Paris)
Co-supervisor:	Pierre Couprie	- Université Paris-Sorbonne
Examinators:	Jean-Pierre Briot	- UPMC (Paris)
	Myriam Desainte-Catherine	- LaBRI (Bordeaux)
	Miller Smith Puckette	- University of California San Diego
Invited:	Jean-Baptiste Barrière	
	Yannis Kyriakides	

Abstract

The reworking of existing music in order to build new one is a quintessential characteristic of the Western musical tradition. This thesis proposes and discusses my personal approach to the subject: the borrowing of music fragments from large-scale corpora (containing audio samples as well as symbolic scores) in order to build a low-level, descriptor-based palette of grains. A fundamental meaning of composing is hence embodied in the acts of filtering, ordering, organizing and querying, whose parameters are handled via digital hybrid scores, in order to equip corpus-based composition with the control of notational practices. Some personal techniques are presented, along with the aesthetic choices that motivated them and the musical works to which they relate. Most of these techniques can be subsumed under a larger exploratory attitude towards music: composition is an interactive, intuitive discovery—more than an 'invention'.

This thesis also introduces the *dada* library, designed and developed with these considerations in mind, providing Max [Puckette, 2002] with the ability to organize, select and generate musical content via a set of graphical interfaces manifesting the aforementioned exploratory approach. Its modules address a range of scenarios, including, but not limited to, database visualization, score segmentation and analysis, concatenative synthesis, music generation via physical or geometrical modelling, wave terrain synthesis, graph exploration, cellular automata, swarm intelligence, and videogames. The library is open-source and extendable; similarly to *bach* [Agostini and Ghisi, 2013], it fosters a performative approach to computer-aided composition (as opposed to traditional off-line techniques): the outcome of all its interfaces can be easily recorded in scores, or used in real time to drive, among other things, digital signal processes, score transformations, video treatments, or physical actuators.

Finally, this thesis addresses the issue of whether classical representation of music, disentangled in the standard set of traditional parameters, is optimal. Two possible alternatives to orthogonal decompositions are presented: grain-based score representations, inheriting techniques from corpus-based composition, and unsupervised machine learning models, providing entangled, 'agnostic' representations of music. As a lateral yet related subject, the thesis details my first experience of collaborative writing within the */nu/thing* collective, in open contrast with the common image of the composer as a Cartesian 'solitary self': although in very different ways, corpus-based composition, generative neural networks and collective practices all cause into question the scope of musical authorship.

Keywords: computer-aided composition, corpus-based composition, music representation, real time, Max, bach, dada, neural networks, machine learning, collective writing

Acknowledgments

First and foremost I want to thank my supervisor, Carlos Agon, for his guidance throughout this research study, and my co-supervisor, Pierre Couprie, for his support.

Secondly, I am not so sure this thesis would have even existed had I not met, some ten years ago, Andrea Agostini, with whom I share a large portion of musical research—and yes, I might have 'borrowed' from him a couple of ideas along the way: I hope I returned them with due credit and with at least some minimal added value. I would like to thank Eric Maestri, whose insights have often been cause for reflection, and the other members of the */nu/thing* collective (Andrea Sarto, Raffaele Grimaldi), which is probably the most purposeful experience I have recently undertook. I would like to thank Marco Momi, Giovanni Bertelli, Carmine Cella for their keen observations that have shed some light on my doubts more than once. I would also like to thank Boris Labbé: the collaboration with him has shaped a portion of the ideas proposed in this thesis.

Importantly, I would like to thank the entire Ircam community (not just researchers, but production, pedagogy and RIMs as well) for making me feel always at home. I am particularly grateful to Moreno Andreatta, Gérard Assayag, Julia Blondeau, Eric Daubresse, Jean Louis Giavitto, Marco Liuni and Yan Maresz for their support and their suggestions on my researches; to Mattia G. Bergomi and Philippe Esling for their assistance and their fascinating intuitions; to Robin Meier for his aid and support on the project *La fabrique des monstres*; and to Léopold Crestel, for helping me tweak recurrent neural networks models.

I am grateful to Fabien Lévy and Mikhail Malt for their observations on my mid-term summary report.

I am especially grateful to Georg Hajdu and Pierre Alexandre Tremblay for their thorough review of this thesis.

Additional credits

Sections 2.3.1, 2.3.2 and 2.3.3 have been previously published, in a slightly different form, in

- Ghisi, D. and Bergomi, M. (2016). Concatenative synthesis via chord-based segmentation for An Experiment with Time. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.

Some portions of sections 3.1 and 3.2.4, as well as the the entirety of sections 3.2.2 and 3.5 have been previously published, in a slightly different form, in

- Ghisi, D. and Agostini, A. (2017). Extending bach: A Family of Libraries for Real-time Computer-assisted Composition in Max. *Journal of New Music Research*, 46(1):34–53.

Some portions of section 3.3.2 have been previously published in

- Ghisi, D. and Agon, C. (2016). Real-time corpus-based concatenative synthesis for symbolic notation. In *Proceedings of the TENOR Conference*, Cambridge, United Kingdom.

Sections 3.4.1, 3.4.2 and 3.4.3 have been previously published, in a slightly different form, in

- Ghisi, D., Agostini, A., and Maestri, E. (2016). Recreating Gérard Grisey's Vortex Temporum with cage. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.

Section 4.4.2 expands and elaborates some ideas developed by the whole */nu/thing* group while writing the dossier for *I mille fuochi dell'universo*.

In addition, some examples were presented in the following talks:

- *Interfaces reactives pour la composition assistée par ordinateur en temps réel*, Collegium Musicae, Ircam, Paris (13/05/2016)
- *Tabula plena: un tentativo di orientamento alla composizione attraverso le basi di dati*, Budrio, Dialoghi sul comporre (28/12/2016)
- *Music and the Digital*, Biennale Digital, Venezia (03/02/2017)
- *Synthèse par RNNs pour "La fabrique des monstres"*, Meridién Ircam, Paris (05/09/2017)
- *Corpus-Based Composition and Interactive Computer Interfaces*, Institute for Computer Music and Sound Technology, Zurich (02/10/2017)

Contents

Chapter 1

Introduction

1.1 Music across music

One of the recurring questions in philosophy of science is whether mathematics and physics are discovered or invented. By extension, one might wonder whether music, and art in general, are discovered or invented [Kivy, 1987].

The artistic dispute is usually tackled by counterposition to the scientific one. A popular quote, credited to a multiplicity of sources and cited in a variety of alternative forms, says that "if Einstein hadn't formulated the Theory of Relativity, then someone else would, while if Beethoven hadn't composed the *Moonlight Sonata*, no one else would". This sentence represents a widespread opinion: science is a laborious yet somehow foreseeable exploration, while art is a pure act of creation. Connected to this idea is another common belief: a scientist can only understand the world from an outside point of view, whereas an artist has somehow within his grasp the sum of all human experience. 'Conscience'—whatever it may mean—is the seal of 'invention'. Artists are unique, scientists are interchangeable cogs.

This point of view entails, incidentally, a certain number of curious corollaries, such as: "Science is not an occupation for a person who wants to make a mark as an individual, accomplishing something only that individual can do" [Lightman, 2005]. In other words: science would be collective, insofar as it needs to tackle issues efficiently (sometimes even 'mechanically'), while art would be individual, insofar as it needs to be expression of a 'conscience'. Similar assertions are common for other fields of study; for instance, composer John Luther Adams writes: "I decided that someone else could take my place in politics; and no one could make the music I imagined but me" [Adams, 2016].

Notwithstanding some elements of good sense, I cannot approve a similar paradigm. On one hand, science being a far from linear process, I am not persuaded that we would have General Relativity today if Einstein hadn't worked on it—but this is a personal, almost irrelevant detail. (And I do believe, incidentally, that great politicians are hardly replaceable, if not with detrimental consequences.) On the other hand, I'm convinced that if John Cage hadn't written his well known *4'33"*, someone else would have written an analogous piece eventually. In addition, there is a certain consensus on the fact that a number of 'fashionable' contemporary music pieces sound so alike, even for well-trained ears; I would be temped to conclude that had their authors not written them, someone else probably would have—if not with the exact same notes, at least in a roughly equivalent form. Sometimes individual artists are interchangeable too.

Vice versa, essentially all mathematicians recognize that creativity is an indispensable stage of their scientific enquiry, starting with the the choice of research subject and the angle of approach. As Alain Connes puts it:

> There are several phases in the process leading to 'find' new math, and while the 'checking' phase is scary and involves just rationality and concentration, the 'creative' first phase is of a totally different nature. In some sense it requires a kind of protection of one's ignorance since there are always billions of rational reasons not to look at a problem which has been unsuccessfully looked at by generations of mathematicians. [Connes, 2005]

The description of the creative process isn't that far off from what a composer may experience while writing the first notes of a new piece. For instance, tackling the incipit of a string quartet also requires 'a kind of protection of one's ignorance': if composers had to ponder the whole amount of existing music literature while starting to write, they probably wouldn't settle on any note. And, of course, writing a string quartet has also tedious 'checking' phases, requiring rationality and concentration only—such as verifying playability for double stops, writing string positions for sequences of harmonics, adding cautionary accidentals, and so on.

If one really believes in an opposition between acts of research and acts of creativity, one should at least be prepared to admit that the two paradigms are embraced by both science and art—in different proportions, depending on the situation.

In his speculative story *Melancholy Elephants* (1984), Spider Robinson goes one step further:

> Artists have been deluding themselves for centuries with the notion that they create. In fact they do nothing of the sort. They discover. Inherent in the nature of reality are a number of combinations of musical tones that will be perceived as pleasing by a human central nervous system. For millennia we have been discovering them, implicit in the universe, and telling ourselves that we 'created' them. To create implies infinite possibility, to discover implies finite possibility. As a species I think we will react poorly to having our noses rubbed in the fact that we are discoverers and not creators. [Robinson, 1984, p. 16]

The science fiction bottomline is fascinating—although to discover does not necessarily imply a finite number of possibilities: for instance, one can search for (and discover) new couples of twin primes even they are conjectured to be infinitely many.

I am tempted to suggest that, in the case of music, the words 'creation' and 'discovery' are labels—perhaps referencing the same core principle?—that we put as placeholders on two different operative paradigms, for lack of better understanding.

The 'inventive' paradigm, usually encouraged in Western music schools and conservatories, is connected to the idea that introspection and reasoning help determining, at each step of the composition process, which elements should be introduced.

The composer, in front of a piece of paper, or even in front of a computer screen, introduces notes and figures one by one, taking the time to introspectively determine what he or she considers, for a given context, interesting, relevant, beautiful. One might speculate that one of the keys to go beyond the white page (a *tabula rasa*) is the very act of 'protection of one's ignorance' that Connes mentioned: the music literature is so stifling that at some point composers need to obliterate it. I recall that one particular friend, several years ago, had an extreme position on the subject, confessing candidly that he didn't like going to any concert, because it was a distraction from his own music.

The 'exploratory' paradigm, by contrast, requires external events to provoke ideas. Such events can be either passively received, or actively searched. The composer is 'listening' (*à l'écoute*), ready to create connections, identify interesting patterns, and work on them. Collages and quotations relate to this paradigm, but the discovery is not necessarily bound to browsing from existing music: one might devise combinatorial strategies where a large quantity of sounds or scores is generated (a *tabula plena*), and specific portions of the generated material are explored later on. The literature is still stifling, but the composer chooses to work on a portion of it, from a certain angle.

One might be tempted to read in the two paradigms some symptoms of the opposition between modernism and postmodernism:

> Influence was a critical issue for modernists. Postmodernists, however, [...] have passed beyond their Oedipal conflicts with their modernist parents, although they may still have an uneasy relationship with them. [Kramer, 2016, p. 15]

Yet both categories are hard to define—postmodernism, especially, being a "slippery, contentious beast", to cite Linda Hutcheon. As a matter of fact, the two paradigms are not in conflict: the 'discovery' paradigm still requires an 'inner self' to resonate with external events—usually via acts of 'surprise' or 'shock'; and conversely, one might well envisage that, when 'scanning' introspectively for ideas, a certain modified combination of the concept we have learned, of the music we have loved, of the scores we have cherished, emerges to mind (and to pencil). Isn't our choice of 'interesting content' an indirect consequence of our aesthetic preferences, our training, our 'base of knowledge'? Our mind is in itself a *tabula plena* [Antiseri, 2005, p. 21].

I imagine myself one day, sitting at my table, taking a pencil in my hand, determined to start writing a string quartet, note by note; and then imagine myself, the following day, collecting the scores of all the string quartets I can find, trying to isolate short figures that impress me, using them to construct meaningful elements, and, modification after modification, eventually transcribing them on paper. These are two operatively different processes—but are they *fundamentally* different? Don't 'point blank invention' and 'processed musical borrowing' eventually reverberate into a similar reservoir—past knowledge, education, inclination, preferences—, only the latter bearing a more explicit tone? What are the boundaries between the two operations? Is their distance worth exploring?

The concept of musical borrowing, in particular, echoes controversies dating at least as back as the mid-eighties, when the term 'plunderphonics' was coined by John Oswald as a category for music made by playing and altering existing audio recordings. The process of using existing music to build new one dates back at least to tropes, paraphrase masses and quodlibets; nowadays, genres such as jazz or hip hop heavily rely on borrowing in order to produce new works. The advent of recording techniques, during the 20th century, has made it easier to cite, reproduce or remix previous works, opening the way for electronics experimentations, sampling, 'turntablism', and the plunderphonic movement itself. In his 1985 manifesto, Oswald declares that:

> Musical instruments produce sounds. Composers produce music. Musical instruments reproduce music. Tape recorders, radios, disc players, etc., reproduce sound. A device such as a wind-up music box produces sound and reproduces music. A phonograph in the hands of a hip hop/scratch artist who plays a record like an electronic washboard with a phonographic needle as a plectrum, produces sounds which are unique and not reproduced - the record player becomes a musical instrument. A sampler, in essence a recording, transforming instrument, is simultaneously a documenting device and a creative device, in effect reducing a distinction manifested by copyright. [Oswald, 1985]

Oswald continues, by stating that "the distinction between sound producers and sound reproducers is easily blurred, and has been a conceivable area of musical pursuit at least since John Cage's use of radios in the Forties".

The digital revolution democratized music production and reproduction [Lehmann, 2017], making editing faster and cheaper: today, virtually all composers can achieve on their own laptop in a few minutes operations that would have required many hours of delicate work on magnetic tapes. More importantly, in our digital era, the interest has shifted from recording supports to encoding types; all digital media are fundamentally stored as binary code: what distinguishes a recording from a photograph, or even a recording from a score, is the way in which information is interpreted. This fact has certainly contributed to bridging the gaps between different disciplines: it is not uncommon that the same physical tools, such as controllers, or algorithmic tools, such as machine learning techniques, may be indifferently applied to music, images, videos or texts.

* * *

My interest in musical borrowing first appeared during my adolescence. I had just discovered Berio's *Sinfonia* and I was amazed: I had the feeling that its quotations were a manner to assign some kind of shared 'meaning' to notes, something as close to a word as I could find in music. I started using quotations in my electroacustic pieces, rather naively at the beginning, then more and more consciously. I decided to tackle the question from the electronic side essentially because I needed

to keep an eye on the possibilities opened by algorithmic tools—and it seemed much easier to deal algorithmically with audio montages than with score montages. It was a first attempt to combine my long-time interest for computer-aided composition with the emerging interest for corpus-based composition. Soon, the number of quotations increased, and I started collecting audio files inside folders on my laptop, one folder for each project. Then, for some projects, one folder was no longer enough: as the amount of data increased, so did the need for a smarter access to searching portions of sound files, according to labels, properties, or harmonic configurations. The audio files had already stopped being quotations by that time—they were simple raw material, 'colors' on a palette.

The present thesis describes the research lines that stemmed from this approach and that I have pursued in the last three years, along with the musical works to which they have lead. I have mainly focused on a digital, interactive, exploratory approach to music, based on a large-scale computer-aided systematization of musical borrowing.

The principal project in which I was able to coordinate an important portion of these ideas is *An Experiment with Time*, an audio-visual installation, also available in concert form with a live amplified ensemble, inspired to the eponymous book by John W. Dunne. In *An Experiment with Time*, I organized large quantities of audio content, sampled across the Western history of music, to be used as a compositional palette: each folder clustered audio fragments matching a given harmonic configuration; composition was also based on the analysis of some of their features ('descriptors'). After the experience with *An Experiment with Time*, it was clear to me that I had to extend the approach to actual databases and to implement an intuitive access to samples—a 'digital pencil'.

Using musical corpora as a palette is deeply inspiring; personally, it is also a way to acknowledge how we all, although in different ways, write 'on the skin of the world'. The range of applications of this concept is wide; after *An Experiment with Time*, two paths seemed to be most promising to me: small-grain concatenations, where the boundary between musical borrowing and usage of primary components (notes, figures, samples, frames) is extremely blurred; and large-grain concatenations, where each component in the montage is well recognizable, and the borrowing is somehow declared. Some of the latter have been temporarily and disorderly collected under the title *Electronic Studies*, and are to me a sort of reminder of how composing can revert to its original meaning of 'putting together'.

With few exceptions, my musical production of the last three years, as well as a certain number of currently ongoing projects, stems from these ideas: *The Well Tempered Sampler* is a work in progress directly related to the chord organization of *An Experiment with Time*; *Any Road* applies concatenations to the detection of speech patterns; *An Urban Dictionary of Popular Music* applies similar ideas in order to build a dictionary of sung words; *Mon corps parle tout seul* turns a mouth, formed by water droplets, into a sound-spitting oracle by means of corpus-based speech reconstruction techniques ('musaicing'); *269 Steps Away From you (269 Steps Away From Me)* is essentially a sequence of concatenated borrowed elements. Also

while working with audiovisual projects, I have tried to take advantage of corpus-based tools: *Orogenesis* brings choir chants into regions of extreme aliasing; *Chute* is constructed starting from a large number of recordings of string quartets.

Dealing with large music datasets is technically challenging for a variety of reasons: for one thing, more often than not, music software is simply unfit to handle big amounts of data, resulting in memory or performance issues that often render computer-aided composition impractical, when not impossible. For another thing, I had to use lossy compression formats in my large datasets—calibrating the quality of the compression to be high enough so that the difference between compressed and original audio was negligible for my own purposes. But it was just a preamble: the real issues with audio quality came as soon as I got interested in machine learning techniques.

Given my passion for large music corpora, machine learning seemed a natural and perspectively promising set of techniques. The project *La fabrique des monstres* stems from a simple curiosity: how could we design an artificial neural network that only takes as input a sequence of digital audio samples (it 'listens'), and is able to learn and reproduce patterns from it? Sample-by-sample learning techniques on raw audio data currently require large amounts of memory and computing time; virtually all state-of-the-art models work best with low sample rates (16kHz being extremely common) and low output bit depths (usually 8). Composers are then faced with a dilemma: should they wait for the technology to improve, or should they engage in a research process, with the risk of ending up producing low-quality audio outcomes, that in ten years' time might sound connotated, or even laughable? There is no right answer, of course. I personally consider this kind of engagement essential, and I live it both as a pioneering experience and as an avant-garde gesture. In ten years' time we will surely have better tools—but they have to stem from somewhere. For *La fabrique des monstres* I chose to assume the low-fidelity electronic outcome, and I tried to handle the music composition accordingly. Sometimes quality is simply not the point.

Artificial creativity is especially inspiring because it questions authoriality to a very large extent. Via relatively simple recurrent network models one can already produce rather convincing samples mimicking the style of a given training set. Curiously enough, if one hears the generated music without paying much attention, one might easily mistake it for the original corpus, even though when one listens more closely, the difference is well apparent. What is at stake is not just an entertaining generator of airport music: as soon as our computers will be able to handle longer memories—which will happen in a very near future—it would probably be hard, if not impossible, to distinguish between 'originals' and 'copies'. Who would be the author, then? How would our copyright framework adapt to the brand new revolution? Perspectives look both scary and exciting.

As I was working on neural networks, some writings by David Cope came to mind. Cope proposed the idea of 'virtual music' [Cope, 2004] as a category of compositions attempting to replicate a style (such as figured basses, or musical dice games, or his own *Experiments in Musical Intelligence*). The neural network models

with which I was experimenting fitted perfectly in the category. Nonetheless, I was at least as much interested in the differences between the style and the outcomes as I was in replicating patterns. Especially exciting were the moments when a vocal vibrato got too exaggerated, when a high soprano note morphed into an oboe sound, when a noise burst came to break a pastoral situation, when the harmony was suddenly yet coherently 'off'. The latter was indeed one of the most unsettling experiences: at times the harmony was both off and pertinent at the same time, generating a strikingly unique and ineffable feeling.

As I was reading Cope's writings again, it was nice to rediscover a certain number of points in common with Spider Robinson's novel and with my own combinatorial attitude:

> Much of what happens in the universe results from recombination. [...] Music is no different. The recombinations of pitches and durations represent the basic building blocks of music. Recombination of larger groupings of pitches and durations, I believe, form the basis for musical composition and help establish the essence of both personal and cultural musical styles. [Cope, 2004, p. 1]

And with an exploratory, trial-and-error approach:

> I think that most composers would appreciate hearing possible solutions to problems, fresh perspectives on a just-composed or just-about-completed passage, or experimental extensions to a new phrase. These composers will always make the ultimate decision as to whether or not to use such music. Even if all the output proposed by [a] program fails to please, it might still inspire, if only by its failure to persuade. [Cope, 2004, p. 344]

I found a third point of contact with Cope's ideas: the belief that a composer may attempt to go beyond his or her being a 'solitary self'. Collective work is a common habit in most contemporary fields, including science, visual arts, cinema, pop and rock music: in all these cases, collective practices have revolutionized the production mechanisms as well as the way of thinking. For several reasons, the milieu of contemporary music has been virtually immune to this revolution: although co-signed works are not new in the history of Western music, they mostly refer to works in which each of the signatories is in charge of a particular section, or an author completes a previously unfinished project, or each of the signatories assumes a specific role. Some attempts to go beyond this situation have mostly been made through technological tools: the *Edison Studio* collective uses audio technologies as key to creatively combine ideas; David Cope himself added collaborative opportunities to his *Experiments in Musical Intelligence* in a software named *Alice*. Even so, the topic of multi-author composition seems to be absent from the collective agenda—the vast majority of today's composers simply not considering it a priority.

In 2010 I have formed, together with five other Italian composers, a collective named */nu/thing*, initially gathered around a blog. In the last two years, we have

engaged in a more ambitious project: truly collective writing, a process where both the signature and the way of working were to be inherently shared. The result is the piece *I mille fuochi dell'universo*, which I found compelling for many reasons, but mostly because I personally do not believe that individuality will be the signature of future generations of composers. *I mille fuochi dell'universo* is both our first attempt to write together and a sort of manifesto, a statement of direction.

All these experimentations are described throughout this thesis, along with the dedicated digital tools developed to support them. I had to refine an approach to audio and to scores which would use live interfaces as pencils, descriptors as rulers and databases as paper sheets. I started developing a certain number of modules, mostly two-dimensional interfaces, now organized and distributed in a library named *dada*, for the Max environment [Puckette, 2002]. The *dada* library, in turn, is based on *bach: automated composer's helper*, an independent project bringing interactive scores inside Max, on which I have been working since 2010 with composer and friend Andrea Agostini [Agostini and Ghisi, 2013]. The *dada* library is open source and provides Max with a set of non-standard user interfaces for music generation, all sharing a performative, explorative approach to composition. A subset of modules explicitly deals with database representations. Most of the communication paradigms, as well as the data representation, is inherited from *bach*. The choice of Max as working environment was prompted by its simplicity of integration with a multitude of processes and devices, a flexibility from which any addition to the system would directly benefit.

The result is a digital lutherie that tends to employ computer interfaces similarly to how a 19th century composer could have used a piano: both as a discovery 'playground' and as a validation environment. Digital tools allow to explore paths, to be surprised by their outcomes, and to ponder, choose, organize, formalize and validate their results. Importantly, interactivity is a choice, not an obligation: in most cases, the 'off-line' portion of the musical process is at least as important as the exploratory playground. Nonetheless, I believe that via interactive representations of music composers can take advantage not just of scores to play (as a means of communication), but of scores to play *with* (as 'instruments' proper).

* * *

Music critic Ivan Hewett claims that "the era of intense formal exploration, when composers sit at their desk, or more likely at their laptops, and explore ever more arcane forms of self-creating grammars for music are coming to an end" [Hewett, 2014]. I can understand why similar postures have found ground to thrive: the twentieth century, in particular, has witnessed an impressive sequence of formal approaches and techniques, some of which have dwelled in a sort of hyperuranion where abstract processes, in some way, both disregarded and justified the sound phenomena—as Grisey [1987] stated, composers have sometimes mistaken the map with the territory.

Yet holding formalization or technology responsible for such mistakes is dangerous. The history of music is also a history of both music formalism and music

technologies, from Pythagorean experiments on intervals to Xenakis's constructions, from the invention of notation to the advent of synthesizers. Are the "intense formal explorations" of 15th century Franco-Flemish school so impenetrable? Are the "self creating grammars" of 20th century spectralism so arcane?

Incidentally, in the contemporary music milieu, I have often heard that composers should be careful not to get 'pushed around' by technology. Sometimes it is said to encourage them to keep reflecting both on the tools and on their outcomes—which I find, of course, compelling and important. At times, however, it is said to proclaim some sort of independence (or even superiority), which puzzles me to a large extent. Throughout the history of music, composers have constantly been influenced (and sometimes even 'pushed around') by technological discoveries, with excellent results. Technology provides new possibilities to express and convey musical ideas, be they in the form of a 17th century violin or an Arduino-controlled vinyl turning table. In the 15th and 16th centuries, printing technologies allowed the spreading of polyphonic music and encouraged experimentation; nowadays, the digital technologies allow for unprecedented treatments in audio and score processing. Developments in technologies invariably foster new ways of thinking.

Since composing also means 'reflecting upon music' [Lachenmann, 2004], formalism is one of the pillars of our Western musical approach. Technology and formalism are simply the canvas on which we operate. One may, of course, put the canvas itself into question—with the awareness that such process is not about expunging the last few decades of arcane contemporary music, but rather about challenging the whole history of Western musical thinking.

Personally, I cannot renounce formalism. What I like in a formal approach is the moment when pieces of a random-like puzzle fall into place in a seemingly natural way, and meaningful structures emerge: the instant in a retrograde canon when certain notes 'miraculously' make sense; the moment when one perceives an appearing monstrosity in a certain accumulation of objects; the discovery of a well known landscape from a collage of small portraits; the constant variation of causal relationships in a polyrhythmic structure.

There's an adrenaline discharge in these discoveries; there is a certain kind of ineluctability leading to the keen perception, for an instant, that some things are unequivocally bigger than us, which makes us wonder what is our place in the universe or how it is possible that time seems to flow. In my opinion, those are questions music should attempt to address, and formal approaches are a very good way to provoke those feelings, by making tangible humanity and abstract mathematics collide, and therefore putting them both in perspective. After all, humans use formalism and technology also as means to go beyond themselves. In Guillaume de Machaut's *Ma fin est mon commencement*, formalism is essential to express a contrast between finiteness and infinity; in Heinz Holliger's *Scardanelli Zyklus*, formalism is a way to attain a monumental level of writing, connecting our space and time with different, higher ones.

Significantly, some aspects of a technological, formal approach have also merged into the composite, ambiguous melting pot often called 'postmodernism', encom-

passing other disciplines, and sometimes even edging to a 'pop culture' approach, ranging from Luciano Berio's orchestral works to modern TV series. There is also a ludic side: as Lewis Carroll indicates, symmetries carry with themselves the playful and yet diabolical taste of spontaneous multiplications of elements. Most times, the game evaporates quickly, and has no aftertaste; but in some situations, it spirals into a vertiginous mechanism that hints to deep questions and high archetypes, and the aftertaste is bitter and indelible. Machaut's interest in retrogradation is well recognizable in Michel Gondry's video for the song *Sugar Water* or in the reversal of Elvis's *Suspicious Mind* that has originated *Ignudi fra i nudisti*, by the Italian rock group 'Elio e le storie tese'; Bach's interest for canonical forms converges into Norman McLaren's *Canon*; Ligeti's polyrhythms will influence a large number of works of art, including Zbigniew Rybczyński's *Tango*, portraying life as a superposition of cycles.

As for this thesis, in honest contrast with Hewett's portrayal, it will represent the wandering of the mind of a solitary composer, sitting at his desk, or more likely at his laptop—exploring. And yet the exploration will increasingly attempt to reduce solipsism by taking advantage of collective music practices; to mitigate egocentrism by assuming that we all stand on the shoulders of thousand giants; to shift the meaning of the word 'invention', by mediating exploration through surprise; to use computer-aided composition software as interactively as a piano, by developing reactive paradigms for score generation and handling; to use scores not just as means of prescriptive notation, but as instruments to play *with*, by exploiting interactive hybrid scores; to cause into question the nature itself of musical notation, by looking for less 'Cartesian' and more parameter entangled representations of music.

I believe that all these activities should contribute to replacing Hewett's sketchy portrayal of a modernist composer with an actually *modern* one.

1.2 Outline

The main body of the thesis is structured around three areas of enquiry.

Chapter 2 examines and discusses my personal views on a corpus-based approach to music composition, analyzing the implications of the different exploration-driven paradigms I have used in my recent works (such as concatenations of audio fragments, database querying and muzaicing), and transferring some of them to the symbolic domain. The chapter also presents my techniques to handle music datasets, along with both the aesthetic principles that motivate them and the description of my recent works that were based on them.

Chapter 3 is devoted to the description of the reactive computer music tools I developed to support my workflow. It explains why a reactive model is important for my research and how real-time computer assisted-composition helps bridging the gap between a 'performative' and a 'speculative' approach. I briefly describe the *bach* and *cage* libraries, which Andrea Agostini and I have been developing and supporting since 2010 and 2013 respectively; then I introduce and detail the *dada* library, which I designed within the framework of this thesis and used in my recent works: its focus is on non-standard real-time interfaces to control analysis, organization and generation of music. Finally, the chapter introduces and discusses reactive hybrid scores (meta-scores), a key tool, in my recent line of work, combining the properties of both prescriptive scores and interactive instruments.

Chapter 4 raises the question whether our representation of music is optimal. The main issue is that notation, especially in a computer-aided composition context, is extremely 'Cartesian', in that it separates musical parameters in an orthogonal way. I will explain why, in my opinion, parameter entangled representations of music might fit better in certain scenarios. In order to start addressing the issue, two paths are explored: on one hand, a grain-based score representation is proposed (based on the ideas exposed in chapter 2 and the tools developed in chapter 3); on the other hand, some machine learning models are discussed, especially in connection with the usage of recurrent neural networks in my most recent works, as a way to obtain abstract, entangled representations of music. As a lateral, and yet very connected, topic, the chapter also motivates my interest in collective writing practices, causing into question the view of a composer as a 'solitary self', as well as the role of authorship.

1.3 A disclaimer

Although this thesis may contain pieces of information pertaining to the fields of study of musicology and computer science, it is neither a musicological nor a scientific thesis. While writing it, I have often felt as an acrobat trying to balance himself on a narrow rope, exposed to the winds from both sides.

Above all, I had the feeling that a certain number of concepts I touched would contain in themselves the potential to open much broader discussions. And yet, such discussions fundamentally belong in a musicological context and are well beyond the scope of the thesis: the respective concepts are illustrated and examined only insofar as they help conveying my view on music, and positioning myself inside the existing panorama of contemporary composition. Ultimately, the aim of this work is not to methodically investigate the intricacies of a musical posture, but rather to express my choices and motivate the beliefs which led me to make them.

As for the scientific side, I have tried to pair in the text technical considerations with the musical reasons that motivate them—or with musical usages that exemplify them. I will provide careful descriptions for all the developed tools and their algorithms, whenever meaningful, but only with the goal to assess their influence in a musical context—and primarily in my own work.

In this regard, another possible misunderstanding might concern the scope and the target of the development. All the tools, techniques and patches presented in this thesis, including the *dada* library, were developed for my own, very personal, purposes, and do not aspire to universality. Any consideration about them is, hence, strictly personal, and pertains to my own aesthetic and usage. This principle has one important exception: the *bach* and *cage* libraries although stemming from the musical background of Andrea Agostini and me, are targeted to a much more general community of composers, a fact that will be kept under consideration during the discussion in Chapter 3. Both libraries, however, predate this thesis by several years, and are briefly presented and discussed only in order to introduce *dada* accurately, and to detail my current work with them.

Finally, this thesis is an attempt to organize my (musical, aesthethic, technical, scientific, methodological) ideas to outline a world view, related to a specific posture—an explorative, corpus-based approach to music—which I find fascinating, for a number of reasons. My curiosity, however, extends beyond it; a posture is not a cage, but rather a (temporarily) privileged viewpoint: there are compelling reasons for enforcing it, but there is no necessity, and other paths might be, of course, just as valid. This thesis discusses explorative paradigms and corpus-based composition because I find compelling aesthetic points in them; however, this does not imply that I systematically refrain from taking any other compositional approach, if more convenient: more often than not, techniques are borrowed across different postures. Ultimately, I hope that my world view is better conveyed by the music itself—which is why I have taken care of adding, whenever possible, audio and video excerpts to illustrate the subjects.

Chapter 2

Towards a Corpus-Based Composition

2.1 Corpus-based composition

2.1.1 Musical borrowing in Western musical tradition

The reworking of existing music in order to build new one is a quintessential characteristic of the Western musical tradition [Burkholder, 1994; Boyle and Jenkins, 2017]: late medieval tropes added new lines to pre-existing chants; Renaissance masses were often composed starting from a *cantus firmus*, typically borrowed from plainchant, or even from secular sources; *quodlibets* combined different popular melodies in order to construct coherent counterpoints, the most notable examples probably being Bach's 30th *Goldberg Variation* and the extraordinary polytonal combination of folk tunes in the 2nd movement of Biber's *Battalia*. An attempt to provide a chronology of uses of existing music can be found in [Burkholder, 1994, appendix 2]. Romantic and post-romantic composers often borrowed music in order to symbolize something (such as the *Dies Irae* in Berlioz's *Symphonie Fantastique*). More recent examples of citationism include Ives's *Central Park in the Dark*, Zimmermann's *Requiem für einen jungen Dichter*, Clarence Barlow's *Piano Trio* [Barlow, 2011] and the third movement of Berio's *Sinfonia*. This latter is so remarkable, in that it transcend the idea of mere collage, "in order to give an understanding of the affinities between stylistically distant musical ideas, [thus creating] an extremely convincing unity of sound, based of the amplest heterogeneity." [Roubet, 2013]

Berio's work has very much in common with Umberto Eco's belief that quotation is a specific trait of postmodernism:

> The avant-garde destroys the past, it disfigures it. [...] But there comes a moment when the avant-garde can go no further, because it has already produced a metalanguage to talk about its own impossible texts (for example, conceptual art and body art). [...] The postmodern response to the modern consists instead of recognizing that the past—since it may not be destroyed, for its destruction results in silence—must be revisited ironically, in a way which is not innocent. [Eco, 1984]

Leeman [2004] argues that Eco's words reveal a caricatural view of avant-garde; modernist approaches often revisit past models precisely while disfiguring them. It is however true that postmodernism gives to borrowing unprecedented scale and

intensity [Beylot, 2004]. Today the boundaries between postmodernism, modernism and avant-gardes are extremely blurred, or even overlap extensively. An analysis of the nature of such postures as well as of their mutual relationships lies well beyond the scope of this thesis. As a consequence, I will refrain from delving into the intricacies of determining if and how the contemporary usage of quotations emanate from a postmodernist approach—indeed, I will refrain from speaking of postmodernism altogether: even though the category is relevant, "it is hard to reach a consensus on what a definitive genre of postmodern contemporary music might even sound like" [Trapani, 2017]. For an extensive analysis of the characteristics of postmodernism in music, as well as the intricacies of its relationships with avant-gardism and modernism, one might read Kramer [2016].

All the aforementioned examples are an indication that, throughout time, composers have held somewhat different approaches to the reusage of existing scores. Burkholder [2004, p. 4] drafts the following comprehensive list of procedures[1]: modelling, variation, paraphrasing, setting, cantus firmus, medley, quodlibet, stylistic allusion, transcription, programmatic quotation, cumulative setting, collage, patchwork and extended patchwork. Ruviaro [2007] engages in a similar task: "To borrow. To quote. To parody. To paraphrase. To model. To make a collage or a mash-up. To allude or refer to. To steal. To arrange. To remix. To transcribe." Investigating the boundaries between these categories is a difficult exercise, which lies beyond the scope of this work.

With the advent of recording techniques, and especially with their widespread diffusion, borrowing and processing music became also a performative, physical activity. Movements such as the plunderphonic group made a manifesto out of this idea, questioning the boundaries of esthetics as well as the ones of copyright. The modern view of DJs as 'creative' seems to be, after all, *mutatis mutandis*, a reverberation of the historical elements mentioned above.

With the appearance of digital technology, music is not just recorded, but also *encoded*. Compared to cassettes or vinyl recordings, longer recordings could fit in smaller supports—even more so when compressed encodings, such as MP3, reached their popularity in the nineties. Roughly at the same time, the interest of the computer music community has shifted from notes (or MIDI) to audio (signal processing), which finally was easier to handle in real-time scenarios. Nowadays, musicians have at their disposal large amounts of music that they can access with virtually no effort whatsoever, and that they can easily modify or process in real-time. The transformation happened so quickly that copyright laws have not managed to keep up, and we, as composers, have been trying to unveil its true potential only in the last few years. In a very real sense, we are still searching for a digital lutherie that would take advantage of the available collections of data for artistic purposes.

[1] The list is compiled with respect to Ives's works, but I believe it also has a more general validity.

2.1.2 A new kind of authorship

Drawing a precise line between legit quotations and plagiarism is especially complicated in music, since, to quote Oswald:

> Musical language has an extensive repertoire of punctuation devices but nothing equivalent to literature's " " quotation marks. Jazz musicians do not wiggle two fingers of each hand in the air, as lecturers often do, when cross referencing during their extemporizations, because on most instruments this would present some technical difficulties—plummeting trumpets and such.
>
> Without a quotation system, well-intended correspondences cannot be distinguished from plagiarism and fraud. [Oswald, 1985]

A legal discussion on the boundaries of copyright is both outside the scope of this work and, most importantly, out of my competence—nonetheless, I believe that the topic should somehow engage, to a certain extent, all contemporary artists.

Musical borrowing is inextricably connected to the matters of copyright, questioning the very nature of legal rights on music production and notation. It is therefore not surprising that artists, groups and theorists who use or encourage reworking of existing music, foster at the same time some degree of remodulation of the legal boundaries of musical ownership. It is deeply problematic that "traditional borrowing practices, long regarded as a composer's liberty, increasingly have become circumscribed by threatened litigation. In the age of digitally recorded music, a composer's use of even a short sequence from a prior recording is likely to draw a lawsuit." [Carroll, 2005, p. 957]. Christian Marclay has a personal theory on the subject: "If you make something good and interesting and not ridiculing someone or being offensive, the creators of the original material will like it" [Zalewski, 2012]. But one should not rely on beauty and interest as judgment criteria (artist still have a right to fail...), nor on owners of the original material liking the new work, putting creators in a continuous state of jeopardy.

As Arewa points out:

> In contrast to areas such as musicology [...], copyright doctrine does not adequately contemplate the entire range of ways in which cultural texts may be interrelated. Consequently, copyright doctrine is typically based on notions of creation that do not take full account of the ways in which creation of cultural texts actually occurs. [...] Copyright discourse, thus, often fails to consider sufficiently the ways in which cultural production is in many cases collaborative, as well as the ways in which cultural texts may interrelate. [Arewa, 2012, p. 33]

An author-based musical copyright focusing on originality and autonomy has much to share with a romantic vision of composers:

> The widespread presence and importance of borrowing in music is often obscured in legal commentary by the pervasiveness of the contemporary

> vision of authorship and originality. [...] Looking at actual classical music practices underscores the fact that borrowing can be a source of innovation, which raises serious questions about legal discussions of copyright that assume a dichotomy between copying and originality and that copying may be antithetical to innovation. [Arewa, 2012, pp. 36-37]

Some countries, such as the United States of America, have 'fair use' exceptions to copyright infringement. Claiming and defending 'fair use' is however delicate and problematic, and different US courts seem to disagree on the subject; in Bridgeport Music v. Dimension Films (2005), the judge famously said: "Get a licence or do not sample"—in other words: borrowing is always piracy, and the law bans piracy. To make things worse, Europe has no 'fair use' doctrine, although unsuccessful attempts have been made in this direction; in my own homeland, Italy, there is no equivalent at all to fair use or fair dealing, and exceptions to authors' rights are generally interpreted restrictively by the courts [Sica and D'Antonio, 2010]. As Boyle and Jenkins point out, at some point we have to say that some level of borrowing is just too small to bother about: it is ironic that, while illegal downloading of recorded music is rampant, "the artistic practice of *making* music has never been so tangled in cumbersome permissions and fees, licenses and collecting societies. [...] The law should serve creativity, not hinder it." [Boyle and Jenkins, 2017, p. 252]. This perspective is not uncommon among composers of my generation; for instance, Johannes Kreidler's 33-seconds piece *product placement* highlights the absurdity of modern copyright doctrine by dumping 70200 forms (one for each sample he used in the piece) in front of the Germany's licensing agency GEMA.

The problem is that, in a very real sense, art is mostly 'collectively owned'. In *Art and Agency*, Gell argues that all cultural production constructs relations not only between art objects, but between objects and persons:

> The anthropological theory of art cannot afford to have as its primary theoretical term a category or taxon of objects which are 'exclusively' art objects because the whole tendency of this theory, as I have been suggesting, is to explore a domain in which 'objects' merge with 'people' by virtue of the existence of social relations between persons and things, and persons and persons *via* things. [Gell, 1998, p. 12]

According to Gell, these relations extend both in space and in time, creating a network of 'retentions' and 'protentions'. Hence, to a large extent, artistic ideas are spread in tree-like, or even graph-like ramified structures—for which the current framework of copyright does not account at all. These considerations should suggest moving, to some degree, towards a stronger form of 'collective ownership', even though this brings up other socio-economic questions. I subscribe to Bruno Ruviaro's view on the subject:

> I would like to suggest a possible expansion, or even distortion, of the concept of borrowing; one based, in fact, in the elimination (or minimization) of the idea of individual ownership as it was discussed above.

> Instead, if a notion of collective ownership is put at work, the very meaning of borrowing undergoes a significant transformation. [...] As a result, the potential centralization of power by individuals or small groups would be minimized, if not eliminated. The borrower has as much right to borrow than any other member of the collective. [Ruviaro, 2007]

This is even more true today:

> It is important that the many new uses enabled by digital technology and the radically changed economics of music production and distribution mark the current moment as unprecedented in many ways. Consequently, we should give little weight to historically grounded arguments claiming that continuous expansion of the subject matter, scope, and duration of rights under copyright has always been, and will continue to be, socially beneficial. [Carroll, 2005, p. 958]

Incidentally, I find it quite emblematic that an epigram as famous as "Good artists borrow; great artists steal" is in itself borrowed by a large number of artists, scientists, entrepreneurs (including David Cope, Douglas Hofstadter, Pierre Boulez and Steve Jobs, only to name a few) and is credited not just to Pablo Picasso, but, although in slightly different forms, also to T.S. Eliot ("Immature poets imitate; mature poets steal" [Eliot, 1997]) and to Igor Stravinsky ("A good composer does not imitate; he steals" [Boyle and Jenkins, 2017, p.52]).

2.1.3 Exploring the *tabula plena*

The act of composing surely takes as many forms as the number of composers. However, for the sake of discussion, I will roughly outline two basic categories of approaches to which I will often refer: an 'inventive' one, where the composer faces a 'white paper' or any of its equivalents (a *tabula rasa*) and introduces musical elements via introspection; and an 'exploratory' one, where the composer comes in contact with sets of existing music (a *tabula plena*) and by selecting and modifying their elements creates new works. One might say that the former is an 'additive' approach, while the latter depend on a 'subtractive' operation.

Traditional academic teaching as well as improvisation practices usually rely on some sort of 'inventive' paradigm—although in very different forms. A certain number of composers after World War II stressed the necessity of starting from a clean slate. As Pierre Boulez put it:

> The 'tabula rasa' was something of my generation. [...] It was not 'tabula rasa' for pleasure. It was necessity, because [previous generations] had, for us, failed to find something important. We did not want to prolong this kind of failure. We were radical in the sense that we tried to establish a new way of thinking. We did not succeed all the time—but it was important for us to begin from scratch. [Duchen, 2012]

When Boulez speaks of a *tabula rasa*, he intends it, first and foremost, as an avant-garde gesture, a rupture with the burdens of the past. In contrast, montages, citationism and musical borrowing mainly refer to postures stemming from the opposite necessity: writing *with* the past, exploring it. This is not only an avantgardist idea, but also as an operational paradigm, a concrete recipe for daily work.

Differently from the 'inventive' approach, the 'exploratory' approach requires the presence of external events (discoveries, surprises...) to trigger or foster creativity. Of course, the line between the two is blurred: invention often requires selecting within a range of possibilities (e.g., choosing notes from the ones playable on a piano keyboard), and exploration still requires a critical amount of introspection to develop personal choices and associations (and hence a view of the world). In addition, more often than not, the two approaches are actually mingled: be-bop improvisations constantly build upon quotations; French spectral composers often rely on sound analysis to choose pitches; some compositional postures are based on a 'translated' recombination of extramusical experiences; and so on. One might also notice that formalization often amounts to bridging the gap between invention and exploration, or be tempted to suggest, as Spider Robinson did in his *Melancholy Elephants*, that invention is exploration in disguise. The picture is hence complex and blurred, and can hardly be compartmentalized. Notwithstanding these important caveats, I believe we can still use the two categories of 'invention' and 'exploration' as rough operational standpoints.

As far as the exploratory approach is concerned, there are multiple ways in which a composer may experience the contact with existing music: the 'external' content can be experienced (e.g., while attending a concert), searched (e.g., while browsing through a discography or a set of scores), or even programmatically constructed (e.g., while generating large amounts of random chords in order to browse through them, later on). I will focus on the two latter cases, and especially on the consequences of enlarging the 'discography' or the 'set of scores' so to include large amounts of music literature. The important side effect in this enlargement is that elements stop being 'quotations' or 'samples', and become more akin to colors on a palette: one can browse them and search for the right one as one would do with Pantone cards. The operation of searching through music dataset and modifying or combining results goes under the generic name of 'corpus-based composition' (or 'corpus-based synthesis'), which should probably constitute a separate category in Burkholder's and Ruviaro's lists. In the context of this thesis, a 'corpus' is simply a collection of scores or samples (possibly to be, in some way, analyzed).

Corpus-based composition is compelling, to me, for a number or reasons. First of all, I find it poetic and meaningful: composing is, in a very concrete sense, writing *with* the world, *on* the world, or even "in the flesh of the world"[2] [Mériaux, 2012]. In an exploratory approach, writing music is a declination of listening to music; and composing is a declination of analysing, sorting, matching, searching, exploring—there is a great amount of creativity in each of these tasks. Besides, an

[2]Translated from the original French: "Dans la chair du monde".

exploratory-driven paradigm is also a way to limit or at least to put into question a certain egotism or centralism of the contemporary composer: it is an explicit and honest way to acknowledge that, in a very real sense, we always write on others: not unlike scientists, we too stand on the shoulders of giants.

From a technical point of view, using large musical corpora is challenging: beyond a certain amount of data, it is hard to uniquely rely on human activity to collect, tag, analyze and assemble pieces of audio or scores. Computer systems may be of help in handling musical corpora and proposing matches or combinations. However, with few exceptions, computer music software is currently unfit to handle 'big data' scenarios: developing dedicated tools and accepting compromises are parts of a standard practice. As a simple example of compromise, due to the size of audio databases, I was often obliged to work with lossy compressed audio formats—which is something I have never regretted; on the contrary, I believe that breaking an electronic music 'taboo' let me focus on the core of my work more purposely.

Some of my personal 'postures' towards corpus-based composition are described below—these are, however, by no means exclusive, and most of my compositional practices belong to more than one category.

Programmatic posture. Composers search datasets for very specific matches, and programmatically handle them, for instance via mixing or concatenation. Constraints are set on the overall form of the result.

Most of my work on the *Electronic Studies* and *An Urban Dictionary of Popular Music* belongs to this category, as well as the discussion in sections 2.4 and 2.6; pieces taking advantage of automatic combination of samples, such as *Mon corps parle tout seul*, also display features of this posture (see section 2.5.2).

Generative posture. Composers produce large amounts of sound files or scores, from which they may later choose specific portions. Content is typically computer-generated, according to patterns or parameter configurations. One the crucial benefits of such approach is that it leaves room for elements of surprise: while listening to the generated results, composer may find unexpected clues that enhance or enrich their poetics—an important feature of a human-machine dialogue.

I have widely used the generative approach throughout most of my works (see for instance *An Experiment with Time* or *La fabrique des monstres*), and it has largely influenced the discussion of sections 2.3 and 4.3.1, as well as the development of the *dada* library (see section 3.3).

Filter-and-discover posture. Composers filter datasets and explore interactively the results. The selection of samples or scores happens usually in a non-programmatic way, so that the posture is not dissimilar to the aforementioned, more traditional 'inventive' paradigm, except for the fact that the material is

browsed instead of conceived. The approach is akin to browsing a dictionary for inspiration, while writing a poem. Intuitive computer interfaces are essential to smooth the creative process.

Projects like *I mille fuochi dell'universo* or *The Well Tempered Sampler* share this perspective, which has also informed the development of some of the *dada* modules (see section 3.3.2).

Notational posture. Composers use scores or meta-scores in order to notate the features results must have; automated, semi-automated or even manual processes are set in place to accomplish the goal. In this case, a 'note' (or any other elementary symbol) represents a general class of entries in a dataset, such as the class of sounds sharing the same fundamental frequency (reverting to the traditional sense of 'note'), but also—more generally—the class of sounds sharing some other significant property (e.g., a specific value of a certain descriptor). In addition, datasets can contain symbolic scores, which can be analyzed and processed with techniques borrowed from the audio domain (such as score granulation or concatenation).

Notation is an essential aspect of my musical practice, and I have taken advantage of it not only in my acoustic or mixed works, but also in most of my electronics works. It might seem unusual to praise notation after having subscribed to an exploratory, interactive view of corpus-based composition; yet I do not think that interaction and notation are in contradiction (the contrary being exactly the prerequisite on which the *bach* project has been carried on, as discussed in chapter 3). Not only is notation an essential technology to develop a 'grammar of ideas': notational models, and, more generally, representations of music, also become true *instruments* for exploration, exactly as a piano might have been an instrument of exploration for a 19th century composer (see section 3.4).

A large part of my recent work variously relates to a twofold contribution of notation: on one side, it enables control over sound-based exploratory models; on the other hand, it can constitute the object of analysis in itself: score database can be created, searched, queried, filtered, opening the way to new symbolic practices, such as score granulation or symbolic concatenative synthesis (also see sections 2.2.2 and 2.8).

Morphological posture. Composers focus on how sounds are shaped through time, and operate accordingly. For instance, retrieving similarities between sound fragments (see section 4.3.3.4), or clustering datasets into families of sounds 'of the same kind', both stem from a similar perspective—influenced, to some extent, by Denis Smalley's spectromorphology and by the study of 'temporal semiotic units' [Favory, 2007].

As Smalley points out:

> In the confusing, wide-open sound-world, composers need criteria

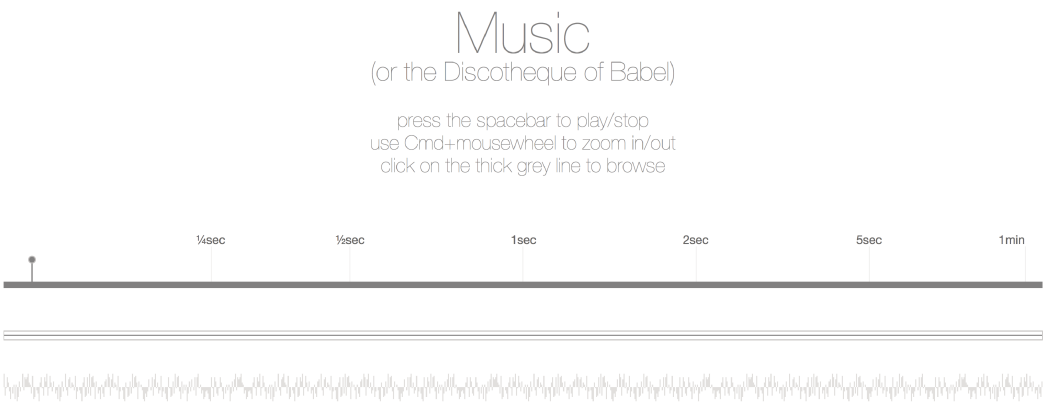


Figure 2.1: Screenshot of Music (or The Discotheque of Babel). *The interface is minimalistic: by pressing the space bar, 'all' music is played; by zooming and clicking on the line one can, in principle, browse the 'discotheque'. The current played buffer is shown at bottom.*

> for selecting sound materials and understanding structural relationships. So descriptive and conceptual tools which classify and relate sounds and structures can be valuable compositional aids. [Smalley, 1997, p. 107]

2.1.4 The Discography of Babel

In the previous section, I mentioned a fuzzy, operative opposition between 'invention' and 'exploration'; yet both paradigms have access to the same 'creative horizon'. If one is actually prepared to wait 'long enough', all music can be discovered—a concept that highly resonates with Spider Robinson's *Melancholy Elephants*.

As an elementary proof of concept, I designed a simple device, named *Music (or The Discotheque of Babel)*, where the listener can theoretically play the entire catalogue of possible digital mono music simply by pressing a spacebar, or can browse such catalogue by clicking on a line (see fig. 2.1). The device, inspired by Robinson's quote as much as by Borges's seminal novel *The Library of Babel*, is nothing more than an elementary combinatorial game, where all audio buffers up to a certain level of quantization in time (sample rate) and amplitude (bit depth) are organized on the line. If b is the bit depth, the line will display, as series of dots, the 2^b buffers composed by a single sample, then the 2^{2b} buffers composed by two samples, then the 2^{3b} buffers composed by three samples, and so on, in lexicographical order. Unfortunately, the interface is hardly usable to actually detect any interesting pattern, given that, even while zooming, one has virtually zero chance of clicking on something different than a white noise buffer (except for some quasi-cyclic buffers that show up as Moiré-like patterns when the zoom factor is a power of two).

The system is embedded in the *dada.music~* module, as part of the *dada* library (see section 3.3). The object supports multiple sample rates and bit depths.

Listening to all buffers up to 1 second, at 44100Hz, 16bit, would require approx-

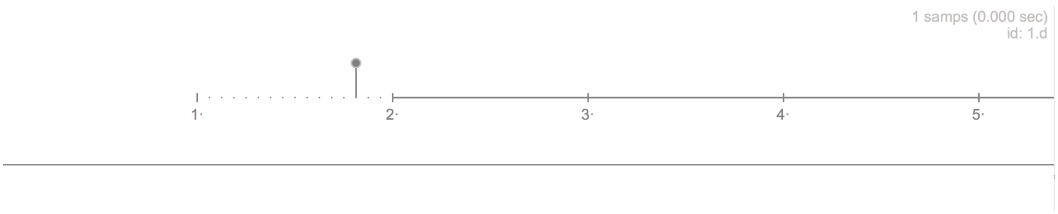


Figure 2.2: Zoom in at the beginning of the dada.music *line, in a 4-bit scenario. The continuous grey line is actually subdivided into regions with different lengths (in samples), which in turns are subdivided in points, each representing a unique buffer configuration.*

imatively $1.84 \cdot 10^{212399}$ years, a number that is not even comparable to the age of the universe. Lowering sample rate and bit depth changes the order of magnitude dramatically, but, of course, not the nature of the combinatorial problem. A 4-bit, 1kHz run through all short impulses (up to a tenth of a second) would still require slightly less than 10^{112} years. Nonetheless, *Music (or The Discotheque of Babel)* is a mental experiment, and it is fascinating to grasp the possibility in a click to discover some exact recording of a given Beethoven symphony.

Of course one could manually write each sample of the buffer, as one writes notes on paper, and such operation is actually not more complex than zooming in properly and selecting some specific sample on the line.

2.1.5 Personal references

Working with composite datasets requires acknowledgment of a tension between unity and heterogeneity. I recall how my first encounter with Berio's *Sinfonia* and its 'unity out of heterogeneity' had a profound impact on my adolescence. I was struck by the very idea that invention could find his path through the usage of existing material; I was fascinated by the idea that composition was imposing a form on objects that already had one. I started working with collages, especially in electronics music, where the technique was easier to handle. I felt that, in a way, citationism was a manner to assign some kind of shared 'meaning' to music content: after all, we probably all share similar references when we hear the first notes of Beethoven's fifth Symphony, or the incipit of Schoenberg's *Pierrot Lunaire*, or Wagner's *Tristan* chord. A quotation was something as close to a word as I could find in music: it was a way to provoke connections, to create levels of interpretation.

My interests have partially shifted since then, but my admiration for *Sinfonia* remains intact, along with the enthusiasm for using existing music to create new music. A certain number of other works have opened the way to profound reflections.

The sequence of automatic orchestrations of a mantra in Jonathan Harvey's *Speakings* had me thinking on how a prototypal way of composing is simply ordering elements in a certain collection. Composing, in this sense, means finding interesting path across patterns—with some assonances to what beautiful scientific theories often are. Lists are often themselves 'poetic', a concept which resonated with my love for many other important 'catalogues'—an incomplete list of which includes:

Aby Warburg's *Bilderatlas Mnemosyne*, Buckminster Fuller's *Dymaxion Chronofile*, Luis Borges's *Book of Imaginary Beings*, Luigi Serafini's *Codex Seraphinianus*, and John Koenig's *Dictionary of Obscure Sorrows*[3].

Christian Marclay also had a profound influence on my thoughts; partly for his turntablist and plunderphonic experiments (such as his *Record without a cover*), but mostly for the simplicity of some of his latest digital works, such as the 24-hours video collage *The Clock*, combining the interest for pure editing of Bruce Conner's *A movie* with the monumentality of Machaut's *Messe de Nostre Dame*. I realized, through Marclay's works, that masterpieces can be based on concepts "as straightfoward as a recipe." [Zalewski, 2012]

Marclay also embodies the fact that a corpus-based approach is not a prerogative of a single art; on the contrary, digital techniques ease the communication between different languages. Image mosaics are almost ubiquitous nowadays, and have lost some of their initial fascination; I am however deeply intrigued, for instance, by textual experiments stemming from the *GTR Language Workbench*[4], by the essential poetry of Ken Murphy's *A History of the Sky*, by the refined arrangements in Lev Manovich's multimedia works (such as *Phototrails* or *The exceptional and everyday*).

2.1.6 Comparison with other compositional researches

Virtually all contemporary composers use quotations in their works, prominent figures in this respect being the aforementioned Charles Ives, Luciano Berio, Bernd Alois Zimmermann and Clarence Barlow. A few composers also have some degree of interest in using organized datasets for specific purposes (such as Mauro Lanza's combinatorial dataset of multiphonics, or Yan Maresz's orchestration tools). Among the latter, some composers, mostly of my own generation, share the more specific interest of tackling composition, and especially computer-aided composition, from a corpus-based point of view. This is a relatively new phenomenon, recently investigated by Malt [2017].

As I have already mentioned, there is a certain overlap of the ideas I have exposed with Johannes Kreidler's work; moreover, some of the compositional ideas I have outlined might be called 'conceptual' from Kreidler's perspective.

Aaron Einbond is interested in combining the paradigm of corpus-based concatenative synthesis with symbolic control [Einbond et al., 2014] and with computer-assisted improvisation techniques [Einbond et al., 2016], with a specific focus on using timbral descriptors to transcribe audio recordings for live instrumental ensemble and electronics [Einbond, 2016]. Christopher Trapani, who has often collaborated with Einbond, also shares similar interests in concatenative synthesis control; his artistic work is however more focused on sonorities and stylistic gestures that carry some specific connotation, working on associations with periods or genres, without borrowing literally from the sources [Trapani, 2017]. Marco Antonio Suárez-Cifuentes was probably one of the first composers to take advantage of real-

[3] http://www.dictionaryofobscuresorrows.com/

[4] https://web.njit.edu/~newrev/3.0/workbench/Workbench.html

time musaicing techniques in order to reconstruct vocal and instrumental sounds via MFCC nearest neighbours. Ben Hackbart, also working on concatenative synthesis and musaicing, has incorporated in his software *AudioGuide* the ability to account for sounds' morphology (time-varying descriptors) and the definition of 'similarity contexts'; most importantly, in *AudioGuide* the samples can be selected simultaneously, so that the nearest neighbour approach is replaced by a search for the 'best' combination of samples matching a given target [Hackbarth et al., 2011]. Hackbart's approach has much in common with the developments of the Orchids project [Antoine and Miranda, 2015], where target samples can be orchestrated with combinations of elementary instrumental or user-defined samples. Frederic Le Bel applies classification techniques to large collections of audio files and computes pairwise distances with respect to a set of reference descriptors, in order to draw and explore a 'map' of the sound collection.

As far as symbolic corpus-based composition is concerned, to the best of my knowledge, I believe that very few composer consistently use the technique in their scores, a prominent example being Bruno Ruviaro's research. Ruviaro's engagement in the practice is inspiring, and portrays him as a sort of 'plungerphonic notationist': he takes care of citing source material in his scores, and mostly releases his works under Creative Commons licences.

2.2 Describing and organizing content

2.2.1 Music descriptors

The characterization of musical data by extracting meaningful information and organizing it according to the resulting description is far from being a recent subject. Traditional labeling of musical pieces as 'studies', 'sonatas' and 'symphonies' was (and is), of course, a first crude way to assign (high-level) tags in order to provide a better way to organize—and hence, to a certain extent, understand—scores. Indubitably, especially in the signal domain, the problem has become more compelling in the last few decades, when the advent of digital media has presented the possibility to convert archives and labels into digital form, allowing a finer and more thorough description. Since then, description and retrieval tasks have become crucial in a wide range of application such as audio editing, music composition, sound effects selection, live mixing, music classification, automatic playlist generation, or sound signaturing for copyright protection [Herrera et al., 1999]. Therefore, unsurprisingly, they have strongly influenced the birth of a relatively new field of research: music information retrieval (MIR) [Downie, 2003].

The problem is two-fold: (a) finding a way to automatically or semi-automatically extract perceptually relevant features from music; (b) organizing the extracted features in a structure which should be, in principle, as versatile and convenient as possible.

As far as the first point is concerned, describing music content by means of features involves procedures and techniques that have been developed in different research areas. Feature extraction can be managed by different non-exclusive means that can range from manual labelling to standard digital signal processing techniques [Wold et al., 1996], computational auditory scene analysis techniques [Wyse and Smoliar, 1995] and statistical techniques [Foote, 1997].

A wide variety of tools have been developed during the last few decades. Many of them are currently available as open-source or freeware projects, including C++ libraries such as *Essentia* [Bogdanov et al., 2013]; Python modules such as *LibROSA* [McFee et al., 2015], *Yaafe* [Mathieu et al., 2010], *aubio* [Brossier, 2006], and *Madmom* [Böck et al., 2016]; Max libraries such as the *Zsa.descriptors* [Malt and Jourdan, 2008], the *AHarker Externals* [Harker, 2014] and *MuBu/PiPo* [Schnell et al., 2009]; Java applications such as *jMIR* [McKay, 2010]; command-line tools such as *ircamdescriptors* [Peeters, 2004]; standalone frameworks such as *Marsyas* [Tzanetakis and Cook, 2000]; Vamp plugins for SonicVisualizer [Cannam et al., 2010]. Many of these projects have multiple bindings; for instance there is a Max *ircamdescriptors~* external, the *Essentia* and *Madmom* modules ship with default command-line applications, and *Marsyas* and *Essentia* are also available as Vamp Plugins. A comparative overview of the majority of the available tools is presented in Page et al. [2012]. A survey on MIR systems can be found in Typke et al. [2005] and Lew et al. [2006].

One can roughly organize sound-based descriptors[5] into three categories:

- *low-level* features, describing basic yet powerful computational properties of the sound (e.g., centroid, flux, zero-crossing rate). Low-level features do not carry musical information *sensu stricto*, but they still bear relevant primary information on the sound characteristics. Low level features can be as simple as statistical properties (such as the skewness) or can require a larger amount of computations (such as the Mel-frequency cepstral coefficients or MFCCs). In any case, there is large consensus on their definition and implementation, which makes low-level features the most reliable descriptors, as well as the less implementation-dependent;

- *mid-level* features, containing information 'cooked' into a musically relevant form (e.g., fundamental frequency estimation, partial tracking, chroma vector extraction, rhythmic density). Mid-level features carry meaningful information for musically trained individuals. Several algorithms have been proposed to estimate them. In most cases there is some consensus on what these features should represent, however there is usually no agreed 'best' way to implement them; as a consequence, mid-level features are often considered to be implementation-dependent. For some of these features, there is little consensus on what they should describe or represent, which makes their very definition somehow problematic (e.g., the 'danceability' estimator [Streich and Herrera, 2005] implemented in *Essentia*);

- *high-level* features, describing the audio in a chiefly symbolical way (e.g., chord recognition, score transcription, melody extraction, rhythm transcription). High-level features are essentially score-based representations of sound, although one may also consider textual representations (e.g., lyrics) as high-level descriptors. Moreover, to a large extent, processes such as mood or emotion detection provide high-level descriptors of sound.

This categorization is partly based on [McKay and Fujinaga, 2006] and [Kitahara, 2010], but it must be noted that there is no general agreement on the subdivision, especially between mid- and high-level features. For instance, information such as genre or musical form, is often considered as high-level information, although not symbolical. The same is true for other type of metadata, such as rehearsal marks, comments or formal indicators.

Descriptors can also be divided, in an orthogonal way, into a set of *global* descriptors, whose meaning concerns the signal as a whole (e.g., attack duration, tonality recognition), and *instantaneous* descriptors, computed for each time frame (e.g., centroid, or chord detection) [Peeters, 2004].

As for the organization of extracted features, some very popular file formats provide users with the possibility to include metadata and descriptions as part of

[5]Information that is not related to the audio signal (such as author, year or other similar metadata) is not included in the categorization.

the file itself, such as MP3 ID3 and ID3v2 tags containing metadata and lyrics; other important frameworks have been proposed, with the intent to provide tools for a better description, editing, tagging and searching of audio content [Herrera et al., 1999]. Visual programming languages, such as Max [Puckette, 2002] or PureData [Puckette et al., 1996], also provide some form of organization, paired with user interface capabilities; as an example, the aforementioned *MuBu* library features containers tailored to represent and query both specific and generic analysis data, in matrix form.

Compared to the relative richness of MIR implementations in the audio domain, the panorama of user-friendly tools for symbolic score analysis is rather scarce. In addition, the wide majority of the symbolic-based tools rely entirely on MIDI files, which are not hierarchical representations of scores and hence might entail quantization issues. In the class of symbolic tools, *jSymbolic* is probably one of the most complete ones [McKay and Fujinaga, 2006], computing a wide set of low- and mid-level features ranging from statistical analysis of pitches to harmonic and rhythmic analysis.

One part of my current research is focused in transferring descriptor-related concepts and techniques to the symbolic domain. Section 2.8 describes some of the results.

2.2.2 Micromontage, granular synthesis, concatenation, musaicing

During the last forty years, and especially with the advent of the digital era, a number of different but related techniques to work on sequences of varyingly small samples have been developed.

Micromontage is probably the earliest. As its name suggests, it is a convenient category for sound collages whose constituents only last for up to a tenths of a second ('microsounds') [Roads, 2004; Vaggione, 1996]. The idea is that when microsounds line up in rapid succession, they induce the illusion of tone continuity that we call 'pitch'. Sometimes, by extension, the term 'micromontage' is used even if the constituents are slighly longer than actual microsounds (e.g., up a few tenths of a second), and hence they retain their own pitch quality.

Among the techniques that have been devised to automate some parts of this montage, granular synthesis is probably the most influential [Roads, 1978]. The microsounds, now called 'grains', are programmatically extracted from the original sound files, layered on top of each other, and played at different speeds and volumes. The parameters for the extraction of grains (region, jitter...) as well as for their concatenation (grain size, density...) can be usually controlled by the user. No size limit is imposed to each grain, nor on the distance between grains: when grains are longer and when their temporal interval is in the order of the tenths of a second or above, the pitch of the original sound is preserved, and the boundaries between 'synthesis' and 'sampling' become blurred. As a personal note, I prefer to speak of 'granular sampling' whenever the morphology of the sampled sounds is still well hearable in the final result.

Corpus-based concatenative synthesis [Schwarz, 2007] directly derives from granular synthesis, and provides mechanisms for sequencing of 'grains' according to their proximity in some descriptor space. It is based on sound analysis, and differs from micromontage in that the descriptor space is formalized and programmatically explorable. Grains are usually extracted from a corpus of segmented and descriptor-analyzed sounds. Exactly as in the case of granular synthesis, one might argue that whenever the morphology of the sampled sounds is still present in the result this technique should be more aptly named 'concatenative sampling'.

Among the existing tools dealing with corpus-based concatenative synthesis or sampling, *CataRT* [Schwarz et al., 2006] is probably the most widely used. Taking advantage of the features in the FTM [Schnell et al., 2005] and (more recently) *MuBu* [Schnell et al., 2009] libraries, it provides tools for sound segmentation and analysis, as well as for the exploration of the generated corpus via an interactive two-dimensional display, both inside the Max environment and as a standalone application.

When the choice of grains to be concatenated depends on constraint solving techniques, one might also speak of 'musaicing' [Zils and Pachet, 2001] (or 'music mosaicing'): similarly to what happen with mosaics, a large sequence of microsounds is created, such that the sequence as a whole satisfies various high-level properties. De facto, the term 'musaicing' is also used to identify concatenative synthesis techniques based on similarity measures with respect to a target sound, which is reconstructed replacing each original 'grain' of sound with a similar one [Malt, 2017].

Since the boundaries between different terminologies are blurred, I will often use the generic term 'concatenation' to refer to any sequence of musical grains selected according to their features.

2.3 Chord-based concatenations

Sections 2.3.1, 2.3.2 and 2.3.3 have been previously published, in a slightly different form, in the article Ghisi, D. and Bergomi, M. (2016). Concatenative synthesis via chord-based segmentation for An Experiment with Time. In Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.*

2.3.1 *An Experiment with Time*

An Experiment with Time is a 46-minutes looping audio and visual installation, inspired by John W. Dunne's essay [Dunne, 1927]. As a second step of the project, part of the electroacoustic tape has been orchestrated in order to obtain a (non-looping) live version of the work, for amplified ensemble, live electronics and video (*An Experiment with Time (reloaded)*, 48 min, 2016). The installation version of the project was premiered in Paris, festival *Manifeste* 2015; the live version was premiered in Milan, festival *Rondò* 2016, by Divertimento Ensemble.

John William Dunne, aeronautics engineer, soldier, philosopher, and fly-fishing lover, wrote *An Experiment with Time* in 1927, discussing an experiment he performed on his own dreams. While comparing the dream images he duly annotated with the occurrences in his daily life, Dunne discovered that oneiric images were connected with events happening both in the past and in the future: premonitions and memories, in roughly the same proportion. This lead him to a develop a theory of time he later called 'serialism'. The direct consequence of his experiment was that linearity of time is a collateral event of consciousness: while dreaming, all moments happen at once.

Taking inspiration from these ideas, and bringing them even further, the video in *An Experiment with Time* shows the writing of an animated diary, during a full year (from January to December). The writer performs Dunne's experiment during the first months (winter); evaluates the results and looks for confirmations from others (spring); realizes that most of his oneiric images are to be found neither is his past nor in his future (summer); which brings him to imagine the existence of thinner imperceptible times and frame rates, and to build a 'supersampling-antialiasing' machine to dilate time (autumn). As a counterpoint, two alter-egos write similar diaries on the side screens, living in temporal cycles with different granularities: a day (left screen) and a life (right screen).

The starting point for the musical writing is a straightforward association between months and major chords, so that the whole year loop is handled like a sequence of perfect cadences in the tempered cycle of fifths (January being B major, February being E major, and so on, till December being F$\#$ major, and hence looping). Although the internal handling of the musical material becomes more complex (different chord types are explored and a few secondary dominants are used occasionally to underline specific passages), everything in the piece is conceived with respect to this simple sequence, which thus represents the skeleton of the whole musical loop.

I have put a certain effort in developing a system relying on a musically notated score driving a large set of audio files (picked from a wide database spanning the history of Western music), segmented and tagged by chord. The notated score is actually a meta-score where each note stands for an abstract chord; the score can be then rendered via concatenative synthesis of chord-labeled samples, as described in the following sections.

Figure 2.3: A frame from An Experiment with Time *(three screens, displayed horizontally).*

Incidentally, the technical and artistic stakes in *An Experiment with Time* extend well beyond the fundamental corpus-based framework, including: the quest for a grey area between writing and reading; synchronized sonification of events as defining feature of the passage of time (as if an 'instant' were the byproduct of a visual and audio synchronicity), inspired both by 'Mickey Mousing' techniques and by some of Yannis Kyriakides's works (such as *Mnemonist S*); the usage of cellular automata to reinforce the impression of a discrete time; stop-motion animation and aliasing effects; polyrhythmic structures; Eadweard Muybridge's chronophotography and its relationship to stop motion techniques; generative, animated dictionaries (Fig. 2.4); Risset-like 'eternal' accelerando or rallentando rhythms, following Stowell [2011], both for audio and for scores (Fig. 2.5); morphing of time into space (Fig. 2.6) and viceversa; study of the action of algebraic groups yielding quasi-canonic behaviors; and more. These topics are not necessarily connected with corpus-based composition, and will not be discussed here (although figures are provided to exemplify some of them).

Related works

- *An Experiment with Time*, audio-visual installation, 46 min (loop), 2015.
  - Project website and full credits: `http://www.anexperimentwithtime.com`
  - Trailer: `https://www.youtube.com/watch?v=a81idHK8-CA`
- *An Experiment with Time (reloaded)*, for amplified ensemble, video and electronics, 48 min, 2016 (Ricordi).

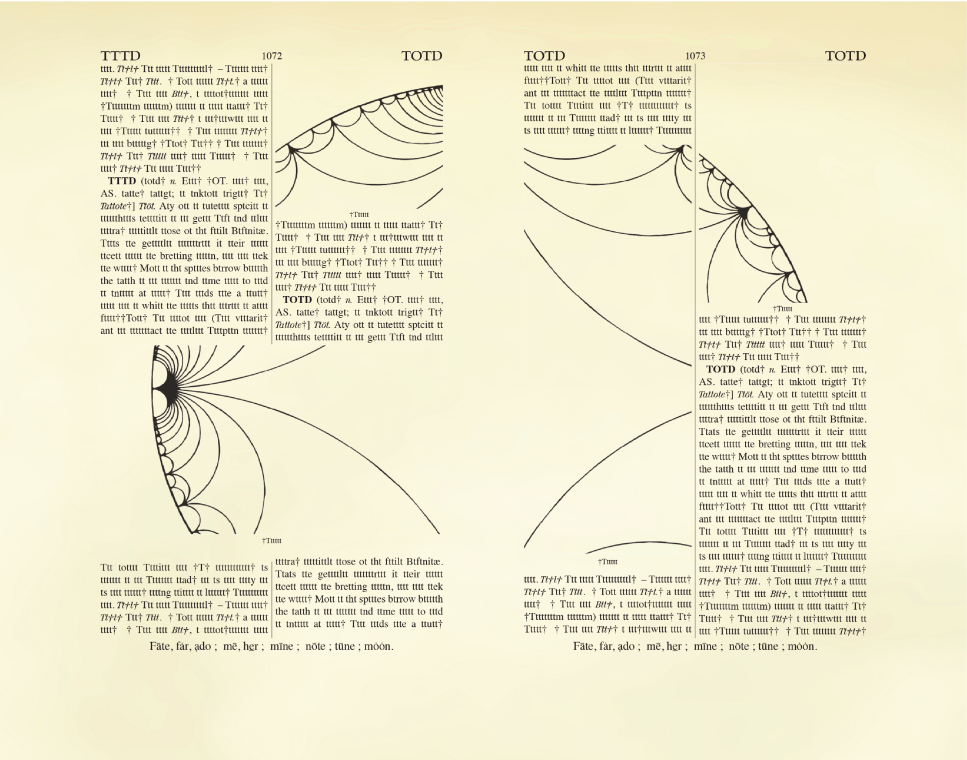


Figure 2.4: Two pages of one of the generative dictionaries used for An Experiment with Time

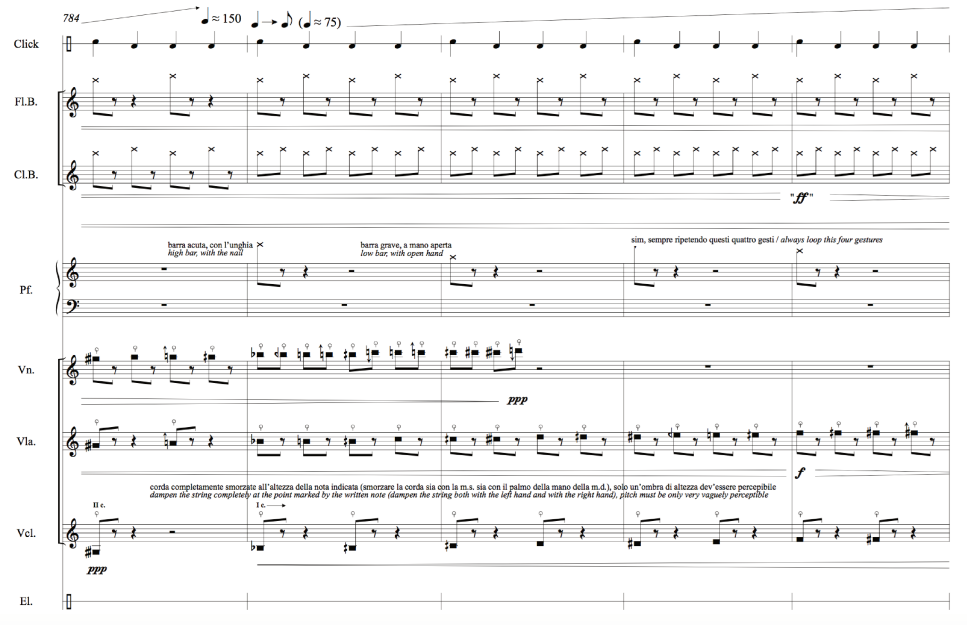


Figure 2.5: A portion of a Risset-like 'eternal' accelerando used in An Experiment with Time (reloaded) *(b. 784)*

2.3.2 Database and segmentation

The database I selected for *An Experiment with Time* was composed by about 3000 tracks of classic, contemporary, rock, pop and jazz music, sampled from the whole history of Western music. This corpus has been chosen so that time is a parameter of the corpus itself. The relationship between the historical time and the musical time creates interesting diffraction patterns. As an example, during June, a radio broadcasts a sort of 'history of C major'[6], composed by C major samples ordered with respect to their composition year. Similar processes, relating to time as well as to other descriptors, are used diffusely throughout the work.

Excerpts

From *An Experiment with Time*:

- G major and G minor concatenations while supersampling the 'blue' letter (19'38"):
  data.danieleghisi.com/phd/vid/AEWT_supersampling.mp4
- A brief 'history of C major' (26'58"):
  data.danieleghisi.com/phd/snd/AEWT_history_of_C.aif|mp3
- Leaves and samples (35'38"):
  data.danieleghisi.com/phd/vid/AEWT_unperceived.mp4

With the help of Mattia Bergomi and NaMi Lab, the harmonic transcription of each song has been computed using the algorithm presented by Mauch [2010]. This effective algorithm allows to set a specific dictionary, in order to select a certain number of chord classes. Each element of the dictionary is indicated by specifying both the bass and chord relative pitch classes; as a consequence, it is possible, for instance, to associate to a major chord only its root form or to identify it with its inversions. In the case of *An Experiment with Time* the chord dictionary was defined in order to detect the classes listed in table 2.1.

This particular choice of chord classes has been motivated by the desire to include the four standard tonal trichords (major, minor, augmented and diminished) and a few of their basic variants.

chord class	*pitch class structure*
N.C.	no chord
maj	(0, 4, 7)
maj/3	(0, 3, 8)
maj/5	(0, 5, 9)
aug	(0, 4, 8)
min	(0, 3, 7)
dim	(0, 3, 6)
6	(0, 4, 7, 9)
7	(0, 4, 7, 10)

Table 2.1: Chord classes in An Experiment with Time.

[6] An officer named Major C. is also a supporting character in the video, hence the word play.

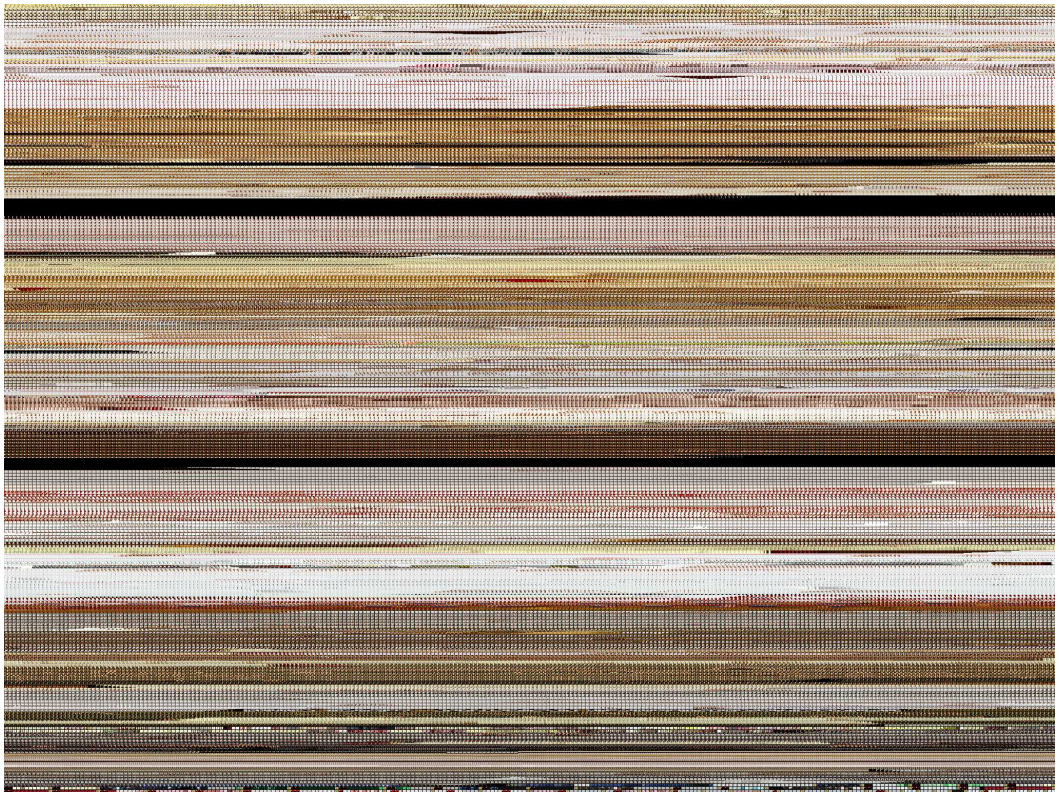


Figure 2.6: All the 69000 frames of the An Experiment with Time *loop (middle screen), resized to tiny proportions and disposed in space (left to right, top to bottom), as they appear on the screen near the end of the loop.*

Thereafter, each audio track has been associated to a file specifying both the onset of the chord in seconds and its class as follows

```
{"chords":[
{"position":0,
"chordname":"N.C.",
"avbpm":139},
{"position":230574.149659,
"chordname":"B\/F#",
"avbpm":139},...
]}
```

Finally, each audio file has been segmented into harmonic grains according to these features. This procedure allowed us to create a new database organized in folders named with a specific pair $(root, class)$ and containing harmonic grains labelled as $chordname\_n\_path\_title$. The file path has been preserved in order to facilitate queries involving the names of the folders containing specific files. The natural number n represents the position of the chord with respect to the entire harmonic sequence of the audio track.

2.3.3 Compositional framework

2.3.3.1 Chord-wise module

The choice of the segmentation was motivated by the desire to compose with chords instead of notes.

I have set up a proportionally notated meta-score (see fig. 2.7) where each note stands for the fundamental of a chord, whose type is specified in the first slot (the *bach* library organizes note metadata inside generic containers named 'slots'). This representation is handier than having to specify all the voices of the chord, as it allows to separately control the two orthogonal and pertinent parameters: fundamental and chord type.

For each note, additional slots carry the information needed for the concatenative rendering: the duration of each unit (or grain), the inter-unit distance (the temporal distance between the onsets of two consecutive grains), the descriptor (if any) according to which the units should be sorted, and the sorting direction. Another slot narrows the choice of units so that a certain descriptor lies within a given range; furthermore, additional slots provide amplitude envelopes (both for each unit and for the whole sequence of units). Finally, a slot is dedicated to filtering the database according to words or parts of words appearing in the file name or path; this is an extremely quick and effective way (via the Unix 'find' command) to narrow the search to a tag or a combination of tags (e.g., 'Mozart', or 'Symphony', ...). All slots and their usages are shown in Table 2.2.

The score is by default rendered in off-line mode: notes are processed one by one. For each note, three rendering stages are necessary: the random retrieval of the sound files satisfying the input criteria (chord, descriptor range, tags); the sorting of such sound files depending on the value of the specific descriptor (if any); the montage of random portions (units) of the sound files into a single sequence, associated with the note. The process can be aborted at any time. Once the score is rendered, the corresponding waveform appears below the score, and stays graphically synchronized with it. The score can be then played, saved, or rendered anew.

This concatenative process can also be accomplished in a real-time fashion, al-

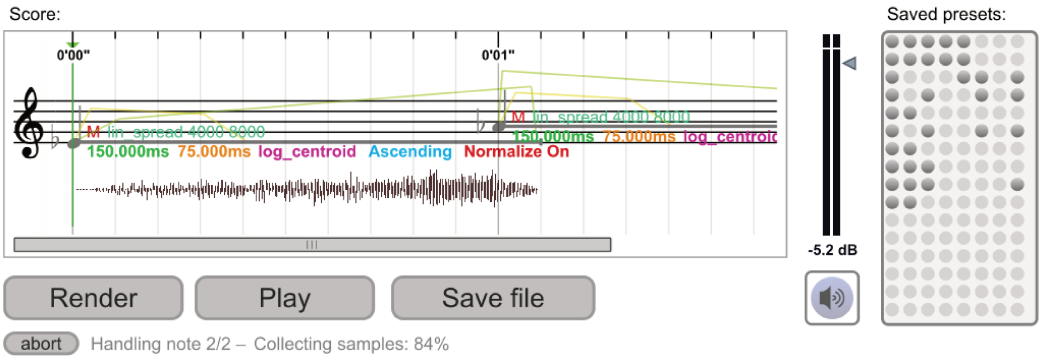


Figure 2.7: Interface of the off-line composition module.

slot number	*slot content*
1	chord class
2	grain duration
3	grain distance
4	grain amplitude envelope
5	global note amplitude envelope
6	filter by descriptor ranges
7	sort by descriptor
8	sorting direction
9	grain normalization

Table 2.2: Slots setup for the offline composition module.

though the retrieval of sound files can take a non-negligible time when switching chords, depending on the number of segmented files per chord. During the writing of *An Experiment with Time*, I mostly used real-time capabilities as an exploratory tool, before turning to the off-line module for actual symbolic writing.

2.3.3.2 Chord 2-grams

The module described above randomly concatenates units of sound files for each note. The inner coherence for the sequence somehow relies on the fact that all units share the same harmonic content, and on the fact that units are sorted according to a specific descriptor. However, no particular continuity is guaranteed when notes

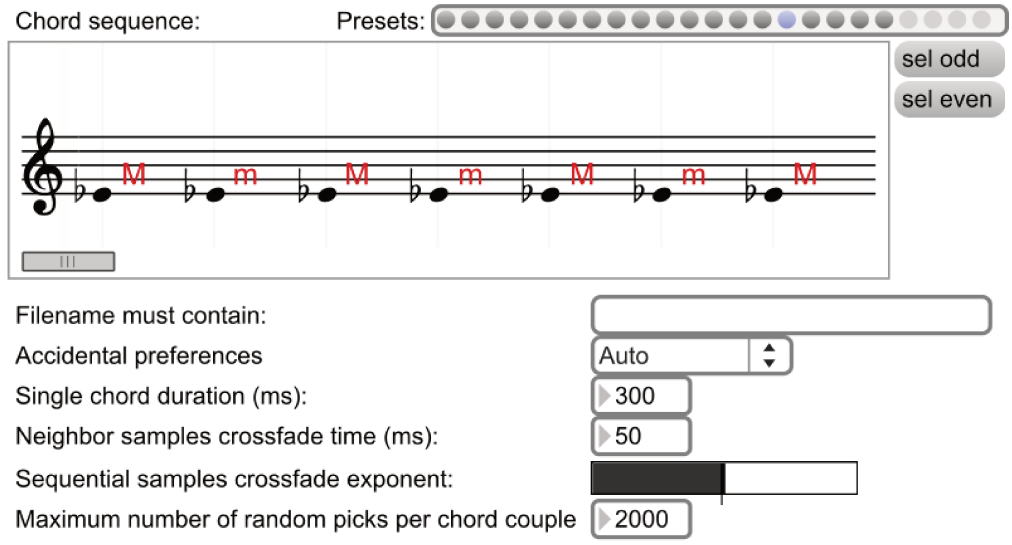


Figure 2.8: A tremolo-like alternation between major and minor chords. During rendering, only transitions between E flat major and minor chords (and vice versa) are retained from the database; for each couple of notes, a transition is chosen and then the sequence of transitions is created via crossfades.

change.

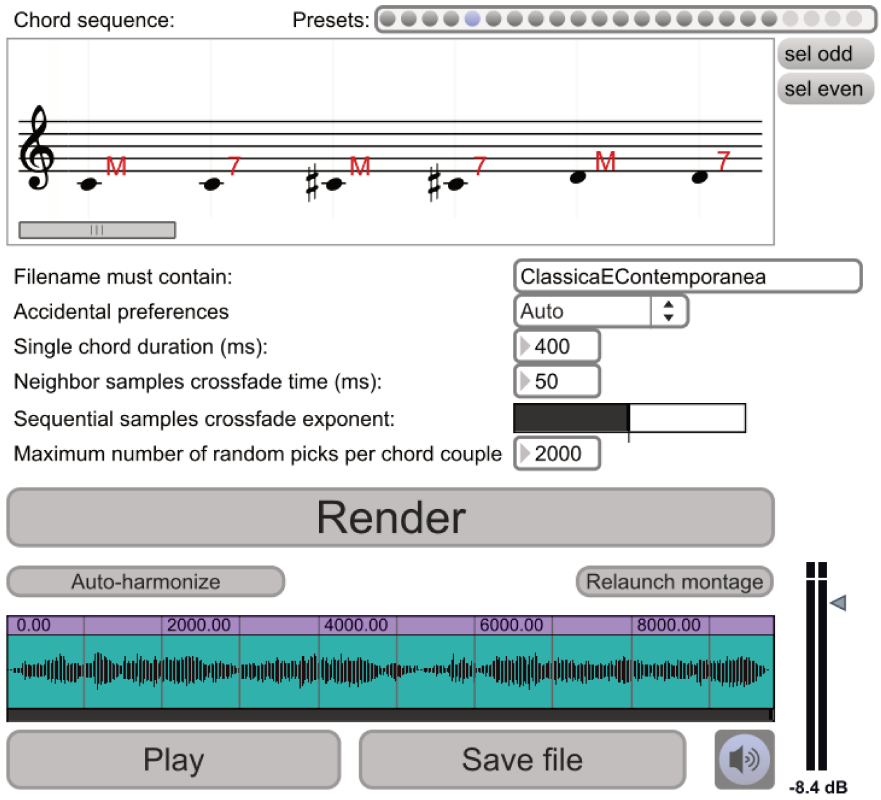


Figure 2.9: Interface of the chord-sequence module.

A specific module has been developed to allow chord sequences to be rendered more smoothly (see fig. 2.9): the user defines a chord sequence in a similar manner as of section 2.3.3.1 (notes represent chord fundamentals and carry meta-information). In this case, however, couples of contiguous chords (2-grams) are rendered at once. For each couple of chords, the algorithm searches for a sound file where such chords show up exactly in the correct order, without discontinuity; this is made possible by the fact that the segmentation process retains in the output name a chord index. All the overlapping couples of chords are then crossfaded in order to create the complete sequence. In this case, a note does not represent a sequence of units, rather a single unit, which on the other hand is guaranteed to join seamlessly with the previous and following one.

A set of basic synthesis parameters can be customized; an auto-harmonize button is set in place to automatically detect chord types depending on the notes of the sequence (this is especially useful when harmonizing scales). Moreover, I have often used this chord-sequence module to programmatically generate and render chord sequences undergoing extremely simple rules, such as a tremolo-like alternations between major and minor chords (fig. 2.8), or sequences of fast, continuous deceptive cadences (fig. 2.9). The latter, towards the end of the work, flows into a

continuous Risset-like glissando composed by a micromontage of small fragments of vocal glissandi (see section 2.4.1).

Excerpts

- A tremolo-like chord alternation (fig. 2.8), from *An Experiment with Time*: `data.danieleghisi.com/phd/snd/AEWT_Eb_tremolo.aif|mp3`
- A sequence of fast deceptive cadences (fig. 2.9), from *An Experiment with Time*: `data.danieleghisi.com/phd/snd/AEWT_inganni.aif|mp3`

2.3.4 Usage in *I mille fuochi dell'universo*

A large section of *I mille fuochi dell'universo*, a collective composition that will be presented in section 4.4, was set to be a kaleidoscopic 8-minutes long descent; for such section I developed a modified version of the previously presented modules. The main difference with the original patch is that the pitch of every note will force a retuning factor for the corresponding samples, so that the overall result is a microtonal and seamless descent. Different versions of this descent were mixed in the final result (Figure 2.10 shows a portion of the descent in a *bach.roll* object).

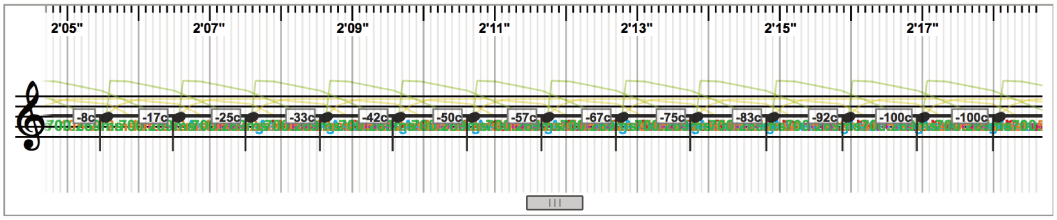


Figure 2.10: Portion of the score for one layer of the seamless descent in I mille fuochi dell'universo.

Excerpts

- A portion of one layer of the descent in *I mille fuochi dell'universo*: `data.danieleghisi.com/phd/snd/IMFDU_descent.aif|mp3`

2.4 A poetic of concatenation

When I first started writing electronic music, I had the impression that montage could be the easiest access to it. My first electroacoustic piece was essentially a montage of heterogeneous recordings. Once I delved into the intricacies of more advanced techniques, I started to consider pure montage as a naïve approach; and yet I could not completely get rid of the fascination of sampling, mixing and concatenating complex sounds. To sidestep the issue, I started using montage on extremely small scales. This didn't have much in common with a 'microsound'-oriented approach (see section 2.2.2): I was not trying to compose pitch and timbre starting from extremely small windows of sound; my aim was rather to gradually reduce the length of the source region, in order to achieve a coherent and yet either zapping-like or loop-like effect. I started to refine each fragment in the micromontage via digital signal processing (DSP); I developed a certain number of notation tools accordingly (the *bach* library was born around that period); I started using organized datasets and musical corpora. I was getting little by little at ease with the framework I was meanwhile constructing: imposing a form onto musical elements that already had one was becoming an interactive and somehow rewarding process. Sometimes the original content was a mere pretext, a loose starting point which I knew would undergo extensive processing before being used in the actual composition.

If I had to trace a path of my musical interest from that point onwards, it would affect three separate factors: since then, the grain size gradually increased (from a scale of tenths of seconds to a scale of seconds), the amount of DSP treatment applied on each grain gradually decreased, the size of the datasets markedly increased (from a handful of sound files to several hundred gigabytes of compressed files). These transitions are noticeable in my latest works. Somehow, the focus has shifted from the control over the grain treatments to the control over the grain sequence.

Today, I tend to think of corpus-based montage as a pure act of composition, in a similar way in which a certain number of composers are interested in additive synthesis as pure act of writing sounds. The duality between the two scenarios is to me quite striking: on the synthesis side, the focus is on absolute control of the parameters; on the corpus-based montage side, the control is deliberately lost, in favour of a degree of unpredictability and surprise. Changing a pitch, a dynamic marking, or an envelope, is a trivial operation while composing for an instrument or for a synthesizer, but it is a non-trivial challenge when dealing with corpus-based montage. One could of course perform corresponding DSP operations with some pitch shifter or gain modifier, but one also has the intriguing possibility of choosing some other sample altogether, matching the new pitch, dynamics or envelope to a greater degree. Modifying a certain query in the dataset is also a way to perform a (highly non-linear and rather unpredictable) pitch shift. And this way, although more complicate, has a number of advantages, first and foremost the aforementioned degree of surprise (see section 2.1.3): the effort of discovering, listening, analyzing, makes of every fragment a truly selected and more abstract object—a 'note', indeed.

2.4.1 Concatenations as trajectories

In this sense, *An Experiment with Time* has been a crucial project in my musical path: it was the first work where I exponentially enlarged the size of the datasets and, in a parallel way, markedly decreased the amount of DSP treatment on musical grains. Nonetheless concatenations generated for *An Experiment with Time* were rarely used as 'foreground', macroscopic objects, and were often employed rather as compound textures, as 'material' to compose with—be it just via an additional layer of amplitude modulation and montage.

But *An Experiment with Time* also featured two notable exceptions to this pattern, two cases where the concatenation was not only than a technique, but also the foreground object itself: the aforementioned 'history of C major' and the vocal glissando appearing at the end of the work. This latter is probably the sharpest sample: it is a concatenation of vocal glissandi, so that the resulting musical trajectory is a continuous glissando throughout four octaves.

Excerpts

- A glissandi-glissando, from *An Experiment with Time*:
  `data.danieleghisi.com/phd/snd/AEWT_glissando.aif|mp3`
- A glissandi-glissando, in Risset-like fashion, from *An Experiment with Time*:
  `data.danieleghisi.com/phd/snd/AEWT_glissando_risset.aif|mp3`

The concatenation, as well as the glissando detection, was carried out mostly by hand—the attempts to set in place an (albeit rough) detection mechanism for vocal glissandi having proven unsuccessful. To gather the considerable amount of glissandi for the montage, I have both consulted a certain number of musician friends and used the Internet as a sort of 'human-based MIR system' [Downie and Cunningham, 2002].

After *An Experiment with Time*, I have been questioning for some time the meaning of concatenations: the ideas that regularly informed my perspective were the ones of belonging, of collective music, of questioning authorship, of 'writing on the skin of the world'. Whenever I imagined to bring this approach to the utmost consequences, I invariably came back to the most radical concept of pure montage.

What I liked in the composite glissando from *An Experiment with Time* was its simplicity, and at the same time its meaningfulness. It was, figuratively, the passing of the torch of all the singers who ever sang a glissando, even if just for a few seconds. It was a *hommage*, of course, but it was well beyond that, the overall shape being so significant; to me, it was music across music.

The key seemed to be the unity of the gesture—as the material composing it was complex enough. And I could try to model such simplicity as the varying of a single parameter in some space. A development of this approach has generated a number of graphical tools to handle geometrical representations of databases (see section 3.3.2); yet even with no visual tools, it is easy to interpret a similar concatenation as the positioning of different 'grains' on a given trajectory, moving with respect to a single, perceptually relevant parameter—in the glissando example: the pitch.

2.4.2 Music across music: *Electronic Studies*

After *An Experiment with Time*, I decided to continue exploring concatenations, trying to keep a single characteristic for a grain to be accounted for, and a single criterion to sort grains. The result is a crude and unfinished collection of short sound files, temporarily and disorderly collected under the title *Electronic Studies*; in addition to the aforementioned glissando, here are some other excerpts:

Excerpts

- A concatenation of hand clapping figures, sorted by decreasing tempo:
  `data.danieleghisi.com/phd/snd/Concat_handclaps.aif|mp3`
- A concatenation of 'money notes'[7], sorted by increasing pitch:
  `data.danieleghisi.com/phd/snd/Concat_moneynotes.aif|mp3`
- A concatenation of words 'kiss' sung by different singers, sorted by increasing length:
  `data.danieleghisi.com/phd/snd/Concat_kiss.aif|mp3`
- A concatenation of words 'tomorrow' sung by different singers, organized via composite criteria (some sections are sorted by increasing main pitch):
  `data.danieleghisi.com/phd/snd/Concat_tomorrow.aif|mp3`
- A concatenation of words 'know', spoken by different people, sorted randomly:
  `data.danieleghisi.com/phd/snd/Concat_know.aif|mp3`

There is a sort of emergence phenomenon taking place as we listen to these excerpts: we do recognize the unicity of each fragment in the sequence, however we also clearly hear that the combination of the fragments represents much more than a random juxtaposition of them: we hear a trajectory in a space. This has very much in common with Christian Marclay's video montages and Lev Manovich's visual interactive spaces.

This also brings to mind a certain number of considerations, which pertain to the way in which concatenations are perceived.

1. Grains usually share a common feature, a sort of recognizable 'aura' (not unrelated to the concept of 'aura' proposed by Lachenmann [2004]); this aura can be immediately spotted, and sometimes it is even harder to describe than to recognize (as in the case of 'money notes').

2. Each concatenation is a path on a map for a certain territory. The listener should have the feeling of following that path curve with a finger.

3. Transitions between grains convey much of the information: a large part of the compositional work lies in crafting them. Transitions reveal and hide at the same time. In most circumstances, my instinct was to make transition as seamless as possible, or to find significant quasi-causal connections on which to base them.

[7] A 'money note' is a slang term of the pop music industry, referring to high, emotionally dramatic notes frequently spotted at a song climax, and usually held for a long time with clear pitch and expressive vibrato.

4. The status of each one of these concatenations remains, to me, a *vexata quaestio*: on one side, I do consider some of them as true miniatures, as self-standing studies (most notably, the vocal glissando); others are well far from being sufficiently sharp, and cannot be considered autonomous studies (most notably, the concatenations of spoken words, which were indeed used as part of a larger project, as described in section 2.5.3). I am planning to organize some of these concatenations, as well as additional ones, in a future larger and more coherent container, a project fully based upon similar techniques.

5. In almost all of these concatenations there is a somehow humorous facet: an angle, a point of view, for which they are, by some means, *funny* or *ironic* [Kramer, 2016, p. 9]. I do not live this as a problem; on the contrary, I find amusement to be an important side of engagement to listening. However, the goal is to prevent the amusement from turning into joke or laughter, and essentially to let the deeper meaning emerge from it.

6. Another way to look at these concatenations is to see them as *games* or *challenges*: listeners are sometimes brought to ask themselves *how much longer* the corresponding process could last. Each of these montages is *going somewhere*, but the ending point is never declared nor shown, and it might be postponed indefinitely.

7. As a corollary of the previous point, the length of these concatenations is a highly delicate subject; it seems to me that in almost all cases they should be *as long as musically possible*—compatibly with the amount of raw material found and with the possibility to keep the process interesting. In the excerpts above, this idea is pursued in very different manners: in the case of the continuous glissando, temporality is essentially determined from the material itself (only glissandi with approximatively a certain slope were selected, and changing temporality would simply imply to change the target slope, or to stretch the result); in the case of 'money notes', the process is developed so to exploit all the grain found for the lower notes, and then gradually sieving the material as the pitch went up, in order to give a feeling of accelerando along the ascent; the case of the concatenations of sung words is still different—potentially, the concatenation of 'tomorrow' should be disproportionately longer than the two minutes it currently lasts (also see section 2.7).

2.5 Speech and corpus-based composition

2.5.1 The utopia of a bridge between speech and music

Having musical instruments talk is a shared dream among the community of composers, and not a recent one. In the 17th century, Jean-Baptiste Lully imitated the speech melodies and dramatic emphasis used by actors [Ranum, 2001], influencing a certain number of 19th and 20th century composers. Among these, Leoš Janáček deserves a special mention: not only did he integrate pitch contours as well as the inflections of Moravian dialect in his operas (such as *Jenůfa*), but since 1879 he carefully kept a collection of transcribed speech intonations [Procházková, 2006].

With the diffusion of recording technologies, and even more rapidly in the digital era, the relationship between music and speech has also been explored from a computational angle, especially in electronic music (Bossis [2005] provides a detailed overview on the subject). Nonetheless, the search for affinities between voice and acoustic instrument has never paused.

Among the numerous examples, some works had a certain influence on my research: Jonathan Harvey's voice orchestral transcriptions in *Speakings*, Fabio Cifariello Ciardi's studies on lip-synced music (such as *Tre piccoli studi sul potere* or *Voci vicine*), Peter Eötvös quartets (such as *Korrespondenz* and *Siren Cycle*), Steve Reich's *Different Trains* and *The Cave*, Peter Ablinger's talking piano (such in *A Letter from Schoenberg* or *Deus Cantando*), PerMagnus Lindborg's *TreeTorika*, Clarence Barlow's works on speech recordings (such as *Orchideae Ordinariae*). Each of these composers finds his own peculiar bridge between audio and spoken word (or between sound and meaning): Ablinger focuses on the threshold of intelligibility of the texts, so that listeners are encouraged to imagine phonemes for the sounds they hear; Harvey and Eötvös both take a much more concert-driven approach, analyzing voice inflections and turning them into chiefly musical gestures; Barlow is interested in sound synthesis through instrumentation (or *synthrumentation* [Barlow, 2011]); Reich employs repetition as a way to turn speech into chant, an effect later studied by Deutsch [1975]; finally in Cifariello Ciardi's case, the synchronization between the image and the sound seems to be a key aspect of the process, partially relying on the McGurk effect [McGurk and MacDonald, 1976].

These works all seem to share a common utopia. It is not a coincidence that so many postures have florished in the last few decades, when technology have allowed composers to take advantage of different tools to transcribe voices into music—including computer-aided composition tools such as OpenMusic (Cifariello Ciardi and PerMagnus Lindborg [Lindborg, 2008]) or *bach* (Eötvös [Sirens Cycle, 2017]). This is, to me, a clear example of how composer aesthetics influence and, at the same time, are influenced by technological advances, in a positive feedback loop.

It is striking to see how transversal the topic appears to be, clustering composers who would have otherwise very little in common. Even more intriguing, rock music has been sharing the very same utopia for a long time: as a notable case, Steve Vai's mimicking of his sister's voice in *So Happy* (in the album *Flex-Able*, 1984) is a

precursor for many of the previously cited works and for a multitude of speech-based instrumental videos that have populated Youtube in the last few years.

Links to examples (verified on 28th April, 2017)

- Peter Ablinger, *A Letter from Schoenberg*:
  https://youtu.be/BBsXovEWBGo
- Peter Ablinger, *Deus Cantando*:
  https://youtu.be/BzcBusxDThM
- Jonathan Harvey, *Speakings*:
  https://www.youtube.com/watch?v=6UJ2RXIEXa4
- Fabio Cifariello Ciardi, *Voci Vicine* (estratto):
  https://www.youtube.com/watch?v=5NeyMhtR3xk
- Steve Vai, *So Happy*:
  https://www.youtube.com/watch?v=Z7Asi870JpI
- Donald Trump Says "China" - Bass Cover by Iggy Jackson Cohen:
  https://www.youtube.com/watch?v=VHtKx2jk40U
- Donald Trump speech-to-guitar translation:
  https://www.youtube.com/watch?v=RD87K5xY-r0

2.5.2 MFCC-based musaicing in *Mon corps parle tout seul*

Influenced and fascinated by these examples, I have been exploring myself the bridge between words and sounds in *Mon corps parle tout seul*, from a corpus-based perspective, where the ensemble or orchestra in charge of imitating a voice is replaced by a collection of recordings of acoustic material.

Mon corps parle tout seul, a joint project with director Daniel Jeanneteau and writer Yoann Thommerel, is an installation where inside a dark room a giant mouth appears in front of the visitors as an oracle, formed by nebulized droplets of water retroilluminated by the light of a projector (see fig. 2.11). Visitors are invited to position themselves on a precise spot in front of the mouth, where the sound, via wave field synthesis techniques, is localized, as if coming precisely from the mouth itself. The acoustic illusion and the scattering of water droplets convey a sensation of 'presence' and create a both intimate and physical relationship with the giant mouth-shaped oracle.

The installation runs as a 12 minutes loop, corresponding to the duration of Thommerel's text read by actress Emmanuelle Lafon. During the 12 minutes, the sounds emitted from the mouth gradually shift from plain voice (beginning), to musically reconstructed voice, till they stop being recognized as phonemens and become concrete sounds, fragment of music, audio synthesis, explosions. The pivotal condition is the following axiom: even inside these mutations, the articulation of sound must always agree with the lip movements, by preserving synchronicities and by matching articulations and envelopes: the oracle is a composite object, and the visual and acoustic parts should never drift. In a nutshell: the mouth shifts from uttering a monologue to spitting out pure sounds: speech becomes sensation,

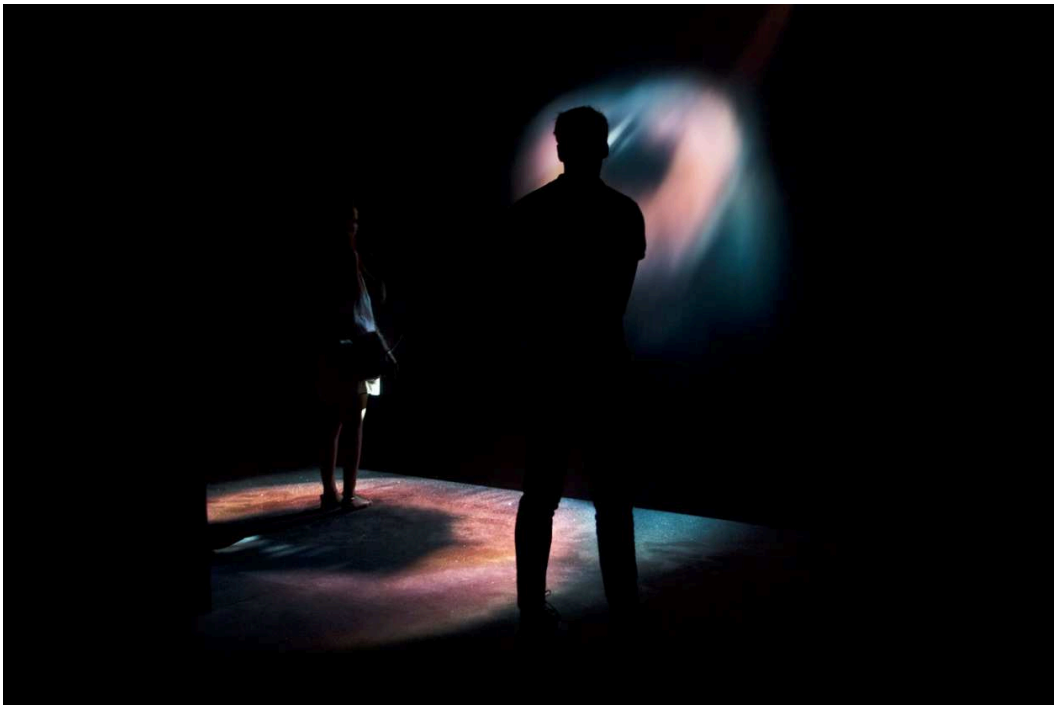


Figure 2.11: Two visitors at the Mon corps parle tout seul *exhibition, the leftmost visitor is standing on the 'sweet spot' for the wave field synthesis acoustic illusion.*

droplets, raw material, a sonorous incarnation. At very precise points during the loop, wind blasts are created from behind the mouth by a person with a simple plastic pad, invisible to visitors, following a score delivered via an ear monitor: the oracle becomes even more touchable, treading on the invisible boundary between installation and live performance.

The smoothness of the transition from voice to sounds is obtained also thanks to a 'voice reconstruction' stage, where mel-frequency cepstral coefficents (MFCCs) are computed for small overlapping windows of Emmanuelle Lafon's voice, and are then matched, via a k-nearest neighbours search, to the closest ones in a set of corpora, taking advantage of the *MuBu* library [Schnell et al., 2009]. Some examples of such matching follow.

Excerpts

- Original extract of Emmanuelle Lafon's voice:
  `data.danieleghisi.com/phd/snd/MCPTS_muzaiking_orig.aif|mp3`
- Mixed with MFCC reconstruction based on Luciano Berio's *Sinfonia*:
  `data.danieleghisi.com/phd/snd/MCPTS_muzaiking_berio_mix.aif|mp3`
- Mixed with MFCC reconstruction based on Edgar Varèse's *Poème électronique*:
  `data.danieleghisi.com/phd/snd/MCPTS_muzaiking_varese_mix.aif|mp3`
- Mixed with MFCC reconstruction based on Franz Schubert's *Nacht und Träume*:
  `data.danieleghisi.com/phd/snd/MCPTS_muzaiking_schubert_mix.aif|mp3`
- A mixing of the previous techniques as used in the installation:
  `data.danieleghisi.com/phd/snd/MCPTS_muzaiking_ex.aif|mp3`

In a later stage, these reconstruction techniques leave the pace to more concrete sounds, meticulously aligned with the speech articulation: a recorded sound of water droplets is lip-synchronized to '*eau qui coule sur lui*', typewriters to '*que ça n'a pa de sens de travailler comme ça tous les jours*', and so on. This 'word painting' layer is however rather discrete, and from there, the sound base also widens, including various music fragments, synthesis sounds, and explosions.

Related works

- *Mon corps parle tout seul*, installation, 12 min (loop), 2015.
  Premiered in Paris, juin 2015 (new version premiered in Gennevilliers, september 2017)
  - Teaser: http://medias.ircam.fr/x724ca0

2.5.3 Concatenation of spoken words for *Any Road*

Speech datasets can also be the starting point for concatenation techniques: this was one of my interests while working on *Any Road*.

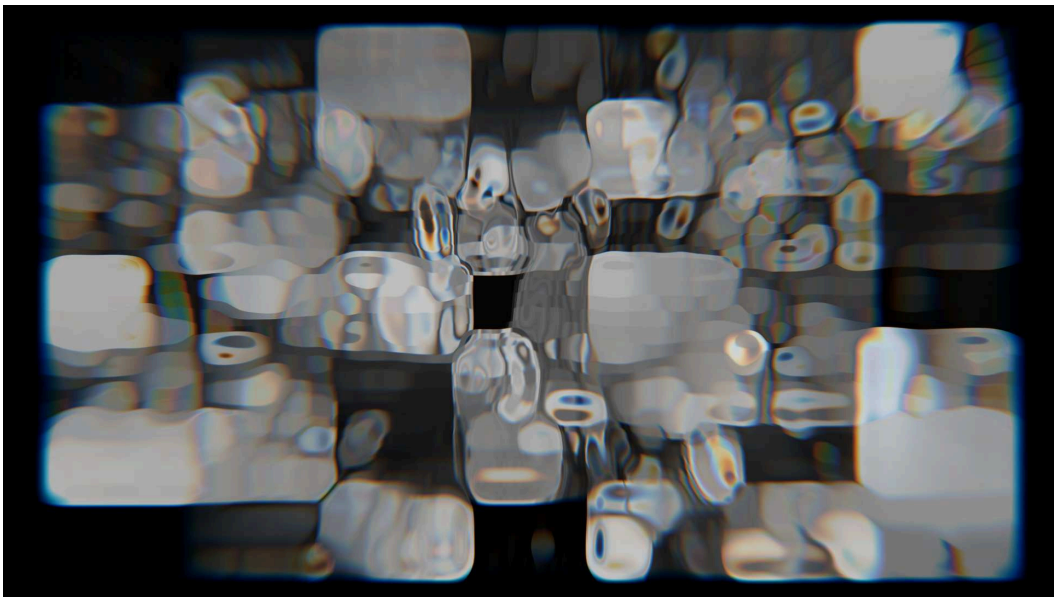


Figure 2.12: A frame from Any Road.

Any Road is a piece for orchestra, electronics and video, commissioned by the French Ministry of Culture, and premièred in Lyon at the *Biennale Musiques en Scène* (2016).

The original project was to develop a videogame to be played live, alongside the orchestral music, by two players located respectively at the left and at the right side of the orchestra, each associated with a loudspeaker; in the videogame, the positioning of elements and targets constituted a true score for each of the players, having its own interactive audio track being diffused via the corresponding loudspeaker. Due to contingent production issues, the project could not be achieved in this form,

and a fixed-media video by Boris Labbé was featured instead. Incidentally, the fact that a videogame can be an true interactive score has inspired the development of the *dada.platform* module (see 3.3.4.4).

Even though the project had shifted from live gaming to fixed media, the basic idea did not change: two loudspeakers represented two gamers, who (taking inspiration from the old Pong arcade game) engaged in a tennis match of sounds and words. Tennis of course has a long tradition of influence on Western composers, having inspired, among others, Claude Debussy (*Jeux*, 1913), Satie (*Le Tennis*, 1914) and Mauricio Kagel (*Match*, 1964). Even more curiously, Arnold Schönberg developed a specific system of music notation, based on a transcription of the events in a tennis match [Bateman and Bale, 2008; Sauer, 2009]. *Any Road* was an attempt to represent a Pong match in a widely-panned electroacoustic scenario, the orchestra being the *trait d'union* between the players.

Related works

- *Any Road*, for orchestra, electronics and video, 12 min, 2016.
  - Project website and full credits: `http://www.anexperimentwithtime.com`
  - Trailer: `https://www.youtube.com/watch?v=a81idHK8-CA`
- *Any Road* (version for ensemble), for large ensemble, electronics and video, 12 min, 2017.
- *Any Road* (electroacoustic version), for electronics and video, 12 min, 2017.
  - Link: `https://www.seditionart.com/boris-labbe-and-daniele-ghisi/any-road`

The speech dataset I used for *Any Road* is composed by a big number of recordings of *Alice In The Wonderland* and *Through The Looking Glass*, selected from the public domain recordings of the website LibriVox[8].

I used the open-source speech recognition system PocketSphinx [Huggins-daines et al., 2006] to locate throughout these recordings instances of specific words. A higher level interface was developed in Max to facilitate the process (see fig. 2.13).

The quality of the PocketSphinx detection algorithm sensibly depends on the size of the dictionary of words used as baseline (the 'dictionary width'). I chose to keep such dictionary rather small (around 1500 words), in order to purposely increase the number of false positives. The identified chunks containing the target word were then collected in a folder and randomly concatenated. Below are two examples of such concatenations (false positives being well noticeable in both of them).

Excerpts

- 'If', from *Any Road*:
  `data.danieleghisi.com/phd/snd/concat_if.aif|mp3`
- 'Know', from *Any Road*:
  `data.danieleghisi.com/phd/snd/concat_know.aif|mp3`

[8] `https://www.librivox.org`

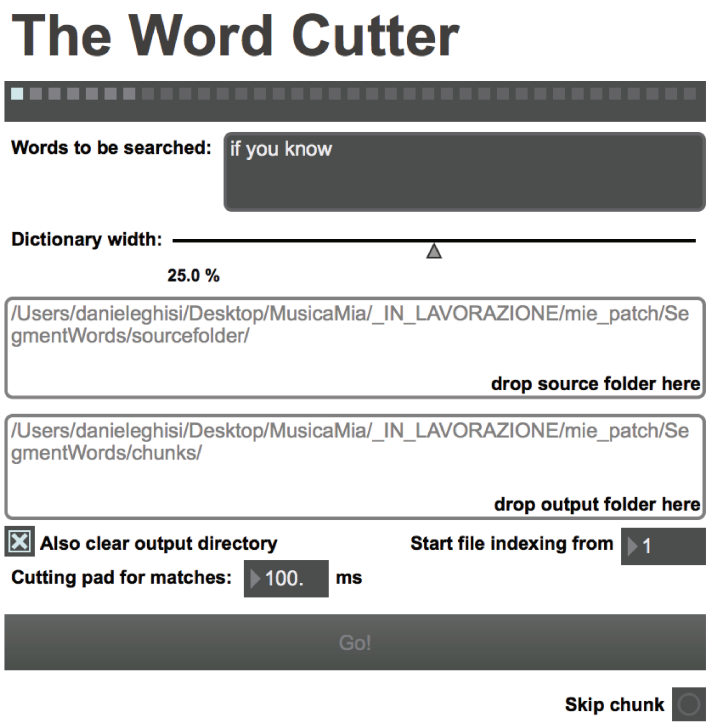


Figure 2.13: Screenshot of the Max interface used to detect occurrences of specific words for Any Road.

Concatenations were used to build, little by little, the sentence '*If you don't know where you are going, any road will get you there*', a classic Lewis Carroll's (mis)quote—actually a paraphrase of an exchange in Chapter 6 of *Alice's Adventures in Wonderland*. Video images, also quotations, are themselves 'panned' in the left and right sides of the video, and become more and more concrete, while maintaining perfect synchronization with the audio events (be they word utterances or sounds).

Excerpts

- From *Any Road* (with video):
  `data.danieleghisi.com/phd/vid/AR_extr2.mp4`

2.6 A query-based approach to musical samples

An Experiment with Time represented my first attempt to increase the amount of samples in my datasets; however, I was still working with simple collections of folders, each containing a collection of files; all queryable characteristics (such as tags or labels) had to be hard coded in the file name, in order to take advantage of Unix 'find' command to sieve the collections. This was far from being handy or easily extensible: for instance, a descriptor-based search in *An Experiment with Time* had to randomly pick and test several files in order to verify whether they matched the specified criteria.

Clearly, the following step had to be integrating a proper SQL database inside Max, so that previously analyzed or tagged samples could be quickly queried. Hence, I provided the *dada* library with the necessary tools: a SQLite database module (named *dada.base*) and its user interfaces (*dada.catart* and *dada.distances*). These modules are described more thoroughly in section 3.3.

Project after project, I fine-tuned the following workflow to deal with corpus-based concatenations. It does not in any way constitute a frame of rules, but rather provides a general setting for some frequently used operations.

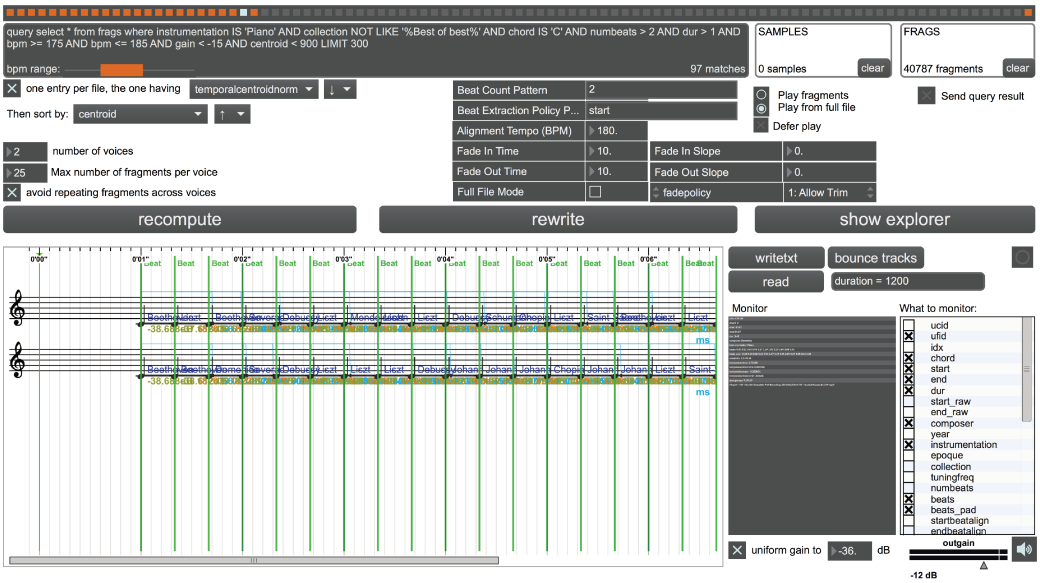


Figure 2.14: A Max patch used as a workspace to generate corpus-based concatenations. Queries are written in SQLite syntax; their results are then sorted by some descriptor, and then concatenated into a sequence of (possibly beat-aligned) fragments, over a certain number of voices. The query results can also be explored via the interface shown in fig. 2.15 and fig. 2.16.

Choosing a corpus. As a very first step, I usually set up a folder or a collection of folders containing the audio files in the corpus. The choice of such folders is, of course, a key compositional choice. Size and audio formats also vary from project to project.

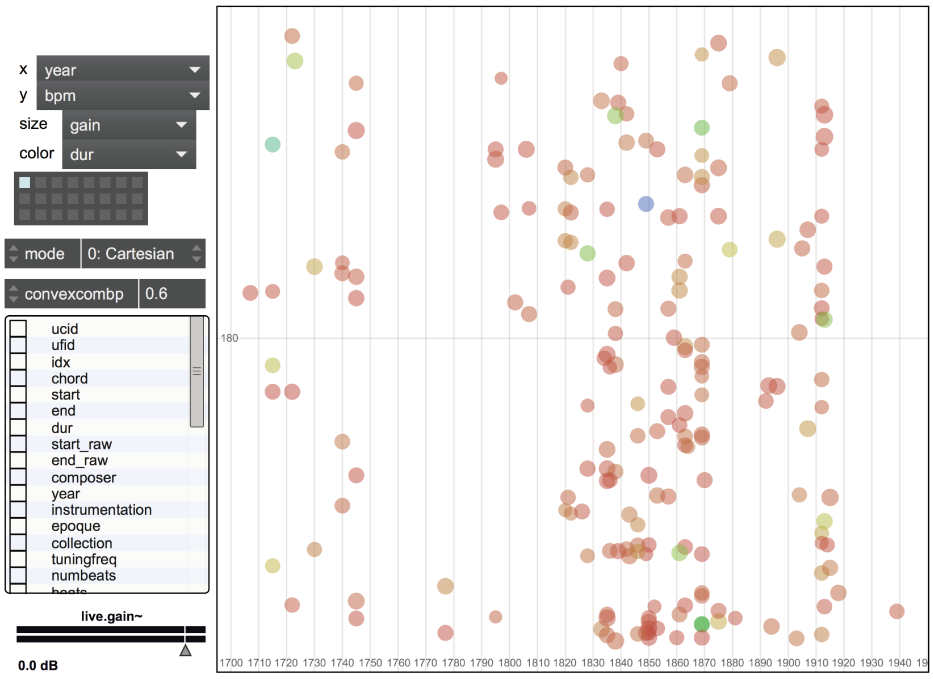


Figure 2.15: A Cartesian explorer for the query results of the patch shown in fig. 2.14. Each circle represents a fragment; according to the choice of features mapped on x and y axes, size and color, the circle is represented in a given position on the plane, with a given radius and color.

Tagging the corpus. Sometimes it is handy to work on ID3-tagged mp3 files. In this case it might be important to verify, fix or add the audio file tags properly. In order to do so, I usually take advantage of the library *id3lib*[9].

Segmenting the corpus. Each file in the collection is segmented according to a given principle, and fragment files are produced. Given the size of the corpus, I usually perform this step via parallelized Unix shell scripts. The open-source command-line tool Sox [2017] often helps, offering easy access to basic audio editing features. I have used, over time, several segmentation principles:

- fixed-window segmentation: fragments are cut via sliding of a fixed-size window on the corpus sound files, with a given overlapping factor;
- chord-based segmentation: cuts happen when chords change—for this purpose, I tend to refer to the algorithm [Mauch, 2010], which was also used in *An Experiment with Time*, as explained in section 2.3.2;
- beat-based segmentation: cuts happen at detected beats—for this purpose, I tend to use the Madmom library [Böck et al., 2016], namely its `madmom.features.beats` function.
- no segmentation: each sample in the corpus is already the smallest musical unit to be considered—this might be the case for impulses, single

[9] `http://id3lib.sourceforge.net/`

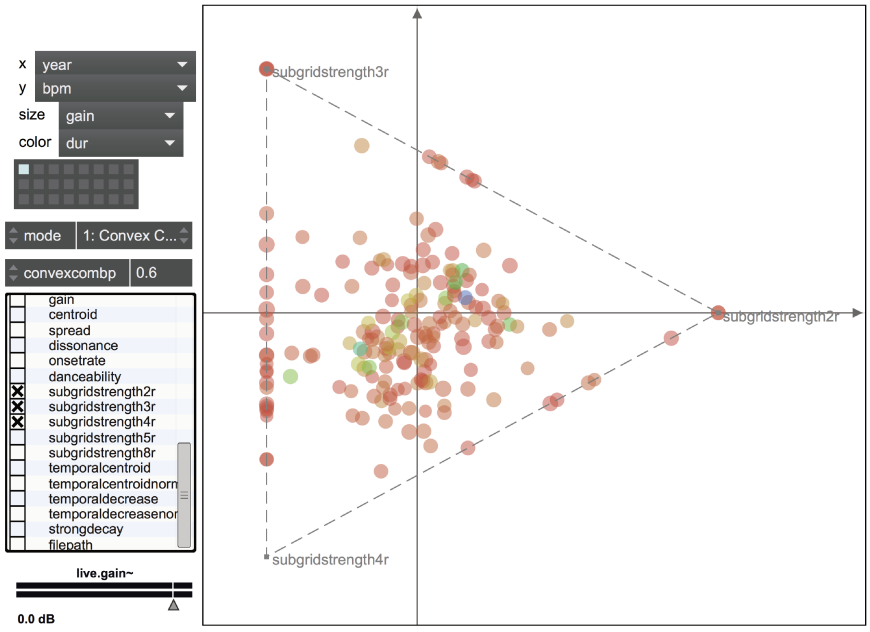


Figure 2.16: A convex-combination explorer for the query results of the patch shown in fig. 2.14. Similarly to fig. 2.15, each grain is mapped onto a circle in the Cartesian plane; however, the position of each circle is not determined by two x and y features, but rather from the relative strength of a given selection of features. Such features label the vertices of a regular polygon (a triangle, in this case), and circles leaning towards one of the vertices will have higher values for the corresponding feature with respect to the other ones. In this case, the displayed features are named `subgridstrength2r`*,* `subgridstrength3r` *and* `subgridstrength4r`*, describing how well the energy pattern inside each beat matches equal 2-, 3- and 4-subdivisions; in other words, if we consider beats to be quarter notes,* `subgridstrength2r` *accounts for how well the rhythm matches eighth notes,* `subgridstrength3r` *accounts for how well the rhythm matches a triplet of eight notes, and* `subgridstrength4r` *accounts for how well the rhythm matches sixteenth notes.*

notes or attacks.

Sometimes keeping a disk version of all the segmented audio files might be cumbersome, or might consume too much space on disk. In these cases, this step and the following one (analysis) can be interleaved and performed together for each file, creating temporary fragments which will be then analyzed and, later, removed by the script.

Analyzing each fragment. The audio fragments created by the previous step are analyzed, and a certain number of features are extracted, often via a combination of python and Unix scripts. The choice of analysis modules and of their parameters of course highly depends on the specific compositional needs. Some of my common choices are:

- straightforward labelling—such as file name, fragment duration or fragment index/onset (so that the grain could be later relocated in the original file). Generally I also define some identifiers for each fragment: an

`ufid` field (*unique file identifier*, an integer number unambiguously identifying the original file), an `idx` field (the 0-based index of the chunk inside the original file), and an `ucid` field (*unique chunk identifier*: an incremental integer unambiguously identifying the fragment, usually defined, for my own purposes, as $\texttt{ucid} = \texttt{ufid} * 10^6 + \texttt{idx}$);

- ID3 tags contained in the file—such as author, title, or any used-defined tag added at point 2;
- standard low-level audio analysis—such as centroid, spread, loudness, and so on. I usually take advantage of the Essentia library [Bogdanov et al., 2013] to perform them;
- higher level audio feature—such as HFCs, MFCCs, spectral complexity, dissonance, and so on (all included in Essentia);
- beat or chord detection—for which I usually rely on the aforementioned Madmom library;
- custom coded features—as an example, based on the Essentia library, I developed a pattern-matching algorithm in order to estimate how well the sample matches a given beat-based rhythmical pattern (see fig. 2.16). Via this system, one can estimate for instance how 'ternary' (or 'swing'-like) the beats are.

I usually prefer to store analyses in a human-readable text file, formatted in as a *bach llll* (each feature wrapped in a level of parentheses, and in turns each fragment wrapped in a level of parentheses). An example could be:

```
(
  ( filename "000001.mp3" )
  ( duration 1.260998 )
  ( chord "C6" )
  ( idx 100 )
  ( title "Rhapsody In Blue" )
  ( artist "George Gershwin" )
  ( year 1924 )
  ( samplerate 44100 )
  ( bitrate 128000 )
  ( gain -30.82905960083 )
  ( centroid 1206.59716797 )
  ( spread 3438790.5 )
  ( zerocrossingrate 0.0548473000526 )
  ( dissonance 0.452602744102 )
...
)
(
  ( filename "000002.mp3" )
  ( duration 1.758005 )
...
)
...
```

Importing the database in *dada*. The file created at the previous step contains all the database information, and can be imported into *dada.base* via the `appendfromfile` command. At each startup, *dada.base* can either reload a

database from a textual file, or attach to a native SQLite3 file (which is much faster, although the file itself is less human-readable).

Performing queries. Any standard SQLite query can then be performed on *dada.base*, in order to only select a specific subset of the fragments. Assuming that `frags` is the name of the table containing the analyzed fragments, here are some examples:

`SELECT * FROM frags WHERE instrumentation IS 'Piano' AND chord IS 'C' AND ABS(bpm - 120) < 5`	Select all C major piano fragments with a tempo of approximately 120 bpm
`SELECT * FROM frags WHERE instrumentation IS 'Piano' AND chord IS 'C' AND numbeats > 2 AND ABS(bpm - 120) < 5 ORDER BY RANDOM() LIMIT 100`	Select at most 100 C major piano fragments, randomly ordered, with at least 2 beats, having approximately 120 bpm
`SELECT * FROM frags WHERE where instrumentation IS 'Orchestra' AND ABS(dur - 2) <= 0.2 AND centroid > 2000 AND gain < -45`	Select orchestral fragments lasting about 2 seconds, having centroid higher than 2000Hz and having intensity less than -45 dB.
`SELECT * FROM frags WHERE where instrumentation IS 'Orchestra' AND dur >= 4 AND temporalcentroidnorm < 0.5 AND onsetrate < 0.3`	Select orchestral fragments lasting at least 4 seconds, having temporal centroid in the first half of the file, and having an onset rate (amount of detected onsets per second) below 0.3
`SELECT f1.*, f2.* FROM frags AS f1 INNER JOIN frags AS f2 ON f1.ufid == f2.ufid AND f1.ucid == f2.ucid - 1 WHERE f1.instrumentation IS 'Piano' AND f1.chord IS 'C' AND f2.chord IS 'Gm' AND f1.numbeats >= 2 AND f2.numbeats >= 2 AND f1.endbeatalign == 1 AND ABS(f1.bpm - 60) < 10`	Select couples of consecutive piano fragments inside a certain file, with approximately 60 bpm, where the first fragment is in C major and the second one is in G minor, provided that both fragments have at least 2 beats, and there is a beat at the very end of the first fragment (in other words: search for C major to G minor cadences across the dataset).

Organizing results into concatenations. The Max patch shown in fig. 2.14 is my current way to perform queries and arrange database results into a concatenation of fragments, represented by notes in a *bach.roll*. The explorer displayed in figure fig. 2.15 and fig. 2.16 shows how query results are scattered with respect to some of the features (also see section 3.3.2 for more information).

Concatenations can be either aligned to the native fragments' beat grid, or to an imposed beat grid (useful for regular concatenations), to which each fragments may adapt. Options are provided for sorting, removing duplicates within the same file, normalizing to a nominal RMS loudness, crossfading and trimming. The final score can be saved, recorded or bounced in off-line mode.

2.6.1 *The Well Tempered Sampler*

The method outlined above is a general framework which I am planning to explore, modify and extend while working on a larger collection of 24 pieces, named *The Well Tempered Sampler*, each uniquely composed starting from audio extracts roughly matching a given major or minor chord. It is a long-term project, and I am still in an exploratory phase; some of the pieces will feature live musicians, in which case either a symbolic database will be used (see section 2.8), or the electronic result will be partially orchestrated—as was the case for *An Experiment with Time (reloaded)*.

As I was testing for the first sketches of the project, two possible paths emerged. The first one is connected with an 'installation'-like situation, where, in a sense, the temporality of the piece emerges from the musical material, and where the compositional practices mostly relate to sorting, ordering and montage. In a sense, in this approach, the dataset is both the material and the main character. The second path, conversely, pertains to a 'concert'-like situation, to which I am probably more used: I would forge the content and form to a much greater extent, using the datasets as a mere palette.

I have tried to sketch the two possibilities with *In C major* (exploring the 'installation'-like format) and *In C minor* (exploring the 'concert'-like format). As a matter of fact, I started the project with the idea that *all* major chords would relate to installation-like situations and *all* minor chords would relate to concert-like situations; however, quite honestly, I am not convinced by this assumption and I am still exploring the middle ground between the two points of view—I would not be disappointed if by the end of the collection they had merged into a unique one.

Excerpts

In C major, from *The Well Tempered Sampler*:

- Excerpt from the beginning:
  `data.danieleghisi.com/phd/snd/WTS1_extr1.aif|mp3`
- Excerpt around 2':
  `data.danieleghisi.com/phd/snd/WTS1_extr2.aif|mp3`

In C minor, from *The Well Tempered Sampler*:

- Excerpt from the beginning:
  `data.danieleghisi.com/phd/snd/WTS2_extr1.aif|mp3`
- Excerpt around 2':
  `data.danieleghisi.com/phd/snd/WTS2_extr2.aif|mp3`

2.7 Concatenation of sung words for *An Urban Dictionary of Popular Music*

In order to explore the conceptual side of the installation-like behavior even further, I have started to work on another project, currently in a preliminary phase, named *An Urban Dictionary of Popular Music*.

A database of rock, jazz and pop music with synchronised lyrics will be analyzed; the 5000 more common words or sentences will be retrieved (from 'a', 'able', 'about', 'above'... to 'zero', 'zombie', 'zone', 'zulu'). The corresponding audio files will be then segmented and concatenated in order to create a database of sounds where such words or sentences are sung. For each word or sentence a montage will be carried out, concatenating a choice of its various repetitions in a musically meaningful way. These word-montages will in turn be sequenced in order to achieve an unique and continuous run through the whole dictionary, in alphabetical order. The dictionary will also feature multi-word phrases (e.g., 'close', 'close my', 'close my eyes', 'close to', 'close your', 'close your eyes', 'closed', 'closer'...), provided that they show up in the list of the 5000 most common expressions. The duration of the portion dedicated to each word or sentence will reflect its relative importance (according to the number of its occurrences in the database). In synchrony with the audio track, the word or phrase being sung will be projected very simply and rather discretely in a corner of the venue.

Ideally, the whole dictionary would span 24 hours or more; it must be in any case *disproportionately long*, as if any instance would only be a tiny portion of the whole project. The installation can be proposed either once (from A to Z) or in a looped form. An alternative form of presentation for the same content might be an actual interactive dictionary, maybe in a web-based context.

Excerpts

- 'Kiss', preliminary study for *An Urban Dictionary of Popular Music*:
  `data.danieleghisi.com/phd/snd/Concat_kiss.aif|mp3`
- 'Love', preliminary study for *An Urban Dictionary of Popular Music*:
  `data.danieleghisi.com/phd/snd/Concat_love.aif|mp3`
- 'Tomorrow', preliminary study for *An Urban Dictionary of Popular Music*:
  `data.danieleghisi.com/phd/snd/Concat_tomorrow.aif|mp3`

The whole project embodies an extreme dilation of a linear sequence of sung words. It will convey a feeling of traumatic obstinacy that we experience in front of something that encompasses us. Each concatenation of a single word will be rather long in itself, as a sort of 'black hole' where perception indefinitely lingers before moving to the next one, as a sort of 24-hours long progress bar, one 'pixel' at a time. Each passage from one word to the following one is already a radical change. Across this stretched time, words will cluster: fragments we recognise will interleave with others we don't know, repeating the same words over and over. Through these different repetitions, a web of similarities and connections emerges. In a sense, the project embodies a sort of non-linear view on popular music, chopping the standard

flow of time in same-word chunks, forcing us to hear the fragments differently. The project also relates both to Jorge Luis Borges's *The Library of Babel* and to Jean-Paul Sartre's *La Nausée*, namely to the character of 'the Autodidact', who passes his time reading every book in the local library in alphabetical order.

On a technical side, a database of lyrics, containing their time annotations, will be queried to find in which song and roughly at which time onset a given word combination shows up. From these results, if the audio file is available, each region can be fine-tuned and exported. It is important to stress that differently from all previous projects, *An Urban Dictionary of Popular Music* would only work if the chosen database contains a large portion of the most famous pop, rock and jazz songs. This underlines a distinct conceptual signature, but also raises questions about copyright: although each fragment would be at most 1 or 2 seconds long, this might still constitute a problem, and a proper way to handle copyright in this context should be inspected before the work being publicly presented. However, as I have already stated in section 2.1.2, I am convinced that similar practices should lie well within the boundaries of a 'fair use' of music material.

2.8 A symbolic approach: a database of scores

An important portion of my current research is targeted at finding equivalences between audio-based and score-based musical paradigms: using hybrid score notation for electronics, on one side, and applying exploratory-driven audio techniques to symbolic notation, on the other one. In particular, the dataset-based approach to montages and concatenations is transferrable to the symbolic domain, extending the concept of score granulation (introduced by Agostini and Ghisi [2012] and then later implemented in the *cage.granulate* module [Agostini et al., 2014]), allowing a finer control on the concatenation of grains, according to some relationships between the grain features extracted during the analysis process.

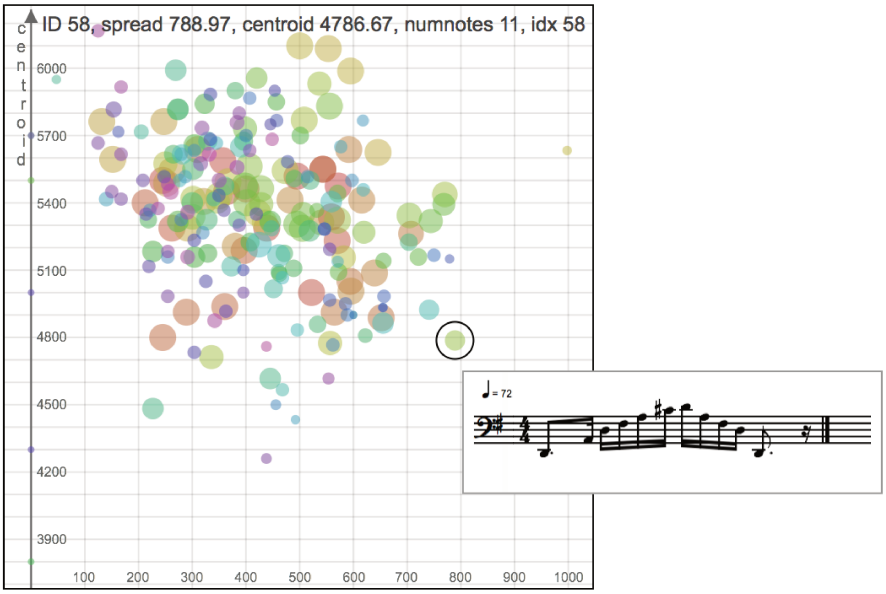


Figure 2.17: Each point represents a measure of Johann Sebastian Bach's first cello suite, organized by average pitch (vertical axis) and average register extension (horizontal axis). The size of a grain represents the number of notes in the measure, while the color represents the measure number (red to violet).

The basic idea is that a dataset of scores can be segmented into 'grains', which are then analyzed. The feature extraction is based on the *bach* lambda loop visual programming pattern [Agostini and Ghisi, 2015], hence making analysis fully customizable: one can extract standard features (such as the average pitch, akin to the 'centroid' descriptor, or the average MIDI velocity, akin to the 'loudness' descriptor) as well as more exotic ones (essentially anything one could program in a Max patch). All information about grains is stored in a database, which can be explored, filtered or navigated via graphical user interfaces. The structure of the system is described more in detail in section 3.3.2. It is worth noting that the interfaces are abstract views on SQL databases, and can hence be used to represent a wide range of items—primarily sounds and scores (in proportional or traditional notation), but also, potentially, any other type of digital data, including images, videos, and text.

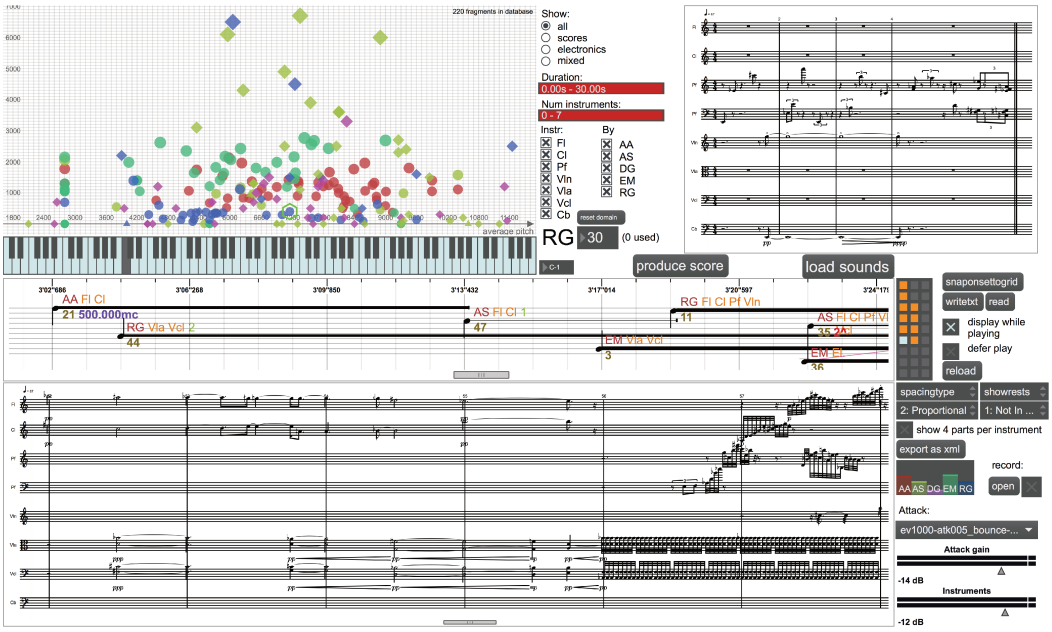


Figure 2.18: Screenshot of the patch used to perform the montage of atomic scores for the last event of I mille fuochi dell'universo.

As a usage example, one might consider the final portion of *I mille fuochi dell'universo*, a collective composition that will be presented in section 4.4, signed by */nu/thing*, a group of five Italian composers to which I belong. The foundation of the last five minutes of the piece, corresponding to the final 'event', is a rough montage, subsequently refined and reworked by hand, of a certain number of 'atomic scores'. The montage is based on a corpus of about 250 acoustic, electroacoustic and mixed 'score grains', created by collecting 50 fragments by each composer of the group, explicitly written for such montage, all complying with a set of rather loose harmonic constraints, and with the additional requests that very few instruments should be used inside each fragment, and that pitches should be kept somewhat localized in the instrumental registers. The requests were set to ease harmonic coherence and to facilitate superpositions of fragments.

In order to carry out the montage, the Max interface shown in fig. 2.18 has been created. The top-left view shows the score grains, organized by average pitch and width of their register; circles represent acoustic scores, diamonds represent electronics scores, triangles represent mixed scores. When hovering with the mouse on a grain, the corresponding score is shown and played (via sample-based simulations). The set of displayed scores can be filtered according to type, duration, number of instruments and author.

The middle, proportionally-notated score with no clef displays the actual montage. Each note represents one of the grains: its length correspond to the actual grain length (so that grains could be trimmed if needed); its MIDI velocity, displayed as duration line width, accounts for modification in the score dynamics; its

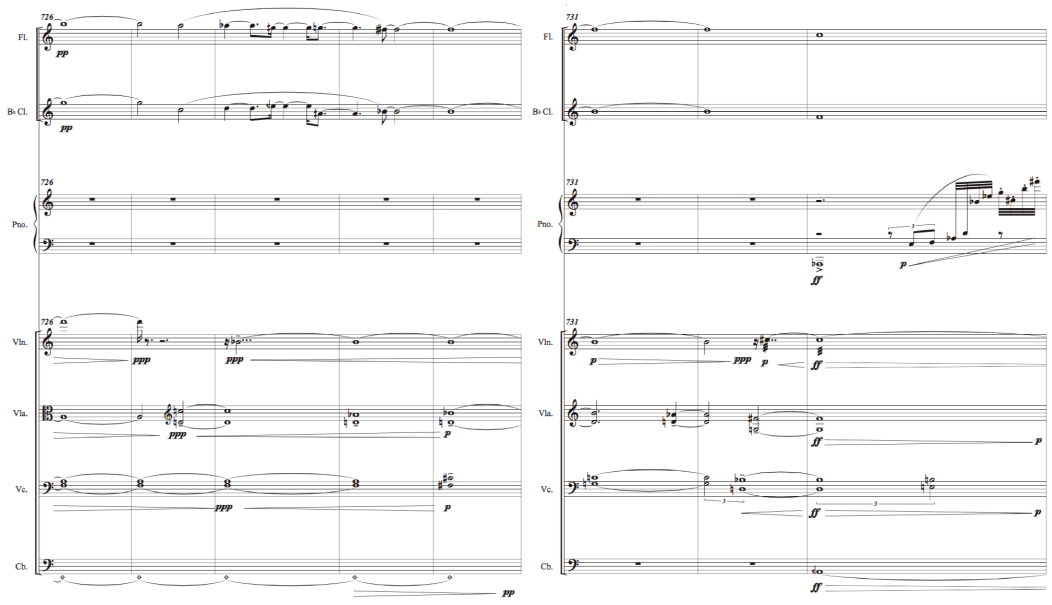


Figure 2.19: Portion of a score draft for the final event of I mille fuochi dell'universo *(measures 726-733). One can notice the similarity with the score in fig. 2.18.*

slots contain information about stretch and transposition factors, as well as the list of instruments to be accounted for. The rendering of the montage is displayed in the score at bottom, with an aptly chosen quantization, and is always kept aligned with the proportional view. It can be played, or exported as MusicXML file.

In the specific case of *I mille fuochi dell'universo*, the technique presented a certain number of shortcomings; first and foremost: the database was too small to be used as a creative palette. And, yet, we were attached to the idea of writing 'atomic' elements ourselves, both as a compositional and as a 'political' gesture: the final event had to be, in a way, our collective signature. Increasing the dataset size by some orders of magnitudes was virtually impossible, as it would require too much individual work. The small dataset size also worsened all coherence problems: very different musical figures appeared in the base, but not enough of them to build a grammar. We ended up, at first, with rather inconclusive montages, prompting memories of a certain twentieth century avant-gardism.

Progressively, as we accorded more freedom to modifying the grains, the technique yielded a valuable starting point for a draft score, which had to be then heavily reworked. Some crude portions of montages are still visible in the final version (see, for instance, fig. 2.19), but most of them have been modified with new ideas and adaptations.

Chapter 3

Towards a Real-Time Computer-Aided Composition

Sections 3.2.2 and 3.5, as well as some portions of sections 3.1 and 3.2.4, have been previously published, in a slightly different form, in the article Ghisi, D. and Agostini, A. (2017). Extending bach: A Family of Libraries for Real-time Computer-assisted Composition in Max. Journal of New Music Research*, 46(1):34–53.*

3.1 Real-time computer-aided composition

A key to interpreting the history of Western music is, arguably, the relationship between composition and computation, embracing otherwise very different experiences and phenomena, including: Greek's theory of musical proportions; the 'harmony of the spheres' and the *musica universalis*; the inclusion of music in the Quadrivium (along with arithmethic, geometry and astronomy) during the Middle Ages; the notational complexity of the *Ars Subtilior*; the highly refined, quasi-algorithmic systems developed by the Franco-Flemish school; Bach's interest in canonic forms; Mozart's palindromes and dice games; the combinatorial complexity of dodecaphonic and serial music; the Fourier analysis of sound as a basis for the works of French spectralism. The catalog is necessarily incomplete; most notably, the advent of computers, in the twentieth century, has generated considerable interest on how to take advantage of the enhanced precision and speed of computation when dealing with music: computer music was born.

The vast domain of computer music research and applications can be roughly divided into two sectors: on the one hand, tools for generating and transforming audio samples; on the other hand, systems for dealing with symbolic data—'notes' rather than 'sounds'. In the latter area, a further distinction can be made between tools for computer-assisted music engraving (Finale, Sibelius, LilyPond...) and tools for computer-aided composition (CAC for short), allowing generation and transformation of symbolic musical data, like OpenMusic [Assayag and al., 1999; Agon, 1998], PWGL [Laurson and Kuuskankare, 2002], or Common Music [Taube, 1991]. Historical surveys of CAC techniques are provided, among others, by Roads [1996], Assayag [1998] and Miranda [2001].

Of course, the boundaries dividing these areas are blurred: for example, some audio sequencers also provide high-quality graphical representation of musical scores and sound treatment, and virtually all of them have the ability of representing and treating MIDI data; modern CAC environments include tools for sound synthesis

and transformation; at least two graphical programming environments, the closely related Max and PureData, have MIDI control and sound generation and transformation among their main focuses, but are at the same time capable to deal with arbitrary set of data, input/output devices and video. Nevertheless, gaps between all these categories still exist, and it is difficult to find tools suitable to work in scenarios combining the peculiarities of several categories. For instance, it is essentially impossible in most environments to drive sound synthesis via a symbolic score, representing not only pitches and MIDI velocity, but also data that are too numerous and complex to be efficaciously represented via MIDI—with the obvious exception of Csound; but the Csound score representation, although rich, clean and coherent, is rather counterintuitive, and it is virtually impossible for composers to 'think' directly within its formalism in the way they would do with paper and pencil, or even with typesetting software.

In order to produce tools capable of bridging these gaps, some years ago, I undertook with composer Andrea Agostini a freelance research project which ultimately led to the development of a library for Max called *bach: automated composer's helper*, primarily meant to provide Max with the ability to treat and display musical scores. The *bach* library is directly inspired by traditional Lisp-based systems (in particular, OpenMusic and PWGL), and shares with projects such as MaxScore [Didkovsky and Hajdu, 2008] and InScore [Fober et al., 2012] the ability to operate with symbolic musical representation in real time. More recently OpenMusic has implemented a 'reactive' mode too [Bresson, 2014], which mimics the notification system of event-driven paradigms. This seems to be a good example of how the whole CAC community is making a conjoint effort to narrow the gap between event-driven ('reactive') and demand-driven ('off-line') paradigms.

The issue is crucial, since real-time properties of a digital environment deeply affect the nature of the very act of composing. As an example, electroacoustic composers expect digital sequencers, synthesizers or graphic programming environments to react *as quickly as possible* to any interface change (in a sort of musical 'what you see is what you get' approach); the same holds true for traditional composers who typeset a score in a piece of software like Finale or Sibelius.

Traditionally, CAC environments have endorsed a different paradigm, conceiving the creation and the modification of a score as an out-of-time activity: a graph of operations can be edited (for instance, via visual programming) but has no effect whatsoever until a certain *refresh* operation is performed, which *renders* the graph and outputs the result (e.g., a score). Yet, there is no deep reason why symbolic processing should only be performed in out-of-time mode; on the contrary, interactivity might be an added value to the musical exploration.

Similar ideas have already been formulated by Puckette [2004] and Cont [2008b]; Andrea Agostini and I have ourselves tackled these arguments in [Agostini and Ghisi, 2013], mentioning a divide between a 'performative' and a 'speculative' approach.[1]

[1] According to Hagan [2014], 'speculative' should be replaced with 'notional', since the former "would suggest that that composers working with CAC do not know what their results will be with each program", whereas "most seasoned composers have some idea of their algorithm's output,

An overview of the questions raised by the real-time/off-line dichotomy can be found in [Seleborg, 2004].

Among the examples of processes easily achievable within *bach*, but hardly programmable in off-line CAC environments, one might cite: live recording of notes into a proportionally notated score; interactive symbolic granulation of an original score into a new one, filled in real time; customizable sequencing, for instance implementing mechanisms of perturbation of the flow of time; reactive meta-score scenarios, where modifying a given note immediately affects the rendering of a certain process, such as the creation of certain sequences of notes. We will encounter these and other examples during the continuation of this thesis.

In traditional CAC environments, there is no relationship whatsoever between the physical time (the time it takes for a composer to obtain results via the computer music tools), and the musical time (the time of the output score). On the contrary, in a reactive environment, a flexible degree of connection can be established: for instance, some portions of physical time could match with the time of some generated score raw material (to be further refined); or the two concepts can coincide, envisaging a symbolic computer-aided improvisation system, whose running outcome is indeed a growing score.

This was the motivation at the core of the development of *bach*. At the time of writing, Andrea Agostini and I have been working on the library for almost seven years; although a few of our desiderata have not been tackled yet, we feel that the library has reached a certain level of maturity. It is currently used as the basis for numerous high-profile artistic and research projects, and we estimate that its current active user base amounts to about 1500 people. Since 2015, we have started to widen the scope of *bach* via a series of extension libraries (the '*bach* family', [Ghisi and Agostini, 2017]), each of which shares with *bach* the core philosophy and the basic programming principles, but at the same time proposes a different point of view on computer-aided composition. The first extension of *bach* is named *cage* [Agostini et al., 2014]; *dada*, the library developed within the framework of this thesis, is the second one.

Section 3.2 briefly describes the scope and the characteristics of *bach* and *cage*; section 3.3 introduces the *dada* library—designed to deal with non-standard interactive interfaces handleing music generation and composition—, and offers some usage examples in my recent works; section 3.4 addresses the issue of bridging the gap between scores and instruments, discussing the concepts of 'hybrid scores' and 'meta-scores', and exhibiting examples of meta-scores used in my recent compositions; finally, section 3.5 outlines the perspectives and future work on real-time CAC within Max.

---

with only occasional surprises". However, we never intended to state that composers have no idea of their algorithms' output; on the contrary, the puzzling word 'speculative' (a replacement for Puckette's even more equivocal word 'compositional') was meant to pertain to the semantic areas of 'contemplation' and 'abstract reasoning', and did not have, in our view, a 'conjectural' or 'theoretical' nuance.

3.2 Previous work: the *bach* paradigm

3.2.1 *bach: automated composer's helper*

Since 2010, the library *bach: automated composer's helper* provides Max with the ability to handle and display musical scores [Agostini and Ghisi, 2013, 2015]. One of its defining features is its seamless integration with the Max environment, which allows it to communicate easily with any other process implemented in Max, or any device connected to it: *bach* is meant to address a wide array of usage scenarios, including traditional computer-aided composition (by means of its data processing capabilities), management of electronic scores, and novel artistic practices taking advantage of the real-time opportunities offered by the system.

The choice of Max as the host environment for *bach* was prompted by several considerations, the most important being the ease of integration with a multitude of processes and devices, including DSP, MIDI, visuals, and virtually any hardware system. Another important consideration was the maturity and stability of the Max graphical user interface and of its graphical API.

At the forefront of *bach* are two interface objects, *bach.score* and *bach.roll*, providing graphical interfaces for musical notation. The difference between the two is that *bach.score* represents time in terms of traditional musical units (hence including notions such as rests, measures, time signatures and tempi), wheras *bach.roll* implements a proportional representation of time, in terms of absolute temporal units (namely milliseconds). While *bach.score* is useful to represent traditionally notated music (including more advanced scenarios, such as polymetric and polytemporal notation), one should take advantage of *bach.roll* in order to represent non-measured music, and also to provide a simple way to deal with pitch material whose temporal information is unknown or irrelevant.

The two notation modules share a wide palette of common features. They can be edited by both mouse and keyboard interface, and by Max messages; they support microtonality with arbitrary resolution; they have sequencing capabilities with variable-speed playback; their notes can carry metadata (such as text, numbers, breakpoint functions, filters, files, spatialization trajectories) inside dedicated containers called *slots*; the metadata can be retrieved during playback, and can be used to control synthesizers and other physical or virtual devices (see fig. 3.1).

The *bach* library also provides Max with two new data types: rational numbers and a nested list structure called *llll*, an acronym for 'Lisp-like linked list'. Rational numbers are extremely important in order to express temporal units such as 1/2, 3/8 or 1/12 (that is, a triplet eight note) as well as harmonic ratios. Lisp-like linked lists, strongly inspired by the tree structure typical of the Lisp programming language (indeed, the whole conception of *bach* has been heavily influenced by existing Lisp-based environments, such as OpenMusic), are essentially lists capable of containing lists within themselves. In its simplest form, an *llll* is equivalent to a generic Max message, but *llll*s are meant to contain hierarchically arranged data: thus, they lend themselves to representing complex collections of information, such as whole musical

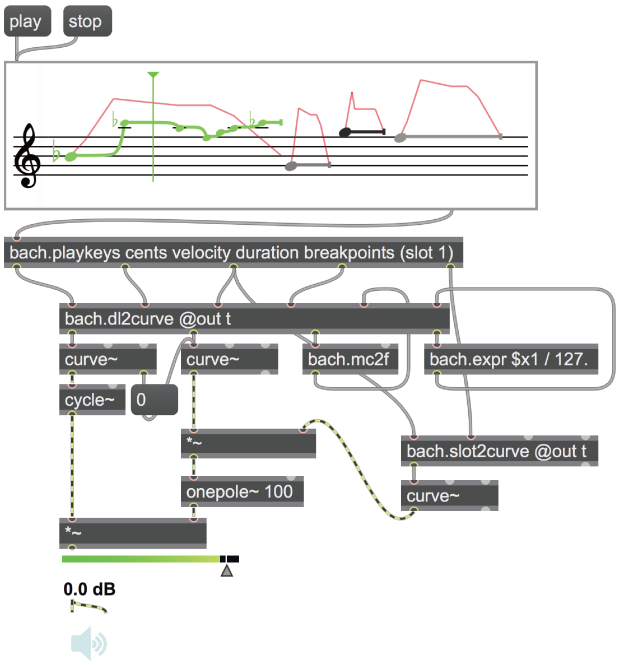


Figure 3.1: A simple implementation of a basic synthesizer controlled by a bach.roll. *Each note contains in its first slot, displayed as a breakpoint function, its amplitude envelope.*

scores, but also sets of musical parameters isolated from the rest of the score (e.g., the pitches of all the notes in the score, with their temporal information removed). Each score in a *bach.roll* or *bach.score* is indeed transparently handled as an *llll* (see figs. 3.2 and 3.3), essentially representing voices, measure, chords and notes via levels of hierarchy, and containing all their parameters [Agostini and Ghisi, 2012]. Beside dedicated messages and user interface operations, one can modify existing scores also by simply altering the corresponding *llll*, or build new scores by creating an *llll* with the correct syntax from scratch, as displayed in fig. 3.4 (c). As a consequence, strictly musical operations such as rhythmic quantization are just extremely specialized operations upon *llll*s (which of course can be performed only if the *llll* itself is structured properly, and if its content is consistent from the point of view of musical notation).

Most *bach* objects are designed to exchange *llll*s with each other, following the usual Max metaphor of messages traversing patch cords, and consistently with the overall philosophy of Max, which, from a formal point of view, mixes aspects typical of functional languages, such as the dataflow paradigm, with other typical of object-oriented programming and, more generally, state machines (most objects actually maintain states, and taking this into account is essential to building virtually any non trivial Max patch). The interaction of *llll*s with the Max ecosystem has some slight intricacies the user has to be aware of; for instance, any *bach* object can output *llll*s in two different formats, called *native* and *text*, according to the setting of the *out* attribute. Each format has its advantages and drawbacks: essentially, the native

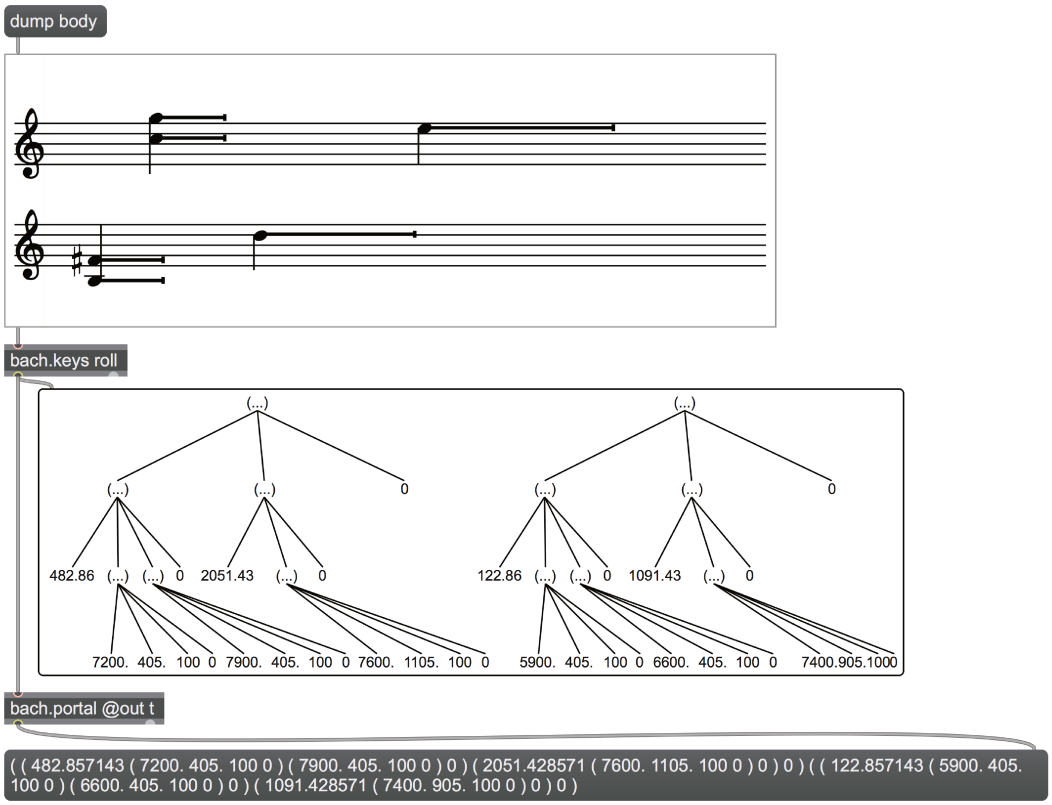


Figure 3.2: The content of a simple bach.roll *displayed in tree form (middle) and in textual* llll *form (bottom). There is one parenthesis level for each voice, one for each chord and one for each note. Metadata such as clefs or keys are not displayed.*

format, which is the default, is more efficient and poses no limitation to the amount of data contained in each *llll*, but its contents cannot be accessed by standard Max objects; the text format, on the contrary, allows seamless communication with all the standard Max objects, but has a 'hard' length limitation and is less efficient. On the other hand, all *bach* objects can indifferently understand both native- and text-format *llll*s. A thorough description of these low-level aspects can be found in [Agostini and Ghisi, 2015].

In addition to the two main editors, the *bach* library contains a few other graphic interfaces (among which a clock diagram, a Tonnetz diagram, and a floating slot window) and a large number of modules operating upon *llll*s, both as generic data structures (e.g., tools for reversing an *llll* or enumerating the elements it contains) and as containers of musical data (e.g., tools for operating on musical scores according to pitch-set theory principles, or for quantizing rhythmically a non-measured score). A complete overview of the modules is proposed in fig. 3.5.

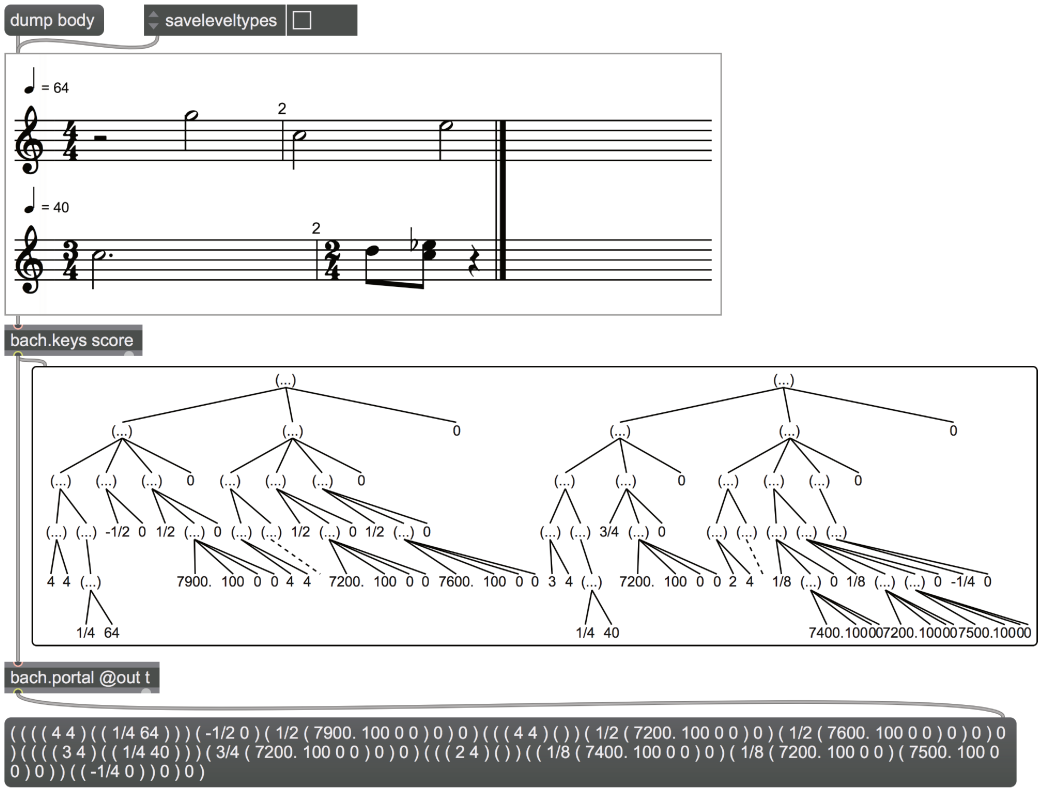


Figure 3.3: The content of a simple bach.score *displayed in tree form (middle) and in textual* llll *form (bottom). Levels correspond to voices, measures, chords, notes; intermediary levels corresponding to beams (or, in principle, to any rhythmic grouping) are also present. Metadata such as clefs or keys or level types are not displayed.*

3.2.2 Comparison with other software

As stated above, the design of *bach* has been explicitly inspired by the PatchWork family of Lisp-based CAC software: PatchWork itself [Laurson and Duthen, 1989], OpenMusic [Assayag and al., 1999] and PWGL [Laurson and Kuuskankare, 2002]. Besides the already discussed similarity between the Lisp tree and the *llll*, there is a correspondence between the 'reductionistic' operational and representational paradigm of *bach* and the aforementioned software tools, based upon the hierarchical arrangement of the different parameters of a score and their individual manipulation[2]. Of course, the real-time nature of *bach*, and its ease of interoperability with the larger infrastructure of Max, give it a distinct area of application of its own. On the other hand, in computationally demanding tasks, such as combinatorial processes, and inherently non-realtime operations, such as batch processing or generation of sound files, traditional CAC environments largely outperform *bach*.

The relation of *bach* to other software tools for dealing with musical representa-

[2]For an instance of a deeply different approach to the treatment of symbolic musical data, focusing on the intertwinement among the score parameters rather than their orthogonality, see the Strasheela system [Anders et al., 2005], and the discussion in chapter 4.

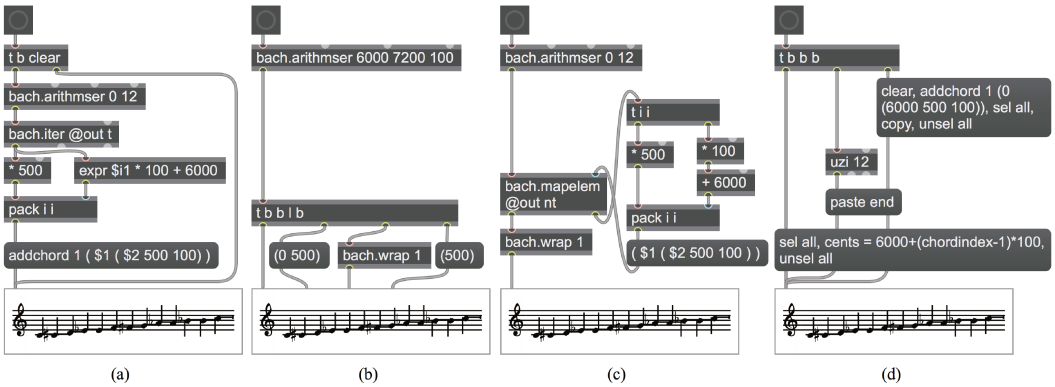


Figure 3.4: Four equivalent ways of generating a portion of chromatic scale in a bach.roll*: (a) adding notes one by one via* addchord *messages; (b) setting the values of each parameter independently; (c) building the* llll *containing the whole score information; (d) using scripting to create a middle C, copy it twelve more times, and assigning the proper pitch to each note. (A fifth way would be introducing the notes one by one via the graphic user interface.)*

tion, or other complex data structures, in real time (such as MaxScore [Didkovsky and Hajdu, 2008], InScore [Fober et al., 2012], FTM [Schnell et al., 2005] and MuBu [Schnell et al., 2009]) should also be remarked. To some extent, *bach* also shares with projects such as Antescofo [Cont, 2008a] and NoteAbility [Hamel, 1997] the goal of providing advanced sequencing capabilities. Each of these tools has its own strengths that distinguish it from all the others, such as the graphical quality of musical typesetting, or the richness of the data representation. The idea at the basis of *bach* is that all its modules form a coherent computer-assisted composition environment, and its very goal is to make live computer-assisted composition, in a responsive, interactive world.

3.2.3 Extending *bach*

In the early stages of the development of *bach*, the scope of the project grew quite rapidly, along with the number of modules included in the library; within a few months, *bach* was provided with a large set of basic tools. After about three years of development, there was a compromise to strike between sustainability and growth of the library. Since many of the ideas we had in our personal wish list had little to do with the scope of the modules inside *bach*, we started organizing the new modules, as organically as we could, in a series of different libraries, each inheriting from *bach* the basic principles of real-time computer aided composition, as well as some datatypes (rationals, *llll*s) and programming patterns (such as the lambda loop).[3]

The simplest way to build Max modules based on *bach* functionalities is to create

[3]This does not mean that the number of modules in *bach* is by now fixed (almost every recent release of *bach* has included some new modules, and this trend is likely to continue), but that the scope of *bach* will not change: handling symbolic data for traditional music representation, display and editing.

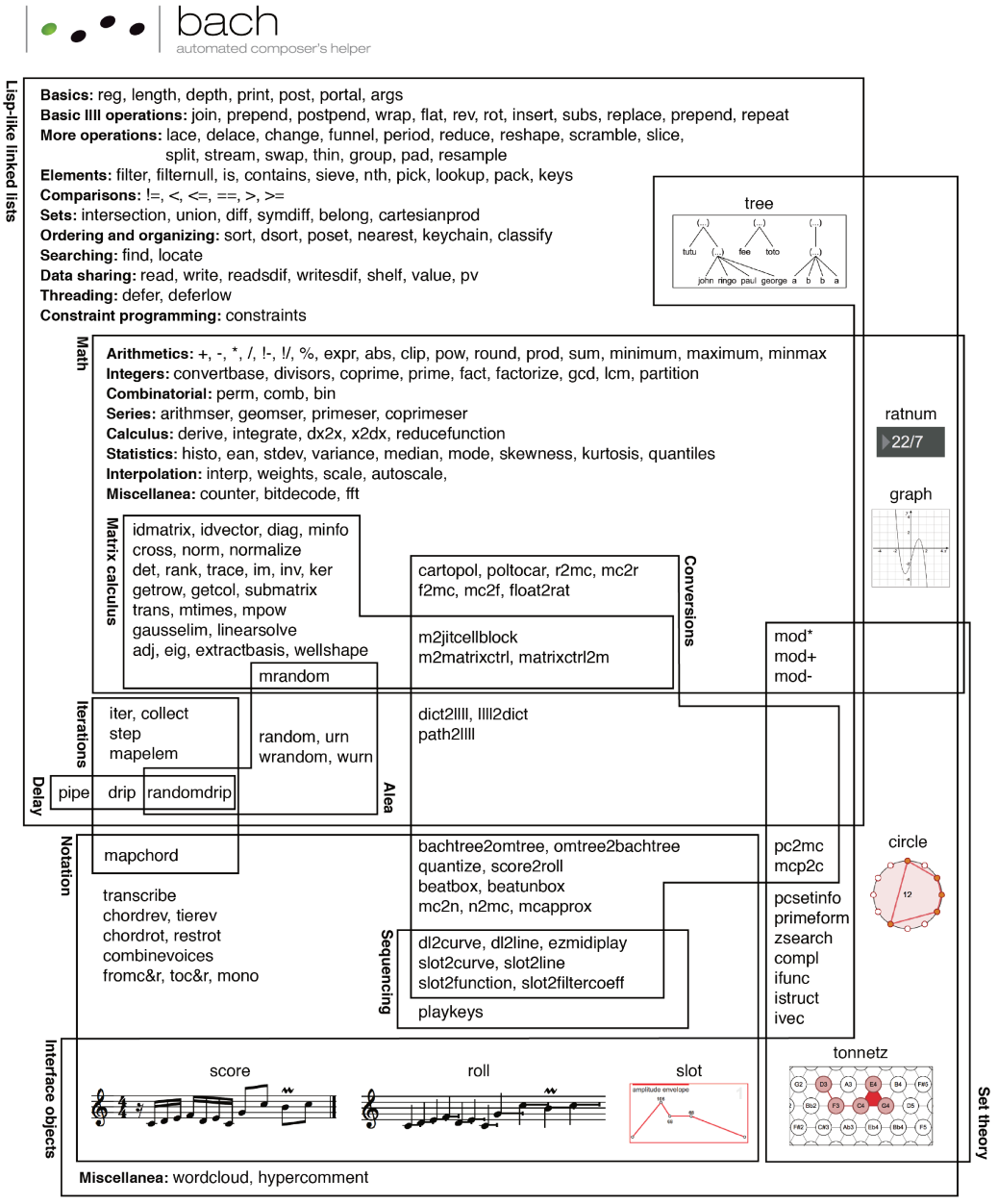


Figure 3.5: Overview of the bach *library (the* bach. *prefix has been omitted from all module names).*

'abstractions', which is: named patcher saved to disk and reusable as a subpatchers. The *bach* library contains two modules, *bach.args* and *bach.portal*, designed to equip abstractions with essentially the same functionalities that Max externals have.

Another possible way of extending *bach* is to take advantage of its public API, which provides the ability of handling *llll*s and rational numbers just like *bach* objects do, and can be freely downloaded from the *bach* website. The API is mostly written in C, because C is the language of the Max SDK upon which it is based, even though it contains some C++ modules for specific purposes. The API can be roughly divided into two major sections:

- Tools for manipulating *llll*s and rational numbers. With the exception of some basic, low-level operations, most of these tools have a one-to-one correspondance with *bach* objects: for example, there are functions called *llll_rev()*, *llll_rot()* and *llll_flat()*, directly matching the *bach.rev*, *bach.rot* and *bach.flat* objects. All these functions provide a large toolbox for building more complex operations upon *llll*s, and are widely used within *bach* itself, in the more complex objects that manipulate musical data.

- Tools for helping the exchange of *llll*s between Max objects: these tools essentially provide an abstraction layer built upon the plain Max object types, and some functions for facilitating the interaction of Max inlets and outlets with the lifecycle of *llll*s. All the objects that manipulate *llll*s should make use of these tools.

It should be added that the API is thoroughly documented, and includes the source code for some example objects: so any C or C++ programmer with a good understanding of the Max API and the basic principles of the *bach* library should be able to build his or her own *bach*-compatible Max objects. On the other hand, the API does not expose currently any feature for working with the internal score representations that are specific to the *bach.roll* and *bach.score* objects, nor for accessing the musical typesetting engine, but this situation might change in the future.

3.2.4 The *cage* library

The first project that was built starting from *bach* is the *cage* library [Agostini et al., 2014], designed to perform a set of standard computer-aided composition tasks (such as, among the others, generation of scales and arpeggios, or computation of symbolic frequency modulation), as well as a number of convenience operations (such as parsing of SDIF files, or audio rendering of a score).

Whereas *bach* features a wide majority of modules accomplishing low-level tasks (such as *llll* processing or formatting) and a certain number of more advanced modules, still performing conceptually basic operations (such as rhythmic quantization or constraint solving), all the modules in *cage* are designed to accomplish higher-level musical tasks. For instance, whereas *bach.rev* performs a reversal on a given *llll*, its counterpart *cage.rev* performs a score retrogradation (see fig. 3.6).

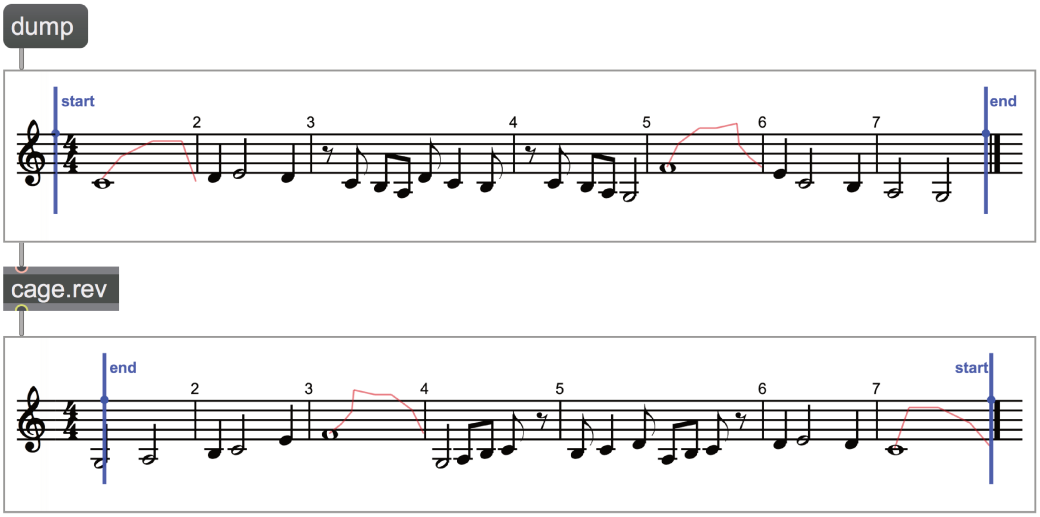


Figure 3.6: The simplest way to perform a score retrogradation is by using cage.rev. *All the temporal meta-information, including markers, temporal slots and pitch breakpoints, is handled properly.*

All the *cage* modules are abstractions containing *bach* modules, along with regular Max objects: hence, they are intrinsically open-source. This was a deliberate design choice: *cage*, supported by the *Haute École de Musique* in Geneva, has a pronounced pedagogical connotation. By double clicking on any module, users can learn how standard computer-aided composition techniques can be programmed in Max; also, they can easily copy and paste snippets of patches, or modify any process in order to tailor it more tightly to their needs.

However, in the simplest case, users do not need to perform any of these advanced operations: most of the tools in the *cage* library are capable of operating directly on the syntax of the *llll* representation of a score, making patching easier and more readable. Most of the time, in order to accomplish a given task, one can simply copy and paste a portion of a module's help file, or even directly experiment with the help file itself. Due to this facility of usage, Andrea Agostini and I have often suggested *cage* as a first approach to *bach*: users can get accustomed to *cage* processes, before delving into the intricacies of *llll* handling or score representation.

As shown in fig. 3.7, *cage* addresses a certain number of standard CAC scenarios, including:

- Generation of pitches according to different deterministic or stochastic criteria: e.g., arranged into scales, arpeggios or harmonic series, or according to probability weights or random walks.

- Management of melodic profiles, similarly to the *Profile* library in OpenMusic and PatchWork. Melodic profiles can be generated according to various principles, or extracted from preexisting sequences of pitches, and subsequently manipulated through compression, inversion, stretching, reversal, approxima-

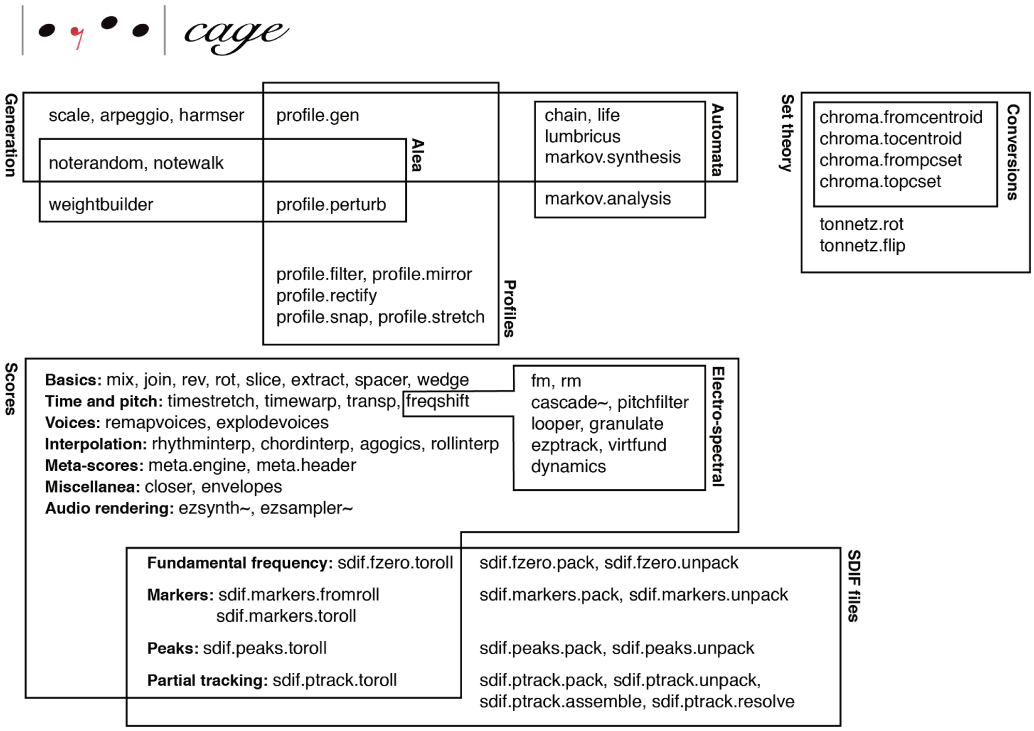


Figure 3.7: Overview of the cage *library (the* cage. *prefix has been omitted from all module names).*

tion to a harmonic grid, perturbation, filtering, and more.

- Helper tools for stochastic processes.

- Score handling, that is, various operations for performing high-level treatments of whole musical scores, including circular rotation, concatenation, transposition, retrogrades, slicing, and mixing; formalization of rallentando and accelerando, and generic temporal distortion of a score; harmonic and rhythmic interpolation; partial tracking (that is, assigning to individual voices notes from a given sequence of chords). This category also features two modules for meta-score handling, where a meta-score is a concept akin to that of *maquette* [Agon and Assayag, 2002], as implemented in OpenMusic, but aiming to be fully integrated with *bach* and Max on the one hand, and more general on the other hand (for a thorough description of the concept of meta-score and its implementation, see section 3.4). Finally, two tools for audio rendering of scores, respectively through sampling and synthesis also fall in this broad category.

- Processes inspired by electroacoustic practices, or emulating operations typically performed on sound: these include pitch and frequency shifting, ring and frequency modulation, virtual fundamental estimation, filtering, looping and granulation.

- Cellular automata and L-systems. These tools allow generating one- and two-dimensional automata (including John Conway's 'Game of Life'), as well as fractals by substitution, with completely customizable rules. On the other hand, they do not provide any graphical interface for such processes. Modules for creating Markov chains also belong to this category.

- High-level tools for musical set theory. This category includes modules performing conversions among pitch class sets, chroma vectors and spectral centroids [Harte et al., 2006], and performing geometrical operations on a Tonnetz representation. Notice that the *bach* library includes a number of low-level and graphical tools for musical set theory, such as a clock diagram, a Tonnetz graphical editor and interface, and a module calculating the interval vector of a chord.

- Management of SDIF files: *bach* includes two objects for reading and writing raw SDIF files, which are complemented by *cage* tools for high-level conversions (e.g., importing into a *bach.roll* object partial tracking, chord sequence or marker analyses a SDIF files, and vice-versa).

On the other hand, since *cage* implements a number of widely used CAC processes, a certain number of modules are explicitly inspired by libraries already existing in other environments, such as the *Profile* [Malt and Schilingi, 1995] and *Esquisse* [Fineberg, 1993; Hirs and editors, 2009] libraries for *Patchwork* [Laurson and Duthen, 1989], which have been subsequently ported to *OpenMusic*.

A comprehensive description of the cage library can be found in [Agostini et al., 2014].

3.3 *dada*: non-standard user interfaces for computer-aided composition

The *bach* and *cage* libraries date back, respectively, to 2010 and 2013. This thesis introduces a new library, *dada*—the third library of the *bach* family. The *dada* library, based on the *bach* public API, collects, documents and organizes most of the tools I have developed for this thesis, namely non-standard musical interfaces for interactive computer-aided composition and music generation. This section outlines the rationale behind the *dada* library, and provides an overview of its modules, detailing, whenever meaningful, if and how I have used them in a musical project.

3.3.1 The scope of *dada*

Although *bach* features a certain number of interactive, graphical objects, all of them essentially implement established representations of music, be they traditional scores or alternative but widespread representations such as the clock diagram or the Tonnetz. This is both a strength and a limitation: it is a strength, inasmuch as it allows *bach* to be a general-purpose, highly adaptable tool; it is a limitation, inasmuch as it limits the scope of *bach* as a toolbox for experimental, non-standard musical practices and research.

The *dada* library is meant to fill this gap, focusing on real-time, non-standard graphical user interfaces for computer-aided composition. Hence, most of *dada*'s modules are interactive user interface modules; nonetheless the library also features a small number of non-UI modules designed to complement the operation of some of the interfaces in the library.

The philosophy behind *dada* is profoundly different from the one which informed *bach* or *cage*: *dada* is to *bach* what a laboratory is to a library. Under the umbrella of non-standard, strictly two-dimensional graphic user interfaces, it is somehow heterogeneous by design. All of its components participate of a graphical, ludic, explorative approach to music; most of its components also refer to the fields of plane geometry, physical modeling or recreational mathematics.

The *dada* library is open-source[4]. A preliminary alpha version (*dada* 0.1) is planned to be released at the end of 2017; nonetheless, *dada* is by design an open box, and additional modules might be added in future releases.

Fig. 3.8 shows the collection of all *dada* modules at the time of writing. The modules included in the *dada* library can be roughly divided into three categories: tools for corpus-based composition, tools for physical or geometrical modelling of music, and tools to handle rule-based systems and games. All interfaces in *dada* share a palette of common messages. Differently from *bach*, all *dada* interfaces also share a palette of reserved keys for particular interface actions (such as 'z' for zooming, 's' for scaling, 'n' for changing the pitch of a note, and so on): keyboard letters are hence used as 'editing tool' switches.

[4] https://github.com/bachfamily/dada, www.bachproject.net/dada

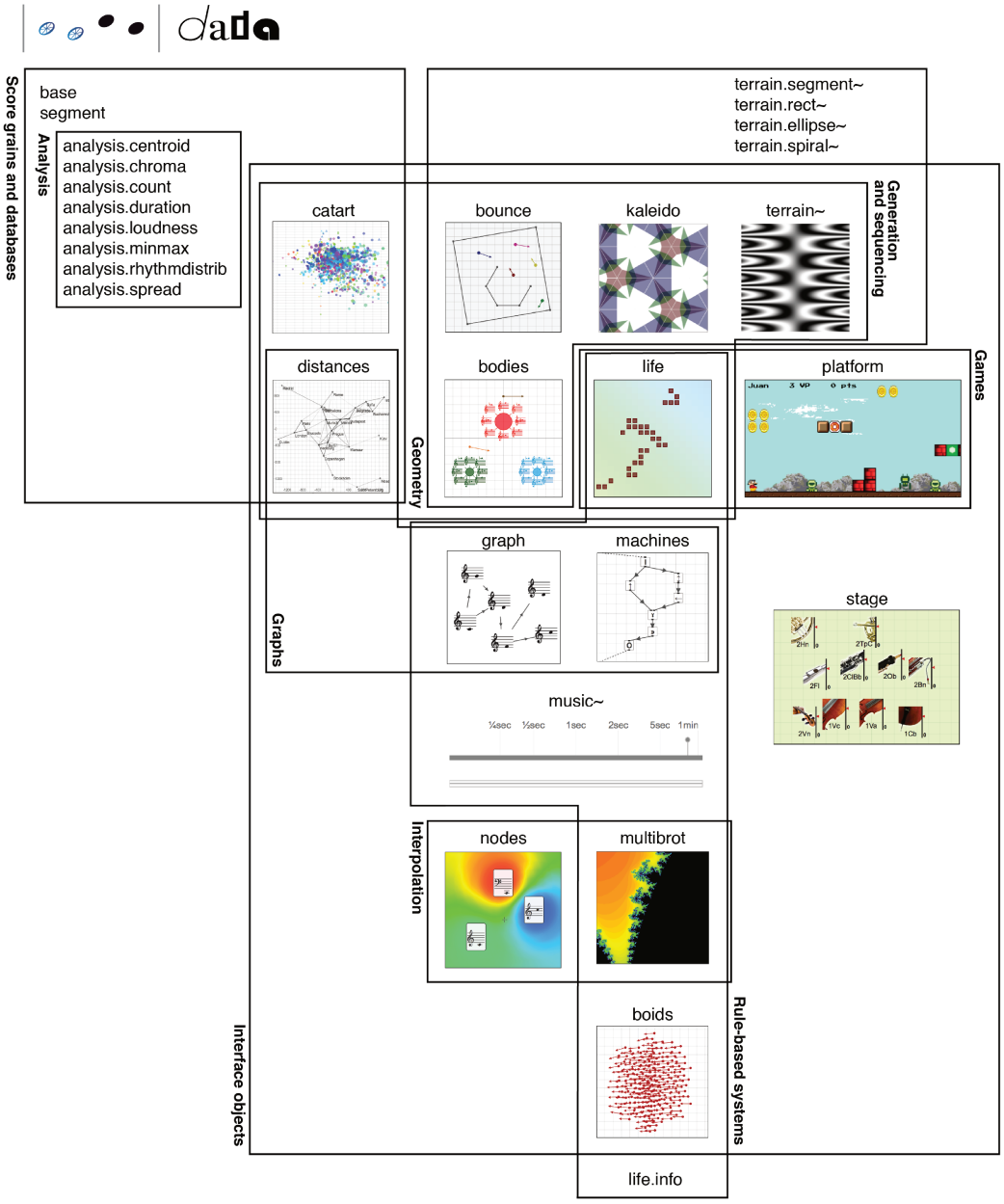


Figure 3.8: Overview of the dada *library (the* dada. *prefix has been omitted from all module names).*

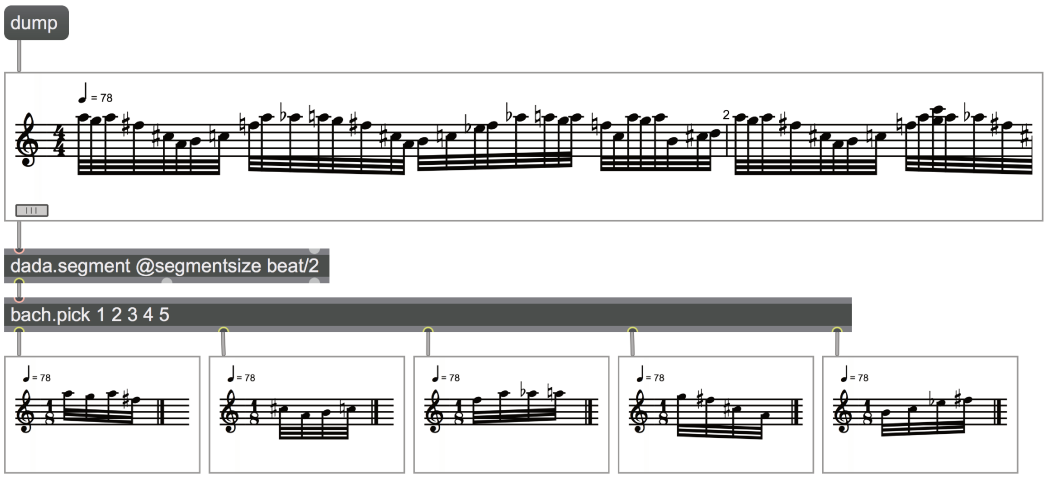


Figure 3.9: Segmentation of a bach.score *into grains having length equal to half of the beat (i.e. an eighth note). The first five grains are displayed in the bottom part of the patch.*

Most modules are designed to be easily used in combination with *bach.ezmidiplay*, to obtain a quick MIDI rendering of the musical outcome, or with *bach.transcribe*, to record the result of the process in symbolic form in a *bach.roll*.

3.3.2 Tools for corpus-based composition

The tools in this category are designed to handle the usage scenarios as illustrated in chapter 2.

The overall system relies on four different modules: *dada.segment*, performing segmentation and feature extraction; *dada.base*, implementing the actual database engine; *dada.catart* and *dada.distances*, two-dimensional graphic user interfaces capable of organizing and interacting with the extracted grains.

3.3.2.1 Segmentation

The *dada.segment* module performs the segmentation of a score, contained either in a *bach.roll* (as proportionally notated musical data) or *bach.score* (as classically notated musical data), in one of the following manners:

- Via markers: each marker in the original *bach.roll* is considered as a cutting point at which the score is sliced. All the slices (grains) are then collected.
- Via equations: a single value (in milliseconds for *bach.roll*, or as a fraction of the bar or beat duration, for *bach.score*) or more generally an equation can be used to establish the size of each grain. In *bach.roll* this equation can take as variable the grain onset, and is especially useful when segmentation needs to be performed roughly independently from the musical content itself. In *bach.score*, voices are pre-segmented into chunks of measures (according to a pattern established via the 'presegment' attribute), and each chunk is in turn

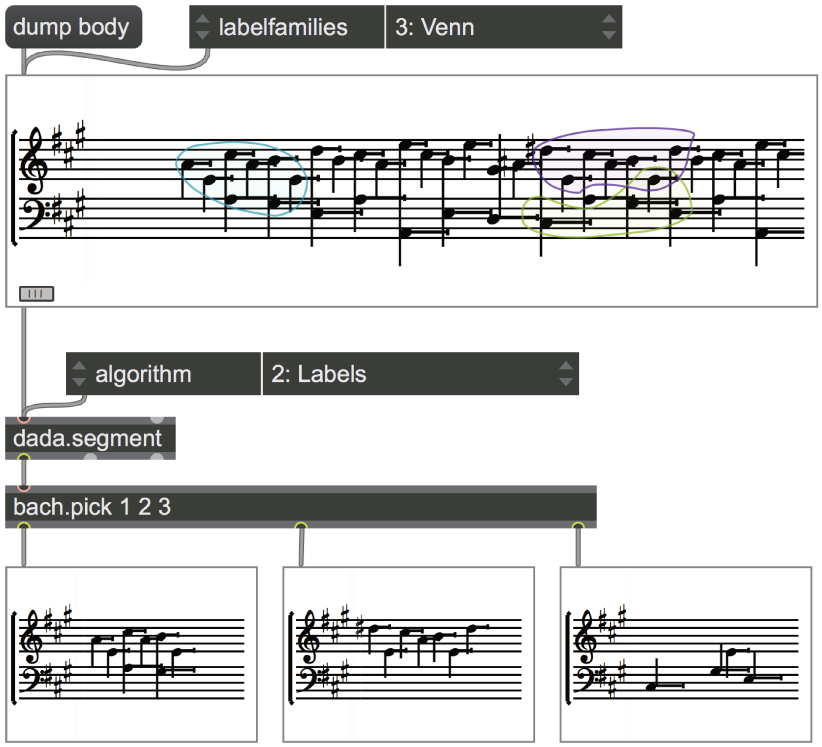


Figure 3.10: Segmentation of a bach.roll *according to label families. Labeled items are automatically enclosed in colored contours in the original* bach.roll*. Notice how families can overlap (in the example above, one note is labeled twice, and hence assigned to two families at the same time). The first three grains (corresponding to the first three label families) are displayed in the bottom part of the patch.*

segmented into grains whose duration is determined by the aforementioned equation—possibly having as variables the measure number, the measure division (beat), and the measure overall symbolic duration (see for instance fig. 3.9).

- Via label families: differently from sound files, scores easily allow non-vertical segmentations, where only a portion of the musical content happening in a given time span is accounted for (see fig. 3.10). If labels are assigned to notes or chords in the original score, a grain is created for each label, containing all the elements tagged with such label.

The segmentation can be performed with overlapping windows, both on proportional and classically notated scores, and standard windowing techniques can be applied to MIDI velocities, if desired (see for instance fig. 4.3).

3.3.2.2 Analysis

Grain analysis is performed during the segmentation process. On one side, *dada.segment* is capable of adding some straightforward metadata to the segmented

module name	*description*
dada.analysis.duration	Get total duration (in milliseconds)
dada.analysis.count	Get item count (for notes, chords, measures, voices)
dada.analysis.minmax	Get minimum and maximum for a given parameter (cents, onsets, durations, MIDI velocities)
dada.analysis.centroid	Get average pitch (also account for note durations)
dada.analysis.spread	Get standard deviation for pitches (also account for note durations)
dada.analysis.loudness	Get average MIDI velocity (also account for note durations and rests)
dada.analysis.chroma	Get chroma vector (also account for note durations and MIDI velocities)
dada.analysis.rhythmdistrib	Get rhythmic energy distribution, in equally divided temporal bins (also account for note MIDI velocities)

Table 3.1: Score analysis modules in dada.

grains, such as their duration, onset, index, label (if segmentation is carried out via label families) and notation object type (either 'roll' for *bach.roll* or 'score' for *bach.score*); in case the grain comes from a *bach.score*, tempo, beat phase, symbolic duration and bar number can also be added.

On the other hand, *dada.segment* allows the definition of custom features via a loopback patching configuration named 'lambda loop' [Einbond et al., 2014]: grains to be analyzed are output one by one from the rightmost (lambda) outlet, preceded by the custom feature name; the user should provide a subpatch to extract the requested feature, and then plug the result back into *dada.segment*'s rightmost (lambda) inlet. Feature names, defined in an attribute, are hence empty skeletons which will be 'filled' by the analysis implementation, via patching. This programming pattern is widely used throughout the *bach* library (one can compare the described mechanism, for instance, with *bach.constraints*'s way of implementing custom constraints [Agostini and Ghisi, 2015]), and allows users to implement virtually any type of analysis on the incoming data.

Some ready-to-use abstractions are provided for quick prototyping (see fig. 3.12), a comprehensive list of the currently provided modules is found in Table 3.1. Terminologies are mostly borrowed from the audio domain, even if they are applied to symbolic data; hence *dada.analysis.centroid* will output an average pitch, *dada.analysis.spread* will output the standard deviation of the pitches, *dada.analysis.loudness* will output the average normalized MIDI velocity, and so on. The reason behind this choice is to underline the duality between this symbolic framework and the digital signal processing approach. Moreover, since analysis

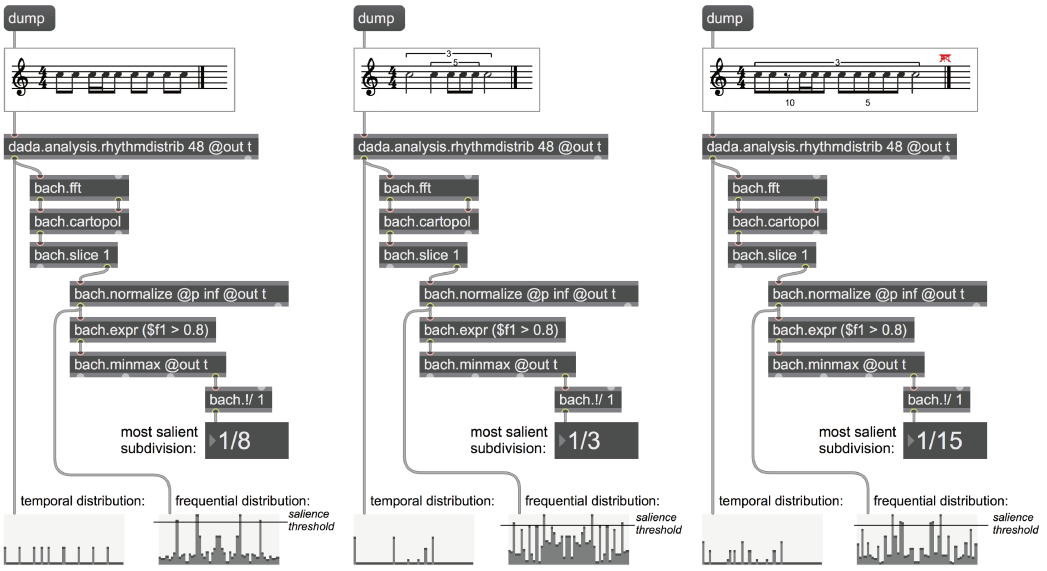


Figure 3.11: Computing most salient subdivisions via dada.analysis.rhythmdistrib *and FFT. The* dada.analysis.rhythmdistrib *module quantizes the temporal energy inside 48 temporal bins;* bach.fft *converts them into their frequential (or spectral) representation, whose amplitudes are retained and normalized so that the maximum will be always 1. Each bin in the spectral representation accounts for a specific regularity in the temporal data. The first amplitude above a given 'salience threshold' (in our case, 0.8) will hence correspond to the most salient subdivision.*

modules are standard Max patchers, it is extremely easy for users to inspect and adapt them to different behaviors. Analyzed features are collected for each grain, and output as metadata from the middle outlet of *dada.segment*. Analysis modules can also be used outside this database creation scenario; as a simple example, see fig. 3.11, where a 'most salient subdivision' is computed for some scores starting from the *dada.analysis.rhythmdistrib* module, via FFT.

3.3.2.3 Database

Once the score grains have been produced and analyzed, they are stored in a SQLite database, whose engine is implemented by the *dada.base* object. Data coming from *dada.segment* are properly formatted and fed to *dada.base*, on which standard SQLite queries can be performed (see figure 3.12).

Some higher-level messages are provided to perform basic operation and to handle distance tables (i.e. tables containing distances between elements in another table, useful, for instance, in conjunction with the *dada.distances* module, as explained below). Databases can be saved to disk and loaded from disk.

3.3.2.4 Interfaces

The two interfaces for *dada.base* are *dada.catart* and *dada.distances*.

The *dada.catart* module provides a Cartesian two-dimensional graphic interface for the database content. Its name is an explicit acknowledgment to the piece of

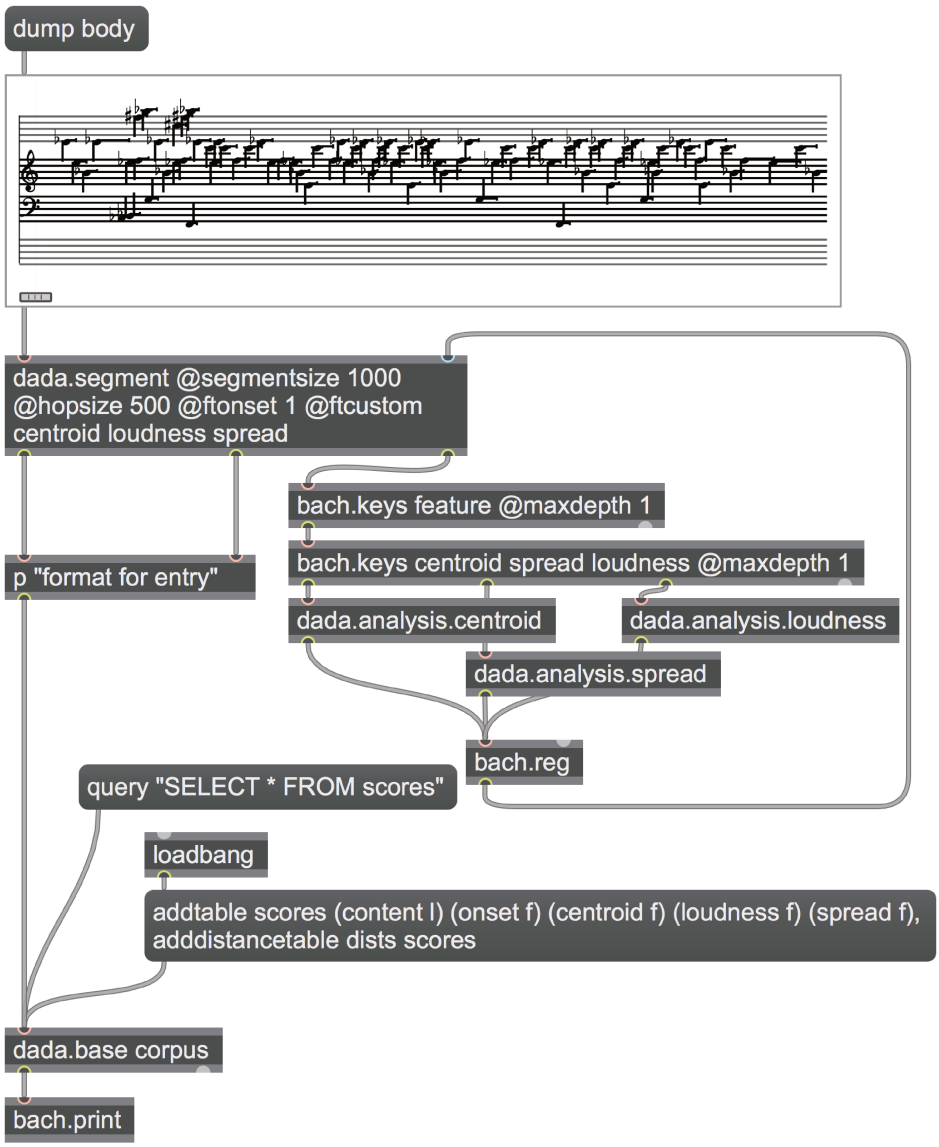


Figure 3.12: Example of segmentation using dada. *When the patch opens, a table named 'scores' is created in the database named 'corpus', collecting all the grains. This table has five columns: the content of the grain (a* bach *Lisp-like linked list representing the score), the onset the grain originally had, its centroid, loudness and spread (all floating point numbers). When the 'dump body' message is clicked, the score contained in the* bach.roll *is segmented and analyzed by centroid, loudness and spread (respectively computed via the* dada.analysis.centroid, dada.analysis.spread *and* dada.analysis.loudness *modules inside the lambda loop). The database is then filled, and standard SQLite queries can be performed on it.*

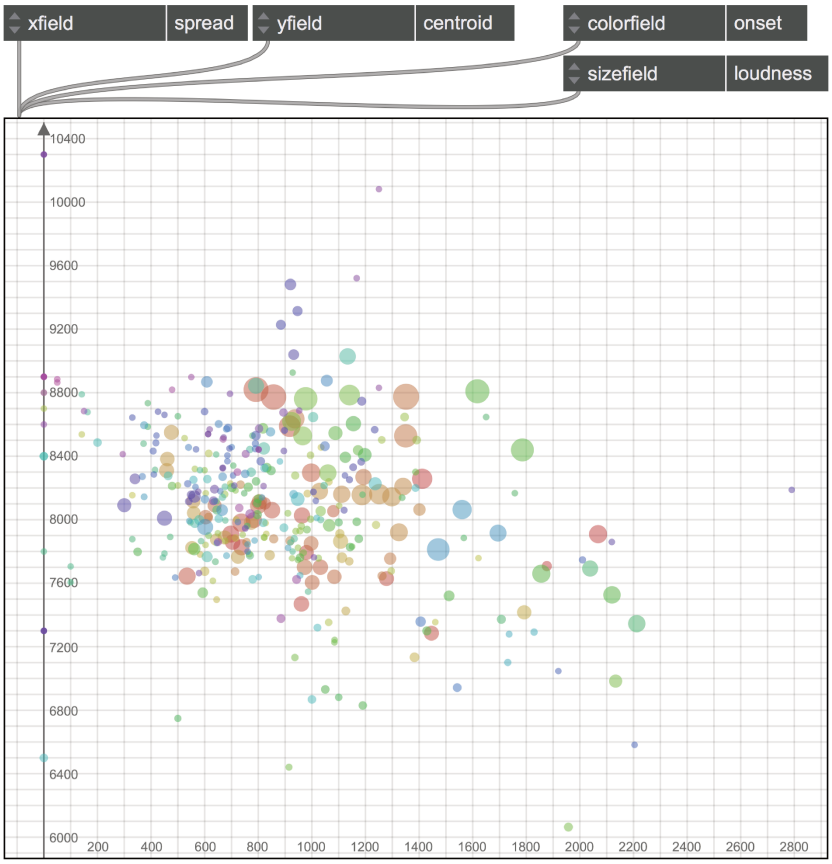


Figure 3.13: The dada.catart *object displaying the database built in figure 3.12. Each element of the database (grain) is represented by a circle. On the horizontal axis grains are sorted according to the spread, while on the vertical axis grains are organized according to their centroid. The colors scale is mapped on the grain onsets, while the circle size represents the grain loudness.*

software which inspired it. Grains are by default represented by small circles in a two dimensional plane. Two features can be assigned to the horizontal and vertical axis respectively; two more features can be mapped on the color and size of the circles. Finally, one additional integer valued feature can be mapped on the grain shape (circle, triangle, square, pentagon, and so forth), adding up to a total number of five features being displayable at once (see fig. 3.13).

The *dada.distances* module provides a distance-based representation of the database content. Points are the entries of a table, characterized via their mutual distances, contained in a different table. They are represented in a two-dimensional plane via the multidimensional scaling algorithm provided by Wang and Boyer [2013]. Edges are drawn only if the corresponding distance is below a certain threshold (see fig. 3.14). The resulting graph is navigable in a Markov-chain fashion, where distances are interpreted as inverse probabilities. As for *dada.catart*, features can be mapped to colors, sizes and shapes.

Both in *dada.catart* and in *dada.distances* each grain is associated with a 'con-

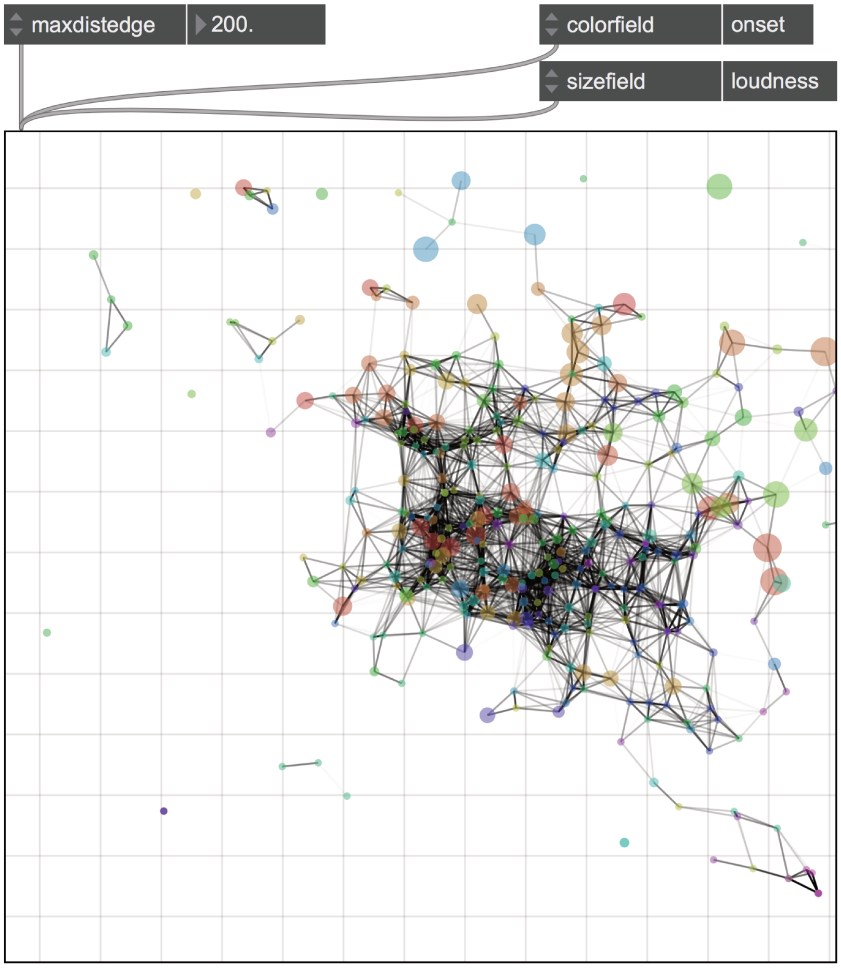


Figure 3.14: The dada.distances *object displaying a portion of the database built in figure 3.12. As for the* dada.catart *case (fig. 3.13), each element is represented by a circle. Grains are only positioned only according to a certain defined distance function (in this case, the distance of their centroids, spreads and loudnesses, as tridimensional vectors), the positioning in the Cartesian plane is carried out via the multidimensional scaling algorithm provided by Wang and Boyer [2013].*

tent' field, which is output either on mouse hovering or on mouse clicking. The content is usually assigned to the *bach llll* representing the score. The sequencing can also be beat-synchronous, provided that a tempo and a beat phase fields are assigned: in this case the sequencing of each grain is postponed in order for it to align with the following beat, according to the current tempo (obtained from the previously played grains).

A *knn* message allows to retrieve the k nearest samples for any given (x, y) position. A system of messages inspired by turtle graphics is also implemented, in order to be able to move programmatically across the grains: the *setturtle* message sets the turtle on the nearest grain with respect to a given (x, y) position; then the *turtle* message moves the turtle of some $(\Delta x, \Delta y)$, choosing the nearest grain with respect to the new position (disregarding the original grain). The turtle is always

displayed in *dada.catart* with an hexagon.

The database elements can be sieved by setting a *where* attribute, implementing a standard SQLite 'WHERE' clause. The vast majority of the display features can be customized, such as colors, text fonts, zoom and so on. In combination with standard patching techniques, these features also allow the real-time display, sequencing and recording of grains (see section 3.3.2.5 for an example).

3.3.2.5 Usage examples

The following paragraphs provide a few usage examples for the previously described system. Some of them were implemented for my own compositions; others were simple exploratory tools which did not lead to any musical piece as outcome.

Corpus-based score and audio montages. The usage scenario which shaped the creation of most modules is corpus-based montage, both in symbolic and audio form. The contents of a *bach.roll* or a *bach.score* object, or the audio contained in a file, are split into grains, according to a variety of highly customizable criteria and approaches. Then, the grains are analyzed, collected in a SQLite database and represented in a two-dimensional graphical visual interface according to a choice of descriptors attached to each grain: in this way, 'similar' grains (according to pairs of descriptors) are placed near each other in the graphical interface. The user can subsequently build a new score, constituted of a montage of the grains, by navigating the interface via mouse or message interaction (see figure 3.13). This system was used to organize and concatenate scores and samples for the last event of *I mille fuochi dell'universo* (see section 2.8). Similar tools are used to navigate through generated audio for *La fabrique des monstres*, as explained in section 3.4.4.

An interactive tonal centroid palette. The patch displayed in figure 3.16 segments (in grains of 1 second each) and then analyzes the first eight Lieder from Schubert's *Winterreise*. During the analysis process we take advantage of the tonal centroid transform proposed by Harte et al. [2006], and implemented in the *cage* library. The horizontal axis displays the phase of the tonal centroid with respect to the plane of fifths, while the vertical axis displays the phase referred to the plane of minor thirds (both range from -180 to 180 degrees). The analysis subpatch computing the phase of the projection of the tonal centroid on the plane of fifths is displayed in fig. 3.15 (the one for minor thirds is analogous). Both colors and shapes are mapped on the Lied number.

One can use this representation as a sort of 'interactive tonal centroid palette': each vertical line refers to a note in the circle of fifths, each horizontal line refers to an augmented chord in the circle of minor thirds. If we focus especially on the horizontal axis, we notice for instance that red circles (belonging to the first Lied, *Gute Nacht*, in D minor) are mostly scattered around the vertical line referring to the D, or that orange triangles (belonging to the second Lied, *Die Wetterfahne*, in A minor) are mostly scattered in the region around A.

A record mechanism is implemented, and the collected notes are displayed at the bottom of the image. The score can then be saved, quantized or exported.

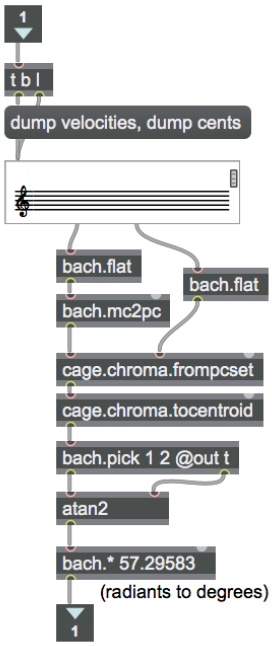


Figure 3.15: The subpatch computing the phase of the projection of the tonal centroid of a bach.roll *grain on the plain of fifths. All the pitches, converted into pitch classes, are weighted with their own MIDI velocity and gathered in a chroma vector, whose tonal centroid is computed via the algorithm proposed by Harte et al. [2006]. The first two components of the tonal centroid (referring to the plane of fifths) are picked, and the angle formed by the vector is computed.*

Path between chords. Starting from a certain number of base chords, a dataset of chords is generated by iterated application of simple operations (addition of a note, deletion of a note, pitch shift of a note within a certain range); the obtained dataset is then sieved with respect to some basic conditions (e.g., absence of octaves). A chord distance is then computed, based on the formula

$$\mathrm{dist}(A,B) = \frac{1}{12}\left(\max(|A|,|B|) - |A \cap B|\right) + \left(\max(|\mathrm{pc}(A)|,|\mathrm{pc}(B)|) - |\mathrm{pc}(A) \cap \mathrm{pc}(B)|\right)$$

where A and B are two sets of pitches, and 'pc' is a function mapping each pitch (and by extension each set) to its corresponding pitch class. This function is designed to work as a sort of 'edit distance' between the sets, yielding 0 if and only if $A = B$, and penalizing modification of pitches different from octave jumps. Indeed, the left term adds a $1/12$ penalty for each note belonging to one set but not to the other, while the right term adds a full penalty for each pitch class present in one set but not in the other.

Chords are hence represented in a *dada.distances* interface, and a Markov path between them, starting from the base chord, is generated (transition probabilities

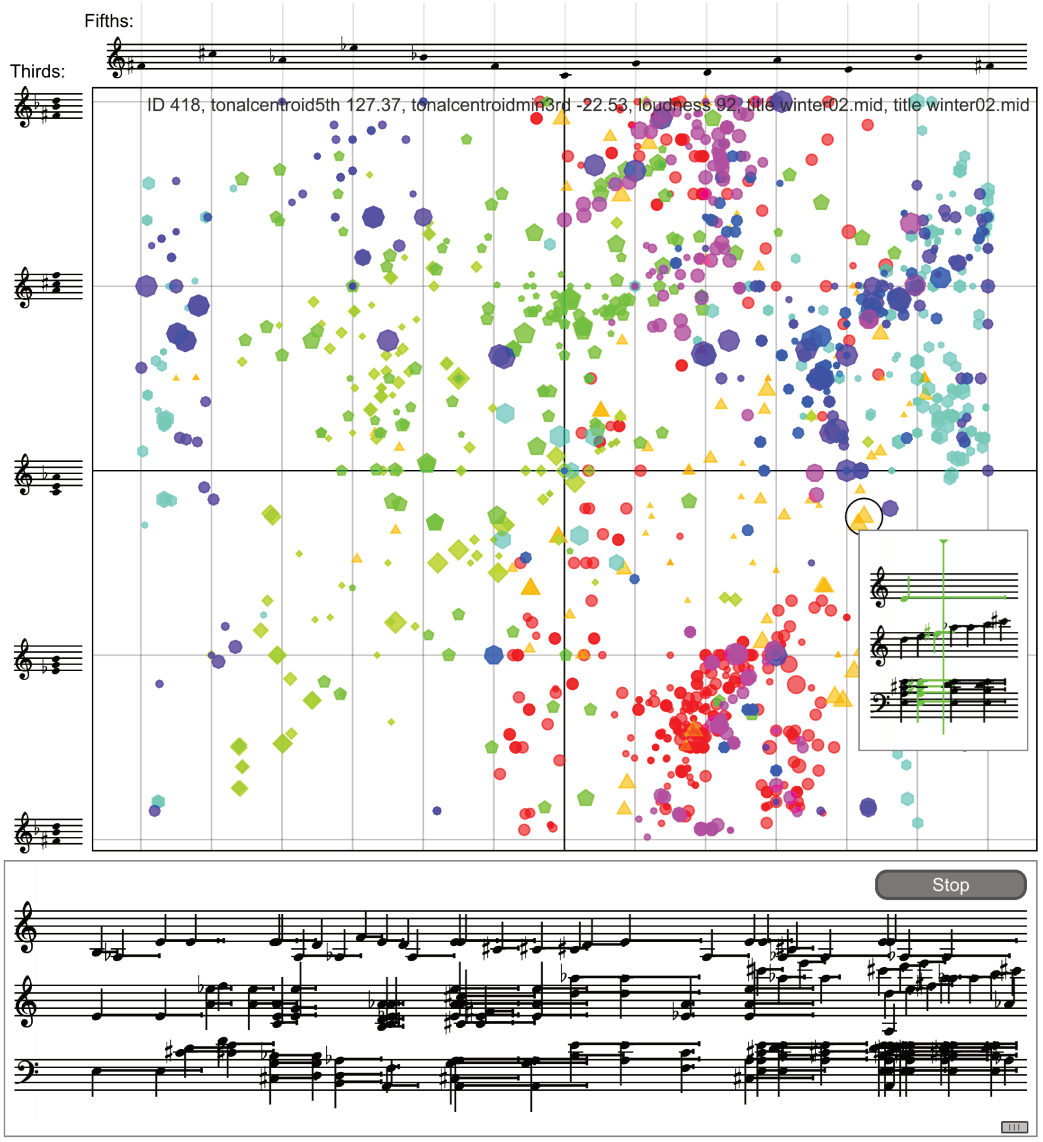


Figure 3.16: A patch displaying the database build from the first eight Lieder of Schubert's Winterreise*, organized by tonal centroids (the phase of the projection on the plane of fifths is on horizontal axis, the phase of the projection on the plane of minor thirds is on the vertical axis). Both colors and shapes identify the Lieder number (1 being the circle, 2 being the triangle, 3 being the square, and so on). When the recording mechanism is turned on, grains can be played via mouse hovering, and the bottommost* bach.roll *contains the recorded result.*

being proportional to the inverse of the distance). A set of features is extracted from each chord, such as its mean pitch ('centroid'), number of notes, extension, density, virtual fundamental, tonal centroid (see previous example), relative triadicity (accounting for how many major and minor triads are in the chord, normalized by the number of notes).A *dada.catart* object displays the same database with respect to a Cartesian choice of these features. The database can be sieved with respect to these parameters. Figure 3.17 shows the overall interface for the tool, which was designed to explore harmonic progressions in *I mille fuochi dell'universo*—and eventually abandoned, in favor of a constraint-based approach.

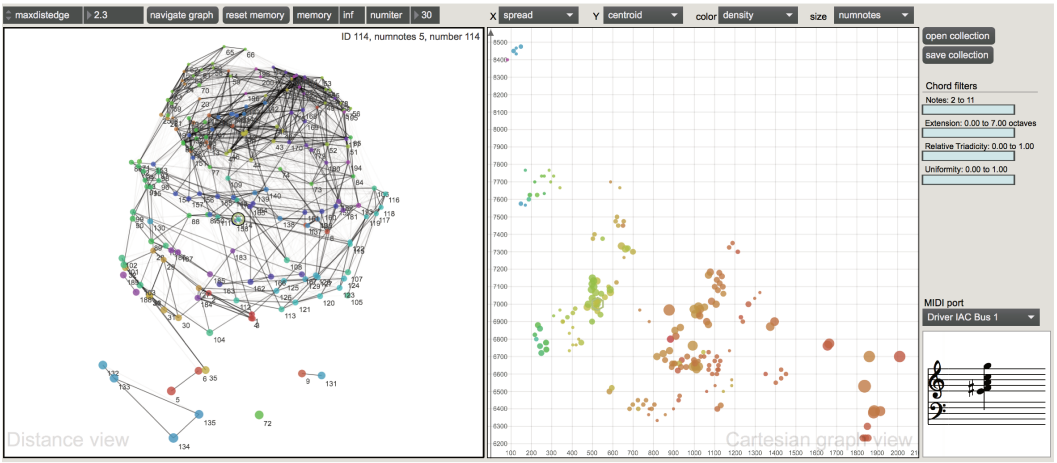


Figure 3.17: Interface for a chord exploration. Each point represents a chord: the left diagram shows chord connections according to a defined distance function (a sort of 'edit distance'), the right diagram shows chord positioning in a Cartesian feature space.

Rearranging beats. As a final example, consider figure 3.18, the first Bach invention (BWV 772) has been segmented by beat. The measure number is displayed on the horizontal axis; the position of the beginning of the grain inside the measure (phase) is displayed on the vertical axis. One can send *turtle* messages in order to navigate through the grains, so that one can read the complete score as it was (patch mechanism at top left corner of the image), or only read the last beats of each measure from last to first measure (top-middle part of the image), or move in random walks across the beats (top-right part of the image).

3.3.3 Tools for physical or geometrical modelling of music.

The interfaces in this group share the idea that objects in space can lead to music generation by means of geometry and motion.

3.3.3.1 Pinball-like bouncing

The *dada.bounce* module suggests a pinball-like scenario, where a certain number of balls move inside a space delimited by a user defined graph, called 'room'. The

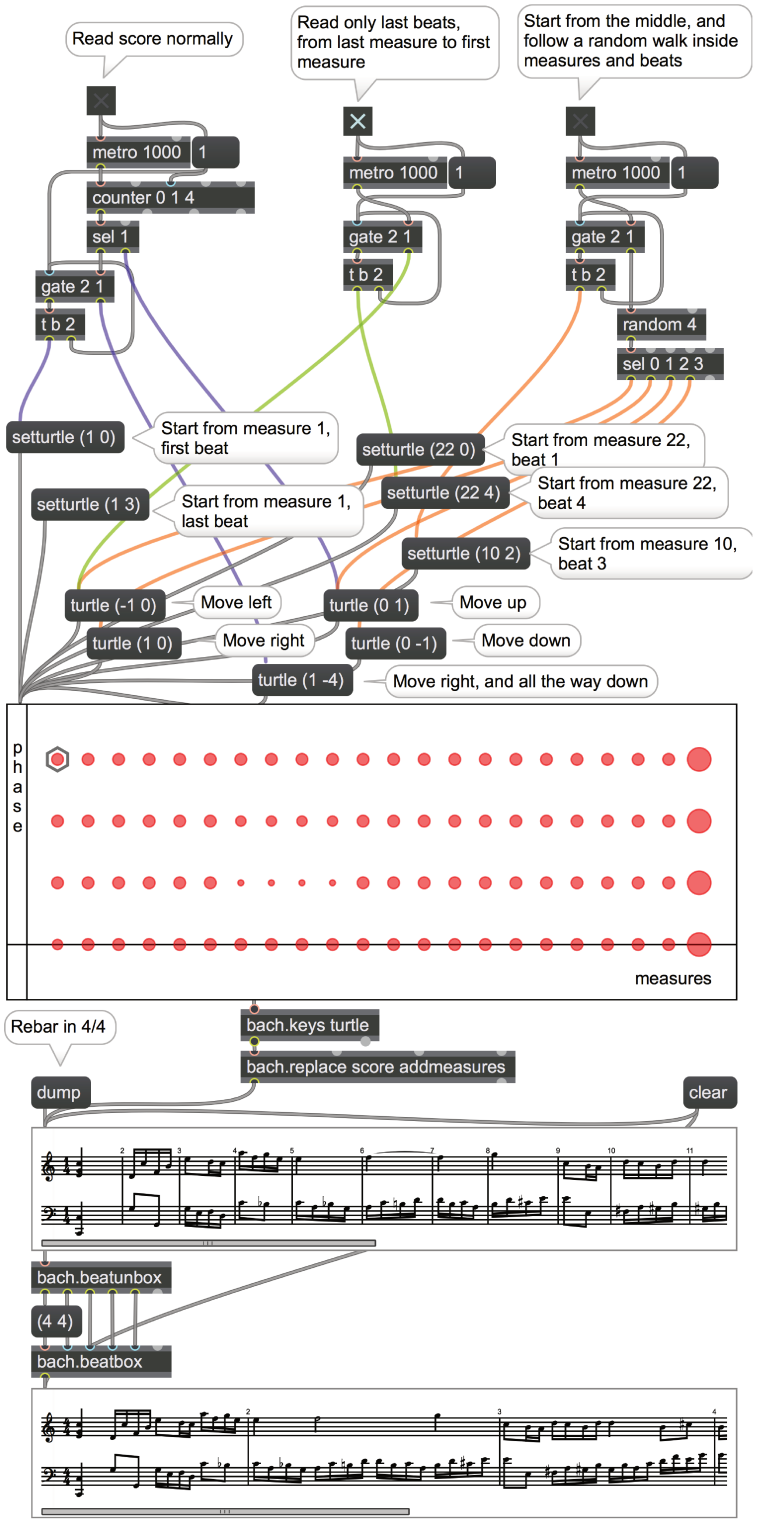


Figure 3.18: An example showing the manipulation of the first Bach invention (BWV 772), segmented by beat, and rearranged so to play and record the last beats of each measure (starting from last measure, and ending with first one). Notice how ties are preserved during the segmentation process (e.g., between measure 6 and 7) of the upper bach.score, *rebarred in measure 2 of the lower one.*

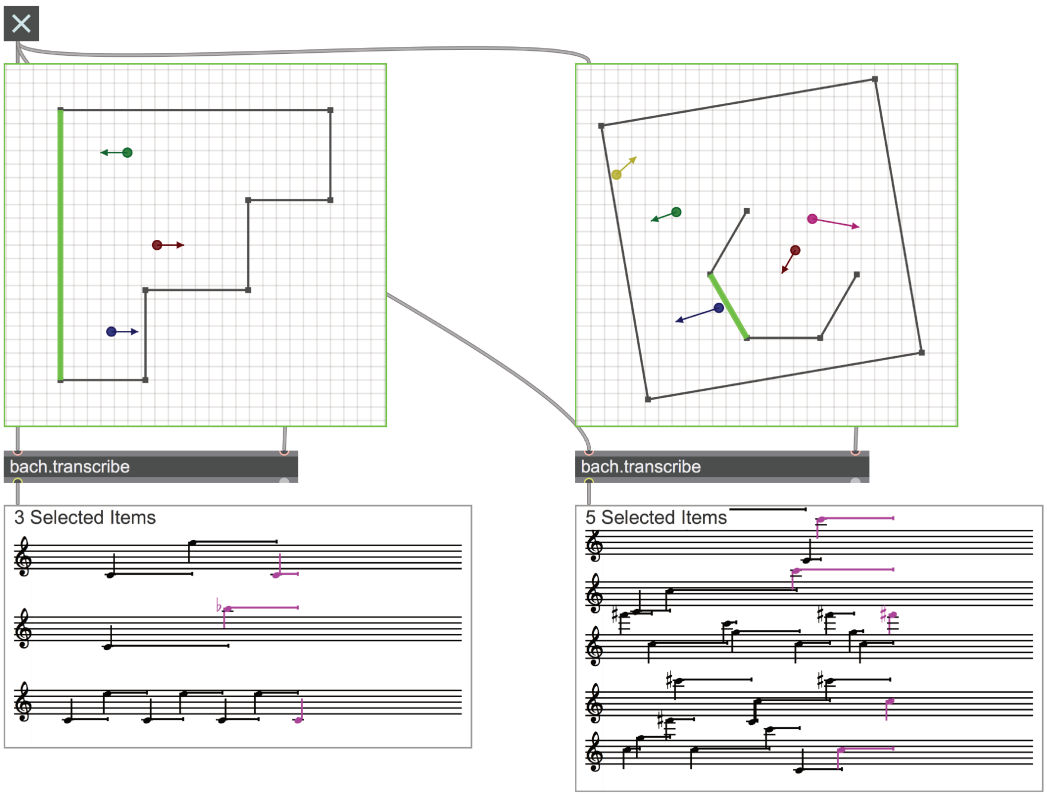


Figure 3.19: Two dada.bounce *objects producing respectively a polyrhythm (left) and a more complex pattern (right). Each edge is mapped on a note which can be played or recorded as soon as the the edge is hit.*

ball movement is uniform (constant speed[5], no gravity), except when a ball bounces off an edge. Each edge contains metadata either as a couple of MIDI pitch and velocity, or as a complex score; such metadata will be output whenever a ball hits the corresponding edge. Information about the collision (identifying the point, the edge and the ball) can be retrieved. Ball and room properties and metadata can be changed dynamically.

Simple room configurations may lead to loops or polyrhythmic patterns; more complex results are achievable by modifying the geometry of the room and the number of balls (see fig. 3.19), or by using feedback loops as programming patterns—e.g., by adding edges at each hit.

As a simple example of usage, consider bars 569 through 580 of *I mille fuochi dell'universo* (described in section 4.4), corresponding to its 995th 'event', where a pattern of string pizzicati is obtained as a retrogradation of the bouncing movement displayed in fig. 3.21. The output score is shown in fig. 3.20.

[5]In order to avoid confusion with MIDI velocities, the term 'speed' is used in this context also to refer to the velocity vector, and not just to its scalar intensity.

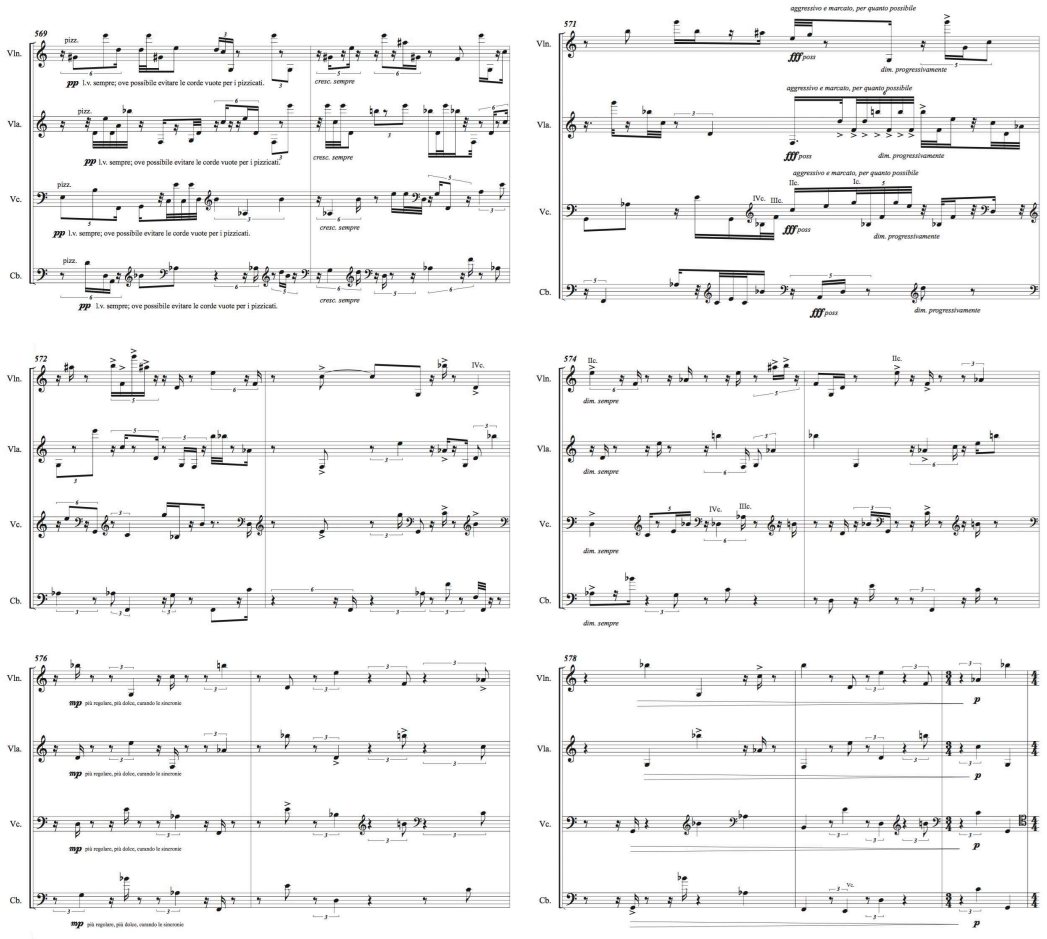


Figure 3.20: I mille fuochi dell'universo*, event 995, strings excerpt, bb.569-580, displaying the quantized version of the bouncing movements of fig. 3.21*

Excerpts

- *I mille fuochi dell'universo*, event 995, bb. 569-580 (ensemble only): `data.danieleghisi.com/phd/snd/IMFDU_ev995_ensemble.aif`|`mp3`. Original patch is displayed in fig. 3.21, string parts are displayed in fig. 3.20.
- *I mille fuochi dell'universo*, event 995, bb. 569-580 (with electronics): `data.danieleghisi.com/phd/snd/IMFDU_ev995.aif`|`mp3`.

3.3.3.2 Gravitation

A different paradigm is enforced by the *dada.bodies* module, modelling a two-dimensional universe with gravity, containing two types of objects: 'stars', fixed circles, from which a certain number of radii stand out, each representing a note (see fig. 3.22); and 'planets', which orbit around the stars, according to a customizable gravitational law, and trigger the playback of radial notes whenever they orbit 'close enough' to a star. The MIDI velocities of the notes are scaled according to

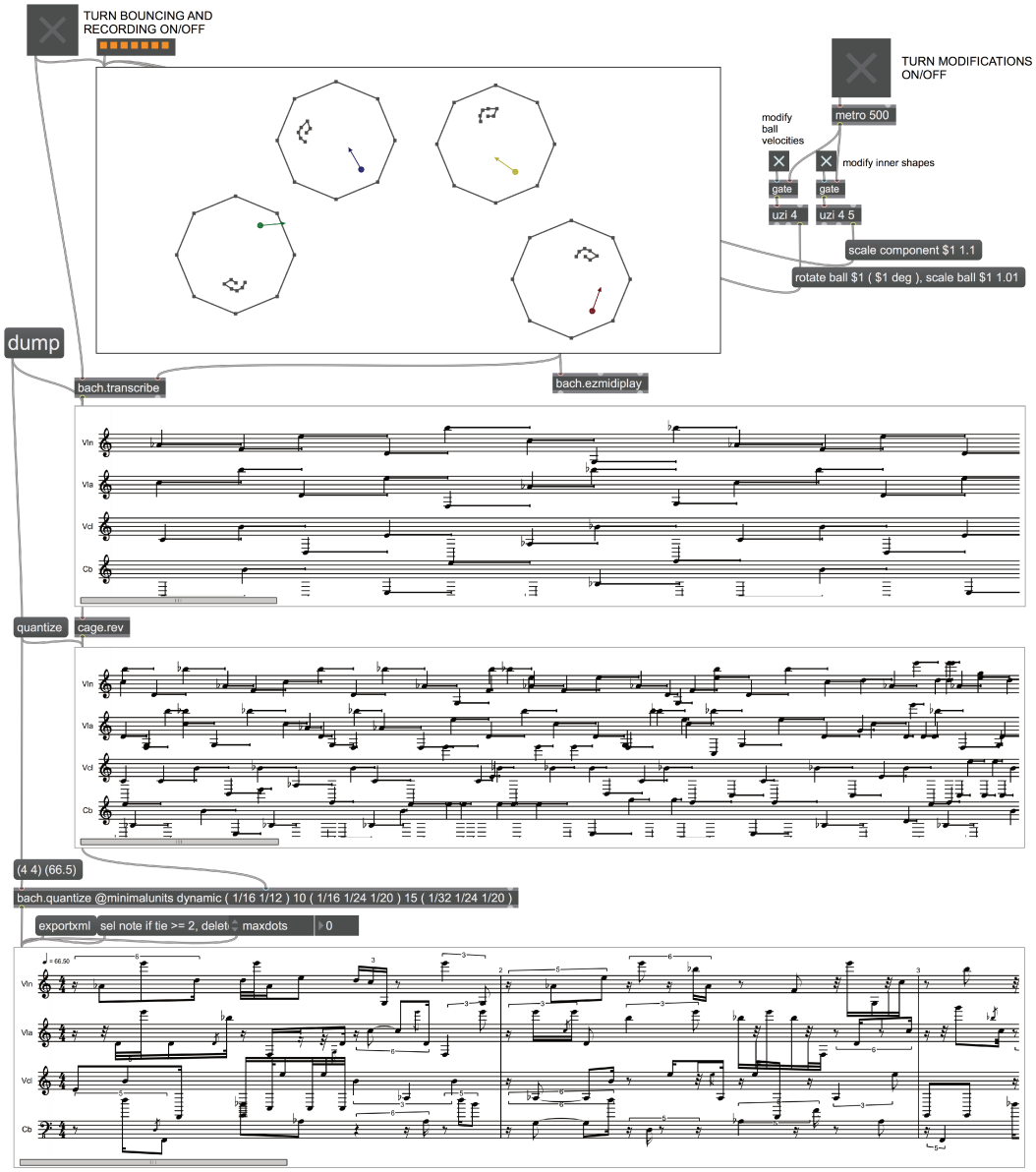


Figure 3.21: I mille fuochi dell'universo *(bb. 569-580): patch used to obtain, retrograde and quantize the bouncing movement originated from a* dada.bounce *object, displaying four balls (one for each string player), each confined in a similar octagon (but each octagon has edges triggering different notes). Each octagon contains inside itself a smaller shape, which, when the rightmost toggle is on, progressively grows in size, so that steady patterns gradually turn into a more chaotic motion. After the retrogradation, the opposite effect is obtained, the four players gradually synchronizing their pizzicati.*

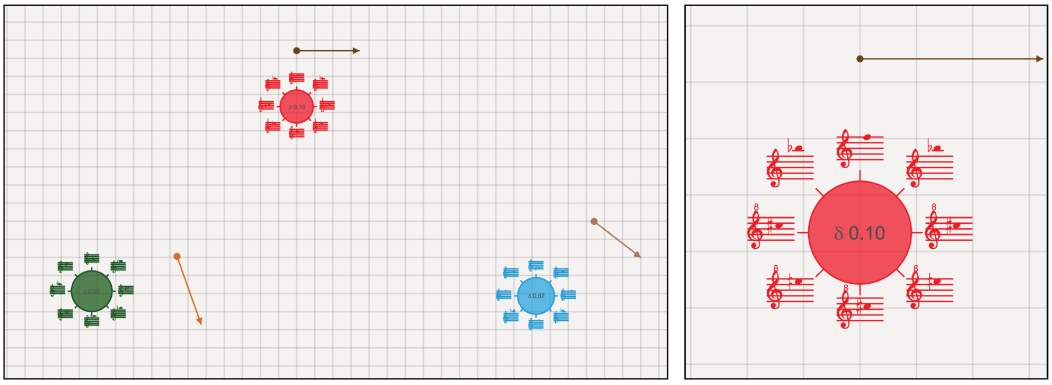


Figure 3.22: Configuration of dada.bodies *gradually distorting the loops of Gerard Grisey's* Vortex temporum*, used in the sixth episode of* Itaca *(also see fig. 3.24). At right: a zoomed version of one of the stars (corresponding to the flute's notes).*

the distances between planets and stars. As a metaphor, one could imagine 'stars' as being 'radial aeolian harps' played by the planets while circling around them.

This model is a convenient representation to handle continuous modification of loops. In a situation with a single star and a single planet, one could set the distances and speeds so that the planet motion around the star is circularly uniform (convenience methods are provided), resulting in a perfectly looping pattern. Modifying the planet position or speed, ever so slightly, results in a time warping operation on the loop. Adding more stars will trigger complex scenarios. Chaotic loops and attractor-like situations can be achieved via this system.

I had already used a crude prototype of *dada.bodies* when I was writing the sixth episode (*'Dei tempi'*) of the collection *Itaca* (2012), for groups of children, ensemble and electronics, which also had a pedagogical overtone; the starting point for the episode were the arpeggi of the flute, clarinet and piano, at the very beginning of Gerard Grisey's *Vortex temporum*. Each arpeggio was assigned to a star; a planet was positioned near each one of the stars in circular motion, so to obtain, initially, a simple reiteration of each arpeggio (hence reproducing the beginning of *Vortex temporum*). However, each planet being also subject to the gravitational pull of the other stars, the loops are gradually distorted (see fig. 3.24 and fig. 3.22). Episode four (*'Intermezzo'*) of the same collection was written using the same technique, with pitches taken from portions of Brahms's 'Intermezzi' (see fig. 3.23).

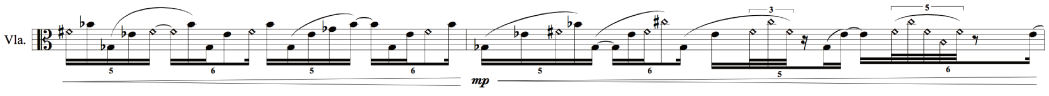


Figure 3.23: A viola pattern in fourth episode of Itaca *(b.126).*

Excerpts

- *Itaca*, section 6 (*'Dei tempi'*), b.171:
  `data.danieleghisi.com/phd/snd/Itaca_DeiTempi.aif|mp3`

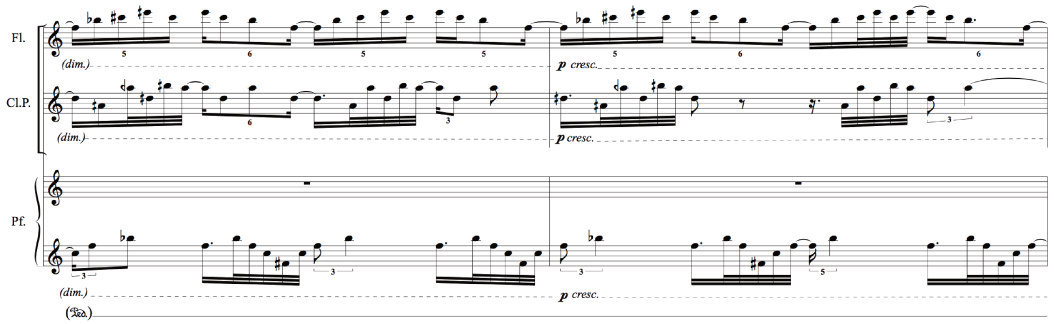


Figure 3.24: Temporal distortion, via dada.bodies*, of the initial loops of Gerard Grisey's* Vortex temporum*, used in* Itaca *(b.179).*

3.3.3.3 Kaleidoscopes

The *dada.kaleido* module traces the disposition and movement of a certain number of polygons in a kaleidoscope-like container. A certain number of shapes (polygon or ellipses) are positioned inside a 2- or 3-mirror chamber. The 2-mirror chamber has a couple of mirrors of equal length hinged at the origin, producing circular 'snowflake'-like patterns. The angle between the mirrors is set by the user via the *count* attribute, an integer number $n \geq 2$ relating to the mirror angle α in the following way: $\alpha = \pi/n$: for $n = 2$ mirrors are at right angles, for $n = 3$ they are two sides of an equilateral triangle, and so on (see fig. 3.25). For $n = 2$ and $n = 3$, a third mirror can be introduced [Gay, 1997, p. 210], closing the triangle formed by the other two, hence extending the tiling to the whole plane.

The shapes inside the chamber can be modified either via the interface or via a set of messages, such as 'move', 'rotate', 'scale' and 'shake'. A combination of rotation with a certain amount of shaking will result in an elementary yet effective modelling of a hand rotating the body of a kaleidoscope.

Users can assign test points on the plane, so that the object may report whenever any of the polygons, during a movement, hits a point (i.e. when the point enters a polygon or any of its kaleidoscopic reflections) or releases a point (i.e. when the point is no longer on the polygon, or on any of its kaleidoscopic reflections). Information about the distances between test points and polygons can also be retrieved, and can be used as control for symbolic or DSP processes.

I have extensively used these features while working on the music for *Orogenesis* (video by Boris Labbé, see fig. 3.26). Each shape is associated with a portion of audio file, which, like a vinyl, is only read, with variable speed, when a certain test point (the 'stylus') is positioned over the shape. The speed, as well as the gain, can be independently coupled with the distance between the test point and the polygons (fig. 3.27). My usage of the patch aimed to exaggerate the speed mapping, in order to enter the territory of digital noise and extreme aliasing.

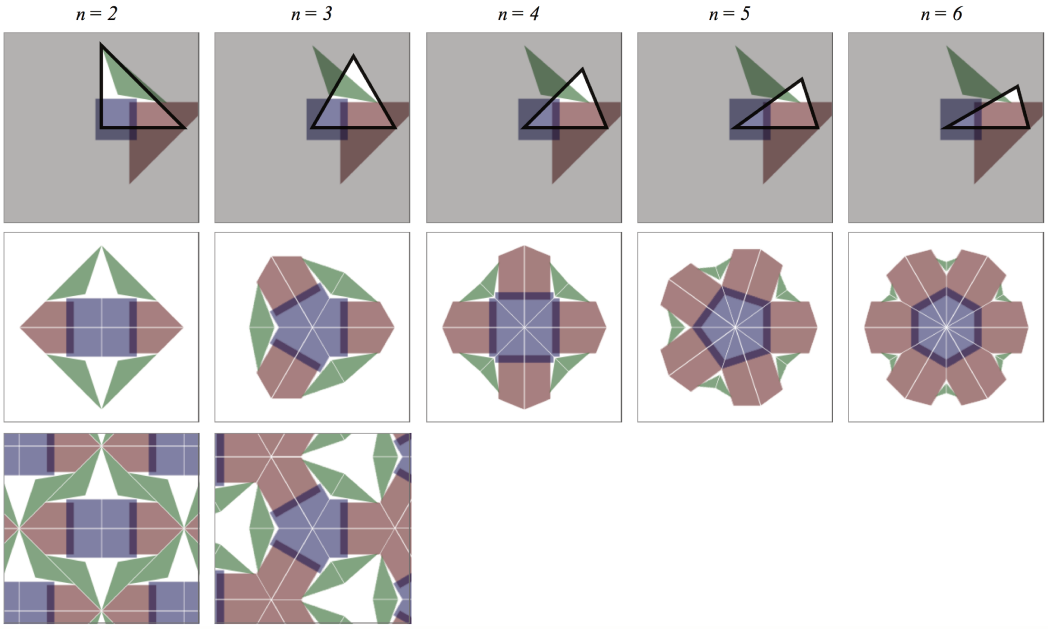


Figure 3.25: Same shapes reflected into different chambers of a dada.kaleido *object, for increasing values of the* count *attributes. Last row shows the 3-mirror version of the patterns, only available for* $n = 2$ *and* $n = 3$.

Excerpts
Some loops obtained for *Orogenesis* via the *dada.kaleido*-based interface:

- `data.danieleghisi.com/phd/snd/orog_loop1.aif|mp3`
- `data.danieleghisi.com/phd/snd/orog_loop2.aif|mp3`

3.3.3.4 Wave terrain synthesis

The *dada.terrain~* module implements wave terrain synthesis [Roads, 1996, pp. 163–167]: a function $z = f(x, y)$ yields the 'height' of the terrain for each point of a plane. Evaluating the function f on a specific path $p : x = x(t), y = y(t)$ produces a one-dimensional function $z = g(t) = f \circ p(t)$, which represents the wave terrain synthesis along the path p. Wave terrain synthesis essentially constitutes an extension of the ordinary wavetable synthesis to bidimensional lookup tables, and it is traditionally implemented in this way, in order to lower computational costs. A typical scenario is when the surface f is a direct product of sinusoids, such as $f(x, y) = sin(n\pi x)cos(m\pi y)$: in this case, by sampling the terrain on circular or elliptic orbits p, one obtains FM-like timbres.

In the *dada.terrain~* module, the function $f(x, y)$ is however not defined via a wave table, and is set via an explicit portion of C code compiled on-the-fly (see fig. 3.28). The wave terrain is displayed so that black corresponds to $z = -1$, white corresponds to $z = 1$, and 50% grey corresponds to $z = 0$.

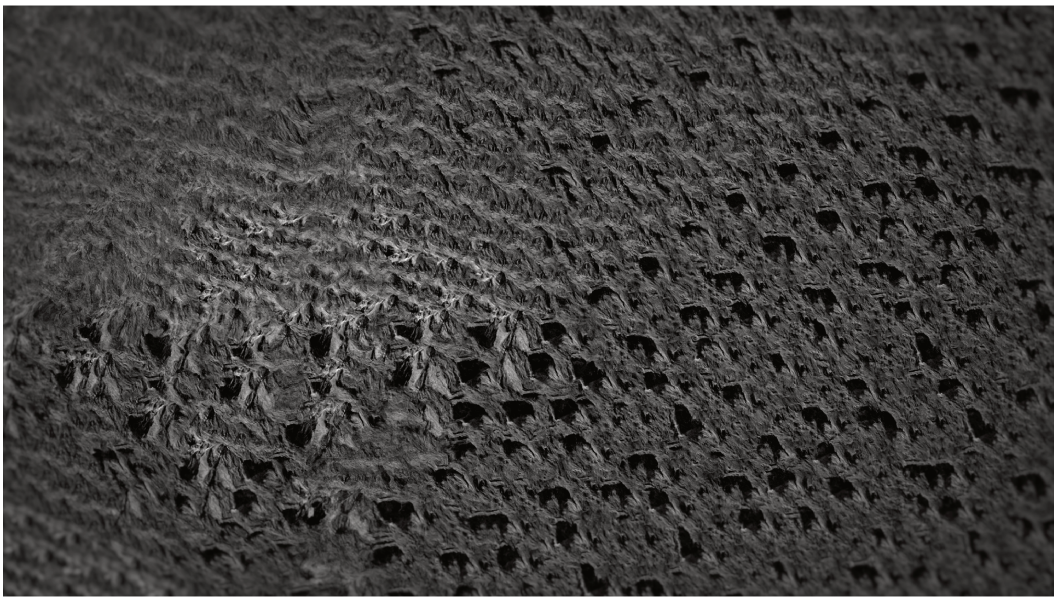


Figure 3.26: A video frame from Orogenesis.

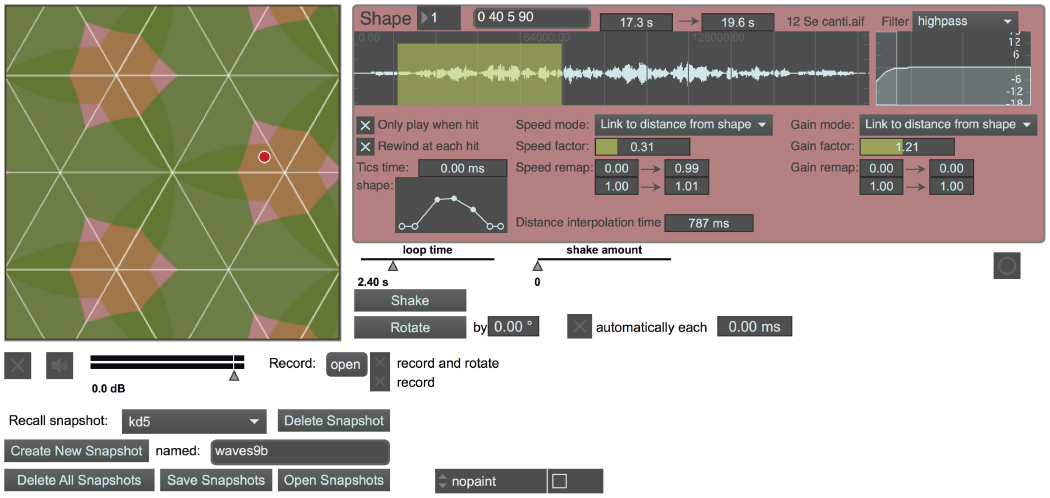


Figure 3.27: Main patch to create the sound material from Orogenesis.

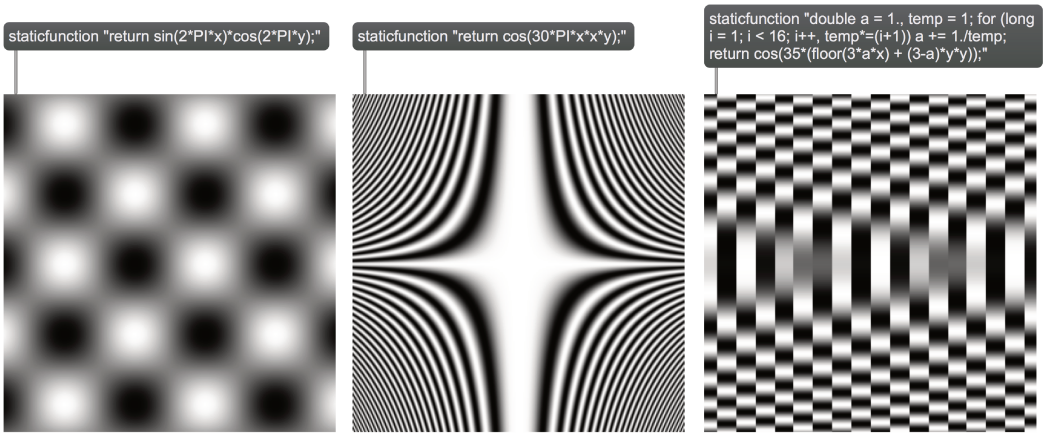


Figure 3.28: Example of the different wave terrains displayed in dada.terrain∼.

Four auxiliary modules help producing the paths p: *dada.terrain.segment∼*, *dada.terrain.rect∼*, *dada.terrain.ellipse∼* and *dada.terrain.spiral∼*, respectively generating cartesian or polar coordinates for segments, rectangular orbits, ellipsoidal orbits and spirals. These coordinates, produced at sample rate, are designed to be used as input for the wave terrain module (see fig. 3.29).

As an example of application, the *dada.terrain∼* module was occasionally used in *I mille fuochi dell'universo* in order to produce high aliased patterns, taking advantage of the inaccuracies of floating point numeric representation and arithmetics. A radial function is defined as terrain, as in fig. 3.29, and then sampled via radial segments starting at the origin. As the function outcome becomes more and more sensitive to accuracy in input position, which is the case for subfigure (d), glitchy effects appear. Patterns vary largely as the parameters of the segment (such as length or angle) are modified ever so slightly.

Excerpts

A collection of 2-second patterns obtained via the wave terrain synthesis of fig. 3.29 (d), by slightly varying one of the parameters of the driving segment (always starting at the origin):

- fixed angle $\alpha = 0.833$, varying segment length (from $L = 1$ to $L = 33$, steps by 1): `data.danieleghisi.com/phd/snd/wtglitch_len.aif|mp3`
- fixed length $L = 1000$, varying segment angle (from $\alpha = 0.833$ to $\alpha = 0.860$, steps by 0.001): `data.danieleghisi.com/phd/snd/wtglitch_angle.aif|mp3`

The *dada.terrain∼* module also supports the a 'buffer wheel' mode, where the terrain is not set via an explicit equation, but is obtained as result of a morphing between radially arranged buffers. Such morphing could be additive (result being a simple crossfade) or multiplicative; the equation for the contribution of each buffer can be set as a portion of C code compiled on-the-fly. As an example, consider fig. 3.30, where four instruments playing the same notes are arranged radially, and a spiral path samples the wave terrain, yielding a morphing between the four sounds.

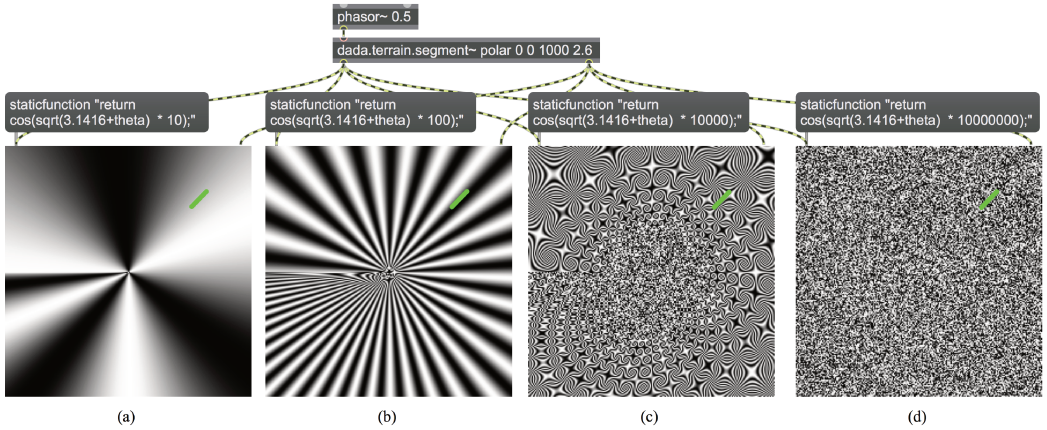


Figure 3.29: Example of wave terrains getting more and more sensitive to the accuracy in input position. The first three terrains provide no audio output (except for DC offsets); however, the floating point arithmetics inaccuracies result in glitches when the terrain (d) is sampled through radial lines.

Excerpts

- Morphing obtained by sampling the wave terrain with the four instruments of fig. 3.30 playing an A3, via a spiral path:
  `data.danieleghisi.com/phd/snd/wtbuffers.aif|mp3`

3.3.3.5 Miscellanea

For the sake of completeness, I shall also mention two other modules:

- the *dada.stage* module is an elementary editor and display for spatial disposition of elements, useful as a simple controller for multi-source two-dimensional panning and mixing.
- the *dada.nodes* module performs interpolations (via inverse distance weighting) among a set of notes disposed on the two-dimensional plane (fig. 3.31);

3.3.4 Rule-based systems, graphs, and music as a game

A certain number of tools explore the relationship between music, mathematics and games, and how this relationship ramifies towards combinatorics, algebra, topology and computer science (the link between canonical processes and topology being of course well known [Hofstadter, 1999], as well as the link between art and games [Huizinga, 1949], further interesting examples can be found in tools such as origami [Andersen, 2012] or juggling patterns [Johnson, 2012; Macauley et al., 2003]).

The modules in this family share two important ideas. The first one is that interesting emergent behaviors may arise from dynamical systems even when their agents adhere to sets of extremely simple rules; this is well known, for instance, in

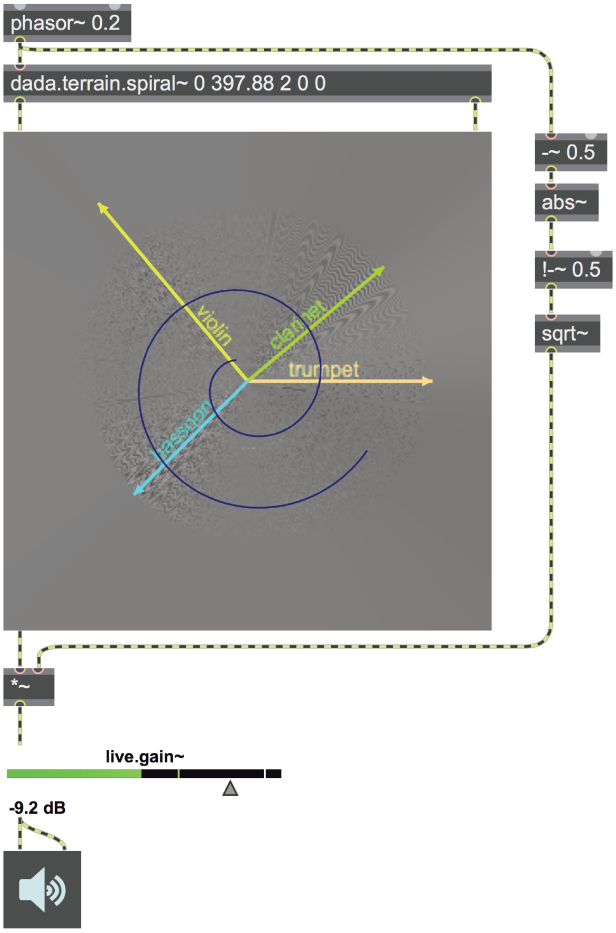


Figure 3.30: Four buffers, each containing an instrument playing an A3 in pianissimo, are arranged radially on a dada.terrain~. *The terrain is then sampled via a spiral path, yielding a morphing between the four sounds.*

the study of cellular automata, swarm intelligence and in Chaos Theory. The second idea is that digital scores may somehow indicate a form of 'gamification', i.e. the usage of game design elements in non-game scenarios. As Paul Turowski points out, there are fundamental similarities between musical scores (in any form) and digital games:

> Musical scores do not constitute the totality of a musical work; they necessarily require human performance, even if only conceptually. To perform music from a score is to actively take part in processes that reify the musical work. [...] Similarly, digital games are an action-based medium; though they exist objectively (at least partially) on some physical medium like a disc, the intended experience is the active interpretive performance of the player. [Turowski, 2016, p. 2]

After all, playing videogames often resolves in following a (graphically notated)

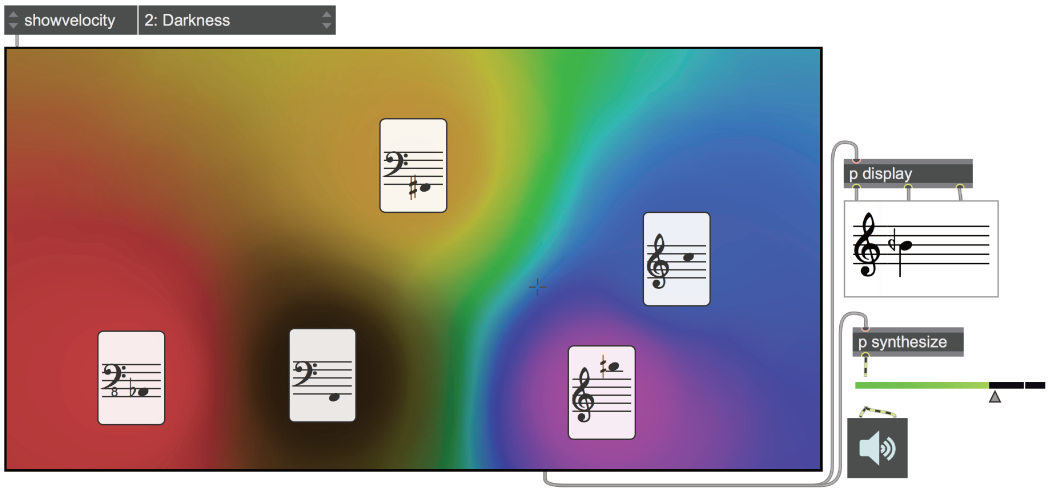


Figure 3.31: A Max patch with a dada.nodes *module, interpolating between a collection of notes, and rendering the result with a sinusoidal wave.*

rhythmical score, not dissimilar to a percussionist playing his or her own part in an orchestra: in both cases, the ability to stay within an acceptable level of precision will affect the outcome. If the score is hard-coded, gamers can progressively learn the precise timing for their actions; if the score is open, gamers are obliged to play *a prima vista*. This idea was, incidentally, at the basis of the original (and later abandoned) concept for *Any Road*: two live gamers follow a (hard-coded) digital score—an actual videogame—to trigger audiovisual events, with the goal to match their timing precisely, so that they fit with an orchestral layer (also see section 2.5.3). Turowski [2016] provides a more in-depth analysis of the relationships between notation and digital games.

There is one last striking connection between games and computer-aided composition, and it involves the concept of 'surprise': Jesse Schell underlines the importance of surprise in games, asserting that "surprise is a crucial part of all entertainment—it is at the root of humor, strategy, and problem solving. Our brains are hardwired to enjoy surprises" [Schell, 2014, p. 37]. Similarly, surprise represents in my workflow the most important feature of computer-aided composition: having outcomes that valuably differ from my expectations is a source of confrontation and inventiveness; in this respect, composing is also, partly, a game, in which, as a composer, I try to find stimuli for my own brain.

3.3.4.1 Cellular automata

The first module in this family is *dada.life*: a graphical interface for two-dimensional cellular automata, on square or triangular grids. Cellular automata are rule-based systems, consisting of a regular grid of cells, each in one of a finite number of states (such as 'alive' and 'dead'). A set of cells called 'neighborhood' is defined relative to each specific cell. Given a configuration of states, a new generation can be created

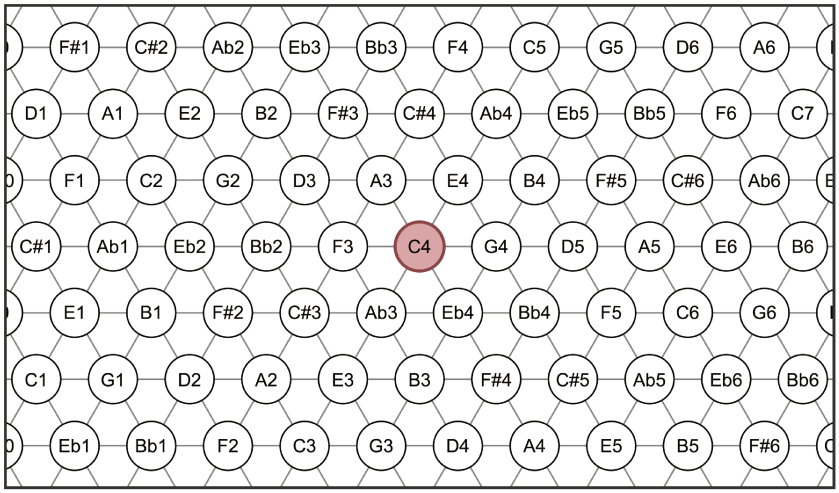


Figure 3.32: A Tonnetz is a two-dimensional pitch lattice with a central point, from which pitches are then obtained by adding two base intervals; in this case fifths (horizontal axis) and thirds (diagonal axes).

according to a given rule, usually a mathematical function, determining the new state of a cell depending on the current states of the cells in its neighborhood. The most famous cellular automaton is arguably Conway's *Game of Life*. Extremely complex patterns can arise in cellular automata, even from simple rules. One might identify some primary types of configurations:

Still lifes: configurations that do not evolve, i.e. configurations for which each new generation coincide with the previous one;

Oscillators: patterns that after a certain number of steps N end up coinciding with the original configuration (the minimal N satisfying this property being called the 'period');

Spaceships: oscillators that move in space, i.e. patterns that after N steps end up coinciding with a translation of the original configuration;

Gardens of Eden: configurations that cannot be reached from any other configuration.

A Max module handling two-dimensional cellular automata was already included in *cage*; nevertheless the *dada.life* object improves the approach, by making it interactive, more customizable and faster. The customization possibilities are not limited to colors and sizes: rules themselves can be defined either via attribute combinations (for simple scenarios similar to Conway's Game of Life) or via a portion of C code, compiled on-the-fly—a more agile approach than *cage.life*'s Max patchers.[6].

I have extensively experimented with *dada.life*'s automata, especially with triangular grids. This choice is motivated by my interest in the Tonnetz structure. A

[6]The fact that *cage.life* is an abstraction is consistent with the design of the whole *cage* project, as explained in section 3.2.4.

Tonnetz is a bidimensional lattice designed to represent neo-Riemannian transformations [Cohn, 1998]. The traditional Tonnetz is generated by repeatedly transposing a central pitch via perfect fifths (horizontal axis) and major thirds (diagonal axis, or vertical axis in square lattice representations); more generally, a Tonnetz can be generated by any choice of base intervals, or can even be modelled upon diatonic scales [Bigo et al., 2015]. Representing a Tonnetz via a triangular grid, as shown in fig. 3.32, is a clever choice, since it connects each note with all and only its closely related ones (the roots of its closely related keys).

One can imagine to use the triangular grid as basis for a two-states cellular automaton, having as cells the Tonnetz elements: cells can be 'on' (playing) or 'off' (silent). Studying the automaton patterns will result in musical sequences. Most importantly for me, studying the automaton oscillators will result in harmonic and melodic loops.

As an application example, consider *Come un lasciapassare*, a piece for orchestra and electronics, subdivided in five small movements. The middle movement is a *passacaglia* based on an harmonic loop determined by an oscillator in a two-dimensional cellular automata, as shown in fig. 3.33.

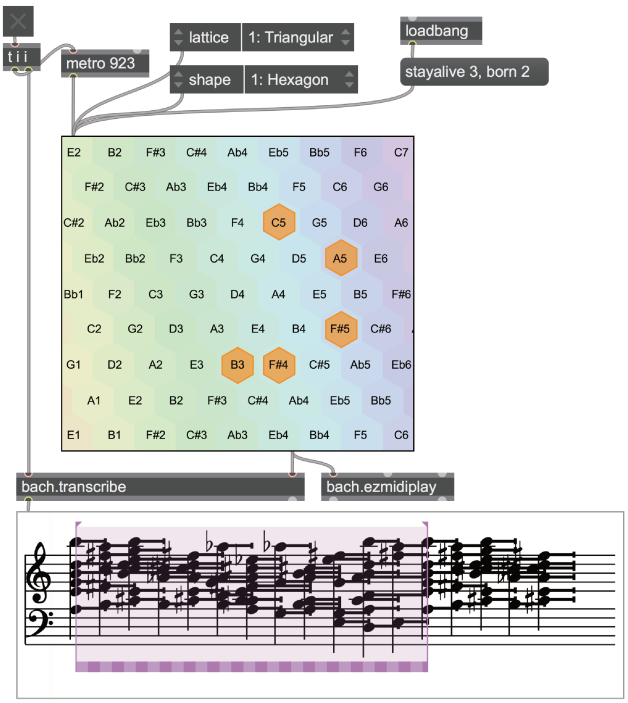


Figure 3.33: The harmonic cycle for the third movement in Come un lasciapassare*, as an oscillator of a two-dimensional cellular automata played on the Tonnetz.*

The harmonic loop is sustained by the vibraphone, and then transferred to a synthesizer towards the end of the movement. Two pianos, one within the orchestra and a recorded one, play the exact same loop respectively in rallentando (from four times to a quarter of the standard speed) and accelerando (from a quarter to four

times the standard speed), also removing some of the chords (the process is depicted in fig. 3.34).

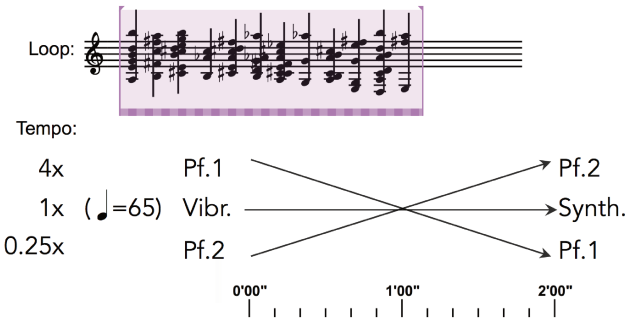


Figure 3.34: Scheme of the harmonic loop arrangement for the third movement of Come un lasciapassare.

The two pianos are widely panned, so that a stereophonic rallentando/accelerando effect is obtained, all across the movement, resulting in a strange palindrome form. To break the symmetry, three equally spaced interventions of a recorded voice recite fragments from Shakespeare's Sonnet No. 123 ('No, Time, thou shalt not boast that I do change'). Last intervention is truncated ('the pyra-'), and only completed at the very beginning of the piece ('-mids'), suggesting that the entire movement should be conceived as if to be played in a circular, looping fashion.

It should be remarked that several composers have shared and share my interest in the application of cellular automata to music; among these, I shall mention Mauro Lanza's *John Conway in Gondola* (where they are played on structures similar to step-sequencers), and Andrea Agostini's *Conway Boogie-Woogie*.

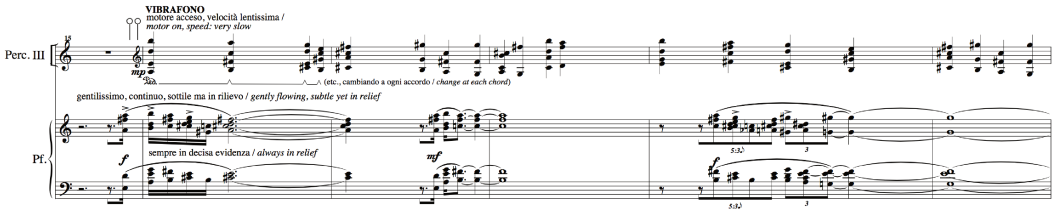


Figure 3.35: Beginning of the 3rd movement of Come un lasciapassare *(b.8): the Vibraphone sustains the harmonic loop; the piano plays the same loop in rallentando, starting at four times the Vibraphone speed, with some chords removed.*

Excerpts

- *Come un lasciapassare*, beginning of 3rd movement (b.98): `data.danieleghisi.com/phd/snd/CULP3.aif|mp3`.

3.3.4.2 Swarm intelligence

The *dada.boids* module investigates swarm intelligence models. The object contains a certain number of 'swarms' or 'flocks', each containing a certain number of 'birds'

or 'particles', singularly represented on the screen as points or arrows. The movement of each particle is dictated by a sequence of higher-level rules, usually in the form of differential equations, accounting for the global behavior of the flock. Particles are traditionally called 'boids' [Reynolds, 1987], a shortened version of 'bird-oid objects'.

In the traditional boids scenario, three rules apply:

Separation: particles steer to avoid crowding local flockmates;

Alignment: particles steer towards the average heading of local flockmates;

Cohesion: particles steer to move toward the average position of local flockmates.

The *dada.boids* module is able to account for such rules, as well as a for the presence of external barriers (obstacle avoidance) and winds. Moreover, each user can define his or her own set of rules, by compiling on-the-fly a portion of C code. Rules can have parameters, defining their position (such as the location of an obstacle), their orientation (such as the wind direction), their intensity (such as the wind speed, or the strength of a barrier), or, more generally, their behavior (such as a threshold for particle separation). Some of these parameters can also be associated to editable graphical user interface elements, such as points, vectors or lines—for instance, users can modify the direction of the wind by dragging the tip of the corresponding arrow, or the position of a barrier by dragging the corresponding horizontal or vertical line.

In addition to their position and speed, particles can have a scalar intensity value, and custom rules can be set to modify intensities along with speeds.

In practice, both built-in and user-defined rules are compiled functions that, for each particle, take as input its state, together with the state of the entire flock (coordinates, speeds and intensities of each particle), and yield as output, according to the current value of their parameters, a speed vector, to be added to the current particle speed (a 'steering' vector), and possibly a value to be added to its intensity. By summing the contributions of all rules, one gets the discrete derivative of the particle speed (and intensity).

Fig. 3.36 shows a screenshot of one of the patches that Andrea Agostini and I developed for the pedagogical project *Ariane#*[7], carried out at the Montbéliard conservatory. The patch takes advantage of *dada.boids*, with a predefined set of rules, as a controller to generate additive synthesis glissandi. Each particle is linked with a sinusoid, whose frequency is mapped on the vertical coordinate. By modifying the rule parameters, complex swarm behaviors appear, and interesting musical patterns emerge.

3.3.4.3 Graphs

The *dada.graph* module is a simple graph interface and editor, also featuring two automatic node placement algorithms provided by the Boost library [Siek et al.,

[7] http://numericariane.net

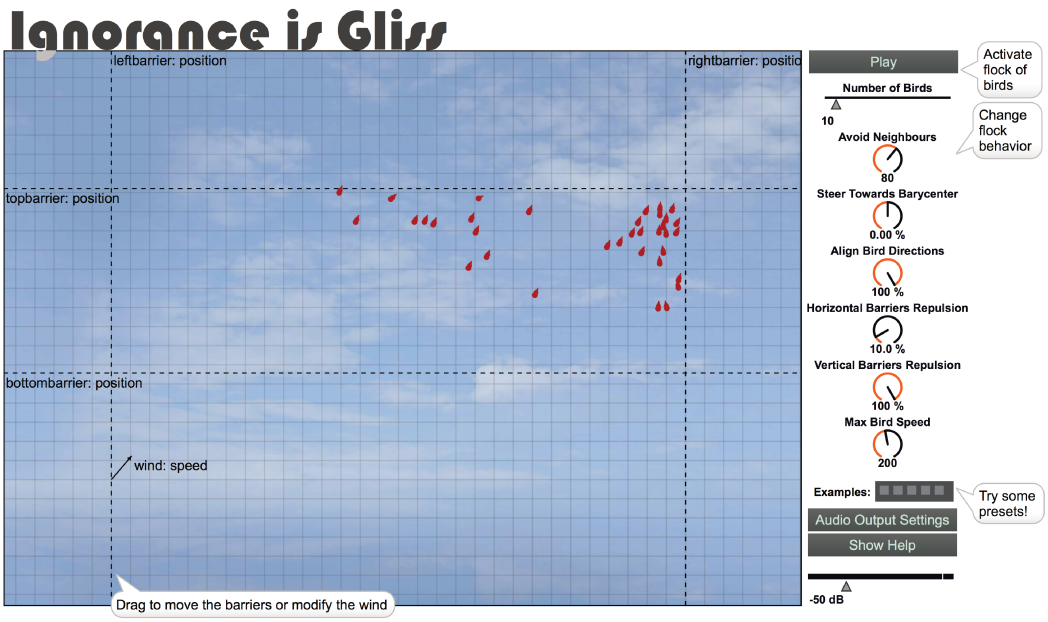


Figure 3.36: A screenshot of the interface of Ignorance is Gliss*, one of the patches developed for the Ariane# project, making use of* dada.boids *in order to generate additive synthesis glissandi.*

2001]: the Fruchterman-Reingold force-directed layout Fruchterman and Reingold [1991] and the Kamada-Kawai spring layout [Kamada and Kawai, 1989]. Similarly to *dada.distances*, the graph can be also navigated in a Markov-chain fashion, starting at a given point, and then choosing each following steps according to the edge probability distribution (weights) and to a desired memory length.

A variation on *dada.graph* is the *dada.machines* module, essentially a graph where each node represents some 'machine', i.e. a simple, prototypal operation to be performed on one or more inputs. By default these operations are elementary symbolic score transformations, such as transposition, retrogradation, circular shift, splitting, merging, and so on; user-defined operations are also supported. In a way, *dada.machines* represents a patch inside a patch, taking a score as input, processing it via the transformation graph, and outputting the result; however its spirit is more peculiar, and it was designed to be used with randomly generated graphs (the 'random' message produces graphs where the number of machines of each type matches a desired distribution). Via *dada.machines* one can apply a performative, exploratory paradigm to music, somehow reversing the functional and ergonomic relationship between algorithm and data.

We are used to operate on data via carefully designed functions, and to modify them if the output result on a certain input is different from what we desire. As an example, to create a symbolic distorted granulation of a given Mozart sonata, one would spend quite some time designing the way the symbolic granulation should be achieved and the type of distorsion modelling needed. Nonetheless, one might reverse the principle, taking a random algorithm for granted, and carefully exploring

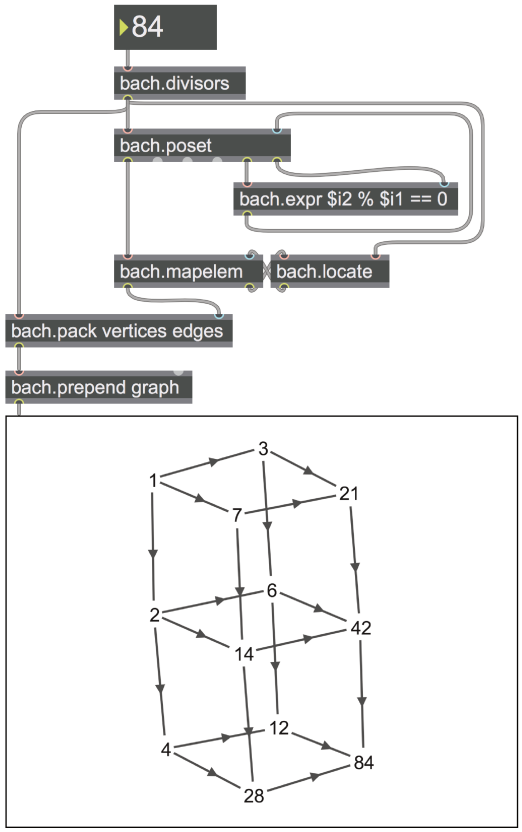


Figure 3.37: A simple patch displaying, via dada.graph*, the lattice of divisors for an incoming natural number.*

input data in order to see if the results are interesting. If the algorithm is 'complex enough', one might attempt to detect simple patterns (such as scales or counting-like patterns) along with more complex ones.

Of course, operatively, it makes little sense to search for a counting machine by tweaking inputs of a complex, random algorithm—which would categorize *dada.machines* module more as a mental experiment than a practical tool. I have often stated that music to me is an exploration—yet usually, I am aware of the global landscape features; with *dada.machines*, on the other hand, I was attempting to produce an exploratory tool where even the landscape itself was somehow unknown.

3.3.4.4 Videogames

Developing a game engine in Max might seem awkward; and indeed there is a large number of environments specifically dedicated to the task (Unity probably being one of the most popular[8]). Max is neither designed nor optimized for such scenarios.

It can however be interesting to have a (crude, primitive) game engine natively

[8] https://unity3d.com

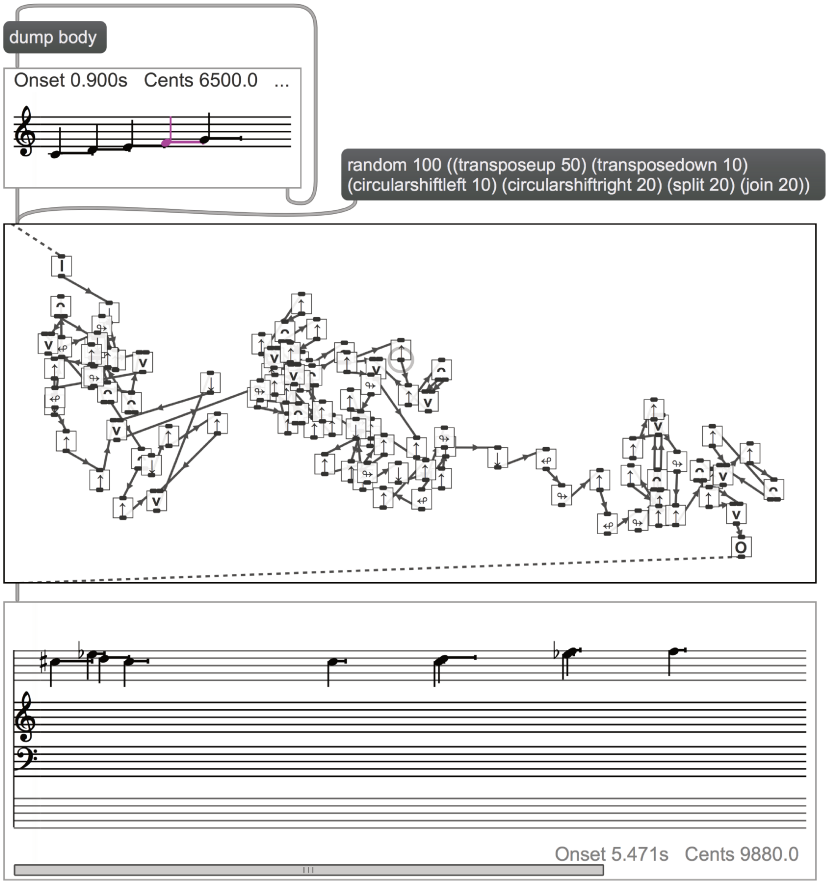


Figure 3.38: A patch featuring a dada.machines *interface, generating network graphs containing 100 machines according to a distribution of some 'atomic' score operations (transposition, circular shift, splitting and joining). The incoming score is processed via the randomly generated graph, and the result is output.*

coded in a Max external, since Max is a general purpose environment, and its visual paradigm can be applied to a large number of scenarios[9] (digital audio, video, lighting, actuators...), making it easier to communicate between different media and techniques.

The *dada.platform* module, allowing the design of graphical interactions inspired by platform videogames, has been imagined and developed with these considerations in mind. Due to the complexity of designing a usable game engine, the module is currently in a prototypal phase, slightly more than a 'proof of concept'. Nevertheless *dada.platform* already supports four categories of objects:

Blocks: fixed objects which can possibly be broken;

Coins: fixed objects which can possibly be taken;

[9]With one very notable, and unfortunate, exception: due to limitations with the Max 'symbol' structure, it is cumbersome, as of today, develop powerful and coherent tools in Max for generative text.

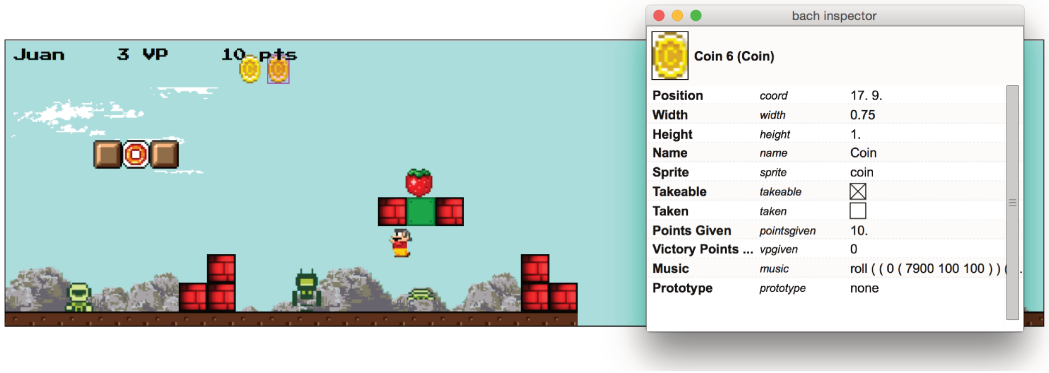


Figure 3.39: A screenshot of a dada.platform *editor, where the properties of the selected coin are displayed in the overlayed inspector.*

Game characters: moving elements which can interact with any other element in a more complex way. Game characters' motion is governed by a crude physical modelling: characters may possess the ability to jump, run, fly, swim, fire, glide, break elements, kill other characters, be killed by other characters. Game characters, in turns, belong to one of the following categories: 'usercontrol' (currently at most one character can be controlled by the user, also called 'hero'); 'idle' (do-nothing characters); 'food' (characters feeding the hero); 'enemy' (characters with the ability to harm or kill the hero); 'bullet' (projectiles potentially killing the hero);

Portals: objects which can dislocate the 'hero' to a new position in the same level, or to a brand new level (possibly saved as text or *llll* file).

All the properties of each object (such as its position, dimension, speed, abilities, image or sequence of images used to display it, and so on) can be set or fine-tuned via a dedicated inspector (see, for instance, fig. 3.39).

Linking game actions to musical events can be done in two ways. On one side, some of the objects' properties are musical scores (in *bach.roll* or *bach.score* syntax), output from a dedicated outlet whenever coins are taken, blocks are broken, and so on. More powerfully, any user action and any game interaction is notified via a dedicated outlet, so that any musical process can be triggered from them, such as sound synthesis, score production, video generation, and so on.

As it is not infrequent for objects in each level to share the same properties (just like identical blocks, coins or enemies), prototypes can be created, in order to easily handle multiple instances of indistinguishable objects.

Even more interestingly, some of the properties of an object can be sequences of instructions, wrapped in levels of parentheses, written in a dedicated scripting language, designed to modify the configuration of the object itself, or of other objects. Instruction sequences are provided whenever a character dies, a block is hit, or a portal is entered, and so on. Script commands allows a wide range of actions, including: breaking blocks, assigning points or victory points, generating new objects,

adding or removing abilities to characters, changing the state of objects, notifying some action, changing level or position in the level, pausing the game, preventing the hero from dying, winning, losing ('game over').

As a simple example, the script

```
(add hero ability fly) (goto level mynewlevel.txt at PipeRev
with (keephero 1)),
```

assigned to a given portal, provides the current hero with the ability to fly, and then loads the level contained in the file `mynewlevel.txt`, at the position of the portal named `PipeRev`, keeping the current hero state (including its properties, points and victory points).

Each game character has a script sequence for its death (the 'death sequence'); as another example, among many others, if one needs to turn a character named 'Juan', whenever he eats a certain fruit, into a character named 'SuperJuan', who, in turn, when killed returns to be a simple 'Juan' (like for the Mario/SuperMario classic Nintendo duality), one might want to assign to the fruit a death sequence along these lines:

```
(change hero (name SuperJuan) (idlesprite superjuanidle)
(walksprite superjuanwalk) (jumpsprite superjuanjump)
(flysprite superjuanswim) (height 1.625) (ext 0.35 0.35
0.825 0.825) (deathseq (dontdie) (remove hero ability die
during 2000) (change hero (name Juan) (idlesprite juanidle)
(walksprite juanwalk) (jumpsprite juanjump) (height 1) (ext
0.4 0.4 0.5 0.5) (deathseq)) (remove hero ability break)))
(add hero ability break).
```

Specific information about keywords and syntax can be found in the *dada.platform*'s help file and reference sheet. I shall just underline, in particular, how the last example is based on the fact that the fruit's death sequence changes the hero's death sequence, which in turns contain an instruction to clear its own death sequence, when triggered.

3.3.4.5 Miscellanea

For the sake of completeness, I shall also mention two other modules:

- the *dada.multibrot* module provides information about convergence or divergence of generalized Mandelbrot sequences of the type $z_{n+1} = z_n^d + c$ on specific input points $z \in \mathbb{C}$;
- the *dada.music*$\sim$ module, upon which *Music (or The Discotheque of Babel)* is based (see section 2.1.4), provides a one-dimensional interface for all the possible configurations of buffers, organized by size in samples. In other words, it virtually provides a way to explore all the mono digital music. In practice,

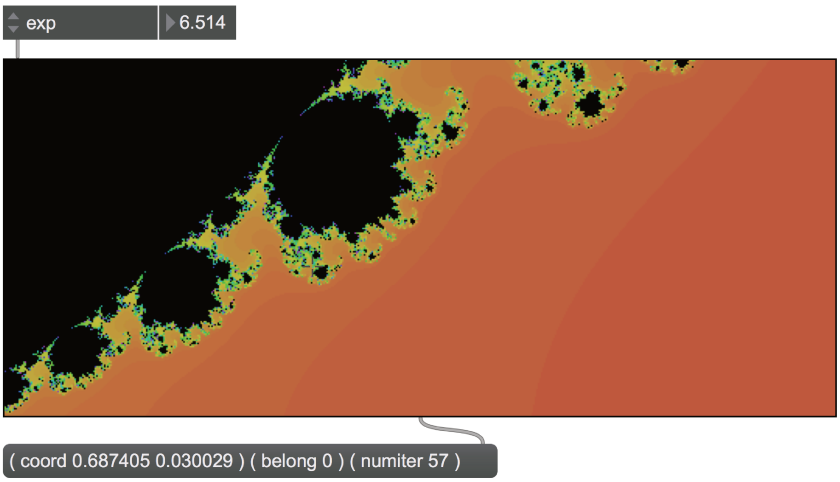


Figure 3.40: A dada.multibrot *interface, displaying the information about the clicked point* $c = (0.687405, 0.030029)$*, for which the generalized Mandelbrot sequence* $z_{n+1} = z_n^{6.451} + c$ *does not converge (i.e. the point does not belong to the corresponding multibrot set), and the divergence is detected after 57 iterations.*

it is essentially impossible not to pick a white noise buffer, except for a few quasi-cyclic Moiré-like patterns showing up when the zoom factor is a power of two). The module relies on the GMP and MPFR libraries for arbitrary precision arithmethic [Granlund et al., 1991; Fousse et al., 2007], and supports multiple sample rates and bit depths.

3.3.5 Comparison with other software

There is some correspondance between *dada*'s geometric approach and graphical sequencers such as Iannix [Coduys and Ferry, 2004] (as a matter of fact, a partial, two-dimensional porting of Iannix into *dada* might be a good addition to the library). On the other hand, the sequencing capabilities of Iannix largely outperform *dada*'s, whose purpose is not sequencing *per se*, but rather a seamless integration with the *bach* and Max environment, allowing, among many other things, live recording of scores.

The *dada* library shares with InScore [Fober et al., 2012] the interest in designing interactive non-standard symbolic representation. The idea of using games to interactively structure musical content resonates with Paul Turowski's researches and works, such as *Frontier* [Turowski, 2016].

Some *dada* modules have been explicitly inspired by, or have correspondances with, specific pieces of software. As stated above, the portion of *dada* dealing with corpus-based composition has borrowed CataRT's representation of audio grains [Schwarz et al., 2006]. The *dada.life* module shares with Louis Bigo's HexaChord [Bigo et al., 2015] the possibility of visualizing trajectories on musical lattices such as the Tonnetz—although the former focuses on the generation of cellular automata, while the latter is tailored for analysis purposes. The *dada.terrain~* module shares

with WAVE[10] and with Stuart James's work [James, 2005] the ability to perform wave terrain synthesis inside Max.

One should also remark the relationship of *dada* with music applications such as Björk's Biophilia, or Brian Eno's generative apps, or with interactive web tools such as some of the 'Chrome Experiments'[11] or of the 'A.I. Experiments'[12] (e.g., 'The Infinite Drum Machine'); all these cases share with *dada* an interest for a tight, creative connection between visuals, gestures and music, and for exploring the grey area between interfaces and musical objects—however, if at least in Björk's case the musical apps are themselves art objects, *dada* modules are designed as simple instruments for composition (with possibly one notable exception: *dada.music~*).

3.3.6 Future work

The *dada* library is still in its infancy, and a certain number of additions and improvements are needed to complete it and to make it more usable.

First of all, thorough testing and optimization are necessary to make the library more stable and the user experience more comfortable. Besides, a Windows porting is also needed (currently the library only works on Macintosh).

One of the most important lines of development would be porting the interfaces on mobile operative systems (tablets, smartphones), where they might take advantage of multitouch support. The most convenient way would be to exploit the Miraweb package[13], developed by Cycling '74, which allows mirroring on web browsers specific interface elements contained in a patch; the possibility to add Miraweb support to third party externals should be explored.

As far as the documentation is concerned, comprehensive help files and complete reference sheets are already provided for each module. However, some video tutorials would be a valuable addition for users who need to get used to the *dada* environment.

The set of tools for corpus-based composition can be improved in a number of ways.

- The number of ready-made analysis modules should be increased, by attempting to bring into the symbolic domain important audio descriptors such as roughness, inharmonicity, temporal centroid, and so on. The relationships between audio and symbolic descriptors could be in itself an interesting topic for further investigation.

- The *dada.segment* module is currently able to perform score segmentation based on markers, equations or labels; however it is not able to infer such markers or labels. One of the interesting topics of future research might the integration of a system for semi-automatic segmentation of scores, and a module for pattern retrieval.

[10] http://www.noisemaker.academy/wave-an-introduction-to-waveshaping-and-wave-terrain-synthesis/

[11] https://www.chromeexperiments.com/

[12] https://aiexperiments.withgoogle.com/drum-machine

[13] https://cycling74.com/articles/content-you-need-miraweb

- *dada.catart* and *dada.distances* are currently able to filter and display a certain table of a given database, but they are not able to interact with it, for instance by adding or deleting entries, or by modifying column values (while dragging the corresponding points on the display). One might imagine an editing mode, where these tools also become capable of performing the aforementioned operations.

- Better communication between *dada.distances* and *dada.graph* should be devised; for instance: *dada.distance* should be able to output the database content in graph form.

The tools for physical or geometrical modelling are probably the modules in *dada* whose development is most advanced; nonetheless:

- One could imagine trajectories in a *dada.bounce* interface to be potentially also affected by a certain gravity field.

- The *dada.terrain~* module should be provided with anti-aliasing capabilities—which were not needed for my particular usage, but which should be a reasonable feature to add for more general applications.

- More generally, one might desire more graphical sequencing tools; for instance, an interface inspired by a two-dimensional version of Iannix [Coduys and Ferry, 2004] might be envisaged.

Finally, a certain number of improvements can affect the subset of tools dealing with rule-based systems and graphs:

- *dada.graph* is already capable of displaying graphs where the vertices are notes; it might also be provided with the possibility of displaying vertices as complex scores, which would open the way for potentially interesting applications.

- Currently, *dada.graph* features two automatic node placement algorithms: the Kamada-Kawai layout, supporting connected, undirected graphs, and the Fruchterman-Reingold layout, supporting (potentially disconnected) undirected graphs. The module employs such algorithms even for directed graphs, however a dedicated algorithm should be provided for this case, deploying the graph direction unambiguously, for instance, as left-to-right or top-to-bottom. Some special graph types, such as trees or partially ordered sets, also require different dedicated algorithms; trees might be displayed via the Reingold-Tilford algorithm [Reingold and Tilford, 1981], while lattice diagrams, such as Hasse diagrams, might be displayed via the algorithm proposed by Freese [2004]. Automatic graph type detection, triggering the corresponding placement algorithm, might be a nice feature to have.

- Another nice feature to have in *dada.graph* would be the computation of minimum spanning trees and shortest paths.

- The *dada.machine* module should support multiple inputs and outputs, and should take advantage of *dada.graph*'s algorithms for automatically placing machines on the canvas.
- The *dada.platform* object is currently little more than a 'proof of concept'. It would be interesting to issue something akin to a 'call for scores' for pieces of interactive music based on it; this would probably also help detecting the bugs and the flaws of the system.

3.4 Meta-scores

Sections 3.4.1, 3.4.2 and 3.4.3 have been previously published, in a slightly different form, in the article Ghisi, D., Agostini, A., and Maestri, E. (2016). Recreating Gérard Grisey's Vortex Temporum with cage. In Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.*

3.4.1 Hybrid scores as instruments

Since the 1950s, the role of musical notation has undergone a number of generalizations (including graphical scores, gestural scores, morphological scores in electroacoustic music). This evolution has been caused, in part, by the development of electronic music. Schaeffer [1966] defined two kinds of score: a descriptive one and an operational one; Seeger [1948] proposed the opposition between prescriptive and descriptive scores. This differentiation is reinforced by the development of computer music: as a matter of fact, both computer programming and composing are mediated through notation [Nash, 2015].

Composers use scores to sketch musical ideas, formalize them into a script and communicate them to performers; computer programmers, on the other hand, mostly use symbol-based formal languages to convey instructions to a computer. In both cases, notation is prescription. Both musical notation and programming are systems of prescription of actions, specifically in the case of musicians that must activate a vibrating body, and tasks, in the case of computers, that activate the vibrating body of the loudspeaker mediating the intentionality of the musician: the combination between the two defines a hybrid dimension of musical scores as partially suggested by Emmerson [2009] and Sorensen and Gardner [2010].

Hybrid scores have a twofold meaning: on the one hand, they are targeted at performers, to whom they prescribe actions which are typically, although not exclusively, aimed at producing sound; on the other hand, they are targeted at computers ('digital performers' [Mathews, 1963]), to which, through information encoded in a programming language, they prescribe the production of sound or symbols, or even more complex tasks [Maestri and Antoniadis, 2015]. In particular, hybrid scores are capable of prescribing (and hence embedding) other hybrid scores within themselves, which makes them very suitable to represent and process abstract, nested musical structures.

Within this conceptual framework, the term *meta-score* will define a hybrid score whose components are not elementary notational items (typically, single notes or chords), but rather processes which can be further prescribed and described as scores in their own terms. So to speak, we might say that a meta-score is a score *of* scores (i.e., a score containing other scores), or a score *for* scores (i.e., a score containing instructions to build other scores), or a score *about* scores (i.e., a score containing descriptions of other scores), hence expanding the fundamental ideas described by Mathews et al. [1974].

Importantly, in a real-time scenario, interactive hybrid scores are not simply

objects to play, but rather objects to play *with*: in this sense, they contribute to narrowing the gap between scores and instruments. Incidentally, the *dada* library (see section 3.3) addresses a closely related issue, insofar as most *dada* modules are interactive digital instruments capable of generating symbolic scores. This attitude is dual to the concept of 'instruments as inherent scores' [Tomás, 2016].

3.4.2 Meta-scores in *bach* and *cage*

Although software systems for computer-aided composition usually focus on the manipulation of basic musical elements, such as notes and chords, or small-scale phenomena, such as melodic profiles, some of these systems provide tools for dealing with higher-level musical objects, treated as containers or generators of lower-level elements, one notable example being OpenMusic's maquette [Agon and Assayag, 2002].

When Andrea Agostini and I had the opportunity to develop the *cage* library [Agostini et al., 2014], we decided to include in it two modules devoted to facilitate the construction of meta-scores, and constituting one of the various subsets of the library itself, namely the *cage.meta* subset. After careful consideration, we came to the conclusion that we would not want to implement a dedicated graphical editor and paradigm (which is what the maquette is, within the OpenMusic environment), but rather to devise a design pattern allowing the usual *bach* notation editors/sequencers (the *bach.roll* and *bach.score* objects) to be used for representing meta-scores, rather than scores proper.

The choice of extending to meta-scores the concepts and tools used for representing traditional scores is motivated by the observation that, somehow, there is no clear boundary between traditional scores and meta-scores. In fact, more often than not, symbols in any traditional score refer to extremely complex processes and activities, be it the skillful control of the friction of a bow on a string, or the triggering of complex sets of envelopes to be applied to the inputs and outputs of a bank of oscillators. Moreover, in historical musical practices, there exist specific synthetic notations representing complex musical figures, such as trills, mordents, arpeggi, gruppetti and other ornamentation marks, or—even more specifically—the numbers and symbols of figured bass. By not striking a dividing line between scores and meta-scores we aim to focus on the similarities, and the continuum, between the two, rather than on the differences. At the same time, we feel that a graphical interface based upon the traditional notational paradigm can be perceived as more 'neutral' than a custom one, and as such is less likely to suggest specific compositional approaches or practices, and more inviting to be bent to each composer's creative needs.

The basic idea behind *cage.meta* relies upon the fact that scores contained in *bach.roll* or *bach.score* objects are hybrid scores, as each of their notes can be seen as a container of various, heterogeneous parameters: a small, standard set of basic, required data which define the note itself in traditional terms (position in time, expressed in milliseconds in *bach.roll*, in bars and beats in *bach.score*; duration,

expressed in the same respective units; pitch; and MIDI velocity), and an optional, user-definable combination of other associated data belonging to a wide array of types (numbers, text, breakpoint functions, lists of file names, and more), contained in slots.

Most importantly, in the *cage.meta* system, each note is associated to a certain process, implemented in a Max patcher file whose name is assigned to the note's first slot. At initialization time, the patchers referred to by all the notes of the score are loaded and individually initialized. At play time, when a note is met, all its parameters and slot data are passed to the patcher it refers to. Although this is not enforced in any way, the idea is that the patcher itself will activate its intended behavior according to these parameters when they are received. Because the duration of a note is passed as one of its parameters, it is possible for the activated process to regulate its own duration according to it—but, once again, this is not enforced by any means, and it is possible to implement processes whose duration is fixed, or depends on other parameters. The same goes for the pitches and the MIDI velocities: the fact that they are passed to the process does not mean that the process itself must use them in any traditional, literal way—in fact, it can as well ignore them altogether.

A *cage*-based meta-score is built in two distinct phases, taking advantage of the modules named *cage.meta.engine* and *cage.meta.header*. The first phase is creating the patchers implementing the processes that will be associated with the meta-score events. Each patcher must contain one *cage.meta.header* module: at play time, parameters from the referring note will not be passed to these patchers through inlets, but through the third or fourth outlet of *cage.meta.header*, according to the rendering mode explained below.

The second phase is setting up the meta-score system, constituted by a *bach.roll* or *bach.score* object (which we shall refer to as the 'score' object from now on) connected to a *cage.meta.engine* object in a 'loopback' configuration, such that the two inlets of *cage.meta.engine* are connected respectively to the leftmost outlet of the score object, from which all the score data can be output as one possibly large *llll*, and its one-but-rightmost outlet, from which the data of each note are output as the note itself is encountered at play time. Also, the first outlet of *cage.meta.engine* must be connected to the leftmost inlet of the score object: in this way, *cage.meta.engine* can perform queries on it and set some of its parameters if required. Finally, a different *bach.roll* or *bach.score* object (according to the type of the meta-score object) can optionally be connected to the second outlet of *cage.meta.engine*, so as to collect a rendered score, according to a different usage paradigm which will be discussed below.

Now it is possible to write the actual meta-score, by introducing the notes, along with their associated patches and slot parameters.

After the meta-score has been written, generated or loaded from disk, the *load* message can be sent to *cage.meta.engine*: this causes the score to be queried for all its file names and instance numbers, and loads each referred patch as many times as required by the different instance numbers found in the score. Immediately after

having being loaded, each patch is initialized, that is, it is sent three identifiers: the engine name, the patcher's own file name, and its instance number. These three identifiers will be used at play time to route the parameters of each note to the correct instance of its referred patch only, while avoiding conflicts with other possible *cage.meta* systems running at the same time, in the same Max session. The instance number handles the polyphonic behavior, in case the same process must be triggered by overlapping notes; it can be either manually or automatically assigned. Furthermore, depending on *bach.roll*'s or *bach.score*'s attributes, markers and tempi can also be sent to all the patches at play time. Receiving notifications for markers can be useful if, for instance, one needs to adapt the behavior of a process to different sections of the score. As a convenient debug tool, selecting a note in the score editor and pressing the 'o' key causes the corresponding referred patch to be opened.

In principle, the outcome of a run of the meta-score is just data, which can be collected in any possible data structure, directed to any possible device or process, and to which any possible meaning can be assigned. Each process, as implemented in the corresponding patcher, receives data from *cage.meta.header* and produces a result which can be routed, for instance, to a MIDI device, or an audio output: but also, according to a less inherently real-time paradigm, to an audio buffer collecting algorithmically-generated audio samples; or to a new score object which will contain the 'rendering' of the meta-score. In particular, we deemed this last scenario to be so typical and peculiar that it deserved some kind of special treatment. More specifically, we expect most musical processes that users may want to implement with the *cage.meta* system to produce either a real-time audio, MIDI or OSC result, or a symbolic result (i.e., a score) to be built incrementally note by note, or chord by chord. As an example of the former case, each *cage.meta.header* patch could contain a synthesizer, generating a complex audio stream depending on the parameters of the associated note; in the latter case, each patch could produce a complex musical figure (e.g., an arpeggio) built according to the parameters of the associated notes, and meant to be transcribed into a final score resulting from the run of the whole meta-score. The latter case can be seen as a special case of the general one, but the complexity of setting up a system for the very basic purpose of generating a score starting from a meta-score prompted us to implement a specific mechanism allowing a process patcher to return to *cage.meta.header* one or more chords in response to a message coming from *cage.meta.header* itself.

More specifically, when the 'playback' attribute of *cage.meta.engine* is set to 1, events coming from the score object are passed to each process patch through the third outlet of *cage.meta.header*, and can be routed to any generic destination (real-time audio, MIDI, OSC, or anything else): for example, the synthesizer implemented in the process patch would set its own parameters according to the data received from the meta-score, activate itself and produce an audio signal to be directly fed to the audio output of Max.

On the other hand, when the 'render' attribute of *cage.meta.engine* is set to 1, events from the score object are passed to each process patch through the fourth and rightmost outlet of *cage.meta.header*, and one or more chords (that is, *llll*s

featuring all the chords and note parameters, formatted in the *bach* syntax) can be returned to the second and rightmost inlet of the same *cage.meta.header* module, in a loopback configuration ('lambda loop'). The *cage.meta.header* module then returns the received chords to its master *cage.meta.engine*, which formats them in such a way to allow an instance of the appropriate object, connected to its second outlet, to be populated with the score being rendered. All this is preceded by a sequence of formatting instructions sent to the destination *bach.roll* or *bach.score*, and generated only if the *render* attribute is on. At the end of the rendering process, the whole rendered score will be contained in the notation object connected to *cage.meta.engine*'s second outlet. So, for example, a patch building an arpeggio receives the parameters of note of the meta-score referring to it (and containing the parameters of the arpeggio, such as starting pitch, range and speed) from the fourth outlet of *cage.meta.header*, and returns the rendered arpeggio, as a sequence of notes, to the rightmost inlet of *cage.meta.header*. The notes of arpeggio are then sent by *cage.meta.header* to the master *cage.meta.engine*, which in turn formats them as messages for the *bach.roll* or *bach.score* object connected to its second outlet. Through this mechanism, this destination *bach.roll* or *bach.score* is incrementally filled and eventually will contain the result of the whole rendering process.

When the 'play' message is sent to the score object, the playback and/or rendering of the meta-score begins: the score object starts outputting the notes it contains according to their very temporality, and *cage.meta.engine* and *cage.meta.header* cooperate to route the data associated to each note to the correct instance of the patcher the note itself refers to. Another possibility is sending the score object the 'play offline' message: in this case, the score object starts outputting the notes it contains in strict temporal order, but with the shortest possible delay between them, without respecting their actual onsets and duration. This is somehow analogous to the 'offline render' or 'bounce' commands that can be found in virtually any audio and MIDI sequencer. As hinted above, this is useful to trigger non-realtime rendering processes, such as, typically, the rendering of a score through the 'lambda loop' of *cage.meta.header*, but also, for instance, the direct writing of samples in an audio buffer, or any other kind of batch operation.

3.4.3 An analysis case study: *Vortex temporum*

As an interesting case study on the subject, Andrea Agostini, Eric Maestri and I have recreated the first 81 measures (corresponding to numbers 1 to 20, according to the numbered sections marked in the score) of Gérard Grisey's *Vortex temporum* in *bach* and *cage*, basing our work upon the analysis by Hervé [2001].

The basic idea behind our exercise is abstraction: we aim at building and manipulating a meta-score featuring operative compositional elements, rather than pre-rendered symbolic processes. For instance, since the pitch choices in *Vortex temporum* are strictly based upon a spectral paradigm, our meta-score will be solely composed of spectral fundamentals.[14] Every note in our meta-score is hence a fun-

[14]It might be worth pointing out that all harmonic series used by Grisey in the part of *Vortex*

damental for some process, and the indices of harmonics that are built upon it and used by the process are contained in the third slot of each note. We implemented both an off-line score rendering and a real-time rendition, the latter through an extremely simple synthesis process: for this reason, each note carries in a dedicated slot an amplitude envelope information, in the form of a breakpoint function.

Each note of the meta-score triggers one of three different processes: an arpeggio, an accented chord, or a long note. We shall now describe them in detail.

The first process is designed to generate all the arpeggio-like figures which characterize and identify *Vortex temporum* from its very beginning. More specifically, the arpeggiator renders all the 16th-notes figures, with the notable exception of the accented piano and string chords at the beginning of each numbered section: these figures have a different musical role (akin to attack transients), and will be rendered by a different module, which will be discussed further.

Besides the fundamental note and the list of harmonic indices, the arpeggiator also receives some additional content, contained in further slots of our meta-score: the duration of each rendered note in the arpeggio (it is, in our case, constantly $1/16$); the number N of notes composing a single arpeggio period (for instance, for flute and clarinet at measure 1 we get $N = 8$, since the arpeggio loops after 8 notes); and the profile for the arpeggio period, as a breakpoint function representing time on the x axis, and the elements in the harmonics list on the y axis. The final point of this function should always coincide with the starting one (to comply with the looping).

Inside the arpeggiator patch, the arpeggio profile is sampled at N uniformly distributed points, each of which is then approximated to the nearest element in the list of the harmonics, which are uniformly distributed on the y axis, independently of their actual value, and subsequently converted into the pitch derived from the stretched harmonic series (see fig. 3.41). All pitches are approximated to the quarter-tone grid, with the exception of piano notes, which are approximated to the semitonal grid.[15]

During real-time playback, harmonics are then output by *bach.drip*, with the appropriate (uniform) time delay between them, depending on the current tempo and the duration of each rendered note. The audio rendering is performed by basic oscillators: the flute is rendered via a slightly overdriven sine tone; the clarinet via a triangular wave; the piano via a rectangular wave. These are of course completely arbitrary choices, only aimed at clearly illustrating the rendering mechanism.

Two accessory processes also need to be described. Firstly, during each of the numbered sections, Grisey gradually filters out some notes from the arpeggi, replacing more and more of them with rests. This deletion only takes place in the flute

temporum that we reimplemented are stretched linearly in the pitch domain by a factor of $\pi/3$. This observation does not appear in [Hervé, 2001], but it seems pertinent in our implementation.

[15] On occasion, the flute part contains 12-TET tempered notes instead of 24-TET tempered notes. This is the case at measure 19, for instance: the natural C 'should' be a quarter-sharp C according to the harmonic series rendering. Grisey probably adjusted these notes to ease and respect the instrumental technique, but we did not account for these 'manual' adjustments in our work.

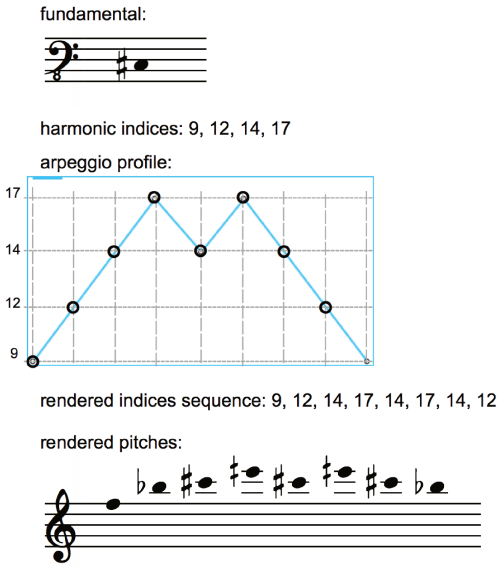


Figure 3.41: The conversion of the arpeggio profile for the flute (measure 1) into actual pitches. The harmonic indices, relative to the defined fundamental, are uniformly distributed on the y axis, and the profile is uniformly sampled on the x axis. The result is then snapped to the nearest harmonic index. The sequence is rendered by retrieving the individual harmonics of the stretched harmonic series built upon the fundamental.

and clarinet parts (also enabling the players to breathe between phrases), and it is roughly anticorrelated to dynamics. Since we do not have a model of the exact process employed by Grisey for performing these deletions, we decided to implement it probabilistically, by adding an 'existence probability' for each note, directly correlated to the instantaneous decibel value of the amplitude envelope: at 0dB, the existence probability will be 100%, while at -22dB the probability drops down to 50%. Secondly, starting from number 11, some bichords show up in the piano arpeggi. Bichords are always composed by neighbour harmonics in the harmonics list, thus preserving the arpeggio profile; hence it made sense for us to map a bichord to a half-integer snapping on the y axis of fig. 3.41.

The second rendering module deals with the previously mentioned accented chords at the beginning of each section. The off-line rendering is trivial: each note is rendered as a chord whose pitches are the specified partials; no additional parameters are required. During playback, each note is rendered as a white noise attack, fed into a narrow-band resonant filter centered around the frequency of the note itself, and subsequently artificially reverberated.

The third rendering module generates the long viola and cello notes (first appearing at number 3). The viola is rendered via a sawtooth oscillator, the cello via a rectangle wave oscillator.

In our line of work, the meta-score, that is, all the notes with all their parameters, is completely written 'by hand' in a *bach.score* object.[16] All the slot values are kept

[16]By manipulating the schemes given by Hervé [2001] one might build the meta-score content

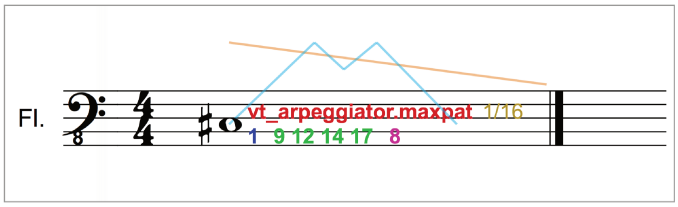


Figure 3.42: The first measure of the flute part in the meta-score. The low C♯ (fundamental) and its 9th, 12th, 14th and 17th stretched harmonics are rendered via the arpeggiator (1st instance). The arpeggio loop has 8 notes (each lasting 1/16th) and follows the 'mountain'-like profile shown in light blue. The amplitude envelope is the descending orange line.

visible for each note; tied notes by default share the same slot value and are output as a single longer note. The first measure of the flute part is displayed in fig. 3.42, with all metadata visible next to the notehead, as text or breakpoint functions. Depending on our desired operating mode, at any time, we can play or render symbolically in real-time any portion of score. We can also render the whole meta-score off-line, in order to have it all output form *cage.meta.engine*'s right outlet. A screenshot of the main patch is displayed in fig. 3.43.

Although our meta-score might appear bizarre at first (all instruments are notated in F clef, transposed one octave lower), it turns out to be extremely pertinent. For one thing, it is immediately clear that all instruments are somehow playing 'meta-unisons' (except for the right hand of the piano), correspondingly to the fact that all instruments are confined within the same harmonic series. When, at number 10, all fundamentals switch to G, such important change is immediately readable in our meta-score, while it would take a larger effort of analysis to grab the same information from the score proper. Our meta-score also reveals very clearly the diminished seventh tetrachord (C♯, E, G, B♭)[17] underlying the whole construction, and abstracts the complexity of the arpeggi to a profile shape and a few indices (leaving most of the score made of long tied sequences of 4/4 notes).

3.4.4 Applications to composition

The previous section showed that meta-scores are valuable analysis tools, allowing significant properties of the musical content to emerge (in the specific case, harmonic relationships and profiles), in a manner loosely reminiscent of how Schenkerian analysis shows hierarchical relationships among notes. Each instance of a process is symbolized by a note (or a chord), making possible possible to represent pitch-based operations intuitively, via their 'most meaningful' pitch (or set of pitches).

These very properties also confer to meta-scores a value in musical writing: not only do higher properties emerge, but they can also be handled rapidly and efficiently. In Grisey's example, one can, for instance, change the duration unit of

itself with *bach* via standard algorithmic techniques.

[17] It should be pointed out that the E does not appear in the portion of score upon which we focused.

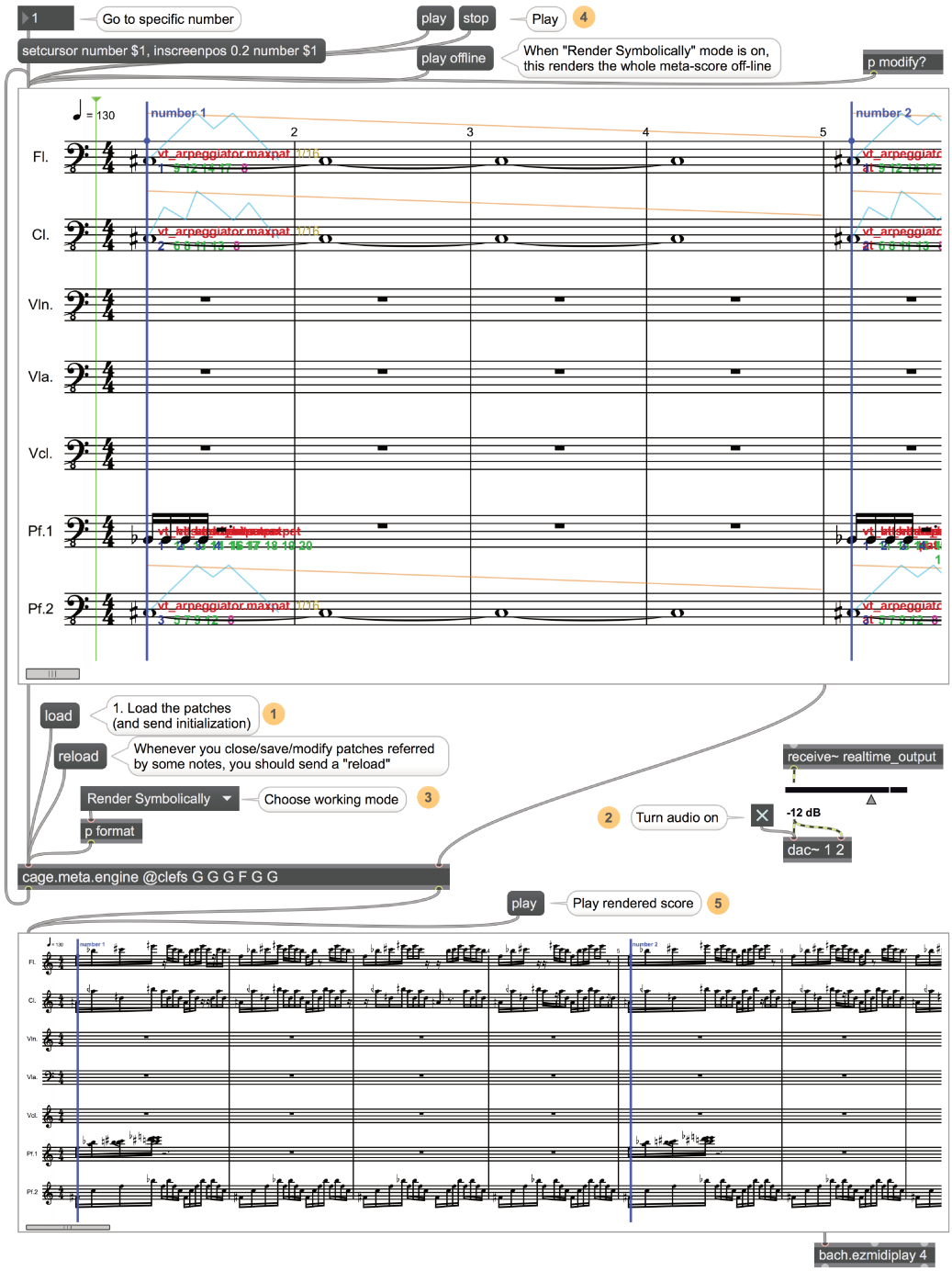


Figure 3.43: A screenshot of the patch containing the meta-score that models the beginning of Vortex temporum.

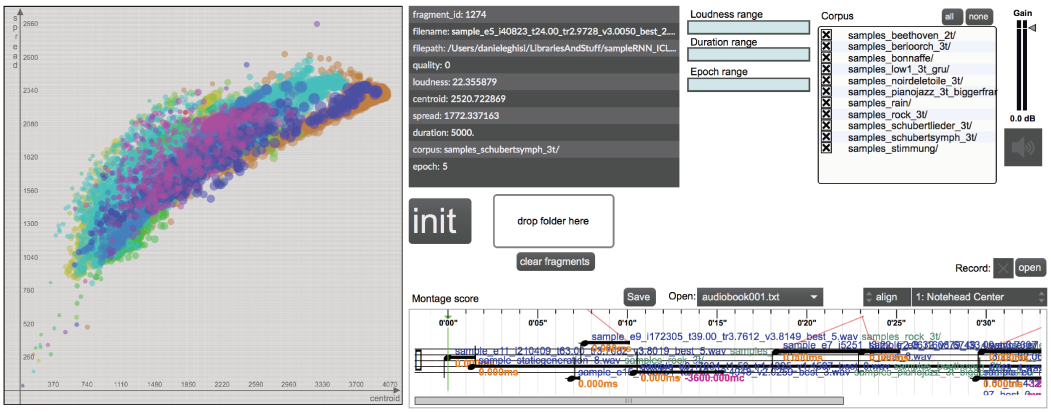


Figure 3.44: Interface used to realize audio montages for La fabrique des monstres*, starting from a database of audio files generated by a sample-by-sample recurrent neural network.*

all the notes in the arpeggi from 1/16 to 1/32, or transpose all the clarinet notes down a tritone, with simple Max messags; one can alternatively apply standard computer-aided composition techniques on higher-level structures (for instance, it would be fairly easy in our case to retrograde or stretch all the arpeggio profiles). Meta-scores are therefore powerful tools for prototyping, exploring and composing.

Most of the Max patch I have been developing for my own music are, to some extent, influenced by these ideas. Some of them make explicit use of the *cage.meta* system, especially the most recent ones, while others implement different, yet analogous, meta-score mechanisms.

Notation for chord-based concatenations. Most of the electronics concatenations for *An Experiment with Time* were carried out via a meta-score system where each note stands for a random concatenation of samples, chosen according to a given chord, having the note as fundamental and the chord type assigned in a slot, according to a number of settings introduced in additional slots. I have described this meta-score usage in section 2.3.3.

Audio montage. A simpler example is constituted by a score where each note represents a portion of an audio sample, similar to a region of a standard modern digital audio workstation. The pitch is either ignored or mapped to a transposition factor; all rendering parameters, such as amplitude envelopes or pan position, are contained in dedicated slots.

I frequently take advantage of this symbolic representation of montages, which turns *bach.roll* and *bach.score* into scriptable sequencers. One of the recent examples is its usage in *La fabrique des monstres* (also see section 4.3.1), where collages of audio files, generated by a sample-by-sample recurrent neural network, are represented in *bach.roll*s. As shown in fig. 3.44, samples can be browsed via a *dada.catart* interface and can be assigned to notes in the score simply by clicking on the corresponding circles while notes are selected. Offsets in audio files can be modified by

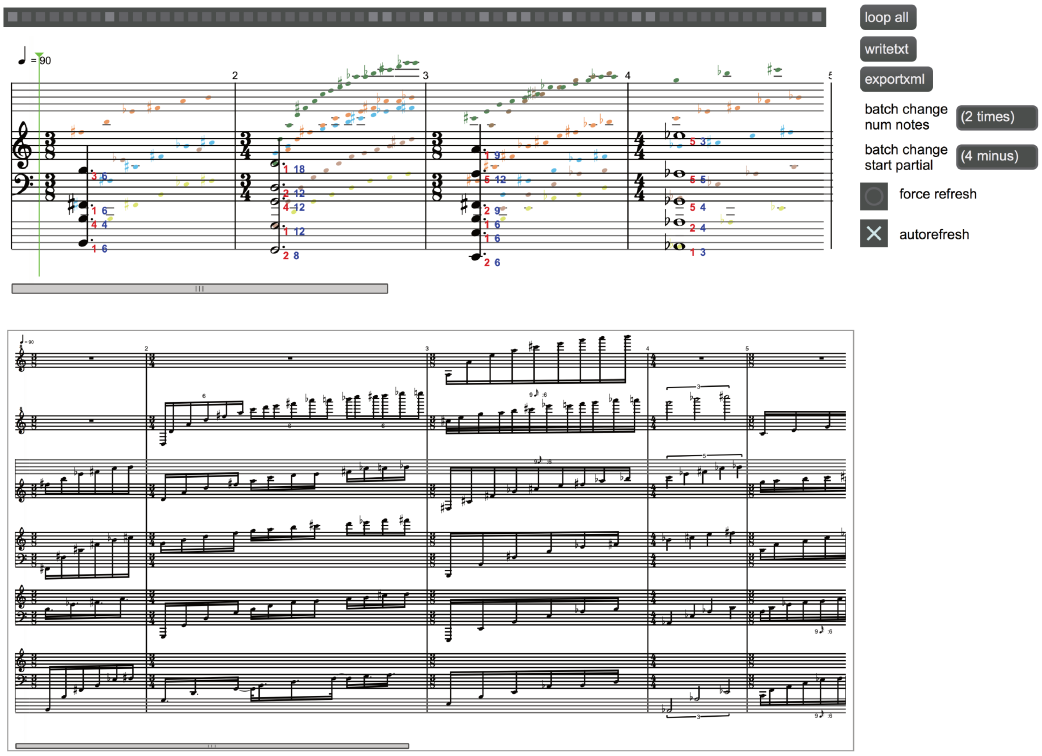
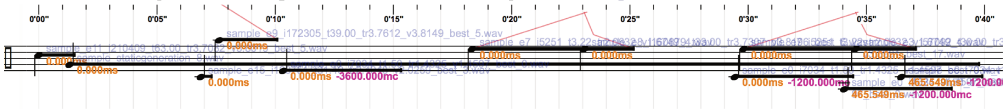


Figure 3.45: Real-time rendering of portions of harmonic series from a traditionally notated score for Any Road *(top). The rendered score, always synchronized in realtime, is displayed at bottom.*

trimming note heads, or by manually changing slot values.

Excerpts

- `data.danieleghisi.com/phd/snd/LFDM_montage1.aif|mp3`



Harmonic pencil. An example of purely symbolic meta-score is the high-level writing system I developed for *Any Road*, enabling to reactively write and render superposed portions of harmonic series (see fig. 3.45). Each note represents the fundamental of an harmonic series, and carries information on the number of harmonics and the index of the first partial; a rendered score is obtained by replacing each note with the corresponding portion of harmonic series.

As hinted at during the case study of *Vortex temporum*, the ability to handle significant higher-level object makes the compositional workflow more intuitive.

Excerpts

- From *Any Road* (with video):
  `data.danieleghisi.com/phd/vid/AR_extr1.mp4`

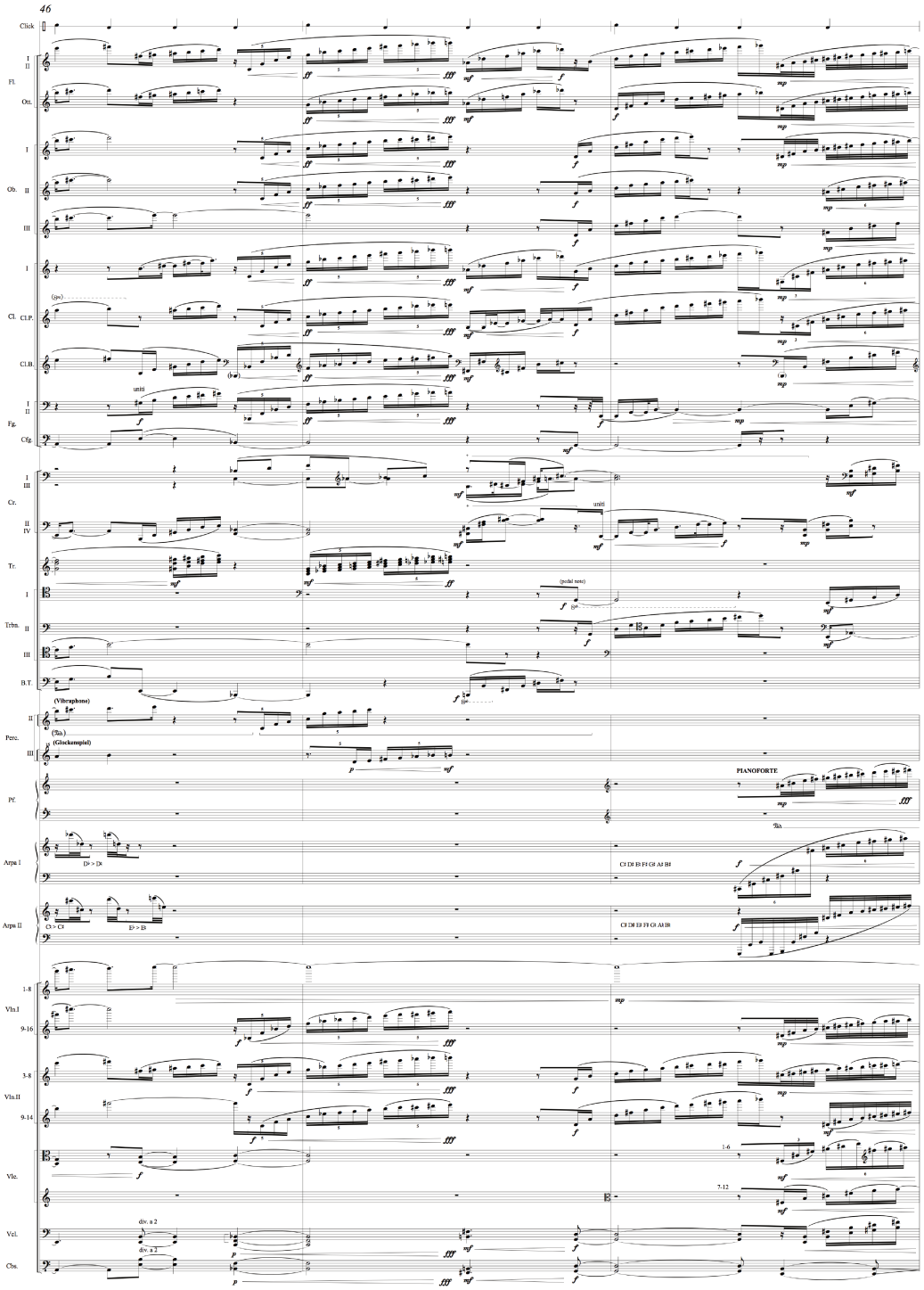


Figure 3.46: A page of the orchestral score for Any Road, *displaying the rendering of the portions of harmonic series (the second half of b. 47 corresponds to the beginning of fig. 3.45).*

Score collage. A meta-score system was designed for the last section of *I mille fuochi dell'universo* to explore combinations of 'grains' into complex score objects. In a very literal sense, the systems enables the fabrication of scores of scores. The underlying score database is constituted by symbolic scores in MusicXML format, electronic scores as audio files, and mixed scores (combining a symbolic component and an audio file). I have described this collage technique in section 2.8.

Notating movements and actions, perturbing time flow. A more complex meta-score scenario has been designed for *Bug (quatuor à corps)*.

Bug (quatuor à corps) is a theatre piece by Ingrid von Wantoch Rekowski, who asked four composers to apply the musical principles of a string quartet writing to theatrical sensibility. Inspired by pioneering works in video art such as Norman McLaren's *Canon* (1964) and Zbigniew Rybczyński's *Tango* (1980), my attempt was to design a certain number of situations, or fragments, strongly based on canonic processes, where actions and words would assume different meanings while performed in canon by different actors. In some situations, the canonic principle would be strictly kept throughout the entire fragment; in other situations, actions and words would at some point diverge into a more chaotic and less formalized writing.

A montage of piano scales constitutes a sort of 'basso continuo', a background framework on which the canonic actions take place. In addition, on the top of this structure, higher-level 'bugs' appear from time to time, looping certain portions of actions, sounds and words, for a certain number of times.

The musical substance being composed by sounds, words, movements, actions, words, I designed a score where each note may represent a sound (an audio file, also possibly including a label to be displayed), a sentence (audio files and text), a movement (timewise interpolation of positions in a bidimensional space), or an action (described using labels). A crude rendering of the actors' position and phrases is presented in real time when the score is played (see fig. 3.47). Scripting tools

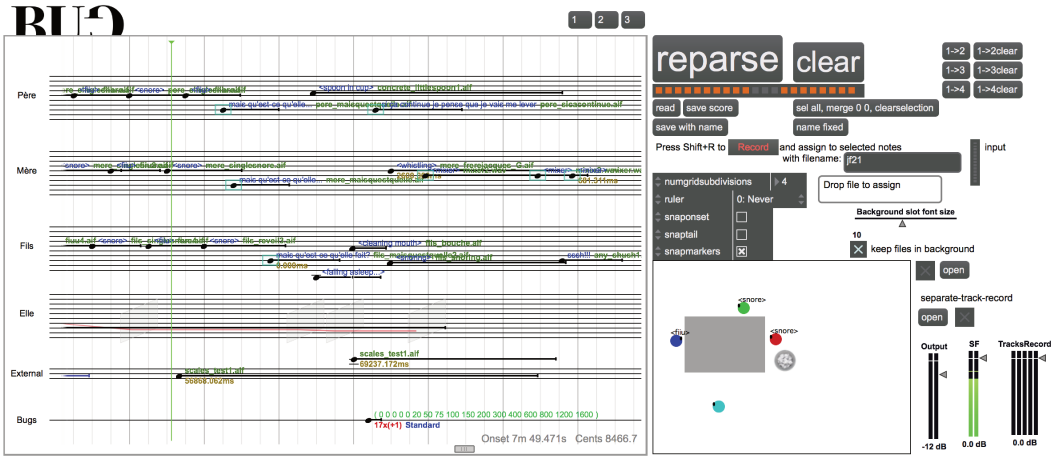


Figure 3.47: Screenshot of the Max patch used to experiment for Bug (quatuor à corps).

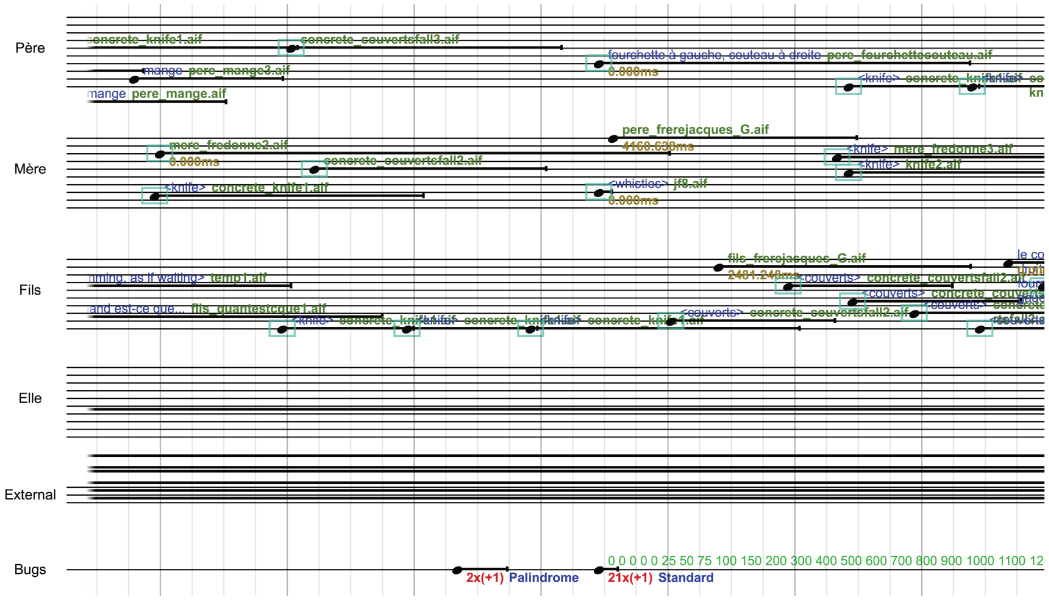


Figure 3.48: A portion of score for one of the situations of Bug (quatuor à corps)*, containing 'bugs'. The audio file containing the corresponding simulation can be found here:* ***data.danieleghisi.com/phd/snd/BUG_ex1.aif|mp3****.*

to automate the creation of canonic voices, by copying and pasting a portion of a given voice, are also set in place.

The score includes a separate 'External' track, containing the 'basso continuo'-like piano scales, and, most significantly, a 'Bug' track, whose notes represent higher-level interventions, looping the portion of score they span, from their onset to their note tail, for a certain number of times. 'Bugs' are timed elements in the scores which affect the time flow itself; they can trigger either standard loops or palindrome loops (sounds and actions going back and forth repeatedly). In addition, the duration of each repetition can be modified individually, in order to obtain, for instance, loops of increasing or decreasing length. During playback, plain 'bugs' trigger standard stop/play messages in order to loop a given portion of score, whereas palindrome 'bugs' require more sophisticated mechanisms of recording and playback. The crude display of actor movements is achieved by dynamically representing the content of a slot of type '3dfunction' into a Max *lcd* editor. The most interesting feature of this score representation, to me, was the possibility to experiment interactively and to search for the most significant positions and parameters of each perturbation, simply by varying the position, duration and slot content of notes in the 'Bug' track—a process that I could not have achieved in any currently available digital audio workstation, due to the combination of sequencing features with programming mechanisms.

Excerpts

- 'Bugs' (audio simulation, see score in fig. 3.48):
  `data.danieleghisi.com/phd/snd/BUG_ex2.aif|mp3`

3.5 Perspectives on the *bach* family

For the time being, *bach*, *cage* and *dada* are the only representatives of the *bach* family, and together they constitute not only a massive development project, but also a research one, with a main focus on integration and usability, touching deeply conceptual topics such as the role of interactivity in musical composition. The set of examples provided in [Ghisi and Agostini, 2017] shows the variety of approaches that can take advantage of the fact that the *bach* family has been designed with this kind of considerations in mind, and at the same time can be seen as a proof of concept of the validity and importance of the research attitude underlying the development of this whole system.

Imagining other libraries to continue the series (and hence the alphabet that *bach* started with 'b') is a thrilling task; Andrea Agostini and I actually have ideas about some categories of modules which might constitute valuable additions to it.

One of these would explore the possibility of linking symbolic representations with audio. Specifically, this might include modules for further narrowing the gap between instrumental and electroacoustic scores through a variety of approaches and tools, such as an audio sequencer completely scriptable via Max messages (just like *bach.score* can be seen as a scriptable score editor) and conceived to be linked to and controlled by *bach*[18]; or a system for converting the contents of a *bach.score* or *bach.roll* object into a Csound score. Another part of this library might be aimed at representing symbolically synthesis processes, possibly reimplementing concepts from Marco Stroppa's *Chroma* system [Stroppa, 2000]. Moreover, tools for manipulating sound samples through a variety of operations can be easily imagined: after all, the only Max data structure explicitly meant to handle sound files, i.e. the *buffer~* object, is not as flexible as one might wish, and several simple operations prove cumbersome to implement. Some progress has recently been made, and Max now allows users to write JavaScript code to create buffers or modify their content. The fact that JavaScript in Max only runs in the low-priority thread is probably a minor issue given the amount of data and CPU time required by such processes. Nevertheless, we believe that simple operations such as reversing, padding, mixing or applying fades to an audio buffer should be achievable with ready-made modules (and not only via JavaScript code), in an straightforward and consistent way. Overall, such a system, which would somehow relate to *bach* like the audio tools in OpenMusic relate to the global library [Bresson, 2006], might allow users to take advantage of the Max graphical programming capabilities to design algorithmic audio machines—something that modern audio sequencers and editors are generally not tailored to do.

Another possible area of research and implementation might involve constraint solving, optimization, machine learning, data mining, and programming paradigms

[18] Although it is true that *bach.roll* can easily represent audio samples and automations through file names and functions attached to notes as slot data, it is by no means a real audio sequencer, because it would be difficult, if not impossible, to add a waveform representation to notes, and because at playtime all the temporal information attached to each note is output at once.

related to artificial intelligence. Although such approaches have been widely used by computer music researchers since decades, very few tools exist that do not require specialized programming skills. Among those, we should mention Bill Vorn's *LifeTools*[19] and the *bios* library for Max[20], all implementing genetic algorithms; the *RLKit*[21], a set of external bringing reinforcement learning into Max and PureData; the OMax system for machine improvisation [Assayag et al., 2006]; and a few systems for solving musical constraints, including PMC [Anders and Miranda, 2011] for PWGL, subsequently ported to OpenMusic as OMCS[22]; Situation [Rueda and Bonnet, 1998] and OMClouds [Truchet et al., 2003] for OpenMusic; the Strasheela system; and the *bach.constraints* object in *bach*, which indeed is a very crude and limited constraint solver, but has already proven useful for tackling several real-life musical problems, including generation of harmonic and rhythmic sequences, fingering for string instruments, and automatic audio montage based on audio descriptors. In the future, it might be interesting to investigate the feasibility of a comprehensive system, implementing both generic and specialized tools (e.g., tools for audio segmentation, or rhythmic and harmonic analysis, and so on) meant to be adopted not only by researchers, but by musicians as well.

[19] http://billvorn.concordia.ca/research/software/lifetools.html
[20] http://bioslib.blogspot.it
[21] http://alab.hexagram.ca/rlkit/index.html
[22] http://repmus.ircam.fr/openmusic/libraries

Chapter 4

Towards a Parameter Entangled Computer-Aided Composition

4.1 An entanglement of parameters

4.1.1 Traditional notation as a Cartesian model

A composer who writes a certain note for flute is probably used to think at the note as disentangled into a certain number of parameters, notated on the staff via independent graphic elements—the pitch mapped on the vertical position of the note head, the duration mapped on the type of flag and head, the intensity mapped on dynamic markings. However, for the flutist that will play that note, pitch, duration and intensity are not exactly independent parameters: depending on the registers, some intensity level might be completely unattainable, or, inversely, notes played *fortissimo* will only be playable for small periods of time. This is indeed well taught in classes about instrumentation and on books about instrumental techniques. As far as notation is concerned, a note is essentially a point in a Cartesian n-dimensional space; however for a flutist, a note is something very different, involving the complex fluid dynamics of a column of air set in motion by blowing, whose properties are also altered via lips and finger movements.

This line of reasoning is obvious—and yet in traditional computer-aided composition we are trained to think of notes exactly as being points in n-dimensional spaces: from the 13th century, occidental music writing has gradually initiated to separate rhythms, pitches and dynamics, using a discrete alphabet for pitches and dynamics, and a tree representation (based, in turn, on a handful of figures) for rhythms. It is not by chance that, even today, didactic examples of CAC usually show how to construct pitches, durations and MIDI velocities somehow independently from each other; for instance, generating a certain melody on a given range, then assigning certain durations and accents.

Partially, this attitude is an heritage of the ancient Greek music theory, with its necessity of quantifying and rationalizing phenomena, leading to the separation of parameters and to the discretisation of data, and of a 'Cartesian' simplification, by which the notational system acquires combinatorial, and hence generative, capabilities [Lévy, 2007].

It should be stated clearly that such disentangled representation of music is a fundamental conquests of Western musical tradition. Via discrete notation, music becomes object of science. An orthogonal representation of parameters favors the

building of simple and elegant algebraic models for music. As Lévy [2014] points out, our 'graphemology' (the ability to transcribe events via signs and signals) has introduced a 'grammatology' (the ability to combine symbols to shape thinking). By using signs to transcribe ideas, we are also unveiling the potential to combine them, which in turns also forges a certain *Weltanschauung*.

Consequently, separating parameters into abstract, independent axes is beneficial, as long as, to quote Grisey [1987], we don't mistake the map with the territory: by using the simplified model repeatedly, there is the particular risk that we eventually forget that it was just a working assumption, a pragmatic necessity.

Due to its simplicity, and for lack of other models, the 'disentangled' paradigm has shaped algorithmic composition since its early days. It was then bequeathed to the Patchwork family of software in the late eighties—a fact exemplified, for instance, in OpenMusic score editors having separate inlets and outlets for each of their classical orthogonal parameters, so that such parameters may be easily queried and entered—and, at the same time, so that the scores themselves could be constructed via a subset of parameters only. The MIDI standard, introduced some years earlier, had been informed by similar ideas. For a survey on parameter representation across different environments, one can read [Anders, 2007, section 2.3].

The *bach* library itself, inspired by the Patchwork family, has adopted the Cartesian approach from its foundations, among other things by maintaining OpenMusic's separate inlets and outlets in *bach.roll* and *bach.score*. It is a choice that today both Andrea Agostini and I regret, to some extent. As an example of environment implementing a somewhat different approach, one can examine the Strasheela project [Anders et al., 2005].

Consider now an electroacoustic composer who was asked to synthesize a simple sound. It is imaginable that he or she would start to construct it via a combination of synthesis techniques, or via sample-based transformations. Composers working in this scenario would be crafting a well rounded object, and it would be at least cumbersome to justify any choice of parameters without taking into account the state of the other ones. This 'trial and error' approach to writing sounds is enhanced whenever the process happens inside a reactive environment—as most synthesis environments are. In a sense, real-time paradigms, by favoring an experimental approach, tend to distance themselves from the reductionism of a mathematical model.

This argument has much in common with the considerations of section 3.1, leading to believe that the very nature of off-line, traditional CAC has somehow endorsed a more 'speculative' and less 'performative' way to model scores. It is probably not by chance that the search for a reasonably standard notation in electronics, let alone for real-time treatments, is still the Holy Graal of modern music notation theory [Manoury, 2007]: traditional notation, an echo of the speculative approach, is unfit to represent complex sounds.

It should however be pointed out that the issue lies partly in the computational models and partly in the limitation of the traditional notation itself. In other words, the loss is twofold: on one hand, classic notation decomposes complex entities into

orthogonal parameters, and by doing so it reduces them to a certain set of their qualities (which is why it is so hard, for instance, to find agreed notations for extended instrumental techniques); on the other hand, traditional CAC environments tend to handle complex scores as if they were a direct sum of separate orthogonal parameters, and not a composite combination of them.

Whereas in the second case the problem appears to lie more in the *modus operandi* than in the actual data representation, the first issue is the inescapable dilemma of notation—a question already raised at the beginning of the twentieth century by a certain number of musicians and researchers. Conductor Leopold Stokowski addressed his concern during the meeting of the Acoustical Society of America in 1932:

> Our musical notation is utterly inadequate. It cannot by any means express all the possibility of sound, not half of them, not a quarter of them, not a tenth of them. We have possibilities in sound which no man knows how to write on paper. [...] Also there would be so much that the composer was trying to express, that he conceived but couldn't write down because of the limitations of notation... One can see ahead a time when the musician who is a creator can create directly into *tone*, not on paper. [Manning, 1985, p. 11]

Stokowski's predictions were opening the route to the realm of sound synthesis, and in doing so he was hinting at the concept of hybrid scores (see section 3.4). And yet the representation of sound that followed was, for the most part, profoundly anchored to the traditional separation of parameters that constitutes the basis of Fourier's theory (frequencies and amplitudes replacing pitches and dynamics). It is somewhat ironic that the interest in noise and timbres triggered by Italian futurists has never truly matched the profound need to notate them more precisely. For instance, Varèse's idea for an electronic notation was akin to a waveform representation, "similar in principle to a seismographic or oscillographic notation" [Manning, 1985, p. 14]—which would of course end up being extremely hard to write manually. Additive synthesis has the potential to represent any complex spectrum, and yet a programmatic notation targeting noisy classes of sounds would be extremely cumbersome to handle; frequency or ring modulations, although representing timbres more concisely, only cover a small class of particularly connotated spectra; sampling-based techniques certainly lack in flexibility and representational power. As a result, after roughly eighty years from those pioneering researches, composers are still searching for an intuitive notation of timbres. We should then ask ourselves whether our representation of sound phenomena into the discrete parameters of rhythm, pitch and intensity is unique and optimal [Lévy, 2014]. Are there other models possessing the grammatological advantages of abstraction without the drawbacks of decomposition into orthogonal parameters?

4.1.2 Why we need some degree of entanglement

The fact that an entangled approach to musical parameters might open new perspectives on computer-assisted composition is exemplified by the following two cases.

A first curious example is the paradox proposed by Sadai [1992][1]. He asked some musicology students to evaluate, with grades ranging from 1 to 10, the rhythmic, harmonic and melodic aspects of the second movement of Beethoven's 7th Symphony (fig. 4.1).

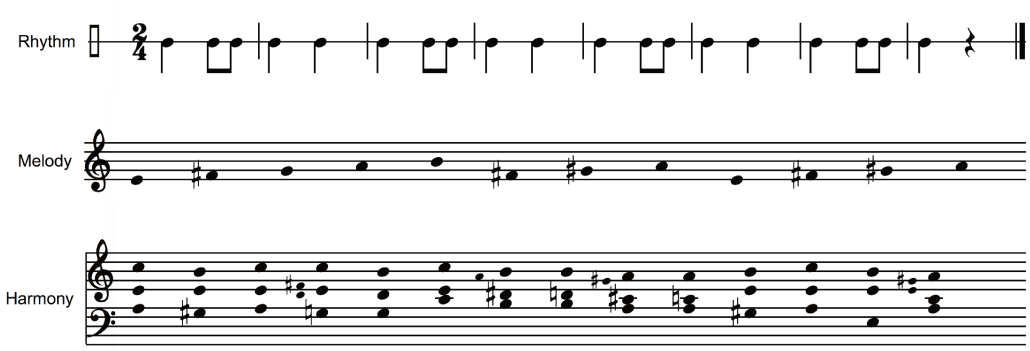


Figure 4.1: Sadai's Paradox.

Evaluation results yielded only 1.6/10 for the rhythm, 2.4/10 for the melody and 3.5/10 for the harmony, which is not really dazzling: taken independently, the three aspects are indeed rather insignificant; and the outstanding musicality lies in their specific combination (for instance, the notation of the harmonic sequence without indication of harmonic rhythm is barely meaningful)—as well as in dynamics, articulations and orchestration.

Sadai's experiment shows that some entanglement of parameters might be beneficial for analysis purposes; and, if one is trained to think of the building blocks of analysis also as building blocks of synthesis, one should therefore conclude that overcoming the limits of disentangled paradigms would be beneficial for composition too.

It might be argued that composers working with CAC can still produce separate parameters whilst having a clear idea of their combination. After all, combining discrete objects while imagining the overall result is exactly what composers are traditionally trained to do, for instance, while orchestrating. There are however other fields where a certain degree of entanglement seems to be an inescapable solution.

Consider the case of a composer who needs to transcribe some proportionally notated material into a traditionally notated score. He or she would face two different but related challenges:

1. finding the position of beats (and downbeats), i.e. tracking tempi and meters (*meter and tempo inteference*)

[1] I owe to [Lévy, 2014, p. 118] the discovery of Sadai's experience.

2. approximating the rhythm to a classically notated one, given tempi and meters (*rhythm quantization*)

These problems are widely studied, and a variety of approaches have been proposed to tackle them, including, but not limited to, hidden Markov models [Takeda et al., 2002], connectionist networks [Desain and Honing, 1989], Bayesian methods [Cemgil, 2004], backtracking algorithms [Agostini and Ghisi, 2015], Monte Carlo methods [Cemgil and Kappen, 2003]. As a matter of fact, the two problems are coupled: on the one hand, the durations cannot be quantized without knowing the tempo (and the meter); on the other hand, the tempo interpretation depends on the quantization [Cemgil, 2004]. A certain number of solutions have been proposed to break this 'chicken-egg' impasse; some algorithms perform the two operations in a connected fashion, such as [Takeda et al., 2002]; not surprisingly, a few of them, namely [Agon et al., 1994] and [Ycart et al., 2016], require a certain amount of interaction by the composer (semi-automatic quantization).

On the other hand, it is rather striking that essentially all of these approaches only focus on rhythmic parameters (onsets and inter-onset intervals), whereas, for tempo inference, intensities, durations, pitches, harmony, and in general any pattern emerging from the combination of these parameters should play a major role in conditioning the feeling of a regular 'beat' (see fig. 4.2).

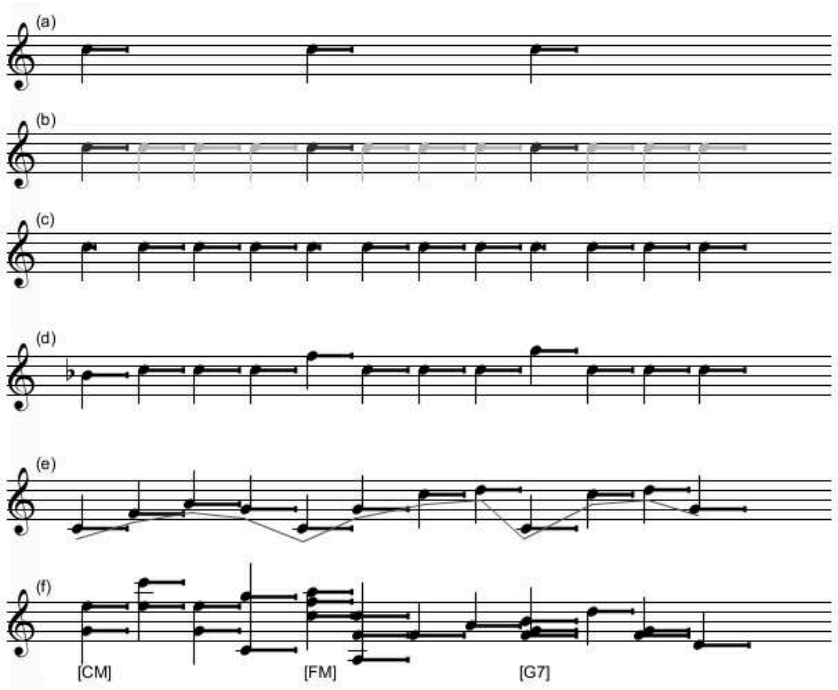


Figure 4.2: The tempo and rhythm perception depends of course on the note onsets and inter-onset intervals (a), but also on their intensities (b), durations (c), pitches (d), melodic contours (e) and harmony (f). In general, the emergence of any pattern involving a combination of parameters will factor into the way we perceive tempo.

Discarding non-rhythmic information is a reasonable assumption in simple case

scenarios, or when the meters and tempi can be adjusted interactively. Moreover, if we assume that additional information (pitches, dynamics, harmonies...) must factor into the algorithm, a web of interrelated dependencies should be unwoven. Fine tuning these relationships might be clumsy: quantization seems to be better learned by examples than by rules. This is why some approaches (such as such as the aforementioned [Takeda et al., 2002]) use vocabularies or sets of examples to model output rhythmic patterns. However, fixed look-up vocabularies usually lack in flexibility and generalization capabilities.

Some kind of entangled representation of data is needed to improve traditional quantization algorithms, which is why some modern approaches take advantage of supervised machine learning techniques, designed to identify and reproduce patterns from raw data. Being essentially data-agnostic, methods relying on artificial neural networks have the advantage of easily including any quantifiable parameter.[2]

These two cases show that some degree of entanglement between musical parameters is often desirable. The next part of this chapter will be dedicated to discussing two partial and incomplete propositions that, following the route traced by each one of these these two examples, try to distance themselves from the disentangled approach: section 4.2 will suggest the usage of a grain-based representation of scores, assuming an essentially traditional representation of musical parameters; section 4.3 will detail my recent experiences with machine learning techniques based on artificial neural networks, as a tool to bypass traditional notation altogether, replacing it with some other abstract, entangled representation of musical data.

[2] I will tackle the implications of machine learning techniques on music more carefully in a later section of this work.

4.2 Grain-based score representations

It should not come as a surprise, by now, the idea that a computer-aided composition based on large score corpora might in a way be beneficial for the integration of musical parameters. The discussion and the examples in section 2.8 should outline a possible framework.

A complex score can be segmented into (windowed, overlapping) grains, using the portion of *dada* described in section 3.3.2. As an example, resuming Sadai's paradox, grading a certain number of grains as in fig. 4.3 would have presumably lead to different ratings for the second movement of Beethoven's 7th Symphony.

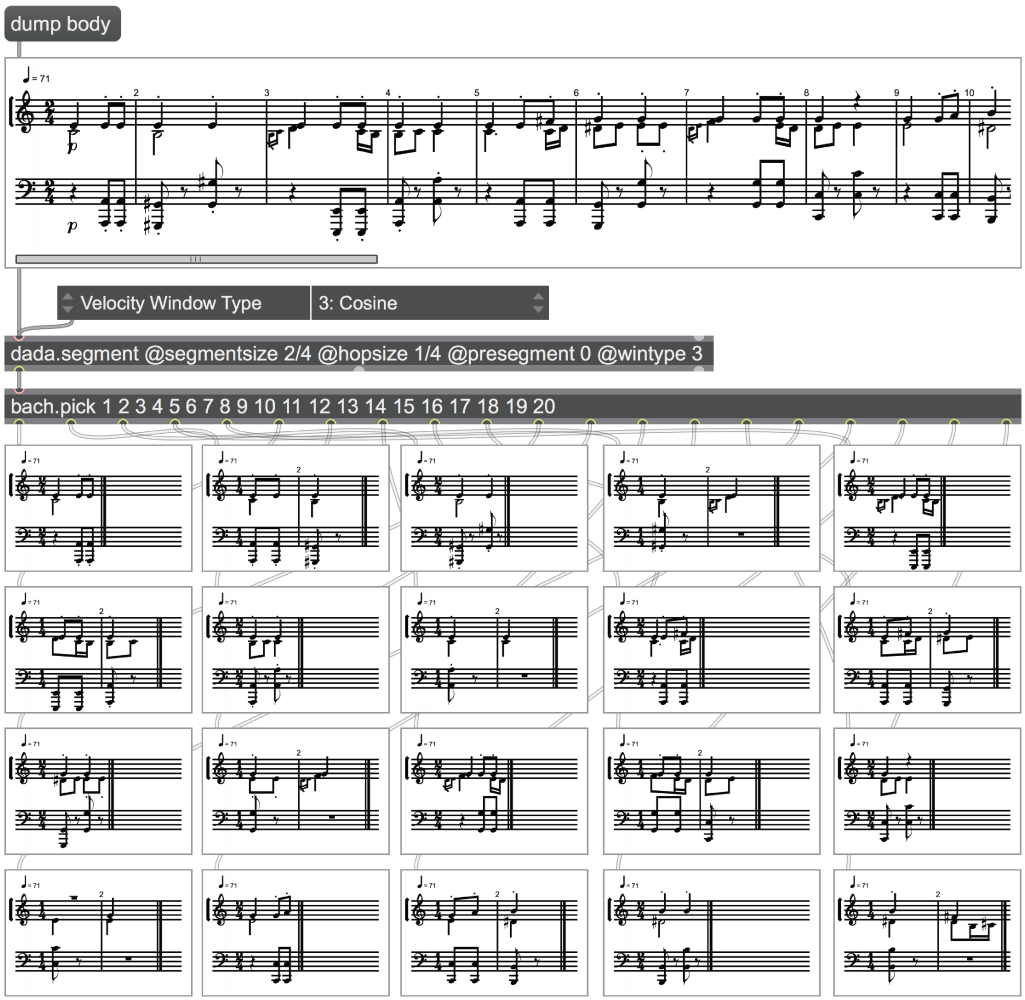


Figure 4.3: Grain extraction from a portion of the second movement of Beethoven's 7th symphony. The first 20 grains are displayed. Although not visible, MIDI velocities are modulated via cosine windows.

Grains can then be analyzed and features can be extracted, in very much the same way as for the digital audio domain techniques. Grains can be combined,

mixed, concatenated; a large number of treatments, such as the ones described in section 3.2.4 can be applied on each individual grain, either directly (e.g., by using the *cage* library) or via the C/C++ functions contained in a subset of the *dada* source code (namely `dada.notation.h` and `dada.notation.c`).

I believe that a grain-based writing might become a natural technique in symbolic composition—as granular and concatenative synthesis have become in the audio domain, for some composers. However, in spite of my interest and usage of symbolic granulation and montage, while dealing with this approach, I have suffered from shortcomings in some respects.

Sometimes grain-based information is not pertinent, especially when the discernment of medium- and long-term time relationships is a crucial feature; for instance, in Grisey's *Le noir de l'étoile*, one might say that the onsets alone are more representative than any short grain of score. This is to me, however, a secondary issue, in spite of which I can still imagine my compositional workflow being widely affected by grain-based score operations.

Another problem concerns splicing. Concatenating seamlessly short portions of audio files is not a trivial task, yet some standard techniques exist (such as simple crossfades). On the other hand, it is more of an effort to concatenate portion of scores (for one thing, instrumental techniques and capabilities should be taken into account), even more so when concatenations include overlapping regions (as one might desire), prompting more important decisions, such as how to handle polyphonic behaviors in monophonic instruments or unplayable double stops for strings. Seamless concatenations are just harder to programmatically employ in the symbolic domain, and the amount of manual work currently needed to make them work seems to almost surpass the benefits.

A third, important, issue involves the size of the datasets. Composers do not have a remotely comparable amount of information for scores as they have for audio—and when dealing with corpus-based composition, more often than not quantity *is* quality. It is true that a certain number of MIDI collections are available online; however, in addition to the fact that their content is often inaccurate, MIDI files represent pitches only via MIDI numbers, leaving virtually no place for microtonality and enharmonic spelling, and implement by design a 'piano roll' format, suitable to represent proportional notation, but much less so hierarchically organized scores. I hardly believe that anyone would handle scores by Berio or Ferneyhough via MIDI files, and, although it is true that one may probably accept MIDI usage with simpler scores, as a working assumption, any framework for grain-based composition with which I may feel at ease should also accommodate more complex notational scenarios. A relatively convenient choice would be to rely on databases of MusicXML scores, a format which represents the *de facto* standard for interchanging notation data; the major issue, in this respect, is the fact that there is simply not enough content available (some promising MusicXML-based projects, such as Wikifonia[3], have been discontinued due to copyright issues).

[3] http://www.wikifonia.org

4.3 Artificial neural networks: a composer's take

Let us revert to the question at the end of section 4.1.1: is there a way to create generative abstract models that do not decompose music into orthogonal parameters?

We have seen in previous section that a grain-based approach to scores has the benefit of entangling parameters, to some extent; however, not only is its generative power limited by the underlying corpus of segmented scores and by the ensemble of techniques to modify score grains, but most importantly the approach does not avoid the decomposition of parameters ontologically pertaining to the notation medium itself.

One might object that this urge to bypass notation clashes against with my deep interest in real-time score handling, discussed in chapter 3—and on some level, it probably does. But I believe that only within these creative oppositions one can search for new meanings of common practices: a new take on notation may emerge from a new take on music representation, which in turn may arise from experiments in radical scenarios where the traditional symbolic approach is dismantled.

Learning abstract representation of patterns from a collection of examples is precisely what machine learning techniques are tailored to do. The term 'machine learning' refers to a family of algorithms that allow computers to automatically construct abstract representations of problems by learning them from data of some kind. More specifically, artificial neural networks (or neural networks, for short) have been developed to automatically detect patterns in data, and then to use such patterns to classify new data or to predict future data. Rather importantly, while classifying, they often build in their deepest layers a latent (possibly entangled) representation, providing interesting information on some (possibly non-standard) characteristics of the input.

Machine learning and neural networks are already widely used in MIR scenarios, for tasks, among others, such as genre recognition, mood detection, music classification or clustering, onset detection, beat tracking, denoising. There is vast literature on each of these tasks; a comprehensive overview of machine learning in music can be found in [McKay, 2010].

Neural network models can be either supervised (labels are entered by the user for each example in the training base) or unsupervised (no label information is needed). Unsupervised neural network models require minimal or no explicit human description of a problem: this reveals crucial in problems where manually introducing expert knowledge would be cumbersome or impossible, for instance because there is no consensus on the formulation of the problem, or because the amount of information to be inserted is too big, or even because human experts can have unconscious biases. In contrast, unsupervised learning does not even require any formalized understanding of the problem.

The number of neural network applications to musical composition is amazingly large—and, most importantly, it is constantly growing, since machine learning is a currently popular and flourishing field of research. One can roughly distinguish

between two classes of approaches:

1. a note-oriented approach, where generation happens on symbolic data. One of the first examples is probably [Todd, 1989]; more modern ones are [Liu and Ramakrishnan, 2014; Huang and Wu, 2016; Roberts et al., 2014; Johnson, 2017]. Training data for symbolic methods is usually extracted from MIDI files; such methods are widely employed on baroque, classical and romantic datasets, but hardly on contemporary ones;

2. a signal-oriented approach, where the information only comes from raw audio data. One of the popular models is arguably WaveNet [van den Oord et al., 2016], immediately followed by a certain number of other models—most notably SampleRNN [Mehri et al., 2016], which I will inspect in more detail in section 4.3.2.

I have focused in the past few years on the very specific case of unconditional sample-based audio generation, which I have used for a recent project, *La fabrique des monstres*. This case is representative of a radical point of view: any other approach would lead to the enforcing of some sort of higher-level representation of music (descriptors or scores), in order to characterize musical signals in an abstract or symbolic way [Cella, 2011]. Using direct signal samples in order to build unsupervised representations of sound is a natural way to maximally enforce the original entanglement request.

4.3.1 Sample-based generative networks for *La fabrique des monstres*

The basic idea for my musical work for the play *La fabrique des monstres*, by Jean-François Peyret, is to explore the possibilities of a musical 'machine' able to 'listen' and somehow 'learn' from different corpora of sounds (starting from Schubert's Lieder and Symphonies, but then widening the range to modern, contemporary and rock music), and creatively reproduce the learned patterns—the learning process being the key, more than its actual endpoint.

The work questions the idea of 'who wrote what'. As it should be clear by now, musical invention to me is a more disguised form of exploration. In this sense, although the corpus of sounds used to train the network is partly based on romantic music, the overall approach is extremely anti-romantic.

As Bruno Bossis writes[4]:

> From the artistic point of view, a human creation whose result (the machine) if not the goal (imitating man) is inhuman does not fail to question. Artificial and natural, despite their apparent opposition, blend in the concept of artificial man. [Bossis, 2005, p. 34]

[4]Translated from the original in French: "Du point de vue artistique, une création humaine dont le résultat (la machine) sinon le but (imiter l'homme) est inhumain ne manque pas d'interroger. Artificiel et naturel, malgré leur opposition apparente, se rejoignent dans le concept d'homme artificiel."

The artificial life, and even artificial life via sound phenomena, is a topos in art. Well before Mary Shelley's Frankenstein, in Jules Verne's *Le Château des Carpathes* the Baron de Gortz tries to have Stilla live and sing again by recreating her voice via electric procedures. The technical object is unsettling and demoniac: it contains the 'soul' of the dead singer. Even among the milieu of contemporary composers, there is some interest in the topic, one of the most prominent examples being Mauro Lanza's catalogue, which often treads on the boundary between automata and living beings, such as in his *Anatra digeritrice (piccola Wunderkammer di automi oziosi)*.

In the case of *La fabrique des monstres*, the research was driven by an unconditional, generative spirit: letting a computer model undergo a learning process and produce a large amount of output results (a reference to Samuel Beckett's *Krapp's Last Tape* is rather explicit) at different points during training; these results would then be organized later into a palette, which I would then explore, and with which I would engage in further composition.

This additional composition step is, above all, a careful selection of interesting segments—the implication being that, once again, composing has more to do with 'discovering' than 'inventing', and that 'listening' is already an important creative task—, combined with basic montage and mixing techniques. The aim is to prevent the result from being unmistakably definable as 'human made'; but at the same time, especially towards the end of the piece, the result should be also hardly definable as 'machine made': as previously stated, it is supposed to raise the question of 'who wrote what'.

During the last year and a half, I have been exploring these ideas together with IRCAM computer-music assistant Robin Meier, supported by IRCAM and NaMi Lab researchers. A certain number of desired properties guided us:

Coherence. The result should sound unitary, non-fragmented, as fluid and organic as possible; no junction points should be identifiable. For instance, when applied on a speech corpus, the generated inflections should sound very natural; when applied on a symphonic corpus, the orchestral generation should sound substantially plausible.

Concatenations of short grains, which usually lead to grainy, jagged results, where the montage technique remains mostly well noticeable, were therefore ruled out. One could imagine to apply further spectral routines to smooth micromontages, but the implications were beyond the scope of our research: we were indeed looking for a 'native', intrinsic way to obtain seamless structural development.

Agnosticism. The algorithm should reduce any *a priori* knowledge on input data to a minimal amount: it should be able to 'listen and learn' without having any information on musical categories (genres, styles...), descriptions (features, descriptors...) or representations (notes, chords, scores...). In other words, the machine should tend to use crude pulse-code modulation (PCM) data, or some of its Fourier-like representations; no audio descriptors nor symbolic represen-

tations should be used, since the first are by design reduced, 'meaningful' representation of sounds, and the latter imply abstract notions of 'note' or 'score'. Given the absence of any choice of a tailored representation, agnostic approaches are sometimes classed as 'brute-force' techniques.

In addition, the agnosticism request suggests leaning towards both unsupervised learning and unconditional network models.

As a side note, it is important to underline that this request is not about any kind of objectivity; no agnostic model is actually 'objective': the type of representation of input data is still crucial, and careful 'human' consideration is put into selecting it. The goal, however, is to keep such representation as faithful as possible with respect to the way we perceive sounds (also see section 4.3.3.3 on this subject).

Generativity. The output should be an arbitrarily long generation of samples or frames; in other words, the temporality of the generated sound should be independent from the duration of the sounds in the corpus.

This ruled out cross-synthesis or hybridation approaches, or any other technique where the temporality of the constituent elements forces the temporality of the result. Predictive algorithms are a better fit for this requirement.

Descriptor-based representation of sounds were, once more, to be rejected: descriptors are analysis features which in almost no case are designed to be used in resynthesis, so that most of them do not admit a musically acceptable reconstruction: direct reconstruction via sound-shaping or source-filtering techniques is usually demonstrative or cumbersome, while nearest-neighbour search generally leads back to a jumbled micromontage.

I did explore, however, MFCC-based representations, but the path was abandoned precisely due to the difficulty of reaching an adequate resynthesis. One might consider using sound-types [Cella, 2011] as an intermediate step between signal and symbol; however, since they are descriptor-based, they also suffer from similar shortcomings.

The request for an arbitrary long flow of generated samples also rules out any method leading to the generation of fixed-size audio fragments. Nonetheless, I still investigated spectrogram-based approaches, due to the effectiveness of convolutional networks; in this case generation will be achieved on fixed-length fragments—the number of which is, however, arbitrary.

Originality. The result should not reproduce exact portions of existing sound files. It should pertain more to the realm of synthesis than to the realm of sampling. This ruled out Factor Oracle-based improvisation models such as OMax [Assayag et al., 2006] or CatOracle [Einbond et al., 2016].

More generally, this ruled out concatenations of long fragments—with the possible exceptions of microsounds, which in turn, in the meaning of Roads

[2004], can be almost seen as single frames of an STFT windowing process. When using single STFT frames in a generative process such as probabilistic sampling or Markov chains, the result would not be entirely 'original' at a microtime scale, but would be still original at a scale where musical perception plays a larger role.

Models that satisfies all the requests presented above are sample-by-sample (or frame-by-frame) unconditional generative models. Hence, our goal for *La fabrique des monstres* was to explore neural network structures that might be apt for sample- or frame-based unconditional audio generation. In the following sections I will detail the results of an exploration.

4.3.2 Recurrent network models

Most of my attention was captured by the application of neural networks to a sequence of pulse-code modulation values: this approach has the advantage to truly allow a sample-by-sample generation, at the price of time-consuming GPU processes, even for relative low sample rates.

Among the simplest neural network models allowing iterate generation of samples are recurrent neural network (RNN) and long-time short memory network (LSTM), trained via backpropagation through time (BPTT). Standard LSTM models, such the ones described by Karpathy et al. [2015] and its Torch implementations [Karpathy, 2015; Johnson, 2016], performed rather well on speech corpora, even when relatively small datasets were provided during learning.

Model: torch-rnn [Johnson, 2016]
Corpus: *Reading of some chapters from Mary Shelley's* Frankenstein *(about 30')*
Parameters: *4 layers, each of size 512, 1650 BPTT steps, dropout: 0.25*
Sound quality: *8 bit, 16kHz*

- Iteration 1000: `data.danieleghisi.com/phd/snd/NN_torchrnn_voice_i1000.aif|mp3`
- Iteration 7000: `data.danieleghisi.com/phd/snd/NN_torchrnn_voice_i7000.aif|mp3`
- Iteration 19000: `data.danieleghisi.com/phd/snd/NN_torchrnn_voice_i19000.aif|mp3`
- Iteration 48000: `data.danieleghisi.com/phd/snd/NN_torchrnn_voice_i48000.aif|mp3`
- Iteration 120000: `data.danieleghisi.com/phd/snd/NN_torchrnn_voice_i120000.aif|mp3`

Quality of result dropped on more complex musical datasets, such as piano pieces, even when increasing the network size and dramatically lowering the sample rate.

Model: torch-rnn [Johnson, 2016]
Corpus: *Beethoven Sonata Op.109 (about 40')*
Parameters: *6 layers, each of size 512, 26000 BPTT steps, dropout: 0.25*
Sound quality: *8 bit, 4kHz*

- Iteration 6000: `data.danieleghisi.com/phd/snd/NN_torchrnn_beeth_i6000.aif|mp3`
- Iteration 15000: `data.danieleghisi.com/phd/snd/NN_torchrnn_beeth_i15000.aif|mp3`
- Iteration 54000: `data.danieleghisi.com/phd/snd/NN_torchrnn_beeth_i54000.aif|mp3`
- Iteration 86000: `data.danieleghisi.com/phd/snd/NN_torchrnn_beeth_i86000.aif|mp3`

DeepMind's Wavenet improved the recurrent model by taking advantage of dilated temporal convolutions [van den Oord et al., 2016]. At the time of writing, Google hasn't released the source code for Wavenet; however, several open-source implementations based on the cited article are available; some of them also feature important improvements in the speed of the generation algorithm, based on [Paine et al., 2016]. Among the latter, I obtained a few timid improvements on the piano corpus via the Tensorflow implementation [Babuschkin, 2016].

Model: tensorflow-wavenet [Babuschkin, 2016]
Corpus: *All Beethoven Sonatas (about 10h)*
Parameters: —
Sound quality: *16 bit, 22050Hz*

- Iteration 12000: `data.danieleghisi.com/phd/snd/NN_tfwavenet_beeth_i12000.aif|mp3`
- Iteration 27000: `data.danieleghisi.com/phd/snd/NN_tfwavenet_beeth_i27000.aif|mp3`
- Iteration 56000: `data.danieleghisi.com/phd/snd/NN_tfwavenet_beeth_i56000.aif|mp3`
- Iteration 92000: `data.danieleghisi.com/phd/snd/NN_tfwavenet_beeth_i92000.aif|mp3`

The best open-source model I could test and adapt, providing unconditional sample-by-sample generation is, arguably, SampleRNN, described in [Mehri et al., 2016] and implemented in [Mehri, 2016]. The model combines memory-less modules (autoregressive multilayer perceptrons) and stateful recurrent neural networks in a hierarchical structure capable to capture sources of variations in the temporal sequences over longer time spans. The generation appears to be more delicate than the previously cited models; depending on the corpus, it may easily end up into noise. Reducing the generation temperature, or changing the kind of data normalization, help to mitigate this inconvenience. When the generation does not steer into a noisy region, it largely outperforms the previous models on all the musical datasets I have used, and has results at least comparable to them on speech datasets.

Model: SampleRNN [Mehri, 2016]
Corpus: *Schubert Symphonies (about 5h)*
Parameters: *3 tier, 3 layers, each of size 512, 2 samples per frame in tier 2, 8 samples per frame in tier 3*
Sound quality: *8 bit, 16kHz*

- Epoch 0: `data.danieleghisi.com/phd/snd/NN_samplernn_schubsymph_e0.aif|mp3`
- Epoch 1: `data.danieleghisi.com/phd/snd/NN_samplernn_schubsymph_e1.aif|mp3`
- Epoch 23: `data.danieleghisi.com/phd/snd/NN_samplernn_schubsymph_e23.aif|mp3`
- Epoch 29: `data.danieleghisi.com/phd/snd/NN_samplernn_schubsymph_e29.aif|mp3`

Model: SampleRNN [Mehri, 2016]
Corpus: *René Descartes,* Discours sur la Methode *(about 3h)*
Parameters: *2 tier, 3 layers, each of size 512, 16 samples per frame in tier 2*
Sound quality: *8 bit, 16kHz*

- Epoch 1: `data.danieleghisi.com/phd/snd/NN_samplernn_descartes_e1.aif|mp3`
- Epoch 14: `data.danieleghisi.com/phd/snd/NN_samplernn_descartes_e14.aif|mp3`
- Epoch 42: `data.danieleghisi.com/phd/snd/NN_samplernn_descartes_e42.aif|mp3`
- Epoch 69: `data.danieleghisi.com/phd/snd/NN_samplernn_descartes_e69.aif|mp3`

Model: SampleRNN [Mehri, 2016]
Corpus: *Late medieval masses (about 18h)*
Parameters: *3 tier, 4 layers, each of size 512, 8 samples per frame in tier 2, 16 samples per frame in tier 3*
Sound quality: *8 bit, 16kHz*

- Epoch 19: `data.danieleghisi.com/phd/snd/NN_samplernn_choir_e19.aif|mp3`
- Epoch 23: `data.danieleghisi.com/phd/snd/NN_samplernn_choir_e23_temp0.9.aif|mp3` (temperature 0.9)

Due to the quality of its results, SampleRNN was selected as main tool for *La fabrique des monstres*, and it was explored in depth. Thanks to Léopold Crestel, a 4-tier version of the algorithm was also tested; its training was however an order of magnitude slower than the 3-tier model, and hence rather impractical to use, yielding results comparable to the 3-tier model in about 5 times the training time.

An important drawback of this class of models is the fact that sample-by-sample prediction takes a large amount of computing time.

4.3.3 Using visual representations of sound

One can take advantage of the state-of-the-art models for convolutional neural networks (CNNs) by converting sounds into images. Such conversion is usually performed by representing a fragment of sound as a grayscale spectrogram, each column being one STFT window, with the darkness of each pixel representing the amplitudes of the bins. The discarded phase information can possibly be reconstructed later via iterative algorithms such as [Griffin and Lim, 1984]. Preserving phases during learning is also feasible, for instance by mapping them on a color scale (such as the amount of red in an RGB representation), or by using a real-valued transform instead, such as the modified discrete cosine transform (MDCT). Nevertheless, this is impractical, as it seems to make the learning process much more difficult.

The prime benefit of using image-based music representation lies in the fact that convolutional networks models are extremely fast, when compared to sample-by-sample RNNs. Differently from standard fully-connected models, CNNs enforce local connectivity patterns (receptive fields) between neurons of adjacent layers, making sub-real-time generation possible. Moreover, CNNs are used so extensively in the visual domain that a large number of reliable models are available, some of which also have gained a certain amount of popularity (such as the well known Google DeepDream [Mordvintsev et al., 2015]).

Two families of algorithms seem to be most promising: generative adversarial networks and autoencoders.

4.3.3.1 Generative adversarial networks on spectrograms

Generative adversarial networks (GANs) are a system of two neural networks competing against each other [Goodfellow et al., 2014]. The first network, the 'generator', is taught to map from a latent space to a particular data distribution of

interest, while the second network, the 'discriminator', is taught to distinguish between instances from the true data distribution and synthesized instances produced by the generator. The generator's aim is to fool the discriminator. GANs have been widely used in computer vision tasks, in conjunction with convolutional techniques (deep convolutional generative adversarial networks, or DCGANs [Radford et al., 2015]).

I decided to test the DCGAN model [Chintala, 2016] on different databases of spectrograms in square form, with 128 pixels per side. The model learns very quickly, and is able to reconstruct voice formants; however, artefacts due to phase reconstruction are well hearable in the audio samples. Some results are given below, both as image and as audio (generated spectrograms are chained with small silence between them). For the sake of completion, in one case, an example of learning via modified discrete cosine transform (MDCT) is given, although its result present annoying spectral artefacts, and the quality is well below the STFT algorithm.

Model: DCGAN.torch [Chintala, 2016]
Corpus: *Reading of some chapters from Mary Shelley's* Frankenstein *(about 30')*
Parameters: *128x128 DCGAN*
Sound quality: *8 bit, 16kHz*

- Epoch 1: `data.danieleghisi.com/phd/snd/NN_dcgan_frankenstein_16khz_e1.aif|mp3`
- Epoch 11 (see fig. 4.4):
  `data.danieleghisi.com/phd/snd/NN_dcgan_frankenstein_16khz_e11.aif|mp3`
- Epoch 11, morphing (see fig. 4.5):
  `data.danieleghisi.com/phd/snd/NN_dcgan_frankenstein_16khz_e11line.aif|mp3`
- With MDCT, epoch 5:
  `data.danieleghisi.com/phd/snd/NN_dcgan_frankenstein_mdct_16khz_e5.aif|mp3`

Model: DCGAN.torch [Chintala, 2016]
Corpus: *A selection of Luciano Berio's orchestral works (about 1h)*
Parameters: *128x128 DCGAN*
Sound quality: *8 bit, 8kHz*

- Epoch 1: `data.danieleghisi.com/phd/snd/NN_dcgan_berio_8khz_e1.aif|mp3`
- Epoch 3: `data.danieleghisi.com/phd/snd/NN_dcgan_berio_8khz_e3.aif|mp3`
- Epoch 9: `data.danieleghisi.com/phd/snd/NN_dcgan_berio_8khz_e9.aif|mp3`
- Epoch 43: `data.danieleghisi.com/phd/snd/NN_dcgan_berio_8khz_e43.aif|mp3`
  (See fig. 4.6)

An interesting feature of GANs is the ability to interpolate in the latent space, which in turns yields a collection of interpolated sounds. This morphing effect is very peculiar and different from standard morphing techniques; for instance fig. 4.5, and the corresponding audio example, show that morphing has captured some characteristics of the voice, and the complex modifications of the formants are well visible in the figure.

Another peculiar feature, as shown by Mikolov et al. [2013a] in the context of evaluating learned representations of words, is the fact that latent space models,

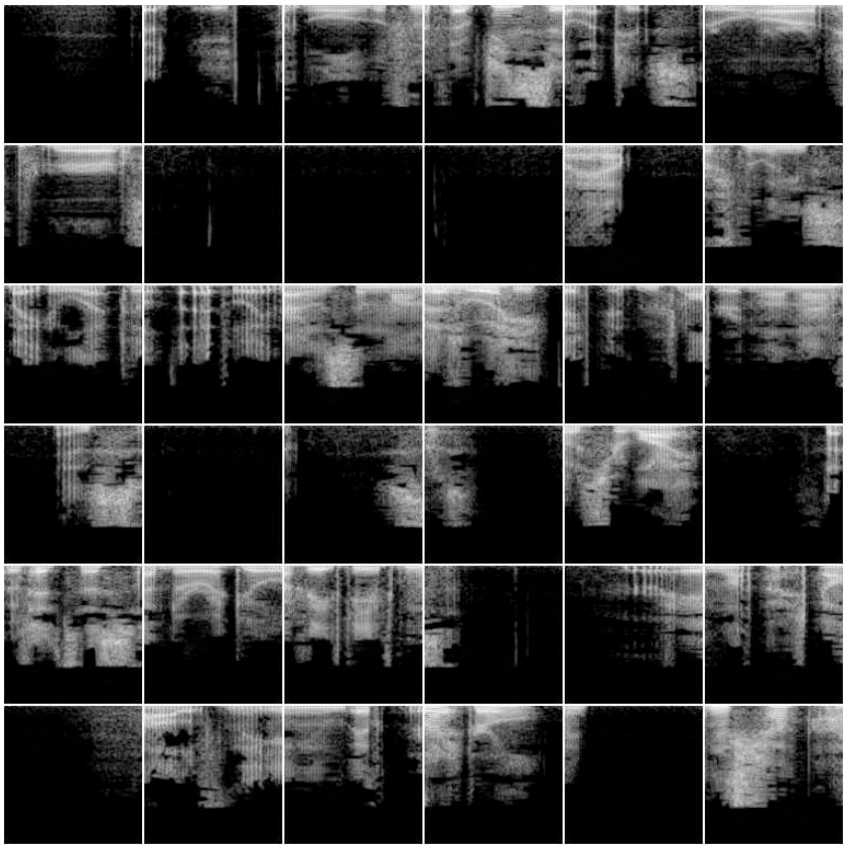


Figure 4.4: Spectrograms generated via DCGANs [Chintala, 2016], trained on a corpus of chapters from Mary Shelley's Frankenstein.

somewhat surprisingly, can reveal rich linear structure in their representation. The classic example is that, if 'vec' represents the function mapping a word to its latent representation vector, then $\text{vec}(Madrid) - \text{vec}(Spain) + \text{vec}(France)$ yields a result whose nearest neighbor is $\text{vec}(Paris)$ [Mikolov et al., 2013b].

One might be interested in developing similar tools for music. This would actually provide an interesting take on some sort of 'meaning' of music from an unconventional perspective. However, in order to generate a spectrogram with specific features, there is no way of determining which initial noise values would produce that result, other than searching over the entire distribution, which is at best impractical.

Another downside of GANs is their tendency to repeat the same few patterns while generating; an example of this is well visible with the generation on the corpus of Berio's orchestral works. The issue is well studied and it is known as *mode collapse*; some techniques have been devised to mitigate its magnitude [Metz et al., 2016].

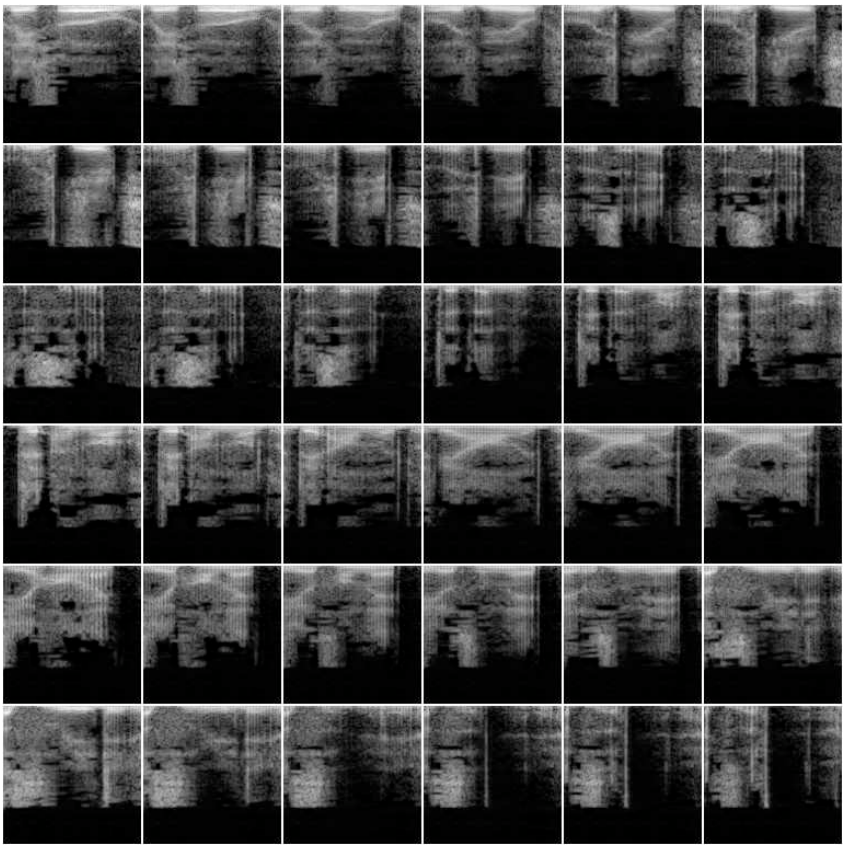


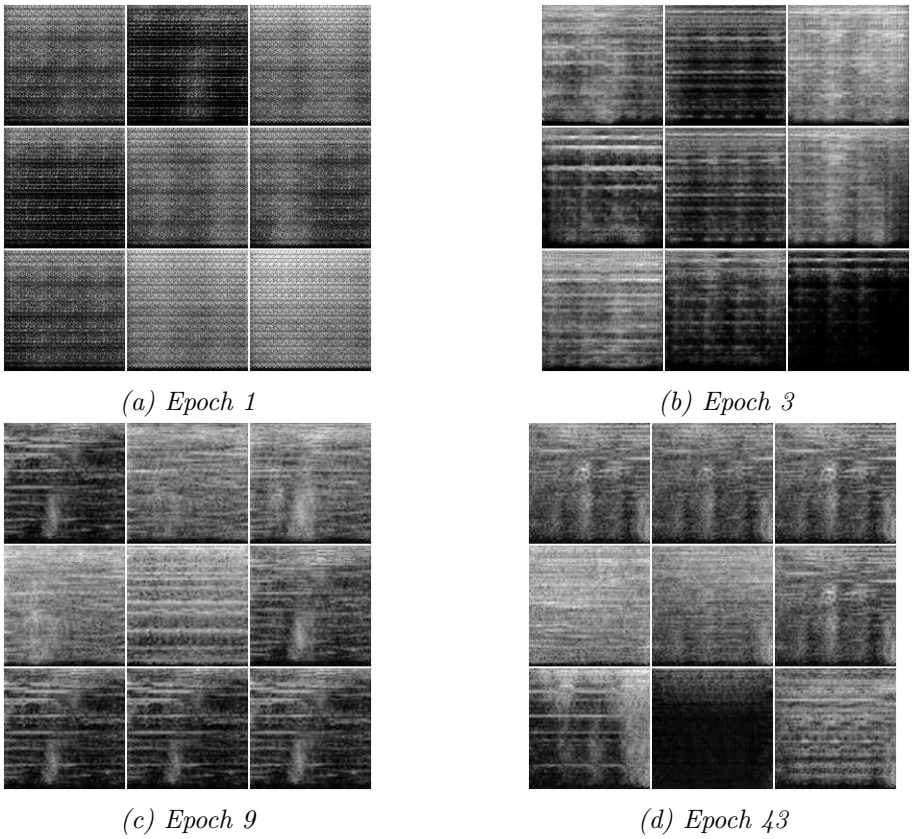
Figure 4.5: The DCGANs model renders a segmented line in the latent space into a morphing between audio segments.

4.3.3.2 Autoencoders on spectrograms

Autoencoders are neural networks performing unsupervised learning of efficient codings for input data. For any introduced sample, the first layers ('encoder') provide a representation in a (usually low-dimensional) latent space; then, the last layers ('decoder') try to reconstruct, from the latent space encoding only, the original sample. Convolutional autoencoders, in particular, obtain the latent space representation via a stacked sequence of convolutional layers. Since their explicit goal is latent space modeling (a low-dimensional representation of data), autoencoders are closely related to compression.

Variational autoencoders merge unsupervised deep learning with variational Bayesian methods [Kingma and Welling, 2013]. Interestingly, disentangled variational autoencoders can learn to separate different features on different dimensions of the latent space, providing a non-conventional (and potentially interesting) disentanglement of musical parameters.

At present, the major downside of autoencoders reside in their blurriness, since they are usually trained with direct mean squared error instead of an adversarial network. Such issue becomes critical on spectrograms, which rely on thin partial



(a) Epoch 1 *(b) Epoch 3*

(c) Epoch 9 *(d) Epoch 43*

Figure 4.6: Spectrograms generated via DCGANs [Chintala, 2016], trained on a corpus Berio's orchestral works.

profiles to represent harmonic components. One is tempted to conclude that the a spectrograms may not be the most suitable representation of music to be used within these models. Hybrid network models have been proposed, which are only convolutional in the time axis, and fully connected in the frequency axis [Stowell, 2016]. Which other two-dimensional representation of sound would be better fit for autoencoders is an open question, briefly addressed in the following section.

4.3.3.3 A quest for a new representation

The representation of sound as spectrogram is inadequate when used in combination with visual machine learning techniques, mostly because such techniques rely on the existence of local patterns, detected by the convolutional filters, whereas, even for simple harmonic sounds, partials on a spectrogram are well spread in the vertical direction (even an interval of octave is represented via disconnected components).

Hence, artefacts such as the blurriness of autoencoders lead to failures in recognizing pitches, especially given the fact that with currently available GPUs one can hardly increase resolution over 256 pixels per dimension. It should not be surprising, then, that noise patterns are better dealt with than harmonic ones.

E4	E5	B5	E6	G+6	B6	D-7	E7	F#7	G+7	A+7	B7	C+8	D-8	D+8	E8	F8	F#8	G8	G+8
Eb4	Eb5	Bb5	Eb6	G6	Bb6	C+7	Eb7	F7	G7	A-7	Bb7	B+7	C+8	D8	Eb8	E8	F8	F#8	G8
D4	D5	A5	D6	F#6	A6	B+6	D7	E7	F#7	G+7	A7	B-7	B+7	C#8	D8	Eb8	E8	F8	F#8
C#4	C#5	Ab5	C#6	F6	Ab6	B-6	C#7	Eb7	F7	G-7	Ab7	A+7	B-7	C8	C#8	D8	Eb8	E8	F8
C4	C5	G5	C6	E6	G6	A+6	C7	D7	E7	F+7	G7	A-7	A+7	B7	C8	C#8	D8	Eb8	E8
B3	B4	F#5	B5	D+6	F#6	A-6	B6	C#7	D+7	E+7	F#7	G+7	A-7	Bb7	B7	C8	C#8	D8	D+8
A+3	A+4	E+5	A+5	D-6	E+6	G+6	A+6	B+6	D-7	Eb7	E+7	F#7	G+7	A-7	A+7	B-7	B+7	C+8	D-8
A-3	A-4	E-5	A-5	C+6	E-6	F#6	A-6	B-6	C+7	D7	E-7	F7	F#7	G+7	A-7	A+7	B-7	B+7	C+8
G3	G4	D5	G5	B5	D6	E+6	G6	A6	B6	C+7	D7	E-7	E+7	F#7	G7	Ab7	A7	Bb7	B7
F+3	F+4	C+5	F+5	A+5	C+6	Eb6	F+6	G+6	A+6	B6	C+7	D7	Eb7	E+7	F+7	G-7	G+7	A-7	A+7
E3	E4	B4	E5	G+5	B5	D-6	E6	F#6	G+6	A+6	B6	C+7	D-7	D+7	E7	F7	F#7	G7	G+7
D3	D4	A4	D5	F#5	A5	B+5	D6	E6	F#6	G+6	A6	B-6	B+6	C#7	D7	Eb7	E7	F7	F#7
C3	C4	G4	C5	E5	G5	A+5	C6	D6	E6	F+6	G6	A-6	A+6	B6	C7	C#7	D7	Eb7	E7
A+2	A+3	E+4	A+4	D-5	E+5	G+5	A+5	B+5	D-6	Eb6	E+6	F#6	G+6	A-6	A+6	B-6	B+6	C+7	D-7
G2	G3	D4	G4	B4	D5	E+5	G5	A5	B5	C+6	D6	E-6	E+6	F#6	G6	Ab6	A6	Bb6	B6
E2	E3	B3	E4	G+4	B4	D-5	E5	F#5	G+5	A+5	B5	C+6	D-6	D+6	E6	F6	F#6	G6	G+6
C2	C3	G3	C4	E4	G4	A+4	C5	D5	E5	F+5	G5	A-5	A+5	B5	C6	C#6	D6	Eb6	E6
G1	G2	D3	G3	B3	D4	E+4	G4	A4	B4	C+5	D5	E-5	E+5	F#5	G5	Ab5	A5	Bb5	B5
C1	C2	G2	C3	E3	G3	A+3	C4	D4	E4	F+4	G4	A-4	A+4	B4	C5	C#5	D5	Eb5	E5
C0	C1	G1	C2	E2	G2	A+2	C3	D3	E3	F+3	G3	A-3	A+3	B3	C4	C#4	D4	Eb4	E4

Figure 4.7: Rearrangement of bins of a STFT window (first column) into a 'frequency-domain' Tonnetz.

In order to use convolutional networks on sounds, one might want to employ, in principle, a representation that undergoes the following properties:

(i) two points near each other should represent 'objects' that are connected on a perceptual level;

(ii) conversely, 'objects' that are connected on a perceptual level should be represented by points that lie near each other;

(iii) the number of dimensions should be as low as possible;

(iv) the representation should be invertible (perfect reconstruction).

Standard spectrograms satisfy conditions (i), (iii) and (iv), but they do not satisfy condition (ii)—straightforward counterexamples is any harmonic series, or any octave interval.

If one wants to tackle condition (ii), one pitch dimension is too tight a constraint, and hence condition (iii) must be loosened. In this case, one might envisage a solution involving a two-dimensional rearrangement of each spectrogram column (each STFT window) so that harmonic partials end up being closer to each other.

The quest for a music representation where close relationships between pitches are mirrored on spatial proximity is not new [Chew, 2000], and it is especially studied in the context of geometrical representation of tonal relationships. A classic approach makes use of the Tonnetz, already introduced in section 3.3.4.1: a bidimensional lattice generated by two base intervals. It is an interesting—and perceptually relevant—representation, inasmuch as it connects each note with its first non-trivial

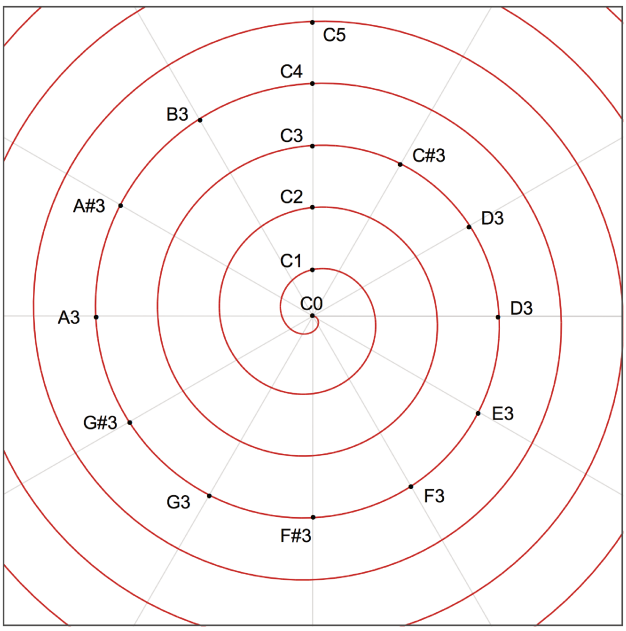


Figure 4.8: Spiral of pitches.

harmonics. Nonetheless, simple octaves are still far apart. The same holds true for comparable structures, such as the helicoidal model proposed by Elaine Chew.

One might consider, instead of a Tonnetz, simplicial chord spaces [Bigo et al., 2015], where dimensions are related to the first few harmonics (octave included), but this would rapidly increase the number of dimensions, violating (iii). Alternatively, one might consider rearrangement of spectrogram column pixels in a 'frequency-domain' fashion, as also described in [Bigo et al., 2015] (see fig. 4.7), essentially duplicating the spectrogram columns to represent harmonic series of a certain number of its lower bins. However, even though in this representation packs adjacent harmonics as close together as possible, it performs extremely poorly on unisons, since pitches now appear multiple times in disconnected regions of the lattice. It is fascinating to imagine a topological space built in a similar fashion, once unisons are identified; in practice, however, this seems to be intractable.

A more concrete approach would be tackling point (ii) only with respect to the relationship of octaves (i.e. restricting the request to the first harmonic). A natural solution is to flatten the helicoidal model onto a spiral, as both described by Lostanlen and Mallat [2016] and independently implemented in the *Snail* tool by Hélie et al. [2016]. Consider a representation where the frequency axis is unrolled along an Archimedean spiral (the center being a low root frequency, and the external region corresponding to the high frequencies), so that each complete turn corresponds to doubling the frequency. All frequency components in octave relationship would then be aligned on the same angles. In this representation, the radial direction represents octave transposition, while the angular direction performs a glissando through the pitches (fig. 4.8). Although most digital signal processing

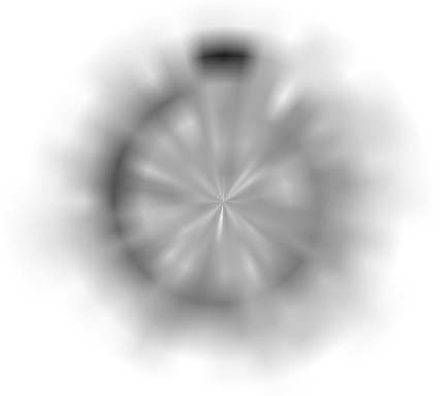


Figure 4.9: A single window of CQT represented in spiral form (dark regions have highest intensity). The window is extracted from the last portion of the test sound `data.danieleghisi.com/phd/snd/spiralrepr_test.aif|mp3` *(notice the crisp high C, and the more noisy A, one octave and a third below it). The CQT is performed over 9 octaves starting from C0, with a resolution of 48 bins per octave.*

methods represent pitch either on a line (e.g., spectrograms) or on a circle (e.g., chromagrams), this spiral model is well known in music theory [Shepard, 1964; Patterson, 1986].

One can rasterize the representation by considering a certain number of samples on the spiral and then bilinearly interpolating among them. Instead of a STFT, a constant Q transform (CQT) can be employed in order to have equal angular sample density. An example of sound, analyzed via *LibROSA*'s CQT [McFee et al., 2015] across 9 octaves, starting from C0 (1200 MIDIcents), with a resolution of 48 bins per octave (i.e. on a eighth-tonal grid) is in Fig 4.9.

When multiple temporal slices are collected, a tridimensional voxel-based representation of sound is obtained. In this representation, radial proximity relates to octaviation, angular proximity relates to frequential neighborhoods, and depth proximity relates to temporal neighborhoods (see fig. 4.10).

Although this tridimensional spiral representation does not eliminate (ii), since every non-octave harmonic is still disconnected from its fundamental, it constitutes a improvement in that respect. The wavelet scattering transform introduced in [Lostanlen and Mallat, 2016] provides improvements to the model, along with an accurate mathematical analysis.

Due to its locality properties, the spiral representation may be better fit to be used in connection with non-supervised tridimensional convolutional networks. It must be remarked, however, that the invertibility request (iv) is essentially lost (one might perhaps devise a better rasterization to overcome the problem). This is an issue for a frame-by-frame audio generation, but it might be less problematic in other MIR applications, such as similarity detection or pattern recognition.

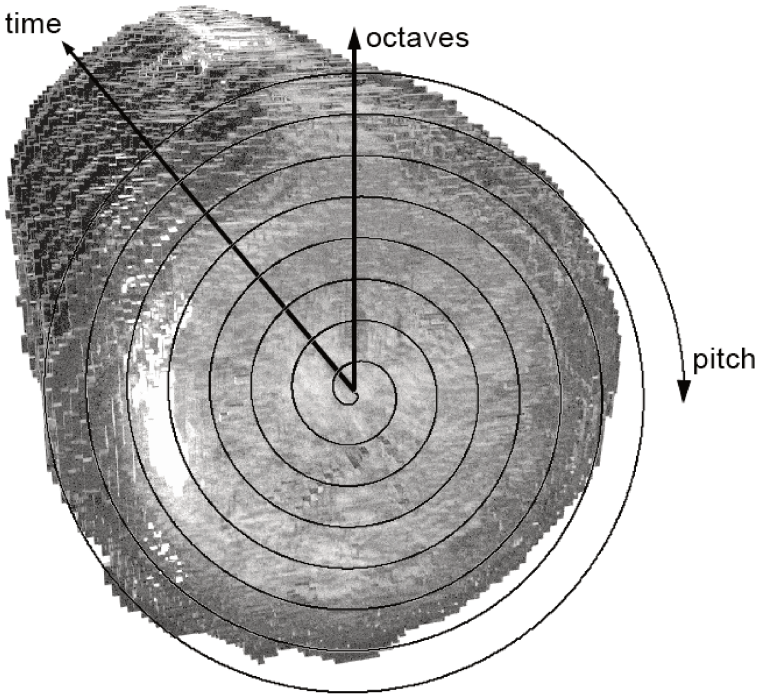


Figure 4.10: Tridimensional voxel structure obtained by overlapping multiple temporal windowed representations as the one in fig. 4.9, and displayed by mapping intensity to transparency (white regions have highest intensity). Notice how the upper C increase its intensity in time.

4.3.3.4 An application: searching samples by similarity

Given my interest in sample-based concatenations, one of the questions that have always intrigued me is querying sound fragments by similarity. It is extremely hard to tackle this issue via scores, where complex parameters such as timbre hardly emerge. It might make sense, however, to use low- and mid-level descriptors to have intermediate representations, and to look for similar patterns in their distributions. Incidentally, this is what most 'audio fingerprinting' algorithm do, by tailoring the appropriate robust descriptors, and the appropriate methods for similarity querying [Typke et al., 2005] (traditional methods rely on indexing and then 'Query by Example' or nearest neighbour search techniques [Roma and Serra, 2015]).

Descriptor-based approaches usually set some assumptions on the characteristics of the input sounds (such as genre, instrument, etc.), hence providing a less agnostic representation than the spiral introduced in the previous section. Therefore, I decided to test the spiral voxel representation with the convolutional variational autoencoder presented in [Brock et al., 2016] and implemented in [Brock, 2016], based on [Maturana and Scherer, 2015]. I have tested the model on a two datasets: a database of about 200 pop and rock songs and a dataset containing around 157

hours of modern and avant-garde music (the 'Avant Garde Project'[5]). Both datasets have been segmented into fragments of about 1 second each, represented as cubes of 32 voxels per side. The input cubes for the autoencoder are similar to the one shown in fig. 4.11. With the in the rock dataset, the intensity of each voxel has been quantized to a single bit; with the Avant Garde Project dataset, two different versions have been tested, respectively with 1-bit and 2-bit quantization.

The autoencoder is trained with a latent dimension of 100; each fragment is then re-analyzed, and its latent representation is stored in a database. A Max interface allows dropping any other sound, selecting one specific second of it, and extracting its latent representation: the twenty closest matches among the analyzed chunks (in L^1 norm) are queried, presented and exported (fig. 4.12).

This research, is at present in a preliminary stage, and better models and larger datasets are needed to improve it. However, even in this rough implementation, the model seems capable of properly matching some characteristics, including pitches and harmonies. Below is a list of audio files, demonstrating the retrieval, in the rock dataset, of the first 20 closest matches for some specific targets. The examples are not cherry-picked, and they are presented with a few comments on their pertinence.

Model: A modified version of voxel-based convolutional VAE [Brock, 2016]

Corpus: *A database with around 200 pop and rock songs*All examples have the target sound as first element, followed by the 20 closest matches ordered by L^1 distance. (Multiple output fragments sometimes belong to the same track.)

- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch1.aif|mp3`:
  pitches seems to be properly matched.
- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch2.aif|mp3`:
  some sort of staticity seems to be preserved.
- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch3.aif|mp3`:
  trying to match a static 'money note' from a database that has no such notes.
- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch4.aif|mp3`:
  trying to match a vibrato 'money note'; the energy and vitality seems to be matched.
- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch5.aif|mp3`:
  trying to match hand claps; first results do not seem to especially relate to the target sound; 8th, 11th and 14th results sound more pertinent. Some result appear to match a sort of loudness-based prosody (e.g., 2nd result—but also,in a different manner, 11th, 14th, 19th and 20th).
- `data.danieleghisi.com/phd/snd/NN_CVAE_samplematch6.aif|mp3`:
  spoken voice seems to be mostly matched.

Also, here are some resulting samples for the Avant Garde Project dataset, always offered in two flavours: the 1-bit and the 2-bit amplitude versions. As one might have anticipated, in general, the 2-bit version seems to guarantee a better quality of results, especially with noisy input.

[5] https://archive.org/details/iaagp

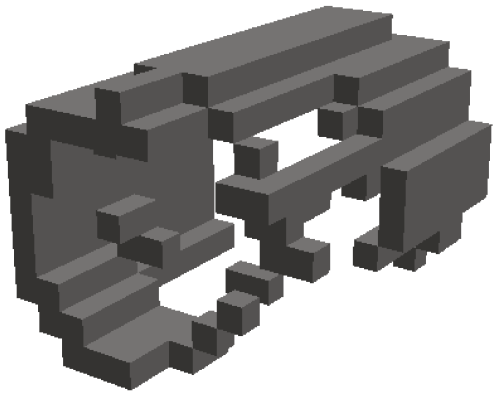


Figure 4.11: Example of a cube with 32 voxels per side and binary voxel intensity, used as input for the variational autoencoder (the displayed voxel correspond to points having CQT intensity above a given threshold).

Model: A modified version of voxel-based convolutional VAE [Brock, 2016]

Corpus: *The Avant Garde Project (around 175 hours)*All examples have the target sound as first element, followed by a concatenation of the 50 or 100 closest matches (in order of L^1 distance). (Multiple output fragments sometimes belong to the same track.)

- `data.danieleghisi.com/phd/snd/NN_CVAE_agp1bit_A.aif|mp3`, `data.danieleghisi.com/phd/snd/NN_CVAE_agp2bit_A.aif|mp3`: the 2-bit version better accounts for the rugosity of the target.
- `data.danieleghisi.com/phd/snd/NN_CVAE_agp1bit_B.aif|mp3`, `data.danieleghisi.com/phd/snd/NN_CVAE_agp2bit_B.aif|mp3`: both version seems to perform equally well on chiefly harmonic targets (notice how higher and lower octaves occasionally pop up in the results, due to the spiral representation).
- `data.danieleghisi.com/phd/snd/NN_CVAE_agp1bit_C.aif|mp3`, `data.danieleghisi.com/phd/snd/NN_CVAE_agp2bit_C.aif|mp3`: hand claps are better matched with the 2-bit version (target was not in the dataset).

4.3.4 Conclusions and perspectives

The work for *La fabrique des monstres* triggered a certain number of reflections on the current and future role of neural networks in computer-aided composition.

A new kind of synthesis. A sample-based neural network approach to audio has the ability to narrow the gap between sampling and synthesis, in a very different way from what Gabor-based approaches do. The idea of windowing the signal into tiny grains of sound, which could then be recombined according to temporal patterns, is replaced with the idea of abstractly learning temporal patterns from the sequence of the tiniest possible grains (the audio samples themselves) in order to reproduce them.

The quality of this approach is limited by computational capabilities; yet one might imagine similar techniques becoming a standard way to produce sounds

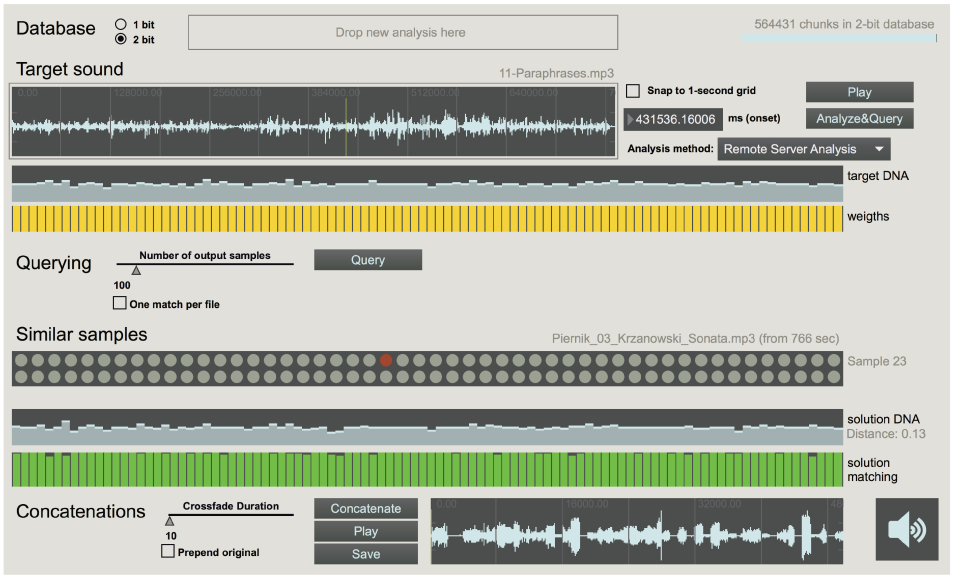


Figure 4.12: Interface allowing the retrieval of closest matches to input fragments.

in some years time: a truly new kind of synthesis. The first prototypal frameworks for audio synthesis based on machine learning techniques are already emerging, such as Engel et al. [2017]; however, virtually all of them remain anchored to the exploration of tone qualities—moving in latent spaces to shape timbre and dynamics. Although this is certainly attractive, recurrent models also enable us to look further, and to synthesize longer portion of sounds: 'phrases' instead of 'notes', and, perspectively, even more complex long-term relationships.

Conditioning. Due to a precise aesthetic choice, all the networks used in *La fabrique des monstres* were unconditional models. One may add conditioning techniques to the recipe, in order to provide this new kind of synthesis with a palette of control parameters. Such parameters may in turn be also handled by higher level networks: for instance, one might add harmonic control (currently an extremely difficult property to be learned from sample-by-sample models, due to the amount of memory needed to model it), and generate a sequence of harmonic structures via an auxiliary recurrent network; or one might condition the generation of a piece for ensemble by imposing the presence or absence of single instruments.

Virtual music. I believe that the application of recurrent neural networks to music generation has very much in common with David Cope's idea of 'virtual music', as a category of compositions which attempt to replicate a 'style', but not the actual notes of the existing music [Cope, 2004]. Virtual music existed in different forms throughout the centuries, notable examples being baroque figured basses and musical dice games. In the specific case of *La fabrique des monstres*, however, I was more interested in the *differences* between the style

and the outcome, than in replicating a style *per se*—what is compelling to me is the moment when the harmony is suddenly off, or when a short white noise burst comes to break a vocal melisma.

Criticism and surprise. While conversing with friends, composers and artists, I have come across two different arguments criticizing the application of neural networks to art. Some reject deep learning on the basis that networks do not actually learn how to produce new art, but rather how to reproduce existing patterns: they do not create, they simply copy. Others reject their usage on the basis that neural networks are essentially 'black boxes', hence nullifying (or at least severely reducing) our control over their outcomes.

Of course deep networks do not perform any sort of 'magic': they are designed and optimized precisely to reproduce existing patterns, rather than to create new ones. This seems to clash, for instance, against Lachenmann's definition of 'art' as "a form of magic, broken in and with spirit" [Lachenmann, 2004].

Cope [2017] argues that "ultimately, the computer is just a tool with which we extend our minds. The music our algorithms compose are *(sic)* just as much ours as the music created by the greatest of our personal human inspirations". The development of art is ultimately connected with the development of technology: instruments (from ancient flutes to electric guitars) and abstract tools (from the invention of notation to machine learning techniques) are intrinsically a byproduct of these developments.

My—extremely personal and very reductive—belief is that creativity is a direct consequence of surprise: we are 'inventing' something the very moment we are being surprised by something else (a thought, a connection, an image, an idea, a sound...). Surprise, as the difference between expectation and event, is the actual information, as a sort of tension between a gesture and its outcome. Creativity is, to me, in a very real sense, a corollary of an (active or passive, conscious or unconscious) exploration. If neural networks were actual 'black boxes' no expectation would be possible, and 'surprise' would be a moot term in this context; however neural networks are more 'grey boxes' than 'black boxes', if one has some experience of the way they work, or a rudimentary understanding of what is happening inside them. Surprises are hence my personal substitute for Lachenmann's 'magic'.

Most of the work for *La fabrique des monstres* was driven by this curiosity: the 'machine' was to generate data unconditionally, and the exploration of the 'maps' of generated content was the core of the compositional process. The key aspect was being able to seize the interesting portions generated by the machine; in other words, the most important ability I had to have while experimenting with neural network models was leaving room (and keeping attention) for surprise. I tried to always be *à l'écoute*, ready to be surprised by some unexpected harmonic change, instrumental quirk or unconventional outcome.

Meta usage. One of the aims of using neural network in *La fabrique des monstres* was to put the learning process in itself on the spot, as object of interest. In a sense, it is a case where a relatively new technology becomes both the portrait and the pencil, not unlike how some movies at the early stages of sound films portray the sound itself as 'main character' (e.g., the opening scene of Rouben Mamoulian's *Love me tonight*).

A modern workspace for audio editing. As previously reported, most of the state-of-the-art MIR algorithms make extensive use of machine learning and neural networks. A few commercial firms have also started to integrate such algorithms in their products (see, for instance, iZotope's *Neutron's Track Assistant*). There is still, nonetheless, an open field of possibilities for significant application of neural networks in common musical tasks.

Neural network models akin to the one we have discussed could be applied in denoising, upsampling, bit depth conversion, dithering, spectral reconstruction, click detection, audio repair. Some experimentations are being performed in these directions (such as [Maas et al., 2012; Kuleshov et al., 2017]), and iZotope developers themselves are delving into some of these possibilities [Wichern, 2017].

As a composer, I fancy a comprehensive workspace where a user-friendly interface would handle these and many more features, so that a 'smart eraser' would reconstruct portions of spectra, a 'smart scalpel' would perform source separation, and so on. Even time stretch and transpositions on acoustic samples would actually be implemented via machine learning techniques: the first would help morphing transients in a plausible way, whereas the latter would help respecting the characteristics of instrumental registers ([Engel et al., 2017] is a starting point)—and extreme pitch shifts should be achievable, without trading too much of their quality.

Enhanced instrumental simulations. One of the hassles of contemporary composers—myself included—is the increasing need of having instrumental simulations of portions of scores. Of course, composers are trained to develop a keen intuition of the acoustic rendering of a score; however, while working with extended techniques and especially while working in combination with electronic sounds, such intuition is not easy to train, and can lead far from the acoustic reality. It is simpler to get a sense of the relative instrumental weights in an orchestral *tutti* than to imagine the rendering of a certain combination of multiphonics and FM synthesis.

Within the context of mixed music productions, composers have sometimes the possibility to record an instrumental simulation of their scores with an ensemble or with selected musicians. Although this practice provides them with the most faithful rendering, it is far from being flexible: further work on the recorded score and on the electronics might trigger changes in the score itself, but between rendering and composing there is no actual feedback, which

would require a series of recording sessions at different stages—too impractical, expensive and time consuming to be feasible.

Ideally a composer might want to strike a balance between the quality and the flexibility of the rendering process, trading some part of the quality in order to have the possibility to render scores more easily and more often—or even in real time.

Virtually all modern music typesetting software include a real-time sample-based audio rendering system—however, this is far too crude and insufficient for the contemporary composer. A certain number of frameworks have been developed in recent years in order to enhance such capabilities; among these one might cite conTimbre[6], including more than 4,000 playing modes. However, being essentially sample-based, all of these tools lack the flexibility of a real musician's playing.

One might imagine a recurrent neural network model which learns sample-by-sample audio rendering of scores starting from a database of symbolic scores for solo instruments (possibly in a MusicXML-compatible format) each associated with multiple audio renderings. This would be an extremely practical neural network application for everyday's compositional practice.

Abstract representation and morphing. Using latent space representations leads to easy ways to achieve morphing and interpolation effects. One can therefore take advantage of GANs and VAEs to develop morphing or interpolation tools. As a crude example, see the interpolation of attacks in a GAN model in fig. 4.13.

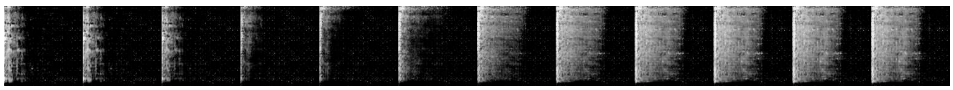


Figure 4.13: An interpolation of short attack-like samples, performed via a GAN model applied on spectrograms. The 8-bit, 16kHz corresponding audio (with reconstructed phases) is available here: `data.danieleghisi.com/phd/snd/NN_dcgan_interp.aif|mp3`

Lack of layman's tools. Since most of the state-of-the-art models are programmed in specially designed languages such as Torch, Theano or Tensorflow, very few musical machine learning tools exist that do not require extremely specialized programming skills—among these, I should cite *RLKit*[7], a set of external bringing reinforcement learning into Max and PureData; *Wekinator* [Fiebrink and Cook, 2010], an OSC-based open source machine learning framework; and *ml-lib* [Bullock and Momeni, 2015], a library for Max and Pure Data focused on machine learning, especially in the context of gesture recognition.

As already stated in section 3.5, it might be interesting to investigate the feasibility of a comprehensive Max library, meant to be adopted not only by

[6] `https://www.contimbre.com`
[7] `http://alab.hexagram.ca/rlkit/index.html`

researchers, but by musicians as well, featuring both generic and specialized tools to handle machine learning along with other programming paradigms variously related to artificial intelligence, such as optimization, constraint solving and data mining.

Questioning authoriality. Generative network models raise a number of questions related to the concepts of musical authorship (who is the moral or legal author of the generated content?) and copyright (are audio files obtained by training networks on a certain class of targets derivative content?). In particular, one observes the blurring of the contours of the 'composer as unique creator'—an interesting phenomenon, reminiscent of Gell's ideas exposed in section 2.1.2. An overview on the intricacies of the relationship between machine learning and copyright in the United States of America can be found in [Sobel, 2017].

4.4 An experiment in collective writing

4.4.1 */nu/thing* and *I mille fuochi dell'universo*

My reflections on the role of the artist both across the history of music and within the context of our society have sparked a desire to tackle the challenges of collective writing. In my practise, such desire is closely related to the topics I have been discussing in this thesis: it has much to do with corpus-based composition (is *music across music* a declination of *musicians across musicians*?) and with technology (which is the best way to share and collaboratively work on musical ideas, scores, sounds, texts or patches?); but most importantly it questions the composer as a 'solitary self'.

Since 2012, I am part of */nu/thing*, a group currently gathering five Italian composers around the blog www.nuthing.eu. The group was founded in an effort to claim a social status for thinking music and making music today, partly in continuity with the European musical tradition of the last few decades, partly in contrast with the muddle of social dynamics that such tradition entails. Our utopia was to break, especially in the inadequate Italian cultural context, the vicious circle between production, fruition and discussion, in order to breath new life into a community of listening and a community of thought.

In 2015, we have accepted the proposition of *Milano Musica* festival to write a collective work. The outcome, *I mille fuochi dell'universo*, premiered in October 2017 in Milan by MdI ensemble, is a 40-minutes long mixed piece, composed by an extremely long rallentando of attacks, each having its own characteristics and its own release shape, with increasing complexity. The variation in the distance between attacks articulates the rallentando (see fig. 4.14): at the very beginning, the distance between the attacks is $1/20000s$ and the output sound is above the threshold of hearing; then the distance widens, originating a downwards glissando; at around 3', the attacks become singularly distinguishable and, as they unravel, they turn into a ribattuto that keeps slowing down; little by little from each of the attacks a peculiar 'release' emerge, which, as its duration increases, becomes a miniature musical world; worlds continue to grow in duration and complexity, from the first musical figures to the very last event, more than 5 minutes long, which, in some sense, constitutes a piece inside the piece. From a conceptual point of view, *I mille fuochi dell'universo* does not end there, and impulses might be thought of as continuing to slow down, metaphorically enclosing everyone's life.

The rallentando *is* the most important signature of *I mille fuochi dell'universo*—a perspective inspired both by Stockhausen [1957] and by Grisey [1987, 2008]. Gradually, from the tiny fragments of the beginning, as the fabric of time relaxes, 'inhabited worlds' emerge, each with its own characteristics. The audience sits between two rows of loudspeakers (an 'electroacoustic tunnel'): sounds are always projected in the same direction, during their release time, traversing the listeners from front to back; the ensemble is located at the frontal end of the tunnel, and is heavily amplified. The piece is purposely imagined for the very reverberant acoustic of the

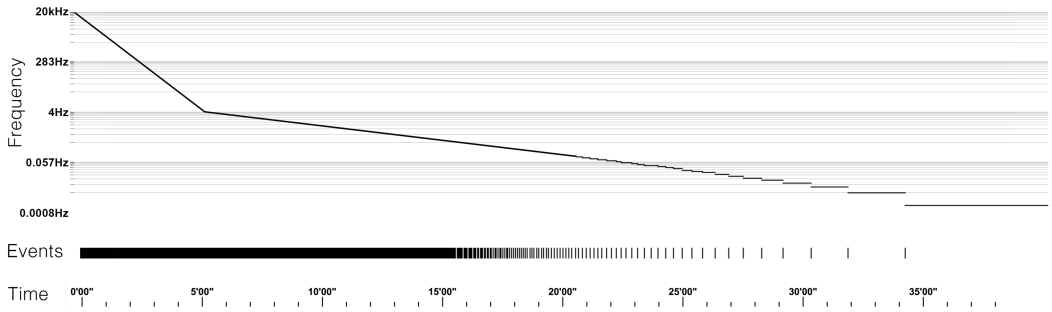


Figure 4.14: Globale structure of I mille fuochi dell'universo. *Events represent the sequence of attacks, frequencies represent distances betweens attacks. The rallentando changes slope at* $4Hz$ *(around* $5min$*).*

Bicocca Hangar in Milan, and hence takes into account the characteristics of the venue.

4.4.2 Towards a collective writing of music

I have already presented a few techniques used for specific portions of *I mille fuochi dell'universo* in sections 2.3.4 and 2.8. However, the main point of interest of the piece is, to me, the attempt to build a truly collective writing. Co-signed works are certainly not new in the history of written Western music; even prominent composers have engaged in collective practices. However, in most cases, these belong to one of the following categories:

- works in which each of the signatories is in charge of a particular section, such as the well known *F-A-E Sonata*, by Dietrich, Schumann and Brahms, or the 1945 *Genesis Suite* by Castelnuovo-Tedesco, Milhaud, Schoenberg, Shilkret, Stravinsky, Tansman and Toch;

- works in which an author completes a previous unfinished project, such as Alfano's ending of Puccini's *Turandot*, or Berio's *Rendering* of Schubert's tenth symphony;

- works in which each of the signatories assumes a specific role, such as Doppler's orchestrations of Liszt's *Hungarian Rhapsodies*, or, more recently, *Regnum Animale* by Lanza and Valle.

What we wanted to achieve with *I mille fuochi dell'universo* was, on the other hand, radically different: a process in which both the signature and the way of working were to be inherently collective. Our core principle, while working on *I mille fuochi dell'universo*, was to share technological tools (scores, audio files, patches, sequencer sessions, texts, etc.), along with conceptual tools (ideas, references, etc.).

Collective practices such as the one in *I mille fuochi dell'universo* are extremely uncommon in contemporary music, but not entirely new. A significant case is the

Edison Studio collective [Cipriani et al., 2004], which in turns stems from a long-standing tradition of groups trying to find a middle ground between collaborative improvisational techniques and composition (including the experiments of John Cage and David Tudor and the *Nuova Consonanza* and *Musica Elettronica Viva* groups). In addition, a large portion of today's cultural production is inherently co-written, including visual arts and cinema. There are also examples in literature, such as the Italian collective of writers *Wu Ming*, as well as in other musical genres, such as the dynamics of pop and rock groups or improvised musics. In addition, the history of Western music includes chiefly collegial practices in its early ages, such as plainchant and medieval schools, and in specific transdisciplinary movements, such as the 20th century Fluxus.

It is not a coincidence that in these situations notation hardly existed (pre-medieval musical practices), was at its dawn (*Ars Antiqua*, *Ars Nova*), was essentially replaced by textual descriptions (Fluxus event scores [Friedman et al., 2002]), or was somehow irrelevant (pop and rock productions, improvisation). In a sense, notation leads to an urge for rationalization, and therefore, in a very Cartesian way, individualization of the musical practice: the composer turns into a solitary self (also see section 4.1.1).

One might be tempted to speculate that notation is, at least in part, what prevents art production to be truly collective. This would also explain, to some extent, why in the twentieth century collectives of visual artists have flourished, while truly common compositional practices have not. Even collectives such as the aforementioned *Edison Studio* seem to agree that notated music is a steeper obstacle than electronics or multimedia content:

> The possibility of being able to discuss work not only on the basis of the concept, as would happen with the composition of a score, but also on the basis of direct audio-visual perception is fundamental. With the use of audio-visual technology, the composers are able to interact with the electronic equipment in order to modify what they are listening to, and can save every tiny detail of what they agree upon. Without this essential prerequisite the management of operations as complex as these would be impossible, unless a subdivision of labour was once again created. [Cipriani et al., 2004]

Our challenge was also to disprove this hypothesis, and to show that it is not notation itself what prevents collective practices to prosper, but rather *access* to notation. A piece of evidence for this is how collective writing (both in scientific and in literary form) has florished since the advent of online collaborative text editors. Can a generation of advanced online score editors (such as evolutions of the currently crude Flat editor[8] [Gurtner, 2016]) and collaborative sequencers (such as OhmStudio[9]), possibly in combination with online computer-assisted composition

[8] https://flat.io
[9] https://www.ohmstudio.com

tools, reshape contemporary music habits? At what point will they incorporate, taking inspiration from the tools already available in the scientific domain, important features for collaborative working such as version branching or pull requests?

Another common characteristic of collective practices seems to be some sort of 'exteriorization': dancers, choreographers, actors, as well as rock and pop groups and improvisers, usually work together in the sense that they *explore together, in time*. It is more difficult to 'explore together', in time, abstract notation or conceptual thoughts, with the same power with which one could share a guitar pattern or a body gesture. And yet, part of our challenge was make music writing collective, without abjuring the possibility of a music of thought and ideas, and without renouncing to the potential of notation.

Chapter 5

Conclusion

This thesis presents my personal, formalized approach to corpus-based composition and interactive music exploration, along with the musical and technological tools supporting it. It is the product of an ongoing line of research which, it contains, beside the music production listed in appendix A, has produced a certain number of contributions, detailed in section 5.1, and has left a certain number of open problems, outlined in section 5.2.

5.1 Main contributions

Corpus-based composition

Corpus-based composition is presented as a natural, exploratory way to write original content, in continuity with techniques employed throughout the Western history of music (section 2.1.1). The proposed approach borrows music samples from large-scale datasets in order to build a low-level, descriptor-based palette of musical grains, possibly aimed at mitigating, to a certain extent, the orthogonality of the traditional decomposition of notational parameters (section 4.2). The approach relies on segmentation, analysis, storage and querying of musical datasets, both in audio and in symbolic form; the composition stage stems from filtering and exploring grains (section 2.6). A simple application is pure concatenation of audio grains (section 2.4), which appears to be a rather promising compositional path. Several usage cases of these techniques in my recent works have been discussed—including the audiovisual work *An Experiment with Time*, possibly the most representative piece of this thesis, condensing a fraction of these ideas and stimulating further research.

The originality of the approach resides both in the amount of data at stake (in the most recent projects, several hundreds gigabytes of compressed audio) and in the construction of an experimental framework, where scores and datasets coexist and control each other—all within the Max graphical programming environment. In particular, a significant contribution is the bridging of dataset-based techniques into the world of symbolic computer-aided composition, opening the route to score-based concatenative synthesis.

Interactive meta-scores

The present thesis provides examples of interactive hybrid scores used to drive compositional processes, both symbolic and electroacoustic (section 3.4), and hence contributes to bridging the gap between scores and instruments. The underlying

idea is that a musical representation, *sensu lato*, is not only a way to convey information, but also an interactive device, controllable in real time, that composers can play *with*.

Exploratory interfaces and the *dada* library

As far as computer music tools are concerned, the main contribution of this thesis is the design and development of the *dada* library, providing Max with the ability to organize, select and generate musical data via graphical interfaces that share a geometrical, exploratory approach (section 3.3). The library, open-source and extendable, constitutes the third member of the '*bach* family' (a sequence of *bach*-related Max libraries for real-time computer-aided composition); it can be downloaded from its official website `www.bachproject.net/dada`. It features graphical user interfaces for a number of scenarios: a portion of the library handles corpus-based composition and analysis, including segmentation and database visualization; the library also addresses, among other things, music generation via physical modelling, probabilistic graph exploration, cellular automata, swarm intelligence models, and videogames.

The *dada* library fosters the idea that music creation can be thought of, to some extent, as a discovery. Among its contributions, I shall mention the *dada.music~* object, and the corresponding *Music (or The Discotheque of Babel)* patch (section 2.1.4), which represent, in my opinion, conceptually troubling hybrid objects.

Two important features of *dada* are its interactivity, promoting a real-time computer-aided composition paradigm, and is its seamless integration with the Max and *bach* environments: the outcome of all its interfaces can be easily recorded in scores, or directly used to drive digital signal processes, manipulate videos, or control physical actuators. As a matter of fact, it is a valuable tool not only for music composition (its primary goal), but also for pedagogical purposes: the interaction with *dada*'s elementary two-dimensional representations enables users to discover musical properties in an intuitive way (such as, for instance, the relationships between rhythms, speeds and proportions).

Neural networks

The application of machine learning algorithms to music is a natural extension of large-scale database techniques. Section 4.3 details one specific usage of recurrent neural networks as sample-by-sample audio generators. The interest of such class of networks is to inquiry the possibilities of 'agnostic' models (taking raw pulse-code modulation values as input, and hence having no notion of high-level musical concept) to 'listen', 'learn' and reproduce patterns. This thesis does not contribute to the subject with any new network model, but it provides a survey of the state of the art, and a discussion, featuring short audio examples, about advantages and shortcomings of each method. The most important implication, as a composer, is that unsupervised sample-by-sample network models contribute to narrowing the gap between sampling and synthesis techniques. In order to better account for closely

related pitches in spatial representations of music, a voxel-based spiral representation is proposed (section 4.3.3.4) and put to test, with some timid results, while trying to detect similarities between audio fragments via convolutional variational autoencoders.

Collective writing

I am convinced that collective practices will be significant lines of research for the remainder of the 21st century. It is a belief I maintain as part of */nu/thing*, a collective of five Italian composers. This thesis motivates such belief and briefly discusses our first case of collective composition in section 4.4.

5.2 Open problems and future work

Corpus-based composition

The ensemble of principles and techniques that one might cluster around the label 'corpus-based composition' is largely still to be explored. New aesthetic needs force the development of new tools, and, vice versa, the improvement of technology and the considerable growth of the available datasets open the field for new musical possibilities. Among the scenarios I plan to explore further, concatenative techniques are both the most elementary and the most promising, hopefully in combination with better detection algorithms—for instance, one could take advantage of neural-based similarity detection to perform large-scale sample-based pseudo-morphing effects (concatenations of similar samples, leading from one sound to another).

One class of problems concerns the development of algorithms inspired by digital signal processing (especially from a sample-based approach), but operating on scores, rather than sounds. The largest shortcoming I have experienced within the framework of corpus-based composition is the lack of a reasonable amount of data about symbolic scores. If large datasets of scores were to become available, most of the symbolic techniques detailed in the thesis might take advantage of their increased size—in a similar fashion to how large-scale audio datasets have changed the quality of descriptor-based interactions. A project targeting the creation of a public collection of high-quality MusicXML scores would be beneficial for the whole computer music community: given its complexity, the task might be better approached from a collaborative angle, similarly to what Wikifonia did.

Grain-based score representations (section 4.2) have a number of advantages, but also leave a large number of open problems. For one thing, the only reliable synthesis techniques are granulation and concatenation—and even with such techniques, complications arise while splicing grains: due to obvious instrumental limitations, actual crossfades are often not an option (for instance, it is impossible for a piano note to begin *dal niente* and then *crescendo*); in practise, one tends to build a rough solution, which is to be meticulously refined by hand. At some point, the possibility to develop radically different synthesis techniques, starting from grain-based representations, should be investigated; neural-based interpolation or morphing might perhaps provide an alternative solution.

Real-time computer-aided composition

The real-time approach to computer-aided composition is a relatively new paradigm, that causes into question many features of traditional environments, including the nature of the CAC music representation itself. In particular, I have come to experience the traditional disentanglement of musical parameters more as a curse than as a blessing (see section 4.1.1). To overcome this, grain-based representations are simply an attempt among others: maybe a modern CAC workspace should implement more 'significant' and intuitive access to complex data. In this sense, extensive work on score descriptors might be beneficial: some preliminary steps have been under-

taken in *dada* to tackle the issue (section 3.3.2), but there is a long path still very much to be explored.

As far as the *bach* project is concerned, the idea of structuring a '*bach* family' to arrange and distribute future Max libraries (continuing the alphabet after *bach*, *cage* and *dada*) holds, for Andrea Agostini and me, as a general organization principle. Perspectives on these new families have already been proposed in section 3.5.

Besides, there are at least three improvements Andrea Agostini and I plan to bring in the core system of *bach* itself. The first improvement addresses the inability to represent diatonic pitches properly, a long standing limitation of traditional CAC environments. Currently, *bach* is able to distinguish between enharmonic equivalents and to interpret symbols as note names, but the mechanism is cumbersome. Even simple operations, such as a transposition by a major third, are currently far from trivial. In a forthcoming version, we plan to implement a new data type specifically designed to represent pitches defined by note names and accidentals: this new data type will be treated, as much as possible, as a numeric type (e.g., minima and maxima will be defined, addition, subtraction and modulo operations will be performable, and so on).

The second improvement addresses the current impossibility to display a page representation of scores (except via extremely cumbersome patching mechanisms): both *bach.roll* and *bach.score* only present a single, horizontally scrollable staff system. In the future, one might have, in addition to the current 'scroll view', two other display settings: a 'papyrus view' (only allowing vertical scroll of staff systems) and a 'page view' (automatically organizing staff systems in multiple pages).

The last improvement refers to the fact that the Max visual programming paradigm is not very suited to clear and maintainable implementations of complex algorithms. Our perception is that, although Max can be scripted through a variety of languages, including C, C++, JavaScript, Java, Lua, Python, and Common Lisp, all these languages bindings are either too revealing of the low-level mechanisms underlying a Max patch (and as such they require a very deep understanding of the inner workings of Max, which makes them impractical for most users) or, on the contrary, suffer from inefficiency issues deriving from them being too abstract with respect to the actual Max data types and programming paradigm (thus requiring expensive data conversion). The implementation of a new, dedicated programming language might contribute to reducing the number of unnecessary visual modules (even to perform relatively simple operations), and also enhance the existing possibilities of selection-based scripting and editing in *bach.roll* and *bach.score*.

At the time of writing, the design for a diatonic datatype is under progress, as is the development of the dedicated scripting language. For more information about this and about future project on *bach*, see [Ghisi and Agostini, 2017].

As a final note, in my opinion, large areas of interests would arise as soon as the power of modern computer-aided composition tools is embedded in collaborative, web-based scenarios.

Exploratory interfaces and the *dada* library

Future work on the *dada* library has already been extensively detailed in section 3.3.6.

Neural networks

The application of machine learning to music generation (and not simply to music classification) is a growing and promising field of study. Some compositional perspectives and implications have been presented in section 4.3.4, including the possibility to have, hopefully in a near future, enhanced instrumental simulations, intelligent morphing, neural-based audio editing workstations, and more. More generally, some work needs to be done in order to make machine learning algorithms accessible not only by researchers, but by musicians as well.

As far as sample-by-sample recurrent generative models are concerned, their memory span is currently relatively small, due to technological limitations; it will be thrilling to see how the quality of results will change once technologies allow training with longer memory. Incidentally, machine learning models will soon question the boundaries of authorship and copyright ownership—a fact that is both challenging and exciting.

Latent space approaches offer representations of musical data where traditional parameters are entangled (or disentangled in an unconventional way), allowing to build a 'high-level mixer', which does not calibrate standard parameters, but rather complex musical properties (an idea I owe to Mattia G. Bergomi). The potential of such representations in connection with musical writing is yet to be investigated: are there intuitive notation paradigms we might take advantage of?

The interest in latent space modelling also entails a quest for a music representation better fit to be used in conjunction with convolutional techniques, which still remains an open line of research. My personal, limited attempts to build a tridimensional autoencoder based on an amplitude quantization of the spiral voxel model proposed in section 4.3.3.3 have shown some timid results, accounting for pitches better than spectrogram-based models. However, results are still too modest, and further work is needed to validate or discard the representation, starting with testing finer quantization for voxel amplitudes. Also, it might be interesting to force a disentanglement of the latent space variables, and see which features each variable affects; this might lead to characterization of an unconventional set of 'fundamental' musical operations (and it would be meaningful, for example, if standard operations such as transposition or time translation could be identified).

Importantly, the validity of latent space representations should also be assessed in conjunction with techniques borrowed from psychoacoustics and gestalt psychology.

Collective writing

Collective music writing is a both exciting and unsettling experience, moved by two equal and opposite forces: the tendency to find a common ground, on one side; and the tendency to live each individual path as a resource, on the other hand. The combination between the two paradigms forces each composer to accept and assume possibly unfamiliar musical inclinations, not only as external objects of a certain value, but as primary sources for the everyday's work. This can lead to potentially unpredictable results, which is a thrilling side effect. I hope that a significant part of my future production will undergo, to some extent, similar dynamics.

The practice is somehow discouraged by the lack of modern real-time collaborative tools. In the case of */nu/thing* and *I mille fuochi dell'universo*, we settled on standard pieces of software (Finale, Logic, and Max) and we decided to save all the data on remote drives (via Dropbox). Conflicts were prevented via a system of 'manual mutexing': every composer was allowed at any time to operate on a private copy of the score or of the sequencer, but whenever he wanted to work on the public version, he had to send a 'lock' notification to the whole group (via a popular instant messaging service), make an incremental copy, and then send a 'unlock' notification, to give anyone access to the new files, via the same mechanism. Not only was this process prone to errors (and indeed at some points we did run into conflicts due to poorly crafted synchronization), but it was also extremely cumbersome to handle: no two people could talk and operate at the same time on the same score, a process that would be trivial had we always worked side by side in the same room. Workarounds included VoIP calls and screen sharing, but they were not nearly as effective as a Google document was whenever we needed to work on texts.

A comfortable framework for collective writing should include a reliable, real-time collaborative typesetting software. Currently, there is nothing of this kind available; the only pieces of software that somehow relate to the description are, at present, too elementary to be used in any high-end music production. Moreover, most of the existing networked collaboration environments seem to be targeted at collective performances more than at collective writing—exceptions hinting to interesting directions being *Diamouses* [Alexandraki et al., 2008] and *Quintet.net* [Hajdu, 2005]. A more convenient piece of software might feature the possibility to interact at the same time on the same score, and might also incorporate concepts such as branching and merging: every composer might be able to make a personal branch of the score at any point, inspect the differences with the main branch, and merge the two, resolving the possible conflicts.

Similar mechanisms should apply to sequencers and to multimedia workspaces. There is some recent interests on these subjects: some sequencers, such as Ohm-Studio, are designed to be used remotely; there is a (crude) porting of PureData in web browsers via Web Audio API; Max itself, via the Miraweb package, provides (limited) mirroring capabilities in a web browser. It would be crucial that all these interests converged into making collaborative working easier.

Appendix A

Catalogue of works

Below is the list of projects on which I have worked during the last three years. These projects are also often cited within the text.

- *Come un lasciapassare* (2015)
  for orchestra and electronics
  duration: 10 minutes
  commissioned by Radio France (*Alla breve*)
  publisher: Casa Ricordi

- *An Experiment with Time* (2015)
  installation for 3 screens and electronics
  duration: 46 minutes (looping)
  full credits: http://www.anexperimentwithtime.com
  commissioned by Ircam/Divertimento Ensemble

- *Mon corps parle tout seul* (2015, rev. 2017)
  electroacoustic installation with projection on nebulized water
  duration: 12 minutes (looping)
  joint project with Daniel Jeanneteau
  text by Yoann Thommerel
  full credits: http://brahms.ircam.fr/works/work/36320/
  commissioned by Ircam

- *Any Road* (2016)
  for orchestra, electronics and video
  duration: 11 minutes
  commissioned by the French Ministry of Culture
  video by Boris Labbé
  publisher: Casa Ricordi
  premiered in Lyon, 04/03/2016, *Biennale Musiques en Scène*

  - new version for ensemble, electronics and video (2017)
    commissioned by Ensemble Intercontemporain
    will be premiered in Paris, 26/01/2018

- *An Experiment with Time (reloaded)* (2016)
  for ensemble, electronics and video
  duration: 48 minutes
  publisher: Casa Ricordi
  commissioned by Ircam/Divertimento Ensemble
  premiered in Milan, 26/01/2016, Divertimento Ensemble, Festival *Rondò*

- *269 Steps Away From you (269 Steps Away From Me)* (2016)
  for bass clarinet, violin and electronics
  duration: 11 minutes
  commissioned by Internationale Fredener Musiktage/Ernst von Siemens Musikstiftung
  publisher: Casa Ricordi
  premiered in Freden, 20/07/2016, *Internationale Fredener Musiktage*

- *Orogenesis* (2016)
  for video and electronics
  duration: 7 minutes
  video by Boris Labbé

- *I mille fuochi dell'universo* (2017)
  for ensemble and electronics
  duration: 38 minutes
  commissioned by Milano Musica
  collective writing with the /nu/thing group[1]
  will be premiered in Milan, 28/10/2017, MdI ensemble, *Milano Musica* festival

- *Bug (quatuor à corps)* (2017)
  for four actors and electronics, theatre piece by Ingrid von Wantoch Rekowski
  duration: about 10 minutes
  premiered in Bruxelles, June 2017

- *La fabrique des monstres* (2017)
  for electronics, theatre piece by Jean-François Peyret
  commissioned by Ircam
  will be premiered in Vidy (Switzerland), 23/01/2018

- *Chute* (2017)
  for video and electronics
  duration: 15 minutes
  video by Boris Labbé
  produced by Sacrebleu Productions

[1] www.nuthing.eu

The researches contained in this thesis have triggered a few additional projects which are currently either sketches or still in progress:

- *Music (or The Discotheque of Babel)*
  Electronics, Max patch
- *The Well Tempered Sampler*
  A collection of 24 short pieces (one for each major and minor chord)
  various instrumentations and durations
- *An Urban Dictionary of Popular Music*
  A 24-hours looping audio installation
- *Electronic Studies*

Appendix B

List of articles

Below is the list of the articles I have published during this thesis. These articles are also often cited within the text.

- Agostini, A. and Ghisi, D. (2015). A Max Library for Musical Notation and Computer-Aided Composition. *Computer Music Journal*, 39(2):11–27.
- Bigo, L., Ghisi, D., Antoine, S., and Moreno, A. (2015). Representation of musical structures and processes in simplicial chord spaces. *Computer Music Journal*, 39(3):11–27.
- Ghisi, D. and Agon, C. (2016). Real-time corpus-based concatenative synthesis for symbolic notation. In *Proceedings of the TENOR Conference*, Cambridge, United Kingdom.
- Ghisi, D., Agostini, A., and Maestri, E. (2016). Recreating Gérard Grisey's Vortex Temporum with cage. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.
- Ghisi, D. and Bergomi, M. (2016). Concatenative synthesis via chord-based segmentation for An Experiment with Time. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands.
- Ghisi, D. and Agostini, A. (2017). Extending bach: A Family of Libraries for Real-time Computer-assisted Composition in Max. *Journal of New Music Research*, 46(1):34–53.

Bibliography

Adams, J. L. (2016). Music in the Anthropocene. In Carl, R., editor, *Postmodern Music, Postmodern Listening*, pages 315–318. Bloomsbury Academic. (Cited on page 1.)

Agon, C. (1998). *OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur.* PhD thesis, University of Paris 6. (Cited on page 59.)

Agon, C. and Assayag, G. (2002). Programmation visuelle et editeurs musicaux pour la composition assistée par ordinateur. In *Interface Homme Machine*, Poitiers, France. ACM Computer Press. (Cited on pages 70 and 111.)

Agon, C., Assayag, G., Fineberg, J., and Rueda, C. (1994). Kant: A critique of pure quantification. In *In Proceedings of the International Computer Music Conference, Aarhus, Denmark. International Computer Music Association.* (Cited on page 131.)

Agostini, A., Daubresse, E., and Ghisi, D. (2014). *cage*: a High-Level Library for Real-Time Computer-Aided Composition. In *Proceedings of the International Computer Music Conference*, Athens, Greece. (Cited on pages 56, 61, 68, 71 and 111.)

Agostini, A. and Ghisi, D. (2012). *bach*: an environment for computer-aided composition in Max. In *Proceedings of the International Computer Music Conference*, pages 373–378, Ljubljana, Slovenia. (Cited on pages 56 and 63.)

Agostini, A. and Ghisi, D. (2013). Real-time computer-aided composition with *bach*. *Contemporary Music Review*, (32 (1)):41–48. (Cited on pages i, 8, 60 and 62.)

Agostini, A. and Ghisi, D. (2015). A Max Library for Musical Notation and Computer-Aided Composition. *Computer Music Journal*, 39(2):11–27. (Cited on pages 56, 62, 64, 76 and 131.)

Alexandraki, C., Koutlemanis, P., Gasteratos, P., Valsamakis, N., Akoumianakis, D., Milolidakis, G., Vellis, G., and Kotsalis, D. (2008). Towards the implementation of a generic platform for networked music performance: the diamouses approach. In *ICMC*, pages 251–258. (Cited on page 167.)

Anders, T. (2007). *Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System.* PhD thesis, Queen's University Belfast. (Cited on page 128.)

Anders, T., Anagnostopoulou, C., and Alcorn, M. (2005). Strasheela: Design and Usage of a Music Composition Environment Based on the Oz Programming Model. In *Multiparadigm Programming in Mozart/Oz*, pages 277–291. Springer. (Cited on pages 65 and 128.)

Anders, T. and Miranda, E. R. (2011). Constraint programming systems for modeling music theories and composition. *ACM Comput. Surv.*, 43(4):30:1–30:38. (Cited on page 125.)

Andersen, E. (2012). Origami and math. http://www.paperfolding.com/math/. Accessed: 2015-13-12. (Cited on page 94.)

Antiseri, D. (2005). *Introduzione alla metodologia della ricerca*. Università (Soveria Mannelli, Italy). Rubbettino. (Cited on page 3.)

Antoine, A. and Miranda, E. R. (2015). Towards intelligent orchestration systems. In *11th International Symposium on Computer Music Multidisciplinary Research, Plymouth, United Kingdom*, page 124. (Cited on page 24.)

Arewa, O. B. (2012). Copyright and borrowing. *Legal Studies Research Paper Series No.10*, pages 33–52. (Cited on pages 15 and 16.)

Assayag, G. (1998). Computer assisted composition today. In *First Symposium on Music and Computers*. Corfu Greece. (Cited on page 59.)

Assayag, G. and al. (1999). Computer assisted composition at Ircam: From Patchwork to OpenMusic. *Computer Music Journal*, (23 (3)):59–72. (Cited on pages 59 and 65.)

Assayag, G., Block, G., Chemillier, M., Cont, A., and Dubnov, S. (2006). Omax Brothers: a Dynamic Topology of Agents for Improvization Learning. In *ACM Multimedia*, Santa Barbara. (Cited on pages 125 and 138.)

Babuschkin, I. (2016). A tensorflow implementation of deepmind's wavenet paper. https://github.com/ibab/tensorflow-wavenet. GitHub repository. (Cited on page 140.)

Barlow, C. (2011). Music Derived from Other Sources. *International Journal of the Humanities*, 9(7). (Cited on pages 13 and 42.)

Bateman, A. and Bale, J., editors (2008). *Sporting Sounds: Relationships Between Sport and Music*. Routledge. (Cited on page 46.)

Beylot, P. (2004). La citation, un espace de problématisation des pratiques artistiques. In *Emprunts et citations dans le champ artistique*, pages 9–15. L'Harmanattan, Paris, Pierre Beylot edition. (Cited on page 14.)

Bigo, L., Ghisi, D., Antoine, S., and Moreno, A. (2015). Representation of musical structures and processes in simplicial chord spaces. *Computer Music Journal*, 39(3):11–27. (Cited on pages 98, 106 and 147.)

Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., and Widmer, G. (2016). madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands. (Cited on pages 25 and 49.)

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. R., and Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. In *ISMIR*, pages 493–498. Citeseer. (Cited on pages 25 and 51.)

Bossis, B. (2005). *La voix et la machine*. Presses universitaires de Rennes. (Cited on pages 42 and 136.)

Boyle, J. and Jenkins, J. (2017). *Theft! A History of Music*. Duke Center for the Study of the Public Domain. (Cited on pages 13, 16 and 17.)

Bresson, J. (2006). Sound processing in OpenMusic. In *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06)*, Montreal, Canada. (Cited on page 124.)

Bresson, J. (2014). Reactive visual programs for computer-aided music composition. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, Melbourne, Australia. (Cited on page 60.)

Brock, A. (2016). Voxel-Based Variational Autoencoders, VAE GUI, and Convnets for Classification. `https://github.com/ajbrock/Generative-and-Discriminative-Voxel-Modeling`. GitHub repository. (Cited on pages 149, 150 and 151.)

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*. (Cited on page 149.)

Brossier, P. M. (2006). The aubio library at mirex 2006. *Synthesis*. (Cited on page 25.)

Bullock, J. and Momeni, A. (2015). ml.lib: Robust, Cross-platform, Open-source Machine Learning for Max and Pure Data. In *Proceedings of the international conference on New Interfaces for Musical Expression*, pages 265–270. (Cited on page 155.)

Burkholder, J. (2004). *All Made of Tunes: Charles Ives and the Uses of Musical Borrowing*. Yale University Press. (Cited on page 14.)

Burkholder, J. P. (1994). The uses of existing music: Musical borrowing as a field. *Notes*, 50(3):851–870. (Cited on page 13.)

Cannam, C., Landone, C., and Sandler, M. (2010). Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1467–1468. ACM. (Cited on page 25.)

Carroll, M. W. (2005). The struggle for music copyright. *Fla. L. Rev.*, 57:907. (Cited on pages 15 and 17.)

Cella, C. E. (2011). *On symbolic representations of music*. PhD thesis, alma. (Cited on pages 136 and 138.)

Cemgil, A. T. (2004). *Bayesian music transcription*. PhD thesis, RU Radboud Universiteit Nijmegen. (Cited on page 131.)

Cemgil, A. T. and Kappen, B. (2003). Monte Carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18(1):45–81. (Cited on page 131.)

Chew, E. (2000). *Towards a mathematical model of tonality*. PhD thesis, Massachusetts Institute of Technology. (Cited on page 146.)

Chintala, S. (2016). DCGAN.torch: a torch implementation of http://arxiv.org/abs/1511.06434. `https://github.com/soumith/dcgan.torch`. GitHub repository. (Cited on pages 142, 143 and 145.)

Cipriani, A., Ciardi, F. C., Ceccarelli, L., and Cardi, M. (2004). Collective composition: the case of Edison Studio. *Organised Sound*, 9(3):261–270. (Cited on page 159.)

Coduys, T. and Ferry, G. (2004). Iannix-aesthetical/symbolic visualisations for hypermedia composition. In *Proceedings of the Sound and Music Computing Conference*, pages 18–23. (Cited on pages 106 and 108.)

Cohn, R. (1998). Introduction to neo-riemannian theory: a survey and a historical perspective. *Journal of Music Theory*, pages 167–180. (Cited on page 98.)

Connes, A. (2005). Advice to the beginner. `http://www.alainconnes.org/`. Collège de France, Institut des Hautes Études Scientifiques and Vanderbilt University. (Cited on page 2.)

Cont, A. (2008a). ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music. In *Proceedings of the International Computer Music Conference*, Belfast, Ireland. (Cited on page 66.)

Cont, A. (2008b). *Modeling Musical Anticipation*. PhD thesis, University of Paris 6 and University of California in San Diego. (Cited on page 60.)

Cope (Accessed: 5th June, 2017). Experiments In Musical Intelligence. (Cited on page 153.)

Cope, D. (2004). *Virtual Music: Computer Synthesis of Musical Style*. MIT Press. (Cited on pages 6, 7 and 152.)

Desain, P. and Honing, H. (1989). The quantization of musical time: A connectionist approach. *Computer Music Journal*, 13(3):56–66. (Cited on page 131.)

Deutsch, D. (1975). *Musical illusions*. Freeman. (Cited on page 42.)

Didkovsky, N. and Hajdu, G. (2008). Maxscore: Music Notation in Max/MSP. In *Proceedings of the International Computer Music Conference*. (Cited on pages 60 and 66.)

Downie, J. S. (2003). Music information retrieval. *Annual review of information science and technology*, 37(1):295–340. (Cited on page 25.)

Downie, J. S. and Cunningham, S. J. (2002). Toward a theory of music information retrieval queries: System design implications. In *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 39.)

Duchen, J. (2012). Pierre Boulez: A very modern maestro. *Independent*. (Cited on page 17.)

Dunne, J. W. (1927). *An Experiment with Time*. Faber. (Cited on page 29.)

Eco, U. (1984). *Postille a "Il Nome della rosa"*. I Grandi tascabili. Bompiani. (Cited on page 13.)

Einbond, A. (2016). Musique instrumentale concrète: Timbral transcription in What the Blind See and Without Words. In Bresson, J., A. C. and G., A., editors, *OM Composer's Book 3*, pages 155–172. Editions Delatour France / IRCAM - Centre Pompidou. (Cited on page 23.)

Einbond, A., Schwarz, D., Borghesi, R., and Schnell, N. (2016). Introducing catoracle: Corpus-based concatenative improvisation with the audio oracle algorithm. In *Proceedings of the International Computer Music Conference*. (Cited on pages 23 and 138.)

Einbond, A., Trapani, C., Agostini, A., Ghisi, D., and Schwarz, D. (2014). Fine-tuned Control of Concatenative Synthesis with CataRT Using the bach Library for Max. In *Proceedings of the International Computer Music Conference*, Athens, Greece. (Cited on pages 23 and 76.)

Eliot, T. (1997). *The Sacred Wood and Major Early Essays*. Dover books on literature and drama. Dover Publications. (Cited on page 17.)

Emmerson, S. (2009). Combining the acoustic and the digital: music for instruments and computers or prerecorded sound. In *The Oxford Handbook of Computer Music*, pages 167–190. Oxford University Press, New York, Roger T. Dean edition. (Cited on page 110.)

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. (2017). Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. *ArXiv e-prints*. (Cited on pages 152 and 154.)

Favory, J. (2007). Les unités sémiotiques temporelles. *Mathématiques et sciences humaines. Mathematics and social sciences*, (178):51–55. (Cited on page 20.)

Fiebrink, R. and Cook, P. R. (2010). The Wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference*, Utrecht. (Cited on page 155.)

Fineberg, J. (1993). Esquisse - library-reference manual (code de Tristan Murail, J. Duthen and C. Rueda). (Cited on page 71.)

Fober, Y., Orlarey, Y., and Letz, S. (2012). An Environment for the Design of Live Music Scores. In *Proceedings of the Linux Audio Conference*. (Cited on pages 60, 66 and 106.)

Foote, J. (1997). A similarity measure for automatic audio classification. In *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*. (Cited on page 25.)

Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., and Zimmermann, P. (2007). MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software (TOMS)*, 33(2):13. (Cited on page 106.)

Freese, R. (2004). Automated lattice drawing. *Lecture notes in computer science*, pages 112–127. (Cited on page 108.)

Friedman, K., Smith, O. F., and Sawchyn, L. e. (2002). *The Fluxus Performance Workbook*. (Cited on page 159.)

Fruchterman, T. M. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164. (Cited on page 101.)

Gay, D. (1997). *Geometry by Discovery*. Wiley. (Cited on page 90.)

Gell, A. (1998). *Boost Graph Library, The: User Guide and Reference Manual*. Clarendon Press. (Cited on page 16.)

Ghisi, D. and Agon, C. (2016). Real-time corpus-based concatenative synthesis for symbolic notation. In *Proceedings of the TENOR Conference*, Cambridge, United Kingdom. (Not cited.)

Ghisi, D. and Agostini, A. (2017). Extending bach: A Family of Libraries for Real-time Computer-assisted Composition in Max. *Journal of New Music Research*, 46(1):34–53. (Cited on pages 61, 124 and 165.)

Ghisi, D., Agostini, A., and Maestri, E. (2016). Recreating Gérard Grisey's Vortex Temporum with cage. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands. (Not cited.)

Ghisi, D. and Bergomi, M. (2016). Concatenative synthesis via chord-based segmentation for An Experiment with Time. In *Proceedings of the International Computer Music Conference*, Utrecht, Nederlands. (Not cited.)

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. (Cited on page 141.)

Granlund, T. et al. (1991). GMP, the GNU multiple precision arithmetic library. (Cited on page 106.)

Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243. (Cited on page 141.)

Grisey, G. (1987). Tempus ex machina: A composer's reflections on musical time. *Contemporary Music Review*, 2(1):239–275. (Cited on pages 8, 128 and 157.)

Grisey, G. (2008). *Écrits : ou l'invention de la musique spectrale*. MF Éditions, Paris. (Cited on page 157.)

Gurtner, C. (2016). Applications of Audio and MIDI API within a music notation editor. Georgia Institute of Technology. (Cited on page 159.)

Hackbarth, B., Schnell, N., and Schwarz, D. (2011). Audioguide: A Framework For Creative Exploration Of Concatenative Sound Synthesis. IRCAM research report. (Cited on page 24.)

Hagan, K. L. (2014). How live is real-time? In *Electroacoustic Music Studies Conference*. Electroacoustic Music Studies Network. (Cited on page 60.)

Hajdu, G. (2005). Quintet.net: An environment for composing and performing music on the internet. *Leonardo*, 38(1):23–30. (Cited on page 167.)

Hamel, K. (1997). NoteAbility Reference Manual. (Cited on page 66.)

Harker, A. (2014). A Set of 80+ Externals for a variety of tasks in MaxMSP. `https://github.com/AlexHarker/AHarker_Externals`. GitHub repository. (Cited on page 25.)

Harte, C., Sandler, M., and Gasser, M. (2006). Detecting harmonic change in musical audio. In *Proceedings of Audio and Music Computing for Multimedia Workshop*. (Cited on pages 71, 81 and 82.)

Hélie, T., Picasso, C., and Calvet, A. (2016). The Snail: un nouveau procédé d'analyse et de visualisation du son. In *Europiano-France*, Tours, France. (Cited on page 147.)

Herrera, P., Serra, X., and Peeters, G. (1999). Audio Descriptors and Descriptor Schemes in the Context of MPEG-7. In *Proceedings of the International Computer Music Conference*. (Cited on pages 25 and 27.)

Hervé, J. L. (2001). *Dans le vertige de la durée : Vortex Temporum de Gérard Grisey*. L'Harmattan, L'Itineraire. (Cited on pages 114, 115 and 116.)

Hewett, I. (2014). Who Killed Classical Music? Radio broadcast, BBC Radio 4, 21/01/2014. (Cited on page 8.)

Hirs, R. and editors, B. G. (2009). *Contemporary compositional techniques and OpenMusic*. Delatour/Ircam. (Cited on page 71.)

Hofstadter, D. R. (1999). *Gödel, Escher, Bach*. Basic Books. (Cited on page 94.)

Huang, A. and Wu, R. (2016). Deep learning for music. *arXiv preprint arXiv:1606.04930*. (Cited on page 136.)

Huggins-daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M., and Rudnicky, A. I. (2006). Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proceedings of ICASSP*. (Cited on page 46.)

Huizinga, J. (1949). *Homo Ludens*. Routledge. (Cited on page 94.)

James, S. G. (2005). *Developing a flexible and expressive realtime polyphonic wave terrain synthesis instrument based on a visual and multidimensional methodology*. PhD thesis, Edith Cowan University. (Cited on page 107.)

Johnson, D. D. (2017). *Generating Polyphonic Music Using Tied Parallel Networks*, pages 128–143. Springer International Publishing, Cham. (Cited on page 136.)

Johnson, J. (2016). Efficient, reusable RNNs and LSTMs for torch. `https://github.com/jcjohnson/torch-rnn`. GitHub repository. (Cited on page 139.)

Johnson, T. (2012). Three notes for 3 jugglers. (Cited on page 94.)

Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15. (Cited on page 101.)

Karpathy, A. (2015). char-rnn: Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch. `https://github.com/karpathy/char-rnn`. GitHub repository. (Cited on page 139.)

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*. (Cited on page 139.)

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. (Cited on page 144.)

Kitahara, T. (2010). *Mid-level Representations of Musical Audio Signals for Music Information Retrieval*, pages 65–91. Springer, Berlin, Heidelberg. (Cited on page 26.)

Kivy, P. (1987). Platonism in music: Another kind of defense. *American Philosophical Quarterly*, 24(3):245–252. (Cited on page 1.)

Kramer, J. (2016). *Postmodern Music, Postmodern Listening*. Bloomsbury Academic. (Cited on pages 3, 14 and 41.)

Kuleshov, V., Enam, S. Z., and Ermon, S. (2017). Audio super-resolution using neural networks. In *ICLR*. (Cited on page 154.)

Lachenmann, H. (2004). Philosophy of Composition: Is There Such a Thing? In *Identity and Difference. Essays on Music, Language and Time*, pages 55–69. Leuven University Press. (Cited on pages 9, 40 and 153.)

Laurson, M. and Duthen, J. (1989). Patchwork, a graphical language in preform. In *Proceedings of the International Computer Music Conference*, pages 172–175, Miami, USA. (Cited on pages 65 and 71.)

Laurson, M. and Kuuskankare, M. (2002). PWGL: A Novel Visual Language based on Common Lisp, CLOS and OpenGL. In *Proceedings of International Computer Music Conference*, pages 142–145, Gothenburg, Sweden. (Cited on pages 59 and 65.)

Leeman, R. (2004). Comment se servir d'un Rembrandt. La citation, le postmoderne et la planche à repasser. In *Emprunts et citations dans le champ artistique*, pages 33–44. L'Harmanattan, Paris, Pierre Beylot edition. (Cited on page 13.)

Lehmann, H. (2017). *La Révolution digitale dans la musique*. Allia. (Cited on page 4.)

Lévy, F. (2007). *Notre notation musicale est-elle une surdité au monde ?* Number 5. éditions Bruylant, Paris. (Cited on page 127.)

Lévy, F. (2014). *Le compositeur, son oreille et ses machines à écrire*. Vrin. (Cited on pages 128, 129 and 130.)

Lew, M. S., Sebe, N., Djeraba, C., and Jain, R. (2006). Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19. (Cited on page 25.)

Lightman, A. (2005). *A Sense of the Mysterious: Science and the Human Spirit*. Knopf Doubleday Publishing Group. (Cited on page 1.)

Lindborg, P. (2008). About TreeTorika: Rhetoric, CAAC and Mao. In In Bresson, J., A. C. and G., A., editors, *OM Composer's Book 2*, pages 95–116. Editions Delatour France / IRCAM - Centre Pompidou. (Cited on page 42.)

Liu, I. and Ramakrishnan, B. (2014). Bach in 2014: Music Composition with Recurrent Neural Network. *ArXiv e-prints*. (Cited on page 136.)

Lostanlen, V. and Mallat, S. (2016). Wavelet scattering on the pitch spiral. *arXiv preprint arXiv:1601.00287*. (Cited on pages 147 and 148.)

Maas, A., Le, Q. V., O'Neil, T. M., Vinyals, O., Nguyen, P., and Ng, A. Y. (2012). Recurrent neural networks for noise reduction in robust asr. In *INTERSPEECH*. (Cited on page 154.)

Macauley, M., Advisor, M. O., and Macauley, M. (2003). Abstract braids and juggling patterns. (Cited on page 94.)

Maestri, E. and Antoniadis, P. (2015). Notation as Instrument: From Representation to Enaction. In *Proceedings of the TENOR Conference*. (Cited on page 110.)

Malt, M. (Juin 6th, 2017). Descripteurs sonores et écriture musicale. http://www.college-de-france.fr/site/philippe-manoury/symposium-2017-06-06-10h15.htm. Symposium "État de l'art / état d'alerte", Collège de France, Paris. (Cited on pages 23 and 28.)

Malt, M. and Jourdan, E. (2008). Zsa. descriptors: a library for real-time descriptors analysis. *Sound and Music Computing, Berlin, Germany*. (Cited on page 25.)

Malt, M. and Schilingi, J. B. (1995). Profile - libreria per il controllo del profilo melodico per Patchwork. In *Proceedings of the XI Colloquio di Informatica Musicale (CIM)*, pages 237–238, Bologna, Italia. (Cited on page 71.)

Manning, P. (1985). *Electronic and Computer Music*. Clarendon Press, Oxford. (Cited on page 129.)

Manoury, P. (2007). *Considérations (toujours actuelles) sur l'état de la musique en temps réel*. Ircam - Centre Pompidou, Paris. (Cited on page 128.)

Mathews, M. (1963). The digital computer as a musical instrument. (3591):553–557. (Cited on page 110.)

Mathews, M., Moore, F. R., and Risset, J.-C. (1974). Computers and Future Music. *Science*, 183(4122). (Cited on page 110.)

Mathieu, B., Essid, S., Fillon, T., Prado, J., and Richard, G. (2010). YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In *ISMIR*, pages 441–446. (Cited on page 25.)

Maturana, D. and Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*. (Cited on page 149.)

Mauch, M. (2010). *Automatic chord transcription from audio using computational models of musical context*. PhD thesis, School of Electronic Engineering and Computer Science Queen Mary, University of London. (Cited on pages 32 and 49.)

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*. (Cited on pages 25 and 148.)

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264(5588):746–748. (Cited on page 42.)

McKay, C. (2010). *Automatic music classification with jMIR*. PhD thesis, Citeseer. (Cited on pages 25 and 135.)

McKay, C. and Fujinaga, I. (2006). jSymbolic: A Feature Extractor for MIDI Files. In *International Computer Music Conference*. (Cited on pages 26 and 27.)

Mehri, S. (2016). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. `https://github.com/soroushmehr/sampleRNN_ICLR2017`. GitHub repository. (Cited on pages 140 and 141.)

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. C., and Bengio, Y. (2016). Samplernn: An unconditional end-to-end neural audio generation model. *CoRR*, abs/1612.07837. (Cited on pages 136 and 140.)

Mériaux, S. (2012). *Dans la chair du monde*. L'harmattan. (Cited on page 18.)

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*. (Cited on page 143.)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. (Cited on page 142.)

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751. (Cited on page 143.)

Miranda, E. (2001). *Composing music with computers*. CRC Press. (Cited on page 59.)

Mordvintsev, A., Olah, C., and Tyka, M. (2015). Deepdream-a code example for visualizing neural networks. *Google Res.* (Cited on page 141.)

Nash, C. (2015). The cognitive dimensions of music notations. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation*, Paris, France. (Cited on page 110.)

Oswald, J. (1985). Plunderphonics, or audio piracy as a compositional prerogative. In *Wired Society Electro-Acoustic Conference*, pages 5–8. (Cited on pages 4 and 15.)

Page, K. R., Fields, B., De Roure, D., Crawford, T., and Downie, J. S. (2012). Reuse, remix, repeat: the workflows of mir. In *ISMIR*, pages 409–414. (Cited on page 25.)

Paine, T. L., Khorrami, P., Chang, S., Zhang, Y., Ramachandran, P., Hasegawa-Johnson, M. A., and Huang, T. S. (2016). Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*. (Cited on page 140.)

Patterson, R. D. (1986). Spiral detection of periodicity and the spiral form of musical scales. *Psychology of Music*, 14(1):44–61. (Cited on page 148.)

Peeters, G. (2004). A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, Ircam, Analysis/Synthesis Team, 1 pl. Igor Stravinsky, 75004 Paris, France. (Cited on pages 25 and 26.)

Procházková, J. (2006). *Janáčkovy záznamy hudebního a tanečního fokloru I.* Etnologický ústav AV ČR, Doplněk. (Cited on page 42.)

Puckette, M. (2002). Max at seventeen. *Computer Music Journal*, 26(4):31–43. (Cited on pages i, 8 and 27.)

Puckette, M. (2004). A divide between 'compositional' and 'performative' aspects of Pd. In *Proceedings of the First Internation Pd Convention*, Graz, Austria. (Cited on page 60.)

Puckette, M. et al. (1996). Pure Data: another integrated computer music environment. *Proceedings of the second intercollege computer music concerts*, pages 37–41. (Cited on page 27.)

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. (Cited on page 142.)

Ranum, P. M. (2001). *The Harmonic Orator*. Pendragon. (Cited on page 42.)

Reingold, E. M. and Tilford, J. S. (1981). Tidier drawings of trees. *IEEE Transactions on Software Engineering*, (2):223–228. (Cited on page 108.)

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34. (Cited on page 100.)

Roads, C. (1978). Automated Granular Synthesis of Sound. *Computer Music Journal*, 2 (2). (Cited on page 27.)

Roads, C. (1996). *The computer music tutorial*. MIT press. (Cited on pages 59 and 91.)

Roads, C. (2004). *Microsound*. MIT Press. (Cited on pages 27 and 138.)

Roberts, A., Hawthorne, C., and Simon, I. (2014). Magenta: Music and Art Generation with Machine Intelligence. `https://github.com/tensorflow/magenta`. GitHub repository. (Cited on page 136.)

Robinson, S. (1984). *Melancholy Elephants*. Penguin short fiction. Penguin Books. (Cited on page 2.)

Roma, G. and Serra, X. (2015). Querying freesound with a microphone. In *Proceedings of the First Web Audio Conference (Ircam, Paris, France), submission*, volume 39. (Cited on page 149.)

Roubet, A. (2013). Emprunts et citations. (Cited on page 13.)

Rueda, C. and Bonnet, A. (1998). Un Langage Visuel Basé sur les Contraintes pour la Composition Musicale. (Cited on page 125.)

Ruviaro, B. (2007). Musical Borrowing, Donatoni, Pousseur. `http://www.alainconnes.org/`. (Cited on pages 14 and 17.)

Sadai, Y. (1992). Le musicien-cognitiviste face au cognitiviste-musicien, ou la méthode de la cognition et la cognition de la méthode. In *Analyse musicale*, number 26, pages 30–33. (Cited on page 130.)

Sauer, T., editor (2009). *Notations 21*. Mark Batty Publisher. (Cited on page 46.)

Schaeffer, P. (1966). *Traité des objets musicaux, Editions du Seuil*. Paris. (Cited on page 110.)

Schell, J. (2014). *The Art of Game Design: A book of lenses*. CRC Press. (Cited on page 96.)

Schnell, N., Borghesi, R., Schwarz, D., Bevilacqua, F., and Müller, R. (2005). FTM - Complex Data Structures for Max. In *Proceedings of the International Computer Music Conference*. (Cited on pages 28 and 66.)

Schnell, N., Röbel, A., Schwarz, D., Peeters, G., and Borghesi, R. (2009). MuBu & Friends - Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP. In *Proceedings of the International Computer Music Conference*, Montreal, Canada. (Cited on pages 25, 28, 44 and 66.)

Schwarz, D. (2007). Corpus-based concatenative synthesis. *Signal Processing Magazine, IEEE*, 24(2):92–104. (Cited on page 28.)

Schwarz, D., Beller, G., Verbrugghe, B., and Britton, S. (2006). Real-Time Corpus-Based Concatenative Synthesis with CataRT. In *Proceedings of the International Conference on Digital Audio Effects*, pages 279–282. (Cited on pages 28 and 106.)

Seeger, C. (1948). Prescriptive and descriptive music-writing. (44 (2)):184–195. (Cited on page 110.)

Seleborg, C. (2004). Interactions temps-réel/temps differé. Master's thesis, Memoire ATIAM, Marseille. (Cited on page 61.)

Shepard, R. N. (1964). Circularity in judgments of relative pitch. *The Journal of the Acoustical Society of America*, 36(12):2346–2353. (Cited on page 148.)

Sica, S. and D'Antonio, V. (2010). The Balance of Copyright in Italian National Law. Comparazione e Diritto Civile. (Cited on page 16.)

Siek, J. G., Lee, L.-Q., and Lumsdaine, A. (2001). *Art and Agency: An Anthropological Theory*. Addison-Wesley Professional. (Cited on page 100.)

Sirens Cycle (Accessed: 29th April, 2017). Sirens Cycle. `http://www.bachproject.net/2016/09/29/sirens-cycle/`. (Cited on page 42.)

Smalley, D. (1997). Spectromorphology: explaining sound-shapes. *Organised sound*, 2(2):107–126. (Cited on page 21.)

Sobel, B. L. W. (2017). Artificial Intelligence's Fair Use Crisis. *Columbia Journal of Law And the Arts*. (Cited on page 156.)

Sorensen, A. and Gardner, H. (2010). Programming with Time. Cyber-physical programming with Impromptu. In *Proceedings of OOPSLA10 : ACM International Conference on Object Oriented Programming Systems Languages and Applications*, pages 822–834, New York. ACM. (Cited on page 110.)

Sox (Accessed: 27th April, 2017). Sox: Sound eXchange. `http://sox.sourceforge.net/`. (Cited on page 49.)

Stockhausen, K. (1957). *...wie die Zeit vergeht...* Number 3. Wien. (Cited on page 157.)

Stowell, D. (2011). Scheduling and composing with Risset eternal accelerando rhythms. In *International Computer Music Conference*. (Cited on page 30.)

Stowell, D. (2016). Example of an autoencoder set up for spectrograms, using Theano and Lasagne. `https://github.com/danstowell/autoencoder-specgram`. GitHub repository. (Cited on page 145.)

Streich, S. and Herrera, P. (2005). Detrended fluctuation analysis of music signals: Danceability estimation and further semantic characterization. In *Proceedings of the 118th AES Convention*. (Cited on page 26.)

Stroppa, M. (2000). Paradigms for the high level musical control of digital signal processing. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-00)*, Verona, Italy. (Cited on page 124.)

Takeda, H., Saito, N., Otsuki, T., Nakai, M., Shimodaira, H., and Sagayama, S. (2002). Hidden Markov model for automatic transcription of MIDI signals. In *2002 IEEE Workshop on Multimedia Signal Processing.*, pages 428–431. (Cited on pages 131 and 132.)

Taube, H. (1991). Common Music: A Music Composition Language in Common Lisp and CLOS. *Computer Music Journal*, 15. (Cited on page 59.)

Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43. (Cited on page 136.)

Tomás, E. (2016). Musical Instruments as Scores: a Hybrid Approach. In *Proceedings of the TENOR Conference*, Cambridge, United Kingdom. (Cited on page 111.)

Trapani, C. (2017). *Convergence Lines: A Musical Distillation of Thomas Pynchon's V.* PhD thesis, Columbia University (New York). (Cited on pages 14 and 23.)

Truchet, C., Assayag, G., and Codognet, P. (2003). OMClouds, Petits Nuages de Contraintes dans OpenMusic. In *Proceedings of the Journées d'Informatique Musicale*, Montbéliard, France. (Cited on page 125.)

Turowski, P. (2016). *Digital Game as Musical Notation.* PhD thesis, University of Virginia. (Cited on pages 95, 96 and 106.)

Typke, R., Wiering, F., Veltkamp, R. C., et al. (2005). A Survey of Music Information Retrieval Systems. In *Ismir*, pages 153–160. (Cited on pages 25 and 149.)

Tzanetakis, G. and Cook, P. (2000). Marsyas: A framework for audio analysis. *Organised sound*, 4(3):169–175. (Cited on page 25.)

Vaggione, H. (1996). Articulating Microtime. *Computer Music Journal*, 20 (2). (Cited on page 27.)

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *Arxiv*. (Cited on pages 136 and 140.)

Wang, Q. and Boyer, K. L. (2013). Feature learning by multidimensional scaling and its applications in object recognition. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on*, pages 8–15. IEEE. (Cited on pages 79 and 80.)

Wichern, G. (2017). What the Machine Learning in RX 6 Advanced Means for the Future of Audio Repair Technology. https://www.izotope.com/en/community/blog/product-news/2017/04/what-the-machine-learning-in-rx-6-advanced-means-for-the-future-of-audio-repair-technology.html. iZotope Blog. (Cited on page 154.)

Wold, E., Blum, T., Keislar, D., and Wheaten, J. (1996). Content-based classification, search, and retrieval of audio. *IEEE multimedia*, 3(3):27–36. (Cited on page 25.)

Wyse, L. and Smoliar, S. (1995). Toward content-based audio indexing and retrieval and a new speaker discrimination technique. *Proc. ICJAI*, 95. (Cited on page 25.)

Ycart, A., Jacquemard, F., Bresson, J., and Staworko, S. (2016). A supervised approach for rhythm transcription based on tree series enumeration. In *International Computer Music Conference*. (Cited on page 131.)

Zalewski, D. (March 12th, 2012). The Hours. How Christian Marclay created the ultimate digital mosaic. *The New Yorker*. (Cited on pages 15 and 23.)

Zils, A. and Pachet, F. (2001). Musical mosaicing. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DaFx-01)*, pages 39–44, Limerick, U.K. University of Limerick. (Cited on page 28.)