REINJECTION MATRICES WITH FAUST LANGUAGE: CREATING COMPLEX STRUCTURES IN REAL-TIME MIXED MUSIC

João Svidzinski
Université Paris 8
Laboratoire Musidanse/CICM
Saint-Denis, France
`svidzinski@gmail.com`

Alain Bonardi
Université Paris 8
Laboratoire Musidanse/CICM
Saint-Denis, France
`alain.bonardi@gmail.com`

ABSTRACT

This article discusses our recent and ongoing research into the composition and analysis of real-time mixed music with Faust language. We will discuss our recent compositions, in particular, João Svidzinski's *Les cris du sixième siècle* for soprano and live-electronics (2016) and Alain Bonardi's *Fil de Soi 1 and 2* (2016-2018) for guitar and live-electronics, in perspective with our analysis, transcoding and recast of the real-time version of Jean Claude Risset's *Inharmonique* (1977). In addition, we will present the principle of matrix reinjection with two modules generated in Faust, an Harmonizer-Delay and a Granulator. It allowed us to compose complex structures in real-time mixed composition. We will first present this principle and then outline a validation of this method by transcoding a PLF 4 routine from the *Inharmonique* original Music V code (with the assistance of its new transcoding) with the same Faust module as in our compositions. That is a creative approach, where the importance of code is at the heart of our compositional and analytical activity in the mixed music repertory.

1. INTRODUCTION

The functional programming language Faust - Functional AUdio STream - offers a good alternative for audio processing in real-time. Moreover, thanks to its strong compatibility and code comprehension, it is a fascinating tool for the preservation of the real-time mixed repertory. Therefore, Faust is used more and more in the computer music context for digital *music making*, especially in the concert activities as part of our research. Our goal is to correlate the analysis of the classical computer music repertory with new compositions. In this context, Faust language is a common tool which links these activities.

Since 2019, we lead the project "*Créer, (re)créer, valoriser la musique faisant appel à l'informatique au XXIe siècle*"[1], with the support of the MSH Paris Nord - Maison des Sciences de l'Homme Paris Nord. The main goal was the analysis, transcoding and recast to a real-time version of Jean Claude Risset's *Inharmonique* (1977) for soprano and tape. The coding of the new version was initiated, in the early 2010s, by the composers-researchers Antonio de Sousa Dias, José Luis Ferreira and Risset himself [1]. In 2019, the two authors of this paper joined the team. Thus, the first release of the new real-time transcoding was performed on November 28th, 2019. In the same program, the four composers-researchers who have worked on the recast of *Inharmonique* presented their own compositions. The two authors of this paper performed João Svidzinski's *Les cris du sixième cercle* (2019) for soprano and electronics and Alain Bonardi's *Dans la Nef de nos songes 1* (2019), an acousmatic piece. These two pieces include Faust modules, especially an Harmonizer-Delay object. The former uses a recurrent approach initiated by Bonardi: the reinjection matrix. That is, a feedback process whose process occurs in several instances of a single Faust object. For example, an Harmonizer-Delay can have multiple instances (that is, audio streams, e.g. channels in parallel). Then, the reinjection matrix can route the output of any instance to the input of any instance, making multiple feedbacks. This method was used and documented for the first time in Bonardi's *Pianotronics 3* (2016) for piano and live-electronics [2][2] The same method was found in our Ph.D. Thesis for the mixed compositions in real-time *Taurus* (2017) and *Tyqué* (2018) [3].

The new real-time version of *Inharmonique*, on the other hand, was transcoded using Csound DSP engines. Indeed, thanks to its syntax, the new version "preserves a flavor or idiomatic writing regarding Music V style, thus becoming a bridge between Music V" and new tools as Max software [1]. However, for the next research stage, we planned to use Faust to generate some passages of *Inharmonique*.

In this paper, we will present our strategy using the reinjection matrix principle with Faust modules in our mixed pieces. Then, the same method will be used to transcode the PLF 4 subroutines in Music V language used in *Inharmonique* [4][3]. At this point, this process intends to be the first step of using Faust to recast classical mixed repertory in computer music, but also to reevaluate our activity as composer.

First of all, we'll present the critical points of composing complex structures in real time mixed music. Jean-Claude Risset and Horacio Vaggione's thoughts will help us to establish our own notion of complex structures. Then, we'll demonstrate our strategy to compose complex structures with Faust modules and with the reinjection matrix principle. Our mixed pieces, Alain Bonardi's *Fil de Soi 1 and 2* (2016-2018) for guitar and live-electronics and João Svidzinski's *Les cris du sixième siècle* for soprano and live-electronics (2019), will exemplify our method. Finally, the same process will be used to transcode Risset's complex structures through Music V's subroutines PFL 4 in *Inharmonique.*

2. COMPLEX STRUCTURES COMPOSITION IN COMPUTER MUSIC

Complex structures are often related, in computer music and electroacoustic composition, to vertical structural construction. These strategies are often based on multilayer approaches.

[1] "Creating, (re)creating, enhancing the computer music in the 21st century".

[2] The Faust codes are available to download in the website http://alainbonardi.net [verified URL August 30, 2020].

[3] For a detailed and practical explanation of PLFs, see [12].

Jean-Claude Risset, as a pioneer in computer music, was one of the first to defend the composition of the sound itself. That is, with computers, "the sound is no longer the acoustic trace of material objects, it can be programmed and constructed" [5]. Risset was interested in sculpting the sound, to manipulate the timbre as a complexity. Since the establishment of digital technology, he has composed pieces whose structure is based on complex harmonic entities. In his early compositions *Suite Little Boy* (1968), and *Mutations* (1969) all the sounds were generated by digital synthesis. Then these techniques, made with Music V language, were published in his canonical "Catalogue of Computer Synthesized Sounds" [6]. Most of these codes were made by superposition of frequency components. That is the case of the instrument-like sounds (as piano, drum, gong and brass) and some non-instruments-like (for example, the generations of paradoxical sounds) [7]. As Risset himself said: "with synthesis, one can compose spectral and timbres just as musical chords, and one can attribute a harmonic function to timbre [...]. The synthesis of sustained tones with arbitrary spectral content has made possible the composition of novel musical structures tailoring timbre to harmony and scale, as exemplified by John Chowning's *Stria*" [8].

If Risset's creed is the composition of sound itself, Horacio Vaggione investigates object-based composition [9]. Indeed, the two viewpoints are quite similar. Vaggione understands the object as "a complex unit that may simultaneously contain different representations, or codes, related, to as many procedures (specific actions) as there are data elements (sounds and time structures) covering many scales or operating levels" [10]. Thus, the object networks are "multi-scale ensembles that include events of different order of magnitude". Furthermore, his notion of object does not only include sound events, it is also relevant to digital algorithms: "The objects may be functions (algorithms), lists of parameters (scores), scripts (successions of actions to be accomplished, or the may be sounds (products as well as sources)" [10]. Finally, Vaggione gives the clue for composing with sound objects: "by a double movement of fragmentation and agglutination several times, in order to create multiples, and to work the parts of these multiples to create others" [9]. Composition with object networks thus leads to create complex structures.

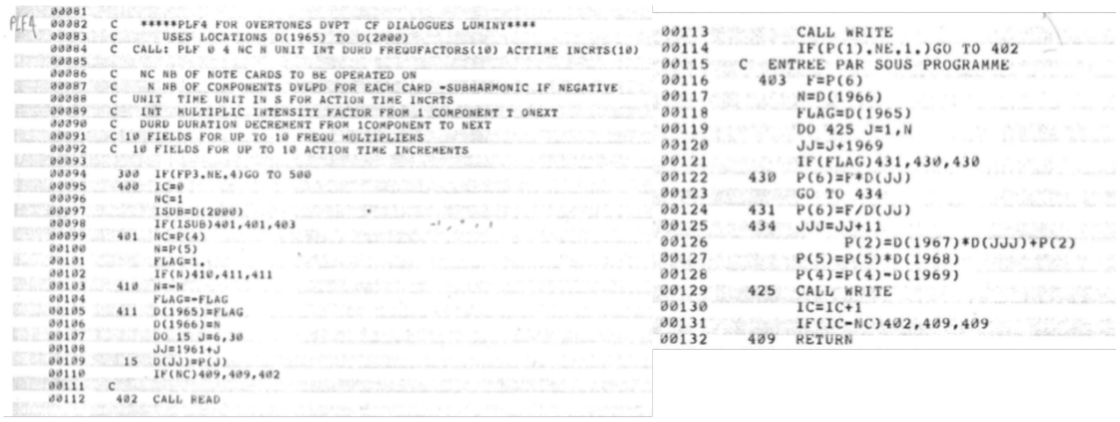
2.1. Complex structures with Music V and subroutines PLF 4

In *Inharmonique* by Risset (1977) for soprano and tape, the way in which the composer creates complex structures is based on PLFs subroutines. That is, a functionality of Music V language that enlarges the generation of events. This code was originally programmed in Fortran. Then as part of our research to transcode and recast *Inharmonique*, the algorithm process was transcribed in JavaScript. In our new real-time version, these codes run inside Max in order to articulate the subroutines in real time.

The Music V subroutines PLF (play it first), available since Music IV version, enable the generation of note parameters by the computer. Thus, these subroutines are useful in saving the human composer from much routine work. It also "let one embed algorithmic control into the sound synthesis process" [4]. For example, with only one operation, it is possible to generate many events. Actually, this language allows very fine and precise control. The internal data of a sound, such as the frequencies and amplitudes of the spectral components, are easily stored and instantiated during the composition process. This makes it easy to integrate synthetic sounds with instrumental sounds, as shown in *Inharmonique*. It was the first step to compose multilayer structures. At the same time, other composers also used PLF subroutines to create complex structures. Curtis Roads, for example, used these routines in his first realization of automated granular synthesis [11].

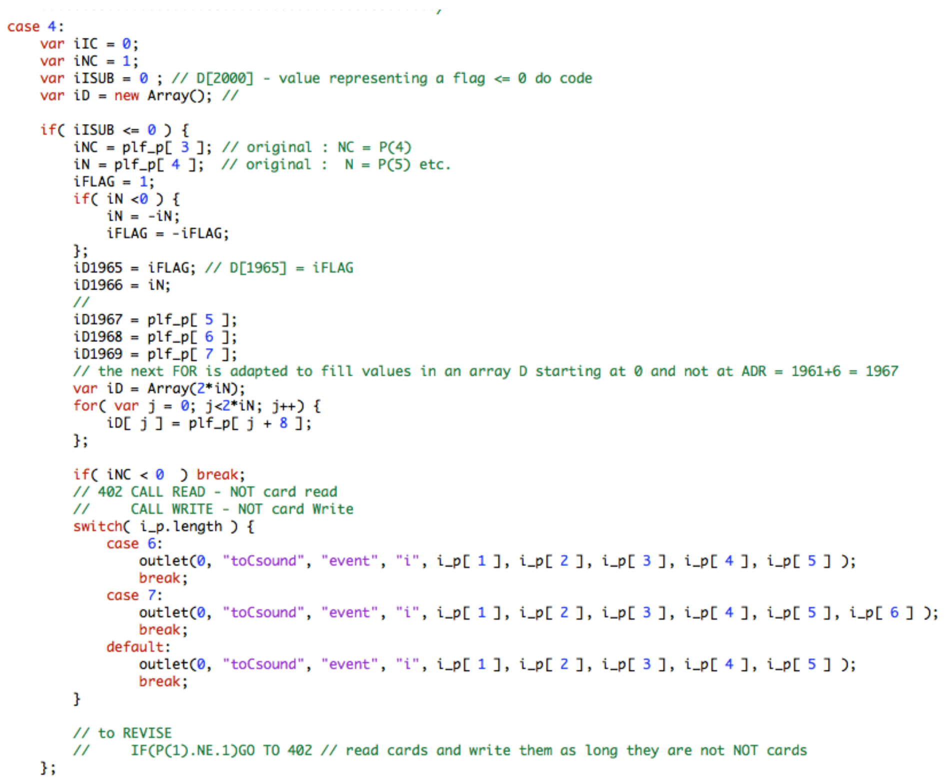
In Risset's works, there are several different PLFs subroutines. It is a method which allows, for example, "to carry out variations on a structure starting from a specification of global parameters different from those of this structure: this program modifies consequently frequency, amplitude and duration on all the components" [12]. In *Inharmonique*, as in many other pieces by the composer, the exact same code as in PLF 4 and 6 are often used. The former for the generation of instrument-like sounds and the latter for the generation of a series of multiple components of a fundamental, that can be shifted in time.

The PLF 4 JavaScript code (Figure 2), coded by Antonio de Sousa Dias, translates the instructions and generates the corresponding i-events. It is based upon several descriptions and is an adaptation of the original code written in Fortran language (Figure 1). Nevertheless, some modifications are made in order to adapt it to a more readable code [13].



```

C    *****PLF4 FOR OVERTONES DVPT  CF DIALOGUES LUMINY****
C       USES LOCATIONS D(1965) TO D(2000)
C  CALL: PLF 0 4 NC N UNIT INT DURD FREQUFACTORS(10) ACTTIME INCRTS(10)

C   NC NB OF NOTE CARDS TO BE OPERATED ON
C    N NB OF COMPONENTS DVLPD FOR EACH CARD -SUBHARMONIC IF NEGATIVE
C  UNIT  TIME UNIT IN S FOR ACTION TIME INCRTS
C   INT  MULTIPLIC INTENSITY FACTOR FROM 1 COMPONENT T ONEXT
C   DURD DURATION DECREMENT FROM 1COMPONENT TO NEXT
C 10 FIELDS FOR UP TO 10 FREQU MULTIPLIERS
C  10 FIELDS FOR UP TO 10 ACTION TIME INCREMENTS

   300  IF(FP3.NE.4)GO TO 500
   400  IC=0
        NC=1
        ISUB=D(2000)
        IF(ISUB)401,401,403
   401  NC=P(4)
        N=P(5)
        FLAG=1.
        IF(N)410,411,411
   410  N=-N
        FLAG=-FLAG
   411  D(1965)=FLAG
        D(1966)=N
        DO 15 J=6,30
        JJ=1961+J
    15  D(JJ)=P(J)
        IF(NC)409,409,402
C
   402  CALL READ
        CALL WRITE
        IF(P(1).NE.1.)GO TO 402
C  ENTREE PAR SOUS PROGRAMME
   403   F=P(6)
        N=D(1966)
        FLAG=D(1965)
        DO 425 J=1,N
        JJ=J+1969
        IF(FLAG)431,430,430
   430  P(6)=F*D(JJ)
        GO TO 434
   431  P(6)=F/D(JJ)
   434  JJJ=JJ+11
              P(2)=D(1967)*D(JJJ)+P(2)
        P(5)=P(5)*D(1968)
        P(4)=P(4)-D(1969)
   425  CALL WRITE
        IC=IC+1
        IF(IC-NC)402,409,409
   409  RETURN
```

Figure 1: *PLF 4 original Fortran language.*



```
case 4:
    var iIC = 0;
    var iNC = 1;
    var iISUB = 0 ; // D[2000] - value representing a flag <= 0 do code
    var iD = new Array(); //

    if( iISUB <= 0 ) {
        iNC = plf_p[ 3 ]; // original : NC = P(4)
        iN = plf_p[ 4 ];  // original :  N = P(5) etc.
        iFLAG = 1;
        if( iN <0 ) {
            iN = -iN;
            iFLAG = -iFLAG;
        };
        iD1965 = iFLAG; // D[1965] = iFLAG
        iD1966 = iN;
        //
        iD1967 = plf_p[ 5 ];
        iD1968 = plf_p[ 6 ];
        iD1969 = plf_p[ 7 ];
        // the next FOR is adapted to fill values in an array D starting at 0 and not at ADR = 1961+6 = 1967
        var iD = Array(2*iN);
        for( var j = 0; j<2*iN; j++) {
            iD[ j ] = plf_p[ j + 8 ];
        };

        if( iNC < 0  ) break;
        // 402 CALL READ - NOT card read
        //     CALL WRITE - NOT card Write
        switch( i_p.length ) {
            case 6:
                outlet(0, "toCsound", "event", "i", i_p[ 1 ], i_p[ 2 ], i_p[ 3 ], i_p[ 4 ], i_p[ 5 ] );
                break;
            case 7:
                outlet(0, "toCsound", "event", "i", i_p[ 1 ], i_p[ 2 ], i_p[ 3 ], i_p[ 4 ], i_p[ 5 ], i_p[ 6 ] );
                break;
            default:
                outlet(0, "toCsound", "event", "i", i_p[ 1 ], i_p[ 2 ], i_p[ 3 ], i_p[ 4 ], i_p[ 5 ] );
                break;
        }

        // to REVISE
        //     IF(P(1).NE.1)GO TO 402 // read cards and write them as long they are not NOT cards
    };
```

Figure 2: *PLF 4 transcoding JavaScript code.*

In short, for each PLF 4 call, five parameters must be defined (number of notes, number of harmonics per note, time unit, gain by harmonic, duration decrement) plus the time increment for each harmonic. Then for each event, the fundamental duration, frequency and amplitude must be set. For each event, all the harmonics behaviors are calculated in relation to the fundamental data. Thus, the PLF 4 makes harmonic scans [*balayages harmoniques*] for each fundamental, the aural result can be harmonic or inharmonic; fast or long. Furthermore, many events can overlap in order to create even more complex phenomena.

2.2. Complex structures composition in computer music

In his fascinating article "Composing in real-time?"[14], from 1999, Risset underlined the limitations and drawbacks of real-time computer composition. Actually, he wrote it at a moment where the real-time technology took up a lot of attention. Since the late 1970s, when Boulez created Ircam, his creed was the establishment of a way to enrich instrumental writing through digital technologies [15]. However, Risset himself pointed out good points: "Real-time time operation is invaluable to produce performance nuance with proper aural feedback" [16]. Actually, Risset's *Duet for one pianist* (1989), using a Disklavier Yamaha piano, is a remarkable example of real-time practice of the late 1980s. Nevertheless, his global thought is that real-time operation is limiting. The same point of view was shared by other composers whose music practice was based on creating complex structures. That is the case of Marco Stroppa's *Traiettoria* (1982-1984) for piano and tape. For that piece, the composer mainly used Music V language [15].

Among Risset's remarks against real-time to compose complex structures, we quote two in particular: "Real-time synthesis systems provide less flexible possibilities than software synthesis: rather than a full choice *à la carte,* they tend to offer a few *menus* for controlling specific sonic parameters through certain gestures". And, "Any digital system can only attain a limited level of sound complexity in real-time. Actually, complexity is often achieved in real-time systems through layering - thus forsaking real-time".

With Faust, however, these remarks were overcome. The Faust power to translate this wide vocabulary to non-domain specific languages (such as C++ for compiling Max objects) proves that, in real-time mix composition, we are able to much more than a "specific menu".

Indeed, the main strategy to make complex structures in real time is to make layering, but with the reinjection matrix coupled with an ambisonic specialization strategy, all the layering process is inside the Faust/Max environment. That is, the creation of complex structures, through overlapping components, is a unique real-time process.

Nevertheless, this procedure is not completely in real-time for a different reason pointed out by Risset. The Faust code must be developed out of time, since the algorithms still have status of compositional objects: they are also complex structures. In fact, this is not a problem, because the aural result will always be the most interesting point for a composer. Moreover, this potential price is paid by the advantageous solutions for obsolescence proposed by Faust. Furthermore, the composition act will always require an amount of freedom from the real-time constraints even for improvisation, as pointed out Risset himself [17].

It is not a question of militating in favor of real-time, but rather of proposing a transverse approach to draw the benefit of each method by making them coexist.

3. REINJECTION MATRIX FOR CREATING COMPLEX STRUCTURES

The reinjection matrix was presented for the first time in Bonardi's *Pianotronics 3* (2016) for piano and live electronics. Moreover, this piece introduces the object-oriented paradigm for the design of real-time processing (with Faust objects working into Max) and control (with JavaScript), using a class-instance approach. The same model has been used in our Ph.D. Thesis [3]. Then, we applied it in *Les cris du sixième cercle.*

In *Pianotronics 3,* the compositional strategy is based on a single generic module Harmonizer-Delay in Faust. Hereupon, it is possible to have several aural results by applying different global and local settings and by association of elementary processes [2]. Moreover, the reinjection matrix plays an important role. As the Faust module has 16 instances, a powerful aural palette is obtained by feedbacking all the instances. Similar to Risset's approaches, in *Pianotronics 3,* thanks to the association of Harmonizer-Delays with reinjection matrix, instruments-like sounds are generated (like drums and bells). However, the whole piece is achieved in real-time only through direct instrumental processing. That is an example of creating complex structures with Faust in real-time mixed music.

In the next section, we will get into this approach by analyzing a recent composition that shares the same process. Then, we will analyze another mixed pièce from the first author of this paper whose principle came from a similar environment.

3.1. Reinjection matrix for creating complex structures - Case of *Fil de soi 1 and 2*

With *Fil de soi 1 and 2* (2016-2018) for acoustic guitar and live-electronics, the composer aims the timbral composition in real-time. Thus, the word "timbre" has a special meaning [18]. Actually, this piece intends to explore a vast world of sonic possibilities of acoustic guitar through digital processing in real-time. To achieve this, the interaction between the acoustic instrument and the electronics is designed as a question/answer approach. With the timbral composition, the composer seeks to enrich the sonic results by changing the acoustic timbral through digital processing with Faust modules: "composing timbres" consists first of all in considering the acoustic guitar as a rich potential generator of sound, opening up a vast space for composition and transformation of the identity of the instrument" [18].

3.1.1. Faust Modules - mTDelHarmo16~ *object.*

The main Faust object used to achieve the "timbral composition" in *Fil de soi 1 and 2* is *mTDelHarmo16~*, compiled for Max. That is a set of 16 processes, each of them including a delay line and an harmonizer by Doppler effect (temporal approach thanks to variable delay) with possible reinjection to any line (Figure 3)

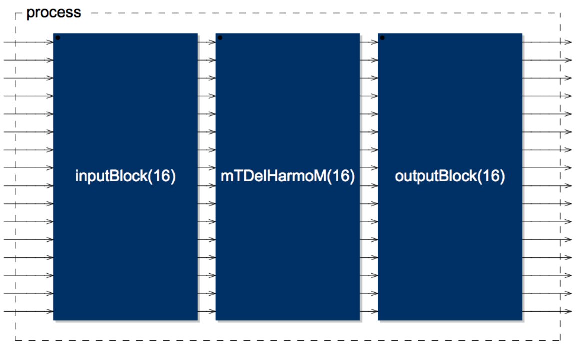


Figure 3: *Faust Modules* - mTDelHarmo16~ *object.*

The central part of this process is the *mTDelHarmoM* block. It includes two blocks *fdToMatrixBlock* and *DelHarmoBlock* (Figure 4). The former deals with rejection (it will be discussed in the next section), the latter processes delaying and harmonizing that are organized in two separate blocks: *DelBlock* and *HarmoBlock* (Figure 5).

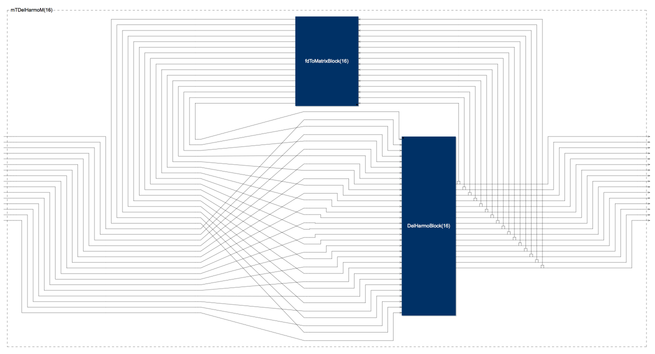


Figure 4: *Faust Scheme of the* mTDelHarmoM *block-diagram.*

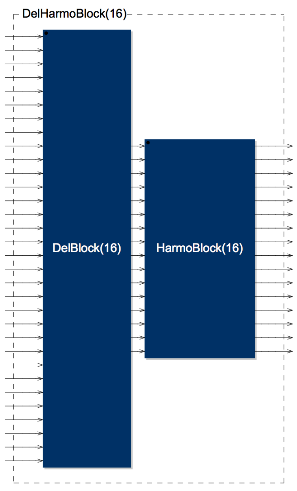


Figure 5: *Faust Scheme of the* DelBlock *and* HarmoBlock *inside* DelHarmoBlock.

The user must define an index value to set the amount of distribution between the harmonizer and the delay. Thus, the two processes are in series. Each delay line is a double overlapped delay enabling the change of duration without clicking. The harmonizer block, for its part, consists of four overlapped variable delays (Doppler effect). All these processes are illustrated below (Figure 6).

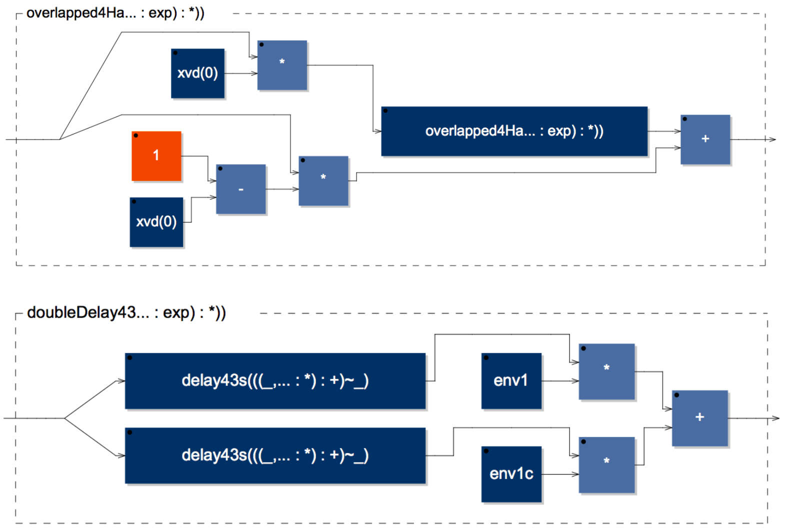

overlapped4Ha... : exp) : *))
xvd(0)
*
1
-
xvd(0)
*
overlapped4Ha... : exp) : *))
+
doubleDelay43... : exp) : *))
delay43s(((_,... : *) : +)~_)
delay43s(((_,... : *) : +)~_)
env1
env1c
*
*
+

Figure 6: *Faust Scheme of* DelHarmoBlock *(below) and* HarmoBlock *(above).*

mTDelHarmo16~ parameters include *DelHarmoBlock* arguments: duration (in milliseconds), and *HarmoBlock arguments:* transposition (in midicents). Furthermore, the global parameters must also be set, as input and output gain and especially the feedback amount. This one plays an essential role for the reinjection process.

3.1.2. Reinjection matrix

The *fdToMatrixBlock* block from the process above (Figure 4) is the main core of the reinjection matrix (Figure 7).

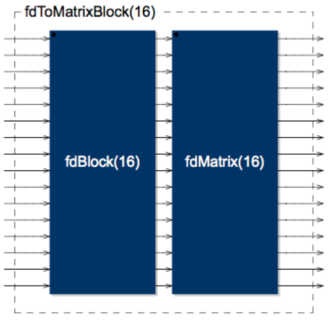


Figure 7: *Faust Scheme of* fdToMatrixBlock *block-diagram.*

With this approach, it is possible to send the output of any line to the inputs of any combination of lines thanks to an internal matrix. Thus, the reinjection is managed by the Max object *matrix~* with the the GUIs object *matrixctrl* (It's also possible to send a message directly to the *mTDelHarmo16~* object with all the matrix coordinates). Therefore, the matrix enables to send any output to any combination of inputs (Figure 8). The internal Faust matrix contains 16 x 16 = 256 toggles (each of them values either 1 or 0).

The Faust code is quite simple, but very effective to accomplish reinjection matrix. A *par* instruction is needed to make the inputs and outputs columns. A special attention is required to feed the correct output index. The user must set all the matrix coordination through the checkbox - e.g. with Max control matrix (Figure 8). Thus, with the feedback value, it is possible to route the correct amount of reinject signal. A special attention is needed to avoid saturation and DC offsets. A good parsimony is required to choose the right quantity of inputs. it is also prudent to put an audio limiter just after the *mTDelHarmo16~* output).

```
toggle(c, in) = check-
box("h:Lines/h:Reinjection_Matrix/v:Del%2c-
->/r%3in");
Mixer(N,out)  = par(in, N, *(toggle(in,
in+out*N)) ) :> _ ;
Matrix(N,M)   = par(in, N, _) <: par(out, M,
Mixer(N, out));
```

3.1.3. *Timbral composition with Faust - aural considerations*

The first seconds of *Fil de soi 1* show the potential of this method. The piece starts with a solitary high note made by the acoustic guitar. Then, thanks to the delay line in association with the transposition, the same sonic material rebounds in the electronic. The whole process generates timbral transformations. Indeed, the first electronics appearance suggests an instrument-like metamorphosis. A percussion-like structure is followed by "small bells-like" sounds. Then, still with a "bell behavior", the guitar note is transposed generating a brief melodic structure.

This example presents a strategy to create complex structures in real-time mixed composition. The association of the *mTDelHarmo16~* object with the reinjection matrix generates a complex phenomenon. Actually, the set of 16 delay lines and their transpositions and reinjections constitutes a "rhythmic-melodic-timbral" cell from the sound of the acoustic guitar. That is to say a timbral composition. Indeed, the reinjection matrix is essential for this process, it causes a time scattering response, but also a timbral transformation, since small amounts of delay can make side effects, such as comb filtering. However, the harmonizer itself contributes significantly in this sense. For example, its window size parameter is essential to generate aural alteration. This is one of the authentic contributions of that piece in the context of creating complex structures.

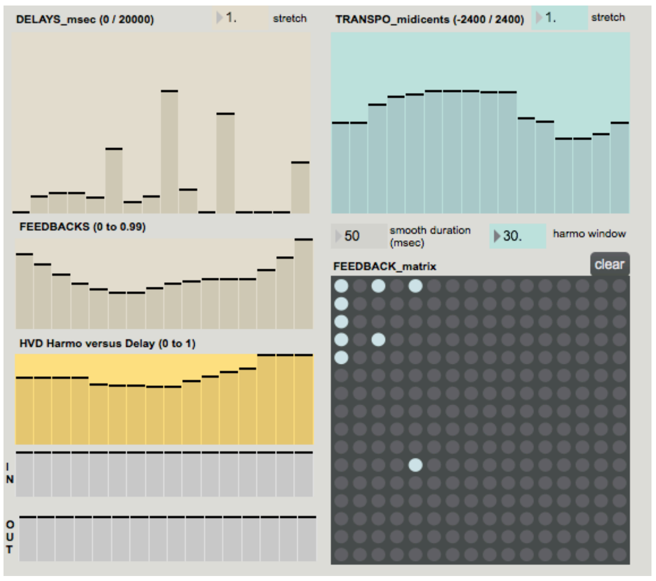


Figure 8: *Example of matrix control with max objects: the first output (horizontal, x-axis) is sent to the five first inputs (vertical, y-axis). The third output is sent to first and fourth inputs. Finally, the fifth is sent to first and to eleventh input.*

3.2. Reinjection matrix to create complex structures – The case of *Les cris du sixième cercle*[4]

Bonardi's *mTDelHarmo16~* object was the starting point for the compositional environment of *Les cris du sixième cercle.* Actually, this piece follows a working flow initiated in our Ph.D. Thesis [3]. However, the motivation using the object was quite different. Instead of making "rhythmic-melodic-timbral" cells, the first attempt goal was to generate a more complex vertical structure whose components are instances of *mTDelHarmo16~* object. To achieve this, the delay line in particular has a different behavior. Instead of making linear lines through well specified durations, as in the first part of *Fil de soi 1,* the delays can be predominantly shortened and can be randomly defined. Thus, each instance output can be assigned to a different instance input through the reinjection matrix with a great feedback amount. The aural result is several layers whose behavior are distinct and temporarily offside.

In order to apply this object in *Les cris du sixième cercle* some changes were needed. First of all, the index value to set the distribution between the harmonizer and the delay was not necessary. Then it was deleted for this piece. All the signals pass through the delay line and the harmonizer without attenuation. Afterwards, the matrix coordination is less dynamic than in *Fil de soi1 1*. Normally, only one message is necessary for the whole pièce. However, all the outputs are reinjected to one different input. One instance never feedbacks to its own input. That is important to avoid side effects, such as comb filter. Then, a pseudo random module in Max is used to generate dynamics durations. Typically, a range value control (with a max value and a min value) is triggered irregularly in time (this time is also controlled by ta range value). Finally, a granular process was developed sharing the same reinjection matrix principle.

3.2.1. *Granular process with reinjection matrix* - jgrain7~

The process below (Figure 9) is quite similar to the *fdToMatrixBlock* (Figure 3). This a main core process where the *grainBlock* is coupled with the rejection matrix. Actually, the whole feedback process is developed with the same code as the *mTDelHarmo16~*.

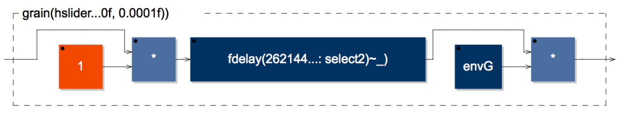


Figure 9: *Faust Scheme of* jgrain7~ *object.*

The *grainG* function is based on a quasi-synchronous granular synthesis. Thanks to its well-adapted real-time implementations, it was able to transport the original Curtis Roads code and applied in Faust[5]. "In quasi-synchronous granular synthesis (QSGS), the grains follow each other at unequal intervals, where a random deviation factor determines the irregularity" [20]. Brief, the granular process is a real-time processing based on delay lines with an amplitude envelope (Gaussian type) whose sound grains follow each other at irregular intervals thanks to a random deviation determined by temporal irregularity. Thus, it is possible to generate various scattered textures.

3.2.2. *Object oriented approach - aural considerations*

For *Les cris du sixième cercle* environment implementation, the scheme below shows this environment. As mentioned above, the motivation using *mTDelHarmo16~* object was quite different. Thus, for that composition, only 7 instances were necessary, instead of 16 in *Fil de soi 1 and 2.* Moreover, a downside was crucial in order to save CPU, considering the addition of the *jgrain7~* object. Therefore, a new object *mTDelHarmo7~* was compiled. This task was easily accomplished thanks to the Faust facilities (especially the online compiler).

The scheme below shows how the environment was developed (Figure 10). Actually, there are three levels of reinjection matrix. First one, inside *mTDelHarmo7~* object itself, Then, inside *jgrain7~*. Finally, feedback of the whole process. That is a reinjection matrix that links the *jgrain7~* outputs with the *mTDelHarmo7~* inputs. This one is not implemented in Faust, instead a Max object *matrix~* takes care of it. A Bypass to connect the *mTDelHarmo7~* directly to the ambisonic matrix was also improved, without passing through *jgrain7~* (this last process was ignored in the schema above[6]).

[4] Concert recording is available in YouTube : https://www.youtube.com/watch?v=f3tdY4RGujM&t=8s[verified URL August 30, 2020].

[5] Actually, Road's code was first developed in the field of ambisonic by Pierre Guillot for the HOA library [19].

[6] The schema also ignores other processes like live direct instrument sound, audio limiting before *mTDelHarmo7~ and jgrain7~*, and all the controls signals.

The entire spatial process is beyond the scope of this paper, but it is important to note that the *Output Matrix in ambisonics* is also a Max *matrix~* object that routes all the 7 signal outputs towards the 7 circular ambisonic harmonics (this process is made at order 3 in 2D ambisonics [21]). The same principle was developed in *Fil de soi 1 and 2*. The *Input matrix*, also a Max *matrix~* object, only routes the acoustic signal to any of the seven instances of *mTDelHarmo7~*. This manipulation is important to control the level of density of the sonic result.

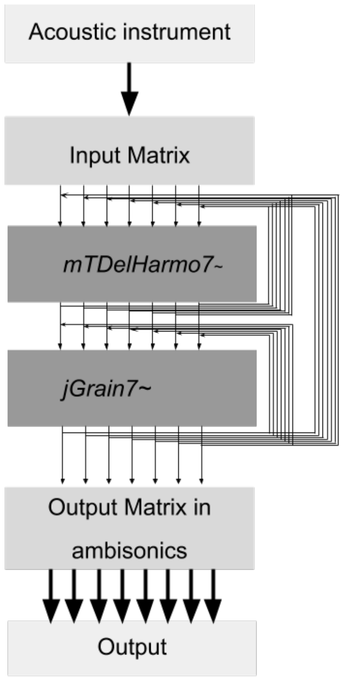


Figure 10: Les cris du sixième siècle *reinjection scheme.*

The aural considerations are particular and quite different from *Fil de soi 1 and 2*. The multitrack session below (Figure 11) shows an example of the environment behavior[7]. Each track corresponds to the output of a level of reinjection from the scheme below (Figure 10). At the top, there is the acoustic direct sound. Then the tracks, the 7 outputs from the *mTDelHarmo7~* (which are also bypassed directly to the *Output Matrix in ambisonics*). Next, the 7 outputs of the *jgrain7~* are routed to outputs *Output Matrix in ambisonics* which correspond to the signals which are sent to the loudspeakers. The micro-temporal decorrelation (thanks to the individual delay values of each instance of the harmonizer [22]). The independent control of each instance is quite visible, especially in the granulator output. The role of the matrix reinjection is obvious in the grainy texture of the harmonizer. That is, a morphology generated by the granulator and reinjected by the overall matrix.

In *Les cris du sixième cercle* the complex structures are more related to Vaggione's idea. Indeed, the example above shows a case of real-time process to generate *micromontages* in several multi-scale layers [23][24]. The canonical object based process fragmentation and agglutination process repeated several times, in order to create multiples, is in our example achieved in a real-time mixed music context.

[7] Actually, the example comes from our piece *Tyque* (2018) for clarinet and live electronics. However, *Les cris du sixième siècle* shares exactly the same environment.

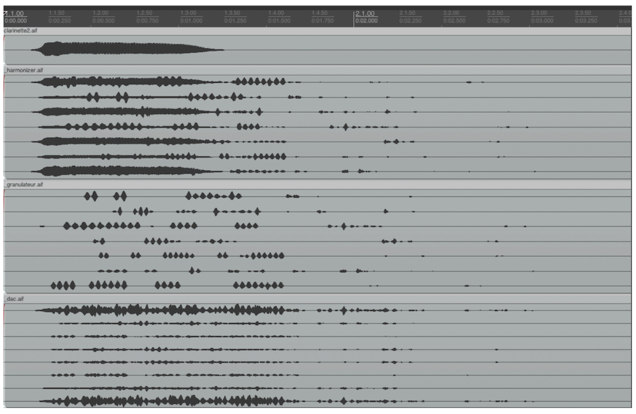
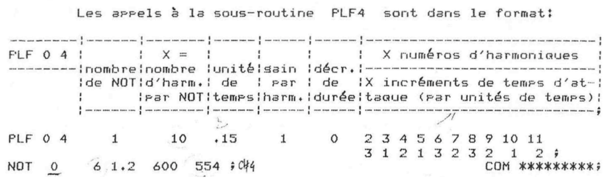


Figure 11: *Micro-scale behavior of the reinjection matrix in* Les cris du sixième cercle.

3.3. Reinjection matrix for creating complex structures - Risset's PLF 4 prototype.

The *Fil de soi 1 and 2* environment shares with Risset's PLF 4 the control of time and transposition of several instances. Thus, it is possible to design the same *Inharmonique* behavior with *mTDelHarmo7~* object. The HH3 sequence in Denis Lorrain's analysis shows an example of Risset's PLF 4 [25] (Figure 12). There are two twin oscillators, each for the two stereo outputs. The first instruction PLF 0 4 calls the subroutine. Then, it specifies 10 instances. Finally, for each instance a index of time increment is defined (those values are multiplied by a time unit, 150 milliseconds). The second instruction NOT plays that subroutine. Inside it, the main argument is the fundamental frequency: 554 Hz.



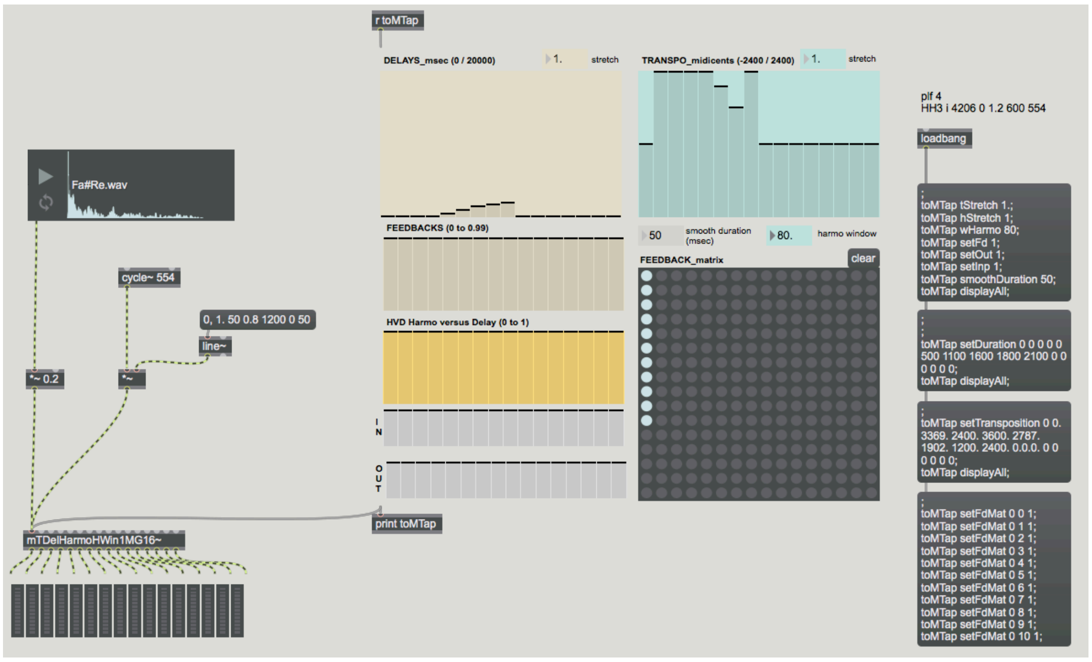
```
             Les appels à la sous-routine  PLF4  sont dans le format:

--------|------|-------|-----|-----|-----|--------------------------|
PLF 0 4 |      |  X =  |     |     |     |  X numéros d'harmoniques |
--------|nombre|nombre |unité|gain |décr.|--------------------------|
        |de NOT|d'harm.| de  | par | de  |X incréments de temps d'at-|
        |      |par NOT|temps|harm.|durée|taque (par unités de temps)|
        |------|-------|-----|-----|-----|--------------------------;

PLF 0 4    1      10    .15    1     0    2 3 4 5 6 7 8 9 10 11
                                          3 1 2 1 3 2 3 2  1  2 ;
NOT  0    6 1.2  600   554 ;C#4                    COM *********;
```

Figure 12: An example of PLF 4 usage in Risset's *Inharmonique*.

The PLF 0 4 behavior was developed in the same *Fil de soi 1 and 2* Max patch (Figure 15). All the harmonic transpositions were calculated in midicents and sent to the Faust object through the *setTransposition* function. Then, all the index of time increment was calculated and sent with *setDuration*. The oscillator with the fundamental is connected in the first inlet: the first instance input. Finally, its output is reinjected to all the 10 instances, each one corresponding to a harmonic. The aural result is quite similar to the original Risset's application.

The goal of this prototype is not to transcode Risset whole piece. Instead, this example helped us to better understand our own use of the environment. For example, the same Risset's PLF 4 can be tested with *Fil de soi 1* acoustic samples. It gives notable aural differences by changing only the harmonizer's windows size. Actually, it is possible to achieve a similar *Fil de soi 1* beginning aural result using the PLF 4 configuration with original parameters. That proves what was already predicted: the importance of this parameter to create sonic complexities. Furthermore, the prototype shows that our composition approach to creating complex structures has the same origin as in *Inharmonique*. However, each composition keeps authentic mark from each research-composer.

.



```
;
toMTap tStretch 1.;
toMTap hStretch 1;
toMTap wHarmo 80;
toMTap setFd 1;
toMTap setOut 1;
toMTap setInp 1;
toMTap smoothDuration 50;
toMTap displayAll;
```

```
;
;
toMTap setDuration 0 0 0 0 0
500 1100 1600 1800 2100 0 0
0 0 0 0;
toMTap displayAll;
```

```
;
toMTap setTransposition 0 0.
3369. 2400. 3600. 2787.
1902. 1200. 2400. 0.0.0. 0 0
0 0 0 0;
toMTap displayAll;
```

```
;
toMTap setFdMat 0 0 1;
toMTap setFdMat 0 1 1;
toMTap setFdMat 0 2 1;
toMTap setFdMat 0 3 1;
toMTap setFdMat 0 4 1;
toMTap setFdMat 0 5 1;
toMTap setFdMat 0 6 1;
toMTap setFdMat 0 7 1;
toMTap setFdMat 0 8 1;
toMTap setFdMat 0 9 1;
toMTap setFdMat 0 10 1;
```

Figure 13: *Fil de soit 1 and 2* Max patch applying Risset's PLF 4.

4. CONCLUSIONS

In this paper we analyzed three pieces *Fil de soi 1 and 2*, *Les cris du sixième cercle* and a prototype of *Inharmonique*[8]. All these examples share the same base of *mTDelHarmo7~* Faust object with the reinjection matrix. Therefore, we demonstrated a creative way to create complex structures in real-time mixed music. However, it is important to note that each composer has his own strategy in music composition. In *Fil de soi 1 and 2*, the composer is looking for a precision, a kind of "determinism". *Les cris du sixième cercle*, on the other hand, gives a special place to the risk, the emergence, or even "chaos". Two apparently distant approaches, but which find their complementarity in the Faust code. Moreover, there are approaches to creating complexities. Or even, quoting Risset, there are examples of composing the sound itself in a real-time mixed music context.

5. REFERENCES

[1] DE SOUSA DIAS, Antonio, FERREIRA, José Luis "Jean-Claude Risset's "Inharmonique" (1977): recast and a real time version proposall", in *EMS 13: Electroacoustic Music in the context of interactive approaches and networks*, Lisboa, 2013.

[2] BONARDI, Alain, "Composition mixte à base de traitements et controles orientés objet : exemple de mise en oeuvre dans *Pianotronics* 3", in *Actes des Journées d'Informatique Musicale 2016*, Mar 2016, Albi, France, p.32-36.

[3] SVIDZINSKI, João, *Modélisation orientée objet-opératoire pour l'analyse et la composition du répertoire musical numérique*, Thèse de doctorat, Université Paris 8, 2018.

[4] MATHEWS, Max et *al.*, *The Technology of Computer Music*, Cambridge, MIT Press, 1969.

[5] RISSET, Jean-Claude, "Composer le son : expériences avec l'ordinateur, 1964-1989", *Contrechamps* n°11, 1990.

[6] RISSET, Jean-Claude, *"An Introductory Catalogue of Computer Synthesized Sounds"*, New Jersey, Murray Hill, 1969, réédité dans le livret de *The Historical CD of Digital Sound Synthesis*, WERGO « Computer Music Currents 13 », 1995, WER 2033-2 282 033-2.

[7] RISSET, Jean-Claude, *Paradoxes de hauteur*, Paris, Rapport IRCAM n°10/78, 1978, accessible via http://articles.ircam.fr/textes/Risset78c/index.html [verified URL February 27, 2020].

[8] RISSET, Jean-Claude, "Looking back to fifty years of digital sound for music", in *Composer le Son : repères d'une exploration du monde sonore numérique*, Écrits, vol 1. Paris, Hermann, 2014.

[9] VAGGIONE, Horacio, "Objets, Représentations, Opérations", in *Ars Sonora* n°2, 1995, p.33–51. http://www.ars-sonora.org/html/numeros/numero02/02e.htm [verified URL February 27, 2020]. (Adaptation française révisée et aug-

[8] Complete concert recording of *Inharmonique* new version is available in YouTube: https://www.youtube.com/watch?v=PcRVusjdO3Y[verified URL August 30, 2020].

mentée de : VAGGIONE, Horacio, "A Note on Object-based Composition", in O. LASKE (éd.), *Composition Theory*, *revue Interface*, vol. 20 n°3-4, 1991, p.209-216).

[10] VAGGIONE, Horacio, BUDON, Osvaldo, "Composer avec des objets, réseaux et échelles temporelles : Une interview avec Horacio Vaggione", in : M. SOLOMOS (éd.) *Espaces composables : essais sur la musique et la pensée musicale d'Horacio Vaggione*, Paris, L'Harmattan, 2007.

[11] ROADS, Curtis, *Composing Electronic Music*, New York, Oxford University Press, 2015.

[12] RISSET, Jean-Claude, ARFIB, Daniel, DE SOUSA DIAS, Antonio, LORRAIN, Denis, POTTIER, Laurent, "De Inharmonique à Resonant Sound Spaces : temps réel et mise en espace", in *actes des Neuvièmes journées d'informatique musicale*, Marseille, 29-31 mai 2002.

[13] DE SOUSA DIAS, Antonio, "Inharmonique (1977) de Jean-Claude Risset. Une proposition de version pour système temps-réel", in *Rencontres internationales du Collegium Musicæ Jean-Claude Risset Interdisciplinarités,* Paris, IRCAM. 2-3 mai 2018.

[14] RISSET, Jean-Claude, "Composing in Real-time", *Contemporary Music Review*, vol.18(3), 1999.

[15] TIFFON, Vincent, *Analyse de Traiettoria de Marco Stroppa*, Anayses, Ircam, http://brahms.ircam.fr/analyses/traiettoria/ [verified URL February 27, 2020].

[16] RISSET, Jean-Claude, "The computer as an interface: interlacing instruments and computer sounds; real-time and delayed synthesis; digital synthesis and processing; composition and performance", in *Jean-Claude Risset Écrits : La munérique, un nouvel artisanat pour la création musicale. Outils et œuvres musicales*, vol 2. Paris, Hermann, 2018.

[17] RISSET, Jean-Claude, "My 1969 Sound Catalogue: looking backk from 1992", in *Jean-Claude Risset Écrits : La munérique, un nouvel artisanat pour la création musicale. Outils et œuvres musicales*, vol 2. Paris, Hermann, 2018.

[18] BONARDI, Alain, "Composer l'espace sonore – Trois approches en recherche-création*",* in *Journées d'Informatique Musicale*, Conférence invitée, 2019.

[19] GUILLOT, Pierre, *Les traitements musicaux en ambisonie*, Mémoire de Master 2, Université Paris 8, 2013.

[20] ROADS, Curtis, *Microsound*, Cambridge, MIT Press, 2001.

[21] COLAFRANCESCO, Julien, *Spatialisation de sources auditives étendues - Applications musicales avec la bibliothèque HOA*, Thèse de doctorat, Université Paris 8, 2015.

[22] VAGGIONE, Horacio, "Décorrélation microtemporelle, morphologies et figurations spatiales", in *Actes des Journées d'Informatique musicale (JIM) 2002*. Marseille: GMEM. Reprinted in A. Sedes (Ed.), Espaces sonores. Actes de recherche, Paris, Editions transatlantiques, 2002.

[23] VAGGIONE, Horacio, "Articulating Micro-Time", *Computer Music Journal*, vol. 20 n°1, 1996.

[24] ROADS, Curtis, "L'art de l'articulation : la musique électroacoustique d'horacio vaggione", in *Espaces composables*. *Essais sur la musique et la pensée musicale d'Horacio Vaggione*, Paris, L'Harmattan, 2007.

[25] LORRAIN, Denis, *Inharmonique : analyse de la bande magnétique de l'oeuvre de J.C. Risset*, Rapport IRCAM n°26, Paris, IRCAM, 1980, disponible via http://articles.ircam.fr/textes/Lorrain80a/ [verified URL February 27, 2020].