RECONSTRUCTING *STRIA*

Kevin Dahan
LISAA (CCAMAN)
Université de Marne-La-Vallée
kevin.dahan@wanadoo.fr

ABSTRACT

We present the process used to conduct a reconstruction of *Stria*, with modern tools and synthesis techniques. From sources to the actual sound "redesign", the methods chosen will be exposed, presenting the difficulties inherent to this particular work. Such a work on reconstructing not only the perceptual nature of an electroacoustic piece also permits to work on the "compositional space", and possibly discovering alternative conception of it. What is questioned here are the ontological properties of a computer music work, its "validity" both in the perceptual and conceptual domain.

1. INTRODUCTION

Stria is an important composition in the history of computer music. Its contribution is important in the field of algorithmic composition, sound synthesis and integration of the notion of space in the compositional domain. All parameters controlling the sound synthesis and spatial dimensions are generated by a set of algorithmic routines and the choice of FM-synthesis permitted a novel approach to the construction of sounds.

The reconstruction work originated after a long period of work on this piece, in which initial transcriptions of the sound synthesis algorithm were tried [3]. However, direct access to the original sources was made possible recently leading to at least another attempt of reconstruction [1].

I will present the different sources that were available in order to conduct an effective reconstruction then discuss the reconstruction process in itself, both from the algorithmic and synthesis standpoint. Finally I will question the very nature of such reconstructive work.

2. SOURCES

Sources available concerning *Stria* genesis are quite numerous [8]. As far as the reconstruction is concerned, there are four main sources, each dealing with a particular aspect of the programming and rendering of the piece.

2.1. SAIL source code

The original code was programmed in SAIL (Stanford Artificial Intelligence Language), a procedural programming language based on Algol. The flow of the program is clear: an entry loop gathering the necessary parameters, a main routine and several subroutines to perform the computation, and a write routine automatically generating the score file used in the subsequent synthesis.

The entry loop is programmed in a "question/answer" manner, with type checking on the input parameters. These high-level parameters are then used to compute the precise numerical values used by the synthesis language.

It should be noted that the main routine can be called recursively, and this particularity has consequences on the result of the computation[1].

2.2. MUS10 source code

The synthesis algorithm for both the main and reverb instrument was implemented in MUSIC10 language, an adaptation of MUSIC IV to the PDP-10 mainframe in use at Stanford University at the time.

The source code for the main FM instrument is complete and intact while there are different versions of the reverb instrument.

2.3. Parameters lists

More importantly, the input parameters used for each section of the work are available. For each of the sections, there is a complete transcript of the "question/answer" session that gives the initial parameters being fed into the main SAIL routines.

All of the sections provide a similar and consequent set of questions, with the notable exception of the last section (called END.MEM). This section was constructed at an initial stage of development of the SAIL code [8] and questioned the possibility of a complete reconstruction [4].

2.4. Score file

The score file for the first section is still available (T0.SCR). The importance of this file should be stressed, as it is the sole point of reference concerning the validity of the output of the main program, i.e. it is the main source to check for discrepancies and errors in the routines. Fortunately enough, the first section includes a recursive call and therefore most of the cases encountered during the reconstruction of the piece are covered.

[1] An analysis and description of the original routines are available in [5].

2.5. Other sources

To complete the list of sources used in the reconstruction, other documents are quite important: the schema of the assembly of the sections, and the corresponding dynamical curve has been provided is very important. Some sections are cut and faded into the following and the overall dynamical construction follows a bell-shaped curve – the non-normalized output produce little dynamical differences between sections.

Figure 1 shows the flowchart of the original code, from the input parameters, down to the section soundfile.

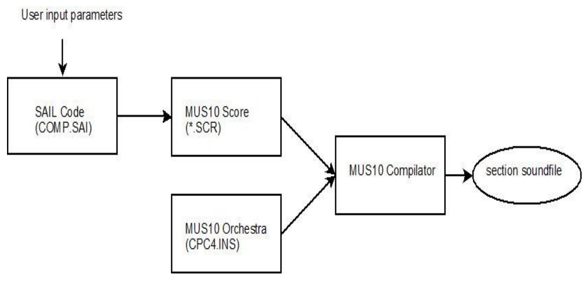

Figure 1: Original Flowchart

3. TRANSCRIPTION

It was clear that new software was needed in order to work on a satisfying reconstruction. Since the SAIL language is very difficult to obtain, the transcription of the SAIL source code was done in Ruby[6]. In the absence of the original implementation documentation, an alternative reference was used in order to have a better understanding of the SAIL language [7]. Due to the inherent dryness of the original question loop, it was decided to provide the user a friendlier interface, a graphical interface (in GTK) was added on top of it, in replacement of the original "question/answers" routine. The choice of the Ruby programming language was purely practical, as it is easily portable and runs without modifications on different platforms[1]. The principal point was to keep close to the original routines, and not to rewrite the code in an object-oriented manner. The flowchart is similar to that of the original one (see figure 2).

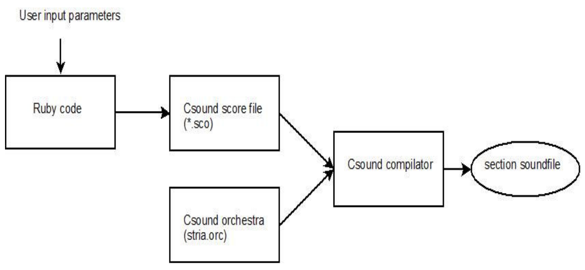

Figure 2: Reconstruction Flowchart

3.1. Graphical interface

The graphical interface provides an easy way to input the numerous (26) parameters needed to perform the computation (see figure 3). It also includes a navigation facility between the different events allowing correction of previous events without leaving the program.

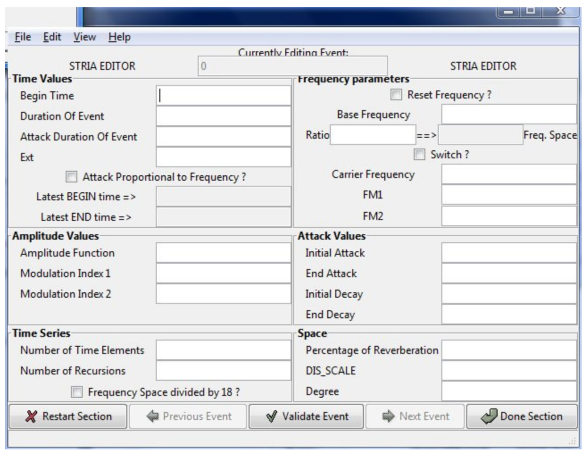

Figure 3: Stria editor

3.2. End section

The difficult part of the transcription was the rewriting and experimentations needed in order to be able to produce the last section. After some work on the parameters, a first modification of the code was found which gave the correct synthesis parameters. The time series however were different from the original and a second modification to the original routine was needed to produce the correct series.

3.3. Impact

The availability of such a tool is not only of use in the context of reconstruction of *Stria*, but may serve also as a basis for exploring more in-depth the complexity of this particular system-piece [3][4], by experimenting

[1] Tests were conducted in linux, windows and OSX environments.

with new parameters, other levels of recursion, and so on. As such, this reconstruction is not only a reconstruction *per se*, but more the reconstruction of system – to use the computing analogy, it provides a class from which to generate new compositional objects.

4. SYNTHESIS

The synthesis algorithm chosen by John Chowning for the main instrument was relatively simple. As for the SAIL source code, it was chosen to stay close to the original methods and techniques used, in order not to complexify even more the reconstruction task.

The synthesis language chosen for the reconstruction is Csound as it has a lot of similarity with the MUS10 language, and for the flexibility it offers in the synthesis domain. There are no portability issues with it either, as implementations are available on most platforms.

4.1. FM synthesis

The reconstruction of the main algorithm was done by strictly following the original version. While there are opcodes in the Csound language allowing a more direct approach to FM synthesis, I chose to use the classical implementation, directly using oscillators.

The instrument used in *Stria* is a basic two-oscillator schema, and is implemented as such in Csound.

4.2. Envelopes

As for envelopes, the original implementation used mostly oscillators and table reading. Again, the reconstruction implementation uses the same algorithm instead of more modern envelope generation procedures. It should be noted that the sources concerning the envelopes were rather imprecise, and limited to small drawings on one of the MUS10 source code. However, after a period of approximations and (listening) trials, correct envelopes were found and implemented.

4.3. Reverb instrument

The part that proved to be more problematic was that of the reverb instrument. The sources available for this part proved not to be accurate, and parameters concerning the delay length were conflicting between versions. Eventually, John Chowning figured out the initial parameters and with numerous trials on the algorithm, a satisfying implementation was done.

The reverb instrument consists of a bank of all-pass and comb filters, feeding 4 variable time delays.

5. CONCLUSION: RECONSTRUCTION OR READAPTATION?

The original version was realized by sections, for practical reason; the synthesis process was very time consuming in 1977 and the quantity of storage needed was problematic. Therefore the original sections used two different sample rates (25600 and 11800 Hz) and the assembly of the whole piece was done using analog technology, as well as the dynamic adjustments.

However, it was clear that these limitations were easy to overcome in a reconstruction; a complete rendering of *Stria* is now computed in less than 3 minutes on an average desktop computer, at a sample rate of 48000Hz. This facility had a direct impact on the reconstruction process, as it permitted to run a number of trials that were impossible, or limited to calculus, in the original work [2].

One of them is the possibility to reunite all sections and compute the piece “as a whole”, without the external help of a mixing software. This has opened the path for an arrangement at the end of the sections which were previously analogically cut and faded.

Moreover the intrinsic potentiality of the software – to replicate using modern technology an electroacoustic masterpiece – it can be used as a demonstrative tool for students in order to show what an electroacoustic composition work can be. By exposing the cohesive nature of the computer implementation, one can therefore demonstrate the path from algorithm to sound in a natural way, as well as opening up a “proven” compositional space for further experiments.

This leads to the question of the true “nature” of a computer music work: while it undoubtly exists in the perceptual domain, it has other “manifestations” in source code, synthesis implementation, and, perhaps more importantly, in the cognitive processes of the composer. The reworking of the sections in order “to find a better overlap” [2] seems to be on the border of the reconstruction and readaptation: while the reconstruction is certainly not different from the original version in its conception and reasoning, these small differences shed a new light on the way composers work and think of their compositions.

6. ACKNOWLEDGEMENTS

I would like to thank John Chowning for his continuous support and patience throughout this project.

7. REFERENCES

[1] Baudoin, O. “A Reconstruction of *Stria*”, to appear in *Computer Music Journal*, 31:3, MIT Press, Cambridge, 2007.

[2] Chowning, J. *Personal Communications*, 2005-2007.

[3] Dahan, K. “Enjeux Esthétiques de la Synthèse Sonore, 1969-2000”, M.A. Thesis, Université de Bourgogne, Dijon, 2000.

[4] Dahan, K. “Surface Tensions: Dynamics of *Stria*”, to appear in *Computer Music Journal*, 31:3, MIT Press, Cambridge, 2007.

[5] Menighini, M. "*Stria*: an analysis of the compositional process", to appear in *Computer Music Journal*, 31:3, MIT Press, Cambridge 2007.

[6] Thomas, D., *Programming Ruby*, 2nd ed., Pragmatic Bookshelf, 2004.

[7] Xydak Corp. *MAINSAIL Language manual*, available at http://www.xidak.com/mainsail/

[8] Zattra, L. "The assembling of *Stria*. A Philological investigation", to appear in *Computer Music Journal*, 31:3, MIT Press, Cambridge, 2007.