Algorithmic Composition with Open Music and Csound: two examples

Fabio De Sanctis De Benedictis

ISSM "P. Mascagni" – Leghorn (Italy)
fabio.desanctis@consli.it

Abstract. In this paper, after a concise and not exhaustive review about GUI software related to Csound, and brief notes about Algorithmic Composition, two examples of Open Music patches will be illustrated, taken from the pre-compositive work in some author's compositions. These patches are utilized for sound generation and spatialization using Csound as synthesis engine. Very specific and thorough Csound programming examples will be not discussed here, even if automatically generated `.csd` file examples will be showed, nor will it be possible to explain in detail Open Music patches; however we retain that what will be described can stimulate the reader towards further deepening.

Keywords: Csound, Algorithmic Composition, Open Music, Electronic Music, Sound Spatialization, Music Composition

1 Introduction

As well-known, "Csound is a sound and music computing system"([1], p. 35), founded on a text file containing synthesis instructions and the score, that the software reads and transforms into sound. By its very nature Csound can be supported by other software that facilitates the writing of code, or that, provided by a graphic interface, can use Csound itself as synthesis engine.

Dave Phillips has mentioned several in his book ([2]): *hYdraJ* by Malte Steiner; *Cecilia* by Jean Piche and Alexander Burton; *Silence* by Michael Gogins; the family of HPK software, by Didiel Debril and Jean-Pierre Lemoine.[1] The book by Bianchini and Cipriani encloses Windows software for interfacing Csound, and various utilities ([6]). We can also recall score generation software like *Cmask*, *Cscore*, *Pmask*, and other ones like *AthenaCL*, *Ceres3*, *Rosegarden*, *Rain*, only to quote some, that can export their outputs in the form of a `.sco` file.[2]

[1] *Cecilia* software described by Dave Phillips is not the actual *Cecilia 5* (`http://ajaxsoundstudio.com/software/cecilia/`) developed by Olivier Bélanger. About *Cecilia 5* we send back to [3]. HPK Composer software family articulates in: *HPKC22*, *HPKComposer 3*, *HPKComposer AV*, *HPKComposerCsound* (see [4]) and *AVSynthesis* (see [5]).

[2] Some softwares can only be used in a Linux environment, other ones are no more maintained. Anyway this *excursus* accounts the widespread custom of pairing Csound to graphic interface software.

Today on Csound site we find links to *CsoundQt*, *Blue*, *Cabbage*, *WinXound*.

Algorithmic Composition and Csound

Algorithmic Composition, as discussed in other place,[3] can be differentiated into constructive and declarative approach. Constructive when the software is used to develop compositional material, declarative when instructions for a complete musical composition are furnished to the software. These definitions fit to instrumental composition, while in electronic music, particularly according to David Cope's CGS category, *Computer Generated Sound*, the boundaries are less defined, so at the same time we can fall into both constructive and declarative approach.

Also Csound owns Algorithmic Composition possibilities and can be used both in a constructive and declarative manner. Giorgio Zucco offers some interesting examples in his book ([10]).

The code underlying the performance of *Solo* by Stockhausen, developed by Francioni ([11]) can be a good example of declarative approach. However Algorithmic Composition is not the main aim of Csound functions, so the alliance with a software such as Open Music appears particularly suitable for this purpose.[4] In following section two examples drawn from my compositions will be treated.

Open Music and Csound

Csound has been used by the writer both directly programming the code, and recurring to graphic interfaces like *HPKComposerCsound* for using simple Csound instruments, but generating complex scores.

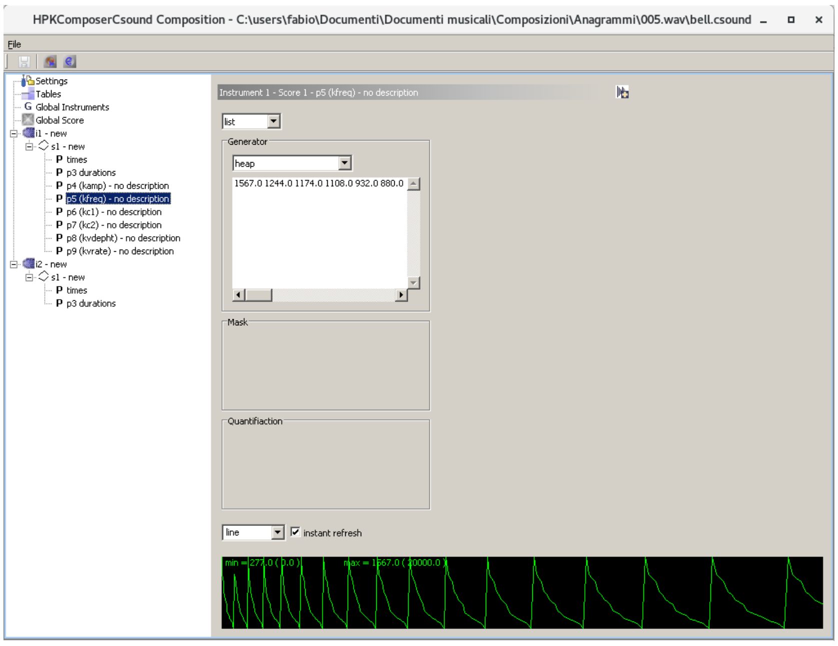
In Figure 1 an example relative to the generation of clouds of bell-like sounds in FM, opcode `fmbell`, used in *Anagrammi*, for Flute and live electronics. Thus the passage to use Open Music as interfacing tool to Csound has been a natural evolutive process. Open Music has been utilized in *Balaenoptera*, for Bass Clarinet and live electronics, and in *Fabula*, for Baritone Sax and live electronics, both quadraphonic works, for realizing electronic materials and spatialization. Below some patches related to Csound will be described.

In *Balaenoptera* Open Music has been mainly used for developing audio material and for its spatialization, by using several libraries, both Ircam and free.[5] Because it is frequent in my compositions to allude to the ambiguity between natural and artificial, Csound has been called into question by Marco Stroppa's

[3] See [7], [8] and [9]

[4] Also PWGL offers interesting examples of use of Csound, thanks to Zucco's PWCsound library, but to respect the limits of this paper it is not possible to show any example in this place.

[5] A more complete description of the use of Open Music for developing audio material in this work is in [14]


HPKComposerCsound Composition - C:\users\fabio\Documenti\Documenti musicali\Composizioni\Anagrammi\005.wav\bell.csound
File
Settings
Tables
Global Instruments
Global Score
i1 - new
s1 - new
P times
P p3 durations
P p4 (kamp) - no description
P p5 (kfreq) - no description
P p6 (kc1) - no description
P p7 (kc2) - no description
P p8 (kvdepht) - no description
P p9 (kvrate) - no description
i2 - new
s1 - new
P times
P p3 durations
Instrument 1 - Score 1 - p5 (kfreq) - no description
Generator
Mask
Quantifiaction
instant refresh

list

heap

1567.0 1244.0 1174.0 1108.0 932.0 880.0

line ✓ instant refresh

Fig. 1. *HPKComposerCsound*: selections of frequencies in an instrument based on `fmbell` opcode.

OMChroma library, creating two instruments *ad hoc* founded on `STKClarinet` opcode.[6] The used Open Music patch can be seen in Figure 2. Here a chord sequence made in Audiosculpt, created by audio analysis of a Bass Clarinet multiphonic, is used as score for Csound and rendered by both `STKClarinet` instruments.[7] Instruments parameters are chosen in random way, pitch by pitch, by `om-random` functions inside selected boundaries (1-5, 100-50, 1-12, 1-12 e 50-128).

Below the beginning of the score of the second instrument:

```
<CsoundSynthesizer>
<CsOptions>
-W -odac
</CsOptions>
<CsInstruments>
; HEADER
sr = 44100
kr = 44100
```

[6] About using Open Music and Csound for audio generation and transformation, see: [15], [16], [17], [18] and [19].

[7] About realizing personal instruments in OMChroma we refer to the very good online documentation, described in [20]

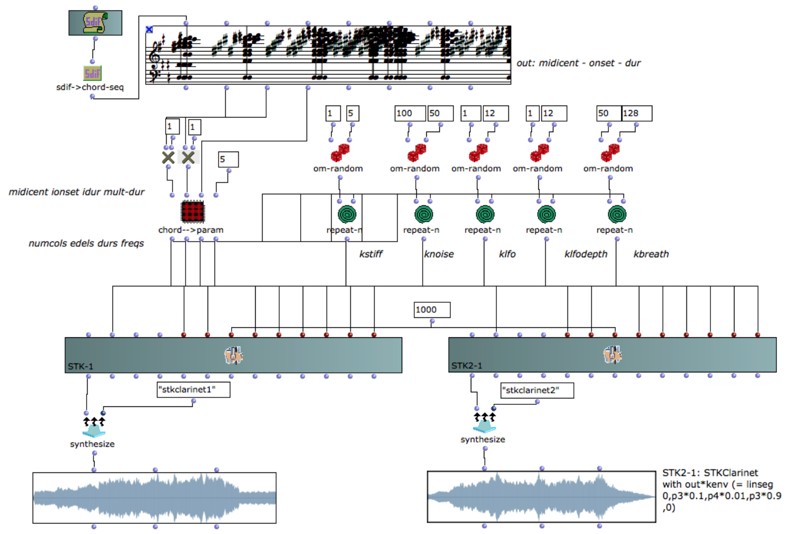


Fig. 2. Opcode STKClarinet as OMChroma class.

```
ksmps = 1
nchnls = 1
;INSTRUMENTS
;==============
instr 1
;==============
;Instrument 1 from file stk2
;asig STKClarinet ifreq, amp, kstiff, kv1, knoise, kv2,
;klfo, kv3, klfodepth, kv4, kbreath, kv5
kenv linseg 0,p3*0.1,p4*0.01,p3*0.9,0
asig STKClarinet p5,p4,2,p6,4,p7,11,p8,1,p9,128,p10
out asig*kenv
endin
</CsInstruments>
<CsScore>
;This synthesis process called my_synt started on 3 26, 2019 - AT 16:10 (24 sec)
; Global Variables: sr = 44100, kr = 44100, ksmps = 1, nchnls = 1
; Defined by chroma classes:
; Loaded tables:
; Generated tables:

;------ Lines for event n. 1 --------
i1 0.000 0.830 1000.000 148.710 5.000 66.000 9.000 1.000 78.000
i1 0.000 0.830 1000.000 299.143 1.000 66.000  12.000 10.000 84.000
i1 0.000 0.830 1000.000 745.136 4.000 100.000 4.000  6.000  91.000
```

```
i1 0.000 0.830 1000.000 886.121 3.000 81.000  8.000  9.000  62.000
i1 0.000 0.830 1000.000 148.710 1.000 58.000  4.000  5.000  77.000
i1 0.000 0.830 1000.000 298.798 2.000 72.000  12.000 11.000 90.000
i1 0.000 0.830 1000.000 745.136 2.000 100.000 7.000  6.000  60.000
i1 0.000 0.830 1000.000 885.098 5.000 73.000  5.000  6.000 119.000
i1 0.166 8.165 1000.000 149.226 3.000 53.000  7.000  5.000 116.000
i1 0.166 8.165 1000.000 297.764 5.000 71.000  3.000  9.000  62.000
```

In *Fabula* we have made a progress towards a more diffuse use of automatic sound spatialization, mainly relying on OMPrisma library by Marlon Schumacher.[8] In Figure 3 an example of a patch, effective with long sounds, that makes a very fast spatialization, acelerating or decelerating, so disintegrating or aggregating the sound.

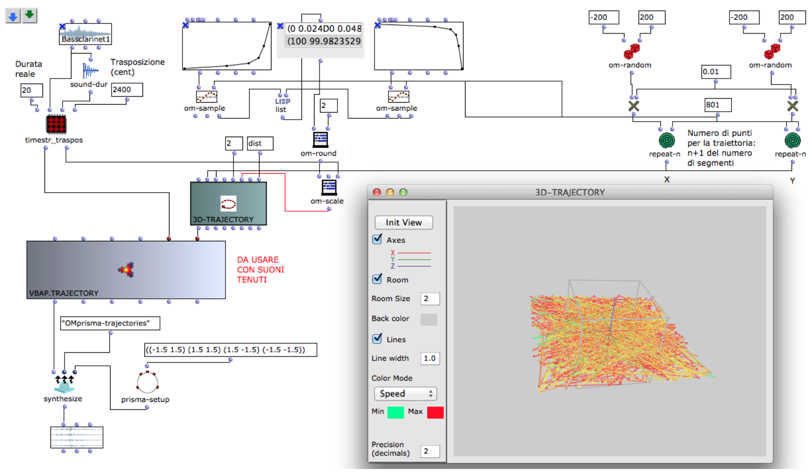


Fig. 3. Quadraphonic spatialization of an audio file acelerating or decelerating, at great velocity, causing sound disintegration or aggregation. Inside 3D-TRAJECTORY object chaotic trajectories and their velocities are visualized.

The curves on the top must be selected for choosing an accelerating or decelerating process, while X-Y coordinates of each trajectory point are random determined by `om-random` functions, in the top on the right of the patch.

Conclusion

We have seen how it is possible to use Open Music together with Csound, mainly using OMChroma and OMPrisma libraries. However that is not the unique way

[8] About using OMPrisma for spatialization and spatialized sound synthesis see: [21], [22] and [23].

to couple Open Music and Csound. Only to quote some: the possibility to export Open Music results in different formats like MIDI and XML permits to utilize very refined musical structures inside Csound; it could be possible to generate score data by algorithmic processes; the possibility of generating personal Csound instruments in OMChroma opens virtual infinite possibilities, not excluding the conversion of historical Csound instruments in Open Music classes, to be used inside the software. This is only the tip of the iceberg.

References

1. Lazzarini, V.: Computer Music Instruments. Foundations, Design and Development. Springer 2017, p. 35.
2. Phillips, D.: Linux musica e suoni. Hops Libri, Milano 2001, pp. 226-304 (or. ed. Linux Music & Sound. No Starch Press, San Francisco 2000).
3. Blanger O.: Cecilia 5, la bote outils du traitement audio-numerique. In: JIM2014 - Journes d'Informatique Musicales, 21-23 mai 2014, Bourges, France.
4. Lemoine J.P.: HPKComposer A Csound 5.0 based Audio Video Composition Tool. Csound Journal, Issue 5, January 1, 2007. Internet: `http://csoundjournal.com/issue5/HPKcomposer.html`.
5. Phillips D.: Composing With Csound In AVSynthesis. Csound Journal, Issue 10, January 19, 2009. Internet: `http://csoundjournal.com/issue10/avs-cs-composition.html`.
6. Bianchini R. and Cipriani A.: Il Suono Virtuale. Sintesi ed elaborazione del suono – Teoria e Pratica con Csound. ConTempo, Roma 2001.
7. De Sanctis De Benedictis, F.: Dall'analisi musicale alla composizione e formalizzazione algoritimica: esempi applicativi con PWGL. In: MUSICHE LIQUIDE, XX Colloquio di Informatica Musicale. Internet: `http://cim.lim.di.unimi.it/2014_CIM_XX_Atti.pdf`
8. Agon C., Assayag G and Bresson J.: The OM Composers Book. Volume One. Editions Delatour France/Ircam, Parigi 2006.
9. Bresson J., Agon C. and Assayag G.: The OM Composers Book. Volume Two. Editions Delatour France/Ircam, Parigi 2008.
10. Zucco, G.: Sintesi digitale del suono. Laboratorio pratico di Csound, Giancarlo Zedde editore, Torino 2012 (English edition: Inside Csound, Giancarlo Zedde editore, Torino 2014).
11. Francioni E.: SOLO_MV_10.1. Solo Multiversion for Stockhausen's Solo [N.19]. Csound Journal, Issue 13, January 19, 2010. Internet: `http://www.csounds.com/journal/issue13/solo_mv_10_1.html`
12. Avantaggiato, M.: Composizione assistita e processi di trasferimento di dati musicali da PWGL a Csound. In: XVIII CIM - Colloquio di Informatica Musicale, Torino-Cuneo, 58 Ottobre 2010. Internet: `http://cim.lim.di.unimi.it/2010_CIM_XVIII_Atti.pdf`.
13. Lanza M., Verlingieri G. and Biagioni N.: LA LIBRERIA OPENMUSIC om4Csound. INTRODUZIONE E PROGETTO DI DOCUMENTAZIONE. In: XVIII CIM - Colloquio di Informatica Musicale, Torino – Cuneo, 58 Ottobre 2010. Internet: `http://cim.lim.di.unimi.it/2010_CIM_XVIII_Atti.pdf`.
14. De Sanctis De Benedictis F.: Electronic sound creation in Balnoptera for bass clarinet, electronic sounds, and live electronics. In: Bresson J., Agon C. and Assayag G. (eds.) THE OM COMPOSERS BOOK. Volume 3, Editions DELATOUR FRANCE/Ircam-Centre Pompidou, Parigi 2016.

15. Bresson J., Stroppa M. and Agon C.: Symbolic Control of Sound Synthesis in Computer-Assisted Composition. In: International Computer Music Conference, 2005, Barcelona, Spain.
16. Bresson J. and Agon C.: Temporal Control over Sound Synthesis Processes. In: Sound and MusicComputing (SMC06), 2006, Marseille, France.
17. Agon C., Bresson J. and Stroppa M.: OMChroma: Compositional Control of Sound Synthesis. Computer Music Journal, 35:2, pp. 6783, Summer 2011.
18. Bresson J. and Nichoin R.: Implémentations et contrle du synthétiseur CHANT dans OpenMusic. In: Actes de Journées d'Informatique Musicale, Saint-Etienne, France, 2011.
19. Stroppa M., Lemouton S. and Agon C.: omChroma: vers une formalisation compositionnelle des processus de synthèse sonore. In: Journées d'Informatique Musicale, 9 e édition, Marseille, 29 - 31 mai 2002.
20. Richelli L.: La Libreria OpenMusic OMChroma - Documentazione online. In: Proceedings of XX CIM, Roma, October 20-22, 2014. Internet: `http://cim.lim.di.unimi.it/2014_CIM_XX_Atti.pdf`.
21. Scumacher M. and Bresson J.: Compositional Control of Periphonic Sound Spatialization: In: 2nd International Symposium on Ambisonics and Spherical Acoustics, IRCAM, Paris, May 6-7, 2010.
22. Bresson J., Agon C. and Schumacher M.: Représentation des données de contrôle pour la spatialisation dans OpenMusic. In: Journées d'Informatique Musicale, 15ème édition, Rennes, 18-20 mai 2010.
23. Schumacher M. and Bresson J.: Spatial Sound Synthesis in Computer-Aided Composition. Organised Sound 15(3): 271-289 Cambridge University Press, 2010.