Synthesis by Parametric Design

Simone Scarazza

Abstract. As a composer I started my research aiming at developing a relationship between the graphic elements and the sound ones. Particularly I focused on the possibility to employ in the sound synthesis, processes and concepts belonging to Parametric Design used in computer graphics. Thanks to this study I built up a kind of library consisting of several models useful for the composition and source of stimulus to master the research toward a graphic approach. In this paper it will shown an example trough Csound.

1 Introduction

In the digital environment the beginning of the Parametric Design dates back to 1963 when Ivan Sutherland, conceiving the Sketchpad the forerunner of Graphical User Interface, introduced new functions in order to create in the patterns, variable and scalable geometries.

Compared to Computer-Aided Design (CAD), the software supporting Parametric Design allows to represent and produce models that, thanks to the synchronism of the parameters, they may grow and can be modified as organisms.

One of the best known software for parametric design is Grasshopper, a visual programming language and environment developed by David Rutten at Robert McNeel & Associates, that runs within the Rhinoceros 3D computer-aided design (CAD) application.

Today the use of graphical digital interfaces and the manipulation of the codex have deeply penetrated the designer expressive language and creative concept. It cooperates in finding solutions in which assisted design doesn't merely consist in supporting its production but it permits to bring new possibilities of useful interacting for the customization of the project and helps to overcome its limits.

2 Sound Synthesis and Graphic Sign

Sound synthesis and graphics have had many points of contact since the beginning of their history. One of the first was "Graphic 1" developed by William Nike in the Bell laboratories in 1961. It is a hardware and software system started for engineering purposes, which was employed by Max Mathews, combined with "Music IV", in order to be able to define sound parameters graphically.

Another important informatics tool to be mentioned is UPIC (Unité Polygogique Informatique CEMAMu) realized by Iannis Xenakis in 1977. UPIC allows to translate drawings, realized with an electromagnetic pen on an electromagnetic board, into music.

Once the drawing has been digitized it is interpreted by computers which combine the graphic form with specific frequency and duration parameters. For example the drawing of an ascending trait corresponds to an ascending glissando.

3 General Description

In order to use Parametric Design as a technique of sound synthesis it has been taken the same model implemented by Xenakis with the UPIC, where in a graphic drawing, definite in a Cartesian coordinate system, the abscissa indicates the time and the ordinate the frequency.

To convert graphic elements, placed in a Cartesian coordinate system into the parameters to generate sound synthesis, the same will be realized in an unit square, (Fig.1) that is a square consisting of the points where both x and y lie in a closed unit from 0 to 1.

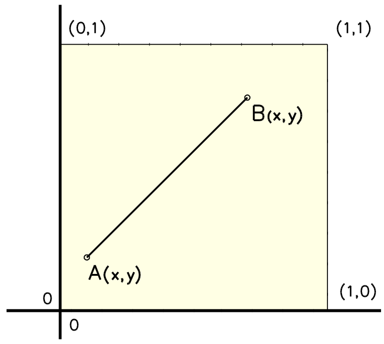

Fig. 1 - Unit Square

In this way, once calculated starting, duration and frequency of the lines, these values can be sized according to the compositional needs of duration, minimum frequency and band-width indicated in the *note-statment* of the score.

Example

```
istart    = Ax
idur      = Bx - Ax
ifreqA    = Ay
ifreqB    = By

------- note statement -----------------------
p1        p2   p3   p4        p5
                    Min.Freq  Band-width
I1        2    10   100       300
```

```
------- sizing --------------------------------
inotestart = p2 + (p3*istart)
inotedur   = p3 * idur
inotefreqA = p4 + (p5*ifreqA)
inotefreqB = p4 + (p5*ifreqB)
```

In Parametric Design one of the most interesting topic to deal with using generative algorithms are geometric patterns. Basically their construction in digital domain results very simple, but the combination between different patterns allows the realization of very interesting shapes.

In the Csound's example that follows, the graphic shape used is composed by the construction lines of Bézier curves.

In this case the shape has been used to create additive synthesis, but in other cases it can be used to control any other parameter of a synthesis algorithm, such as the frequencies of a filter bank, FM parameters, or in granular synthesis, speed, volume, and frequency of grains.

4 Implementation

For the realization of the Csound program we need to start from a graphic model to understand how to automate the generative process. (Fig.2)

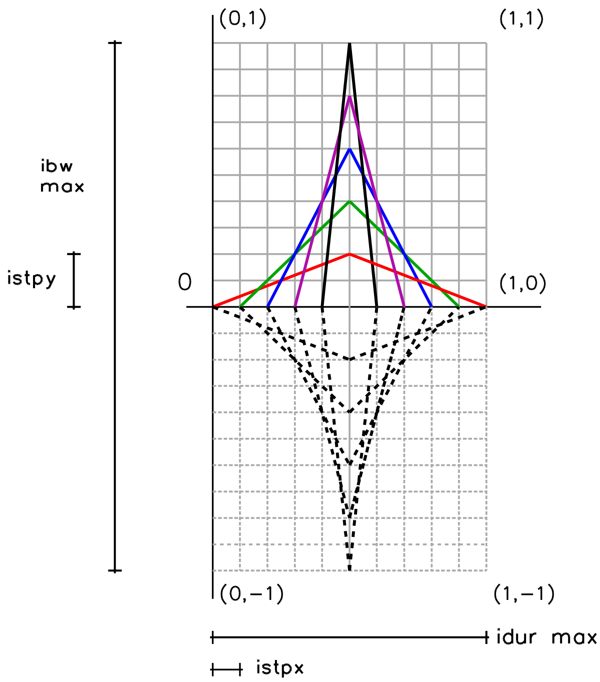

Fig. 2 - graphic model in a Cartesian coordinate system

As shown the drawing is mirrored on the x-axis so, in the first step, it is possible to consider only the positive plane to implement the algorithm. The graph shows a *n* number of poly-lines[1] that have an ascending direction to the middle of the diagram and a descending direction until their end. Each poly-line respect to the previous one has an increase in glissando frequency of a certain step, a delayed start of a certain step and a proportionally decreasing duration.

Thus, it is possible to implement the algorithm creating an iterative structure.

The program is based on two instruments that work together: instrument 1 generates the control parameters for instrument 2 that contains synthesis algorithm.

Instrument 1 will be implemented by a loop (`loop_it` opcode), which will be iterated according to *n* number of required poly-lines. The Loop will iterate *n* times mathematical calculation between the initialization variables and a step value in order to generate a note-list for instrument 2.

In the note-statement of the score file, that calls instrument 1, `P4` will indicates the global amplitude, `P5` the central frequency, `P6` the band-width and `P7` the number of the poly-lines.

Inside the algorithm, `P7` plays a key role: it will indicate the number of called instances of loop, it will divide the global amplitude to share it on each poly-line and it will define the steps values (`istpx,istpy`) that are inversely proportional to `P7.`

Instrument 2 contains the control variables for the glissando, an anti-foldover logic filter, an amplitude envelope, two sinusoidal oscillators and a DC filter.

The control variables for glissando are two: one for the positive plane, that *sums* the central frequency with the band-width and one for the negative plane, that *subtracts* the central frequency with band-width. In this way the drawing will be mirrored.

If the note-list, generated by instrument 1 has negative frequency values, the anti-foldover logic filter avoids this problem bringing them to 0Hz. Consequently to eliminate the 0Hz, at the end of the audio process, another filter has been implemented with the `dcblock2` opcode.

Example of Program Code

```
<CsoundSynthesizer>
<CsInstruments>

sr        = 44100
ksmps     = 32
nchnls    = 1
0dbfs     = 1

instr     1 ;-------------------------------------------

istart    = 0
idur      = 1          ; duration
ibw       = 0
```

1 poly-line in computer graphics is a continuous line composed of one or more line segment

```
inop      = p7          ; number of poly-line
istpx     =.5/ inop     ;(1/p7)*0.5
istpy     = 1/ inop     ; 1/p7
;--- Event note -------------------------------------
ieamp     = p4 / p7     ; Amp / number of  poly-line
iefreq    = p5
iestart   = 0
iedur     = p3
iebw      = 0           ; band-width

event_i   "i",2,iestart,iedur,ieamp,iefreq,iebw

;--------- loop parameters ---------------------------
indx      =0  ;
incr      =1  ;
inumber   =inop  ; number of iterations
;----------------------------------------------------
sequence:   ;loop start process

ibw       = ibw+istpy        ; band-width
iebw      = ibw*(p6*.5)      ; sizing

event_i   "i",2,iestart,iedur,ieamp,iefreq,iebw

istart    = istart+istpx     ; starting point
iestart   = istart*p3        ; sizing
idur      = idur-(istpx*2)   ; duration
iedur     = idur*p3          ; sizing

loop_lt indx,incr,inumber,sequence      ;loop
endin

instr 2  ;--------------------------------------------

idur      = p3
kglis     linseg p5, idur*.5, p5+p6, idur*.5, p5
kglisneg  linseg p5, idur*.5, p5-p6, idur*.5, p5

;------- Anti-foldover filter  -----------------------
if (kglisneg <= 0) then
    kglisneg = 0
endif
;----------------------------------------------------
kenv      linseg 0,idur*.5,p4,idur*.5,0

a1        poscil kenv, kglis, 1
a1n       poscil kenv, kglisneg, 1
aout sum a1,a1n
```

```
adcb dcblock2 aout ;------ DC filter -----------------
out       adcb
endin
</CsInstruments>
<CsScore>
f1        0  4096  10   1

; P1     P2   P3   P4    P5     P6     p7
i 1      3    30   0.5   2000   3000   10

</CsScore>
</CsoundSynthesizer>
```

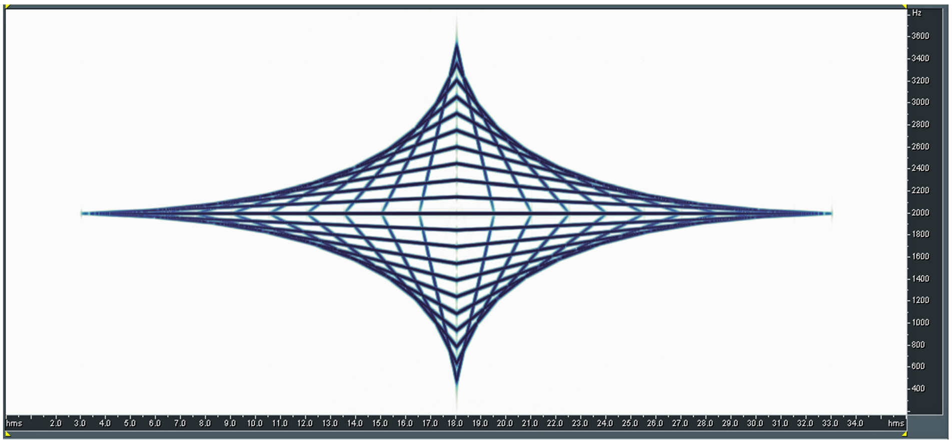

Fig. 3 - Spectrogram of the sound file written by the program

5 Conclusion

In this example an essential program has been realized to clarify as much as possible its working logic but it is possible to obtain substantial transformations adding p-fields, that will intervene on loop parameters; or applying in instrument 2 whatever synthesis technique on the oscillators used for the sound generation.

Working with this kind of program it's possible to create morphologies which possess their own consistency because parametric software permits their adaptive management and to control, through tiny parameters, a great amount of data.

The use of Parametric Design to generate control parameters, provides an interesting alternative to the most used technique by generation of data trough random distributions.

My next step will consist in developing a collection of User Defined Opcodes in order to be able to easily recall each algorithm and interface it with the others.

References

1. Gérard, Ma., et al.: The UPIC System: Origins and Innovations. Prospectives of New Music Vol.31 No.1, 258-269 (1993)
2. Jabi, W.: Parametric Design for Architecture. Laurence King Pub (2013)
3. Woodbury, R.:Elements of Parametric Design. Routledge (2010)
4. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)