STRIA_CS 2.5

Multilevel Interactive Sound Synthesizer

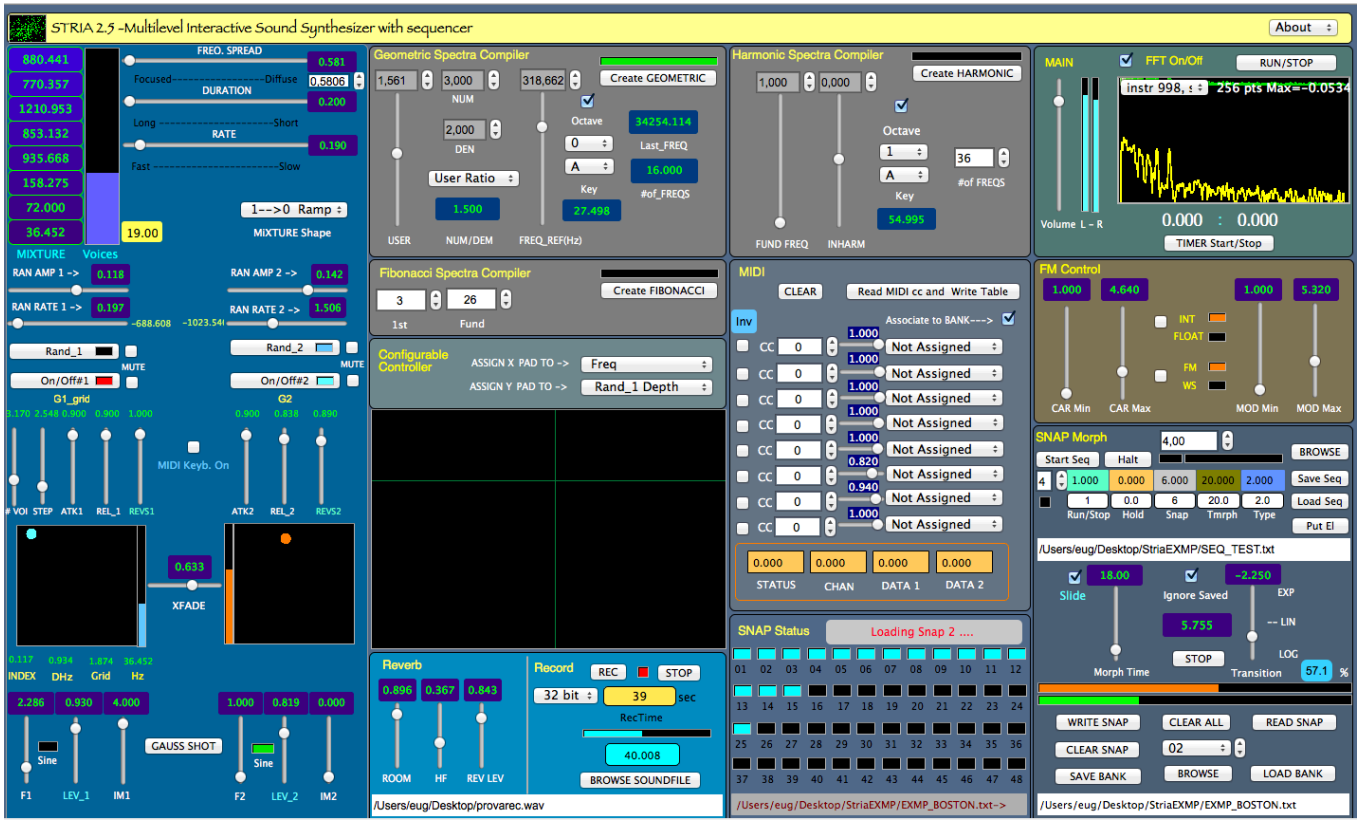
A Csound5 orchestra written by Eugenio Giordani using CsoundQT Front End

USER'S MANUAL

Introduction and acknowledgments

Stria is the title of a very famous composition by John Chowning and I would like to say that this computer music piece represents the manifest of Frequency Modulation technique applied to audio synthesis, invented by Mr. Chowning himself. The Stria Csound orchestra is a very, very little tribute to that great person and also to Max Mathews who was the inventor and father of the well known *Music-n* sound synthesis language generation. Among the different implementations of this type of language, Csound (developed for the first time by Barry Vercoe at MIT Media Lab), is certainly one that has grown enormously in recent years. In addition, this work was realized in three different version of Csound language: the first one was realized using CsoundAV by Gabriel Maldonando and FLTK graphic library, subsequently ported on standard Csound5 and finally converted into the easy to use GUI of CsoundQt by Andrés Cabrera, a simple but interesting cross-platform application for Csound, featuring a highlighting editor with autocomplete, interactive widgets and integrated help. CsoundQt is the natural replacement for MacCsound (Matt Ingalls) , and aims to be an efficient and powerful complete development environment for Csound.

This Csound orchestra was conceived for and started as an exercize for the students of electronic music at LEMS (Laboratorio Elettronico per la Musica Sperimentale – Conservatory of Music G. Rossini – Pesaro - Italy) during the last few years. The main goal of that exercize was to deal with multiple instrument call from inside another instrument that has been possible after the introduction of opcodes such as "*event*", "*schedwhen*", "*schedkwhen*" and so on. At the same time, Stria was musically inspired by some very interesting ideas about the use of tuning systems. I am deeply grateful to Walter Branchi (my early electronic music teacher), whose great knowledge of the tuning systems and music has prompted to me the main guidelines of this project. This work is also dedicated to him.

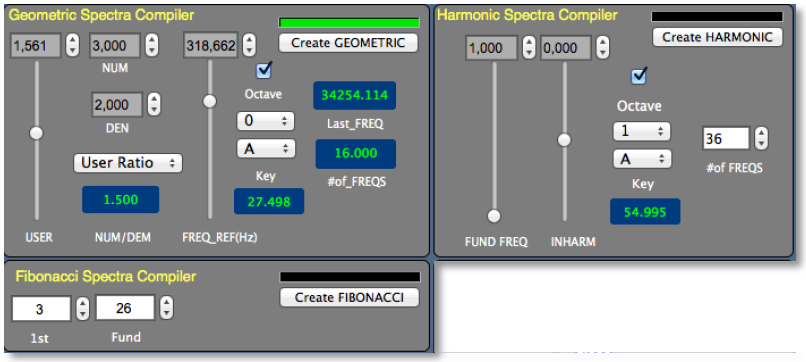


Screenshot of "StriaCS" Control Surface

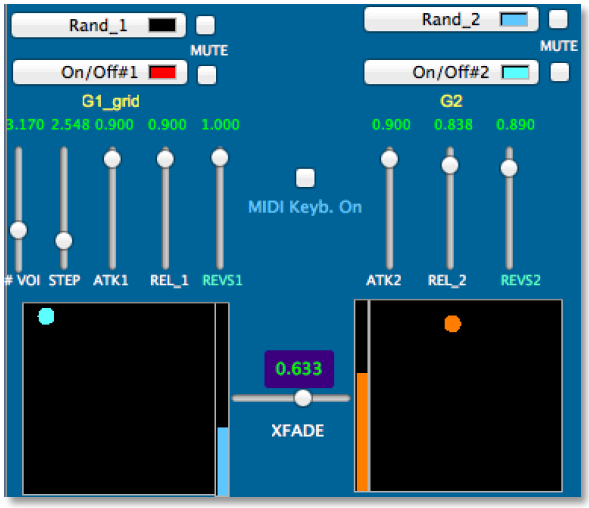
INSTALLING AND RUNNING STRIA

Stria is a Csound orchestra running in CsoundQt (Andrés Cabrera).

- Select your output device
- (No input is necessary)
- Select MIDI device if you plan to control STRIA by external or virtual MIDI device
- Press RUN on CsoundQt Menu or Run/Stop button on GUI
- Stria plays sound only if first of all you have created a pitch system :
  press one of the three buttons : GEOMETRIC / HARMONIC / FIBONACCI
- After having created a pitch system you can press On/Off#1 or #2 and move the relative XY controller. Enjoy Stria.



Pitch system compiler



Start sound generators

OVERVIEW

Stria Csound Orchestra (hereinafter referred to as *StriaCS*) is essentially a sound synthesizer working in real time using CsoundQt, a cross-platform[1] Csound frontend and controlled by an interactive graphic interface and external MIDI at the same time. The generated sound is mainly composed of time-continuous bands of frequencies using concurrently types of sound synthesis and manipulation:

a) Simple FM synthesis
b) Additive synthesis
c) Granular-like synthesis concepts
d) Wavetable synthesis basis
e) Subtractive-like synthesis

All the generated sound come from two separated sound source, we may call :

a) the main multilevel grid-based generator (GM1_Grid)
b) the second multilevel simple generator (GM2)

In addition, the orchestra includes two random generators and a simple reverb.

We can think of *GM1_Grid* as a virtual bank of oscillators and relative amplitude envelopes, generated continuously and overlapping in time (*note blast*). A controlled timer triggers "notes" in a granular style using two main controlling parameters:

- DURATION (the time lasting a single note or grain object)
- RATE (the rate at witch notes or grains are generated)

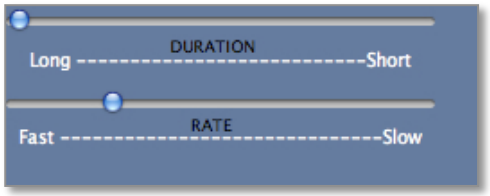


Figure 1

The fastest the rate, the greater is the number of overlapping notes generated by GM1_Grid. Similarly, the amount of overlaps depends also on the note duration. The maximum level of overlaps is reached when the *duration* control slider is on the position Long (turned completely left) and the *rate* control slider is on the position Fast (turned completely left). You can move both the sliders during the performance in order to obtain the desired degree of complexity. The ranges of the two parameters are as follows:

a) DURATION range : [0.2 sec - 3.0 sec]

b) RATE range : [0.1 sec - 0.5 sec]

At the minimum rate of 0.1 sec the GM1_Grid generates 10 notes per second and at the opposite value of 0.5 only two notes per second are produced.

[1] StriaCS runs on a MacBook PRO (OSX 10.8.4 and QuteCsound ver. 0.73 – Csound5 ver.5.19)

The basic frequency of the generated notes is controlled mainly by the position of the horizontal component of a XY joystick on the virtual control surface, as shown in fig.2. The current X controller value (*index*) is displayed on the left bottom side of the joystick (in the fig. 2 shows the number 0.0391). The position of the X in the joystick controls the deterministic component of the base frequency of every generated note. Each note keeps the base frequency value throughout its duration (the actual frequency of each note).

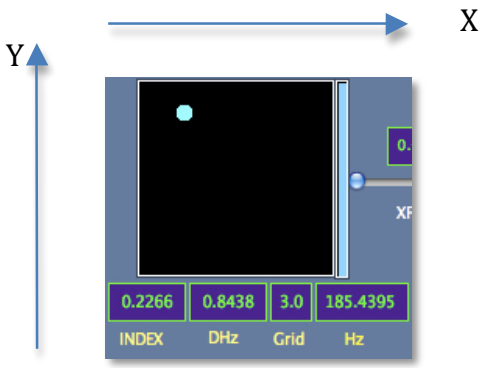


Figure 2

The actual frequency of each note is formed by two components: a *deterministic* part plus a *randomic* part, the first of them, as stated above, is controlled by the X axis of a joistick. The randomic component is controlled by two variables that works together: the coarse range of the random component is controlled by an horizontal slider over the duration and rate sliders as shown in figure 3 while the Y component of the joystick scales that value from minimum value (Y = 0) to the value itself (Y = 1).

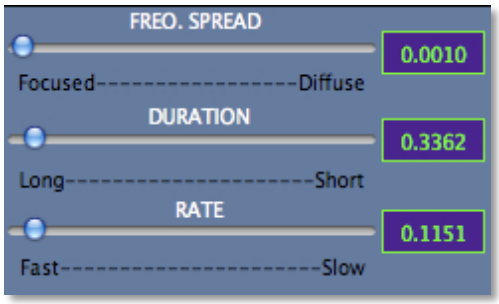


Figure 3

The randomic component of the base frequency is then controlled by:

a) FREQUENCY SPREAD (slider control)
b) DELTAHz (DHz) (Y control on joystick)

When the *frequency spread* is at the minimum value (Focused), all the notes generated have almost the same frequency (not exactly the same). As you move the slider to the right position (Diffuse), each new generated note will have larger frequency displacement in respect to the base deterministic value set by the X control on the joystick. For example, if you start with a simple sine wave at frequency f_0 and gradually move the slider control

from the focused to diffuse position, you will obtain a gradual increase of a frequency band centered on f_0 as sketched in figure 4.

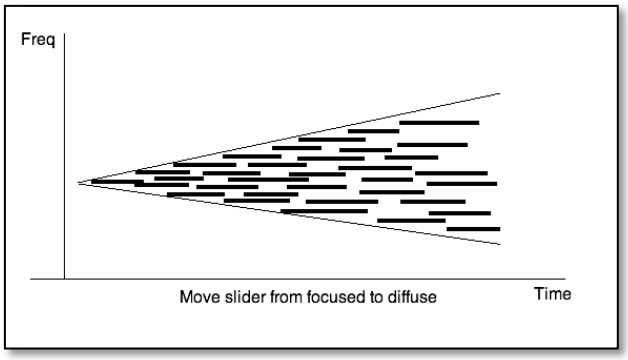


Figure 4

This is the basic concept on which *Stria* works. The multiple generation of overlapping notes and its frequency dispersion produces narrow bands of frequencies (focused pitched sound) that are time uncorrelated so the resulting sound is very rich internally and almost never boring. One can move through different degrees of band thickness, from a very pitched steady and focused sound to a fat chorus-like sound and vice versa. Every single note is amplitude controlled by a smooth symmetrical envelope of which you can control the attack and release time. In order to obtain a smooth sound you have to set those parameters at the maximum level.

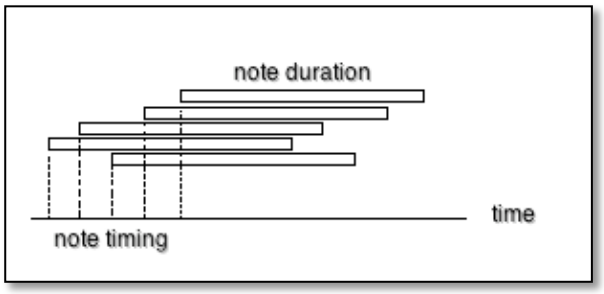


Figure 5

In figure 5 is shown the disposition in time of a sequence of notes of a certain duration triggered in time and overlapping one to each other. The level of overlap depends on the ratio of note duration and the rate of note generation.

In the next pages we will discover many other interesting aspects and possibilities of StriaCSO , starting from the pitch system management to the timbre transformation and external MIDI control.

THE PITCH SYSTEM

The GM1_Grid generator has two other important features that make it particularly interesting and powerful.

The first one is the possibility to generate the sound using two different techniques:

1. Frequency Modulation (FM) Synthesis
2. Additive Synthesis

FM synthesis consists of a simple patch of modulating/carrier oscillator set with a modulation index controlled by a prefixed envelope while in the Additive Synthesis we have the possibility to add up to eight components for each generated note. At the same time, you can choose continuously from a set of waveform obtained with a Fourier summation (GEN 10). In other words, you can make an additive sound using sinusoidal/non-sinusoidal waveform and during the performance you can smoothly move from one to each other.

The second feature that increases complexity and coherence to the generated sound, is that you can create your pitch space by the way of three different criteria of construction. Doing this, you determine a pitches grid and all the frequency changes are derived from that basic selection. You can create a "geometric pitch space" based on the ratio of two numbers that are in turn the numerator and denominator of a specific pitch interval. For example if you specify the two numbers as 3 and 2, you will create a pitch space grid based on the natural interval of a perfect 5th. Because you can choose any relationship, you can virtually have infinite pitch spaces.

Let us explain in detail the necessary steps to build a pitch system (a pitch space grid) starting from figure 6.

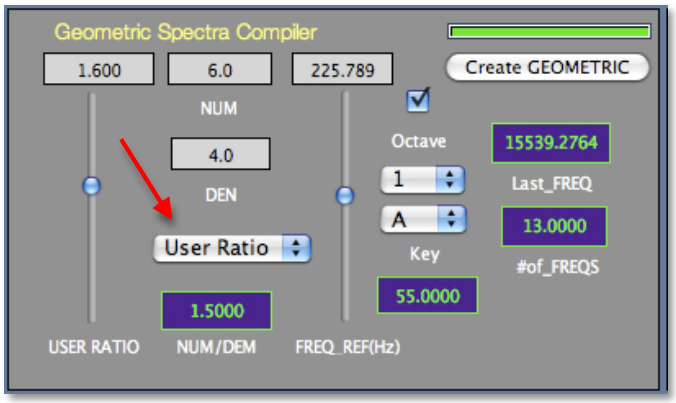


Figure 6

In this figure you can see the **Geometric Spectra Compiler** subpanel. In this subpanel are shown all the controls you can use for the purpose to create a certain pitch grid. Basically you have two main parameters to manage:

a) the interval ratio
b) the base frequency

The *interval ratio* defines a coefficient of multiplication that serves to build up the whole pitch grid while the *base frequency* is the starting point from which the grid takes its origin.

In order to define the interval ratio you can specify it in two different ways:

- to specify directly the ratio in terms of a quotient quantity (i.e. 1.6 also rapresented by the ratio 8/5)
- to specify the ratio in terms of numerator / denominator (i.e. 6.0 / 4.0)

You can choose one of the two ways making a selection on the menu widget (indicated with the red arrow in figure 5).

Once you've chosen the type of interval, you must select the base frequency for the grid. For this purpose you have two different ways to do it:

- specify the directly the value of frequency in Hz using the slider control FREQ_REF(Hz)
- specify the key note and the relative octave using the little two menu widgets KEY

You can choose one of the two ways using the check box widget OCTAVE: if checked the frequency is specified in term of note and octave. If unchecked, the frequency is specified directly in Hz. To complete the operation and generate the pitch grid you have to press once the button widget Generate GEOMETRIC. After this last action the linear led lamp over the button will be green. With the parameter setting shown in fig. 5 you will create a geometric pitch grid based on the interval 1.6 that start at frequency A1 (55 Hz). The grid will consists of a total of 13 frequencies (see display value widget #of_FREQS) starting from 55 Hz up to 15481.1279 Hz as sketched in the next table:

Table 1

#of_FREQS	Hz	operation
13	15539.276	$55*1.6^{12}$
12	9675.702	$55*1.6^{11}$
11	6047.313	$55*1.6^{10}$
10	3779.571	$55*1.6^{9}$
9	2362.232	$55*1.6^{8}$
8	1476.395	$55*1.6^{7}$
7	922.746	$55*1.6^{6}$
6	576.716	$55*1.6^{5}$
5	360.488	$55*1.6^{4}$
4	225.280	$55*1.6^{3}$
3	140.800	$55*1.6^{2}$
2	88.000	$55*1.6^{1}$
1	55.000	$55*1.6^{0}$

You can obtain the same result using the next recursive formula:

Eq. 1

$$f_i = f_{(i-1)} * \text{iterval\_ratio}$$

where f_i is the i-th frequency (i.e. $f_2 = f_1 * 1.6 = 55.00 * 1.6 = 88.00$).

When you press the button *Create Geometric*, the pitch grid is stored into an internal table up to 2048 location. In the exampled shown in the table greatest frequency is 15539.276 Hz and the table will contains only 13 values. For smaller values of the interval ratio, the grid will contain more and more frequency (i.e. for interval ratio = 1.038 will be generated 152 frequencies). The program always will stop the grid generation before the actual Nyquist limit is reached (sr / 2).

The control surface contains other two types of pitch grid compiler so in total you have three different pitch grid compiler:

a) **Geometric** Pitch Grid Compiler
b) **Harmonic** Pitch Grid Compiler
c) **Fibonacci** Pitch Grid Compiler

The Harmonic Pitch Compiler, as for Geometric, is located on the control surface inside the Geometric Spectra Compiler panel, as shown in figure 7.

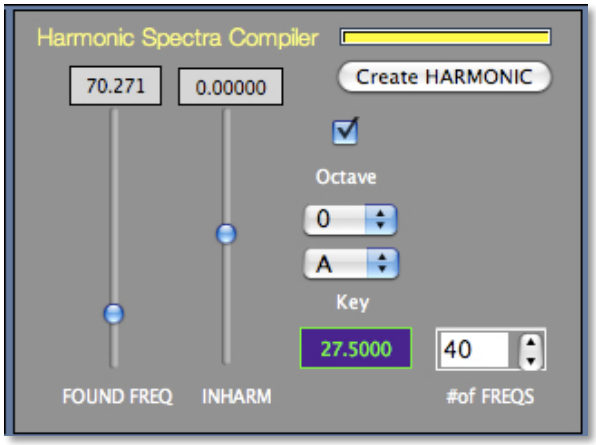


Figure 7

In this panel are shown all the controls you can use for the purpose to create a certain harmonic pitch grid. In a similar way as the Geometric Grid, you have two main parameter to mange plus a third:

a) the base frequency (FUND FREQ or Octave)
b) the inharmonic factor (INHARM)
c) the number of frequencies (#of FREQS)

The Harmonic Grid Compiler creates a grid of #of FREQS frequencies according to the natural series of integer numbers (1,2,3…etc) starting from a fundamental frequency (FUND FREQ) . There are for this purpose, two different ways to do it:

- specify directly the value of frequency in Hz using the slider control FREQ_REF(Hz)
- specify the key note and the relative octave using the little two menu widgets KEY

You can choose one of the two ways using the check box widget OCTAVE: if checked the frequency is specified in term of note and octave. If unchecked, the frequncy is specified directly in Hz.

To complete the operation and generate the pitch grid you have to press once the button widget Generate HARMONIC. After this last action the linear led lamp over the button will be yellow. With the parameter setting shown in fig. 6 you will create an harmonic pitch grid of 40 (see display value widget #of_FREQS) frequencies based on the series of natural numbers start at frequency A0 (27.5 Hz).

Table 2

#of_FREQS	Hz	operation
40	1100.0	27.5*40
39	1072.5	27.5*39
	.	
	.	
	.	
5	137.5	27.5*5
4	110.0	27.5*4
3	82.5	27.5*3
2	55.0	27.5*2
1	27.5	27.5*1

In addition you can create *stretched* or *compressed* harmonic spectra putting the INHARM parameter at a non zero value. Stretched spectrum is obtained for values greater than zero while for compressed spectrum you have to select values less than zero.

The stretched or compressed spectra are created using the following formula:

Eq. 2

$$F_n = F_0 * n^{(1+exp)}$$

where:

Fn = n-th frequency
F0 = fundamental frequency (FUND FREQ)
n = index of Fn frequency
exp = factor of expansion/compression (INHARM)

The inharmonic factor (INHARM) can vary from -0.01 to 0.01. When INHARM = 0 the generated frequency grid is pure harmonic. Using the same parameter for the previous table but with INHARM = 0.00634 the grid will result as follows:

Table 3

#of_FREQS	Hz	operation
40	1126.021	$27.5*40^{(1+0.00634)}$
39	1097.694	$27.5*39^{(1+0.00634)}$
	.	
	.	
	.	
5	138.901	$27.5*5^{(1+0.00634)}$
4	110.971	$27.5*4^{(1+0.00634)}$

3	83.076	$27.5*3^{(1+0.00634)}$
2	55.242	$27.5*2^{(1+0.00634)}$
1	27.5	$27.5*1^{0}$

The third mode for generating pitches grids is based on the famous Fibonacci[2] Series. In the Fibonacci sequence of numbers, each number is the sum of the previous two numbers, starting with 0 and 1. Thus the first nubers of the sequence are:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ……. Etc.

The ratio of two consecutive Fibonacci numbers numbers converges on the Golden Mean Ratio (approximately 1:1.618) as its limit.

The computer music composition Stria by John Chowning was based in almost every structural aspect on this ratio.

StriaCS uses this sequence of numbers for compile the grid pitches.

The Fibonacci Pitch Compiler, as for Geometric and Harmonic, is located on the control surface inside the Fibonacci Spectra Compiler panel, as shown in figure 8.

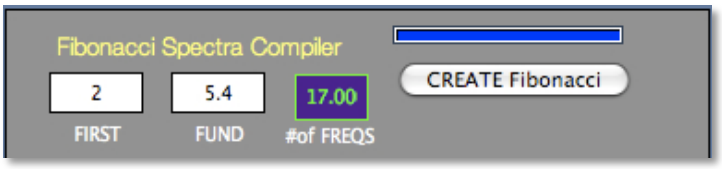


Figure 8

In this panel are shown the few parameter you can use for the purpose to create a certain Fibonacci pitch grid. In doing this you have to specify:

a) the first member of the series (FIRST)
b) the fundamental frequency (FUND)

With the parameter setting shown in fig. 7 you will create a Fibonacci pitch grid of 17 (see display value widget #of_FREQS) frequencies starting from 5.4 Hz as in the following table: (*to be completed*)

#of_FREQS	**Hz**	**operation**

[2] *Leonardo of Pisa (c. 1170 – c. 1250), also known as Leonardo Pisano, Leonardo Fibonacci, or, most commonly, simply Fibonacci, was an Italian mathematician, considered by some "the most talented mathematician of the Middle Ages". (Wikipedia)*

This page is purposely left blank

THE FM / WS AND MIXTURE MODE

As stated in the overview section, the main grid-based generator GM1_Grid is capable of generates blast of notes whose base frequencies derive from the actual pitch grid. At the basic level, the GM1_Grid create simple band of tones centered on the actual value of the X pointer and distributed over a FREQ. SPREAD interval, fine controlled by Y pointer of the relative joystick. Each tone consists of a cloud of notes controlled by a symmetrical and smoothed envelope. The length of each note is controlled by the parameter DURATION (from 0.2 to 3.0 secs) while the rapidity of note scheduling is controlled by the parameter RATE (from 0.2 to 0.5 note/sec) as shown in fig. 1.

Each note consists of an alternatively Frequency Modulation (FM) or simple wavetable (WS) based synthesis tones.

By selecting FM mode, the basic notes are generated by a classic couple of modulating/carrier oscillators controlled by independent amplitude and modulation index envelopes as shown in figure 9.

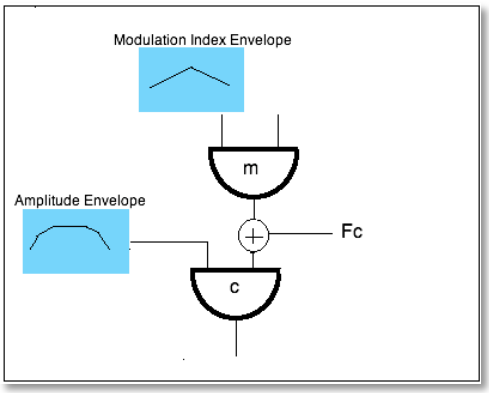
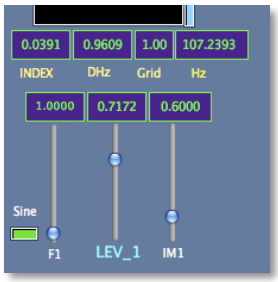
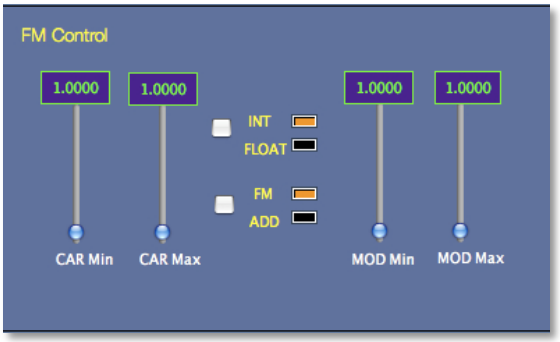


Figure 9

The triangular shape of the modulation index envelope, creates a tone color change starting from pure sine at the very beginning, increasing to maximum complexity at the center, corresponding to the value of maximum modulation index IM1 (cfr. fig. 10 a) and back to sine at the end of the envelope. In the figure 10 b you can see the subpanel of FM c:m ratio and working mode switches.



a)



b)

Figure 10

The modulation index can vary from a minimum value (namely zero) up to a maximum of 4. The reason of this limitation is to prevent or reduce aliasing distortion when using high frequencies on the grid and or the maximum number of voices in the mixture. You can vary the attack and release of the carrier amplitude envelope moving the sliders ATK n and REL n, both within the values 0.05 – 0.9 sec (see fig. 11). During the course of one note generation, the modulation index always vary with triangular shape from zero to the actual selected value as shown in figure 12.

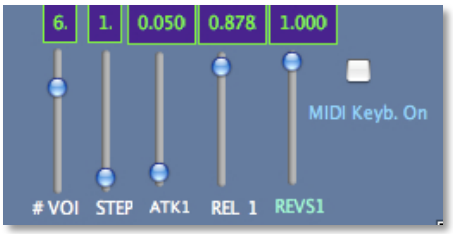


Figure 11

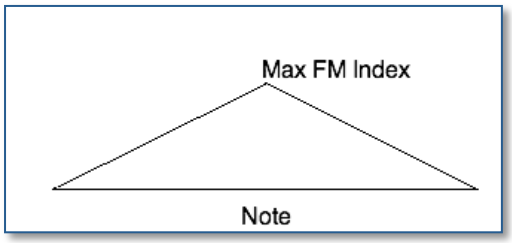


Figure 12

As you know, in the basic FM synthesis it is important to select the ratio of carrier to modulating ratio frequencies (c:m). In figure 10 b, as mentioned above, you can see the controls for set this values. Note that you have not just two controls (sliders) for this purpose but four. In fact in this implementation, the value of c:m is selected randomly inside the ranges you will specify.

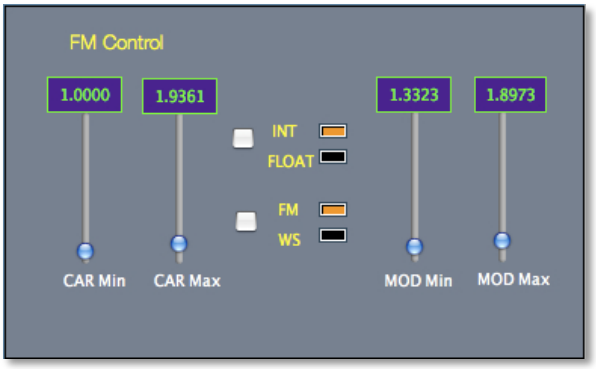


Figure 13

In the case shown in figure 13 the two vertical sliders on the left are selecting the carrier frequency in the range 1.78 to 5.16 while the two vertical sliders on the right are selecting the modulating frequency in the range 1.840 to 3.280. This values are thinked as normalized frequency values so the actual values of frequency is obtained as a multiplication of this values with the frequencies of the pitch grid. Since the orange led INT/FLOAT is on the INT position (checkbox unchecked), the program will select, for each generated note, the integer part of random values for carrier and modulating frequency. So in this case, for the carrier frequency, the possible values are 1,2,3,4 or 5, while for the modulating frequency the possible values are 1, 2 or 3. For this reason the resulting spectrum is always at least harmonically related. On the contrary, when the INT/FLOAT is on the FLOAT position (checkbox checked), the random values are selected as floats numbers inside the same interval. In this case the sound will be more inharmonic. You can switch from FLOAT to INT whenever you want but the change will affect only the new

generated notes so for a short portion of time the two types of spectra will cohabit and overlap.

The other checkbox and the relative led is used to switch between complex (frequency modulation) and simple (wavetable) synthesis. In other words, checking the FM/WS checkbox, each note is generated by simply reading a pre-selected wavetable. Also in this case, only the new generated notes will be affected by the synthesis change.

Just regarding of the WS sound generation, you can select among different wavetable specified in the orchestra. Actually are available eight basic waveform (see the BASIC WAVEFORM section code): the simplest waveform is a sine wave while the others are more complex. The tables are generated using the GEN10 routine (see Csound manual for this topic) and thus is very simple change their shape by changing the relative parameters. The wavetables are associated with the vertical sliders F1/F2 as shown in figure 14.

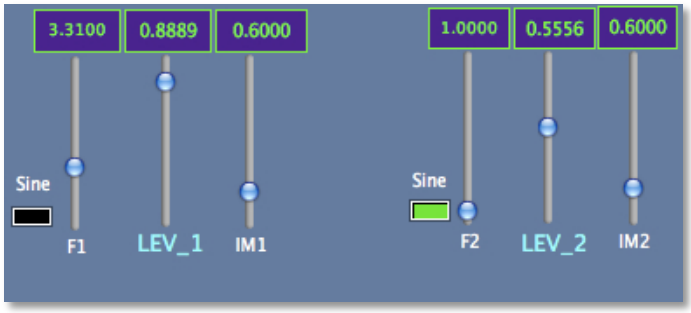


Figure 14

In the same figure 14, the slider F1 is set to the value of 3.31 while F2 is at the init value of 1 and, as one can easily notice, the F1/F2 values are float numbers. This means that the selected wavetable is a *linear combination* between table 3 and table 4. When you move slider F1/F2 the timbre changes smoothly between each table to each other. You are advised to use caution when selecting very complex waveforms (high numbers for F1/F2 control) to avoid possible aliasing effect. When the slider is at the minimum position (value = 1) a green led (Sine) lits. For all the others values, the led is switched off.

MIXTURE Mode

For all the sounds produced in the manner described above, one can be think as “special kind of notes” with a simple (WS) or complex (FM) timbre. In addition, the program create some blurred notes blast that can be focused or not in frequency by superimposing a great number of overlapping notes with small random frequency deviations. If all this is not enough, StriaCS is able to create a “special kind of chords” by the superposition up to 8 notes (each one with the individual timbre and with the focusing attribute): this is what we would claim the *Mixture mode.*

The Mixture mode is automatically activated by setting the parameter #VOI (see the relative vertical slider in figure 11) at values greater than 1. When #VOI = 1 the chord degenerates in a single (even complex) note. The value set with this control determine the number of active voices in the chord so you can move the slider from 1 up to 8. The Mixture mode is allowed only for the main multilevel grid-based generator (GM1_Grid).

The basic note of the chord (a kind of chord root) is coincident with the frequency of the X control of the left most virtual joystick while the other notes of the chord are automatically derived from the actual selected pitch grid. So, for example, if you have created a geometric grid based on the ratio 1.5 (perfect natural fifth) starting from 100 Hz as the lowest frequency of the pitch grid and you have selected 8 voices (the maximum value) for each chord, each chord will be made of 8 notes whose frequencies are shown in the left most part of figure 15, starting from bottom to top. In the rightmost part of the same figure you can see the position of # VOI and the (value = 8) and the position of the X component of the joystick that is at minimum level (value = 0). The four displays just under the joystick show in turn the normalized value of pitch address (INDEX), the delta frequency (DHz), the effective number of the grid pitch element (Grid) and the value in of associated frequency (Hz).

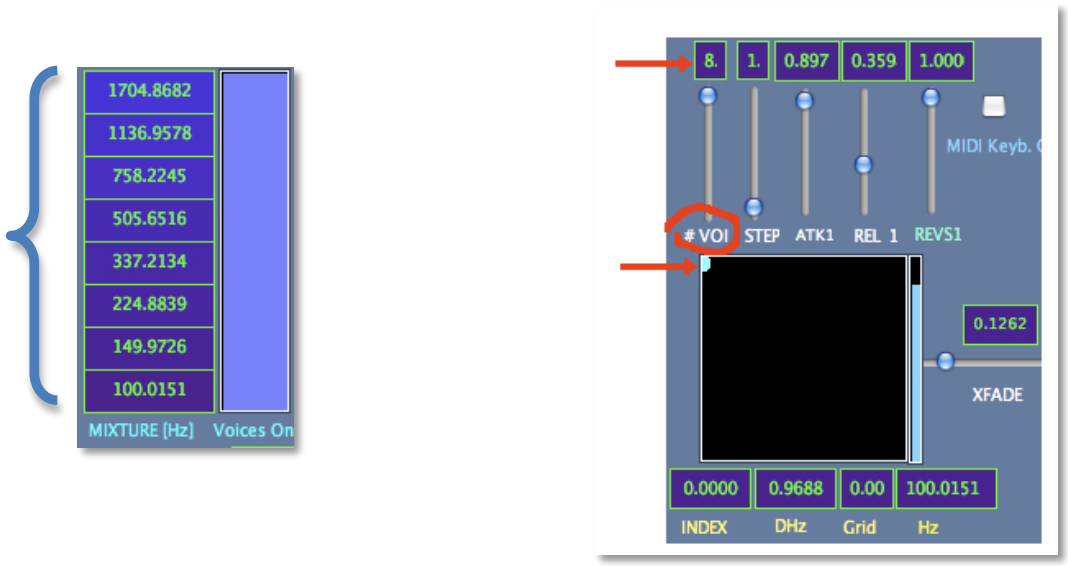


Figure 15

On the right side of the displayed frequencies column, a large blue colored bar shows graphically the number of active voices in the actual chord.

Moving the X joystick control the chord will change accordingly to the addressed values of the pre-computed pitch grid. It is important to notice that, if the X control is in a different position (i.e. in the position or INDEX = 1), the chord will have as root value 149.9726 Hz. In other words, the chord root starts from the index value (X position) inside the pitch grid. The higher the chord root, the brighter and higher will be the resulting sound quality.

In addition to the root chord, you can select its “density” specifying the distance between the chord notes via the STEP parameter and its relative slider. As you can see, in the rightmost part of figure 15, that slider is at minimum level (STEP = 1): this means that each note in the chord is selected doing a unit step in the pitch grid. But, for example, if you select STEP = 2, the root note will be the same but the others are selected skipping one value, as is schematically illustrated in figure 16. In this figure are represented four different notes placement depending on different values of parameter STEP. By varying the STEP value (from 1 to 9), you will change the distribution (or density) of the chord notes.

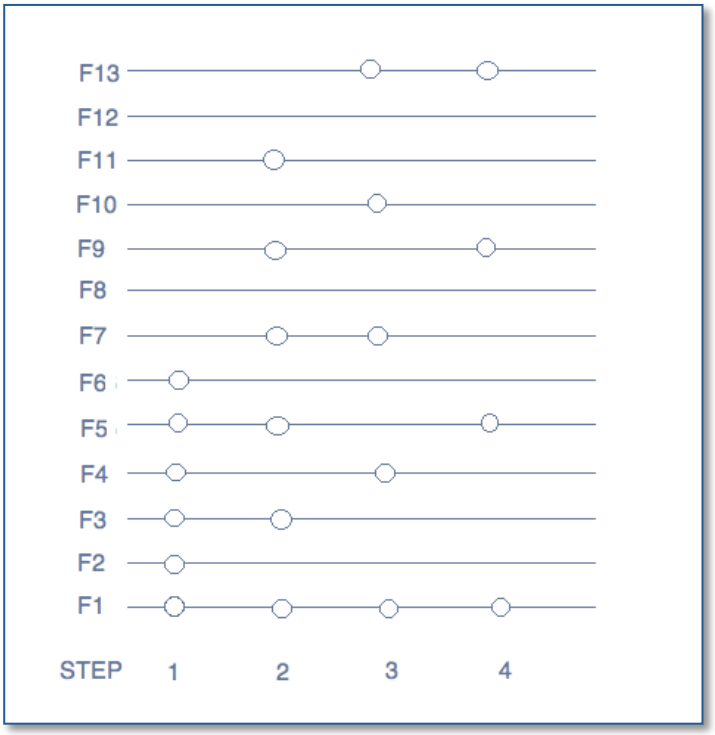


Figure 16

This effect produced by the incrementing of the STEP parameter is particularly interesting when using the harmonic grid. Notice that, with high values of STEP, one or more notes of the chord may fall outside the grid: in this case, the notes in excess are simply ignored.

Any change in frequency is always mapped on the actual grid, but since it will affect only the new notes, you will ear a kind of smooth sound timbre transformation rather than a simple change in pitch. In that sense StriaCS is more than any other thing a tool for create spectra transformations. The response time on changes depends essentially from two parameters that control the attack and release time of the internal note envelope. In the right side of fig. 15 there are two sliders for that purpose:

a) the attack time (ATK1 / 2)
b) the release time (REL 1 / 2)

You will obtain more smooth transformations when both the parameters are at maximum level but it is possible to achieve any desired combination of attack and release proportion.

SNAPSHOTS AND BANKS

Any combination of parameter setting can be saved in order to retrieve for a later use. For this purpose 48 snapshots are available, each one containing almost all the parameter set.

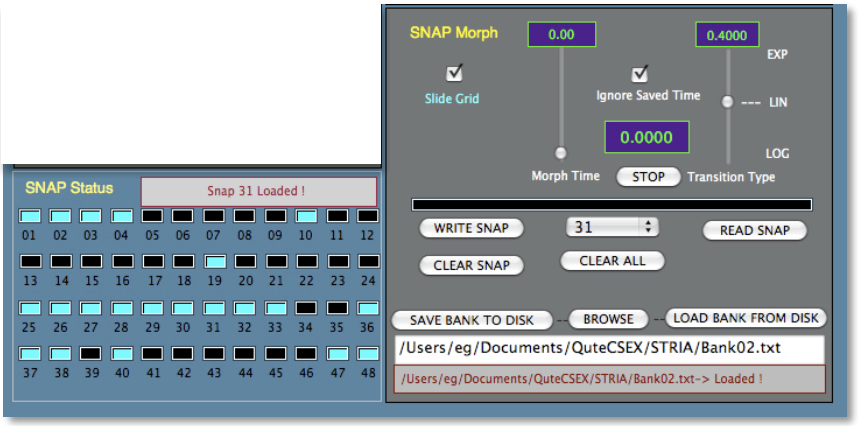


Figura 17

The most left panel contains the 48 led to know how many and which snapshots have been stored. You can overwrite over a saved snapshot or decide to write data on an unused one (switched off). At the top of the same panel you will find the SNAP Status that report the most recent store/load operation (in progress) or just done. To select a snapshot number (to read from or to write on) you have to click inside the numeric menu in the center of the right panel (in the figure above, it is showing the number 31). So, before reading or writing a snapshot you have to select which number you are going to use. After the selection, you can write (WRITE SNAP) or read (READ SNAP) the selected snapshot. If the selected number corresponds to an empty location (led off) a warning message is displayed in the SNAP Status. If the number corresponds to a stored snapshot (led on), you can retrieve all parameters stored in it. It is possible to clear all the snapshots using the CLEAR ALL button (all led will be switched off and each snapshot will be marked as empty) or to clear a specific one using the CLEAR SNAP button (the actual selected snaphot will be marked as empty ahe corresponding led will be switched off).

All the snapshots are stored in the RAM memory of your computer. If you want to save permanentely the data on disk you have to save those data in a bank. You can directly type the name of the selected file on the white window, just under the three buttons (Save, Browse and Load Bank) or using the BROWSE button to navigate among the file list and directories. Once you have selected the file (using one of the two options) you can save or load the 48 snapshots on or from the selcted file. Under the white window for the file selection is located a status window (light grey) for the load/save operation.

The stored file containg a specific bank of snapshot is an ASCII format file so you can acces and eventually carefully modify later using a simple text editor.

MORPHING BETWEEN SNAPSHOTS

Among the several features of StriaCS, on of the most interesting is the possibility of create smooth transitions between two different saved snapshot. This is not just a simplified cross-fade between two sounds but more properly a structural transformation of timbre and sound articulation. In other words, we are able to create a true *sound morphing* process in a selectable variable time and variable shape. This option can create an incredible number of timbral trajectories between a starting and ending timbral state. This means that we can know the initial and final state of sound but not the in between. Just within this transformation that we can discover sounds we can not think in advance.

The first step to make a sound transformation is to have two snapshots availables. The morphing is possible:

a) between two saved snapshots
b) between the actual configuration (not yet saved) and a saved snapshot

The morphing process is in practice a special case of the reading a saved preset (as explained in the previous section) plus some parameters setting in the SNAP Morph Panel. The parameters you have to set are:

a) the morphing transition time in seconds (MORPH TIME slider)
b) the mode of transition : Linear, Log or Exp (TRANSITION TYPE slider)
c) the mode of grid transition (SLIDE GRID check button)
d) the mode of transition set time (IGNORE SAVED TIME check button)

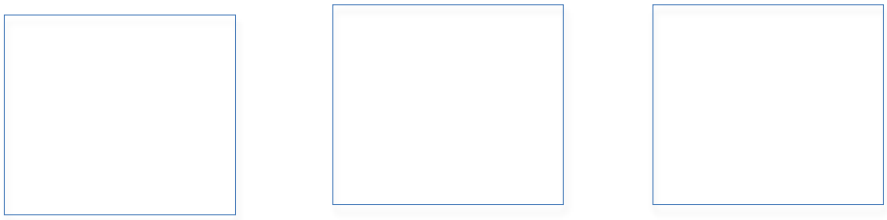


Figura 18 (a,b,c)

In figure 18 are showed three different mode of transition rapresented by different functions. The leftmost (a) means a slower initial transition in respect to the end one. To graduate this non linear curve the TRANSITION TYPE slider must have values less than zero up to the maximum negative values -1. The more negative is the value, the slowest will be the transition at the beginning and the fastest at the ending. The center function is used for a linear transition and it will be obtained only for the value of the parameter set to zero. The rightmost function is the opposite of the first one. The more positive is the value, the fastest will be the transition at the beginning and the slowest at the ending.

Each saved snapshot can contain also the associate transition time of the morphing so when you recall a snaphot to make a morph with the current sound, the transition will be done using the saved time. But it is possible to not use that information and do the transition with a different value of time. It depends on the state of the IGNORE SAVED TIME check button: when the button is checked, the transition will be done with the actual value of the Morph Time slider (that is to say ignoring the saved time information); when the button in unchecked, the transition will be done with the proper saved time.

During the transition, a progress bar will be activated so you can check the state of morphig process. In addition, a time display will be continuosly updated. If you want to make an instataneous transition (a classical snapshot recall), you have to put the MORPH TIME to zero.

NOTE: during the transition, almost the widgets (sliders, bars and displays) on the control surface will move according to the target values. In this time it is not possible to manually control the widgets until the transition process is completed.

Since any saved snaphot contains also the data for building the pitch grid, you can choice two different modes of operation:

a) the morphing process does not involves pitch grid transformation
b) the morphing procss involves pitch grid transformation

The mode of operation depends on the state of the SLIDE GRID check button: when button is checked, the morphing process will include the transformation of the pitch grid; when the button is unchecked, the morphing process will retain the previous pitch structure.

During a morphing process you can stop the process (pressing the STOP button) at any point of transition but reloading the same snapshot will cause a new morph between the “frozen” state and the selected snapshot. This feature is very interesting because during a morphing process yo can freeze any interesting situation that arises and save it for create new snapshots: in that way, you have an extraordinary tool for create many variation of sound quality allowing you to investigate on the parameter set producing that particular sound.

THE MIDI SECTION

Just working but manual in progress

THE TIMELINE

Realized works with StriaCS

Conclusions and Future developments