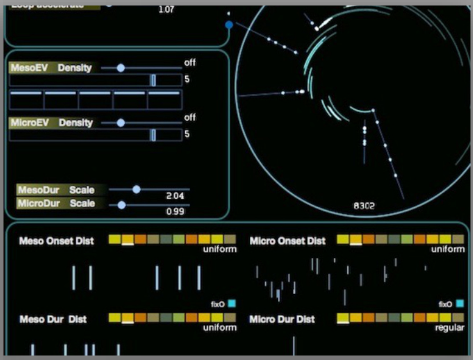
Cosmosƒ FX7.1 manual

CosmosFX is conceived, designed & programmed by Sinan Bökesoy

in VST/AU programming platform: *C++ / JUCE ,*



all rights reserved ©Sinan Bökesoy, sonicLAB™

www.sonic-lab.com

---

installation of *Cosmosƒ FX7.1*:

Cosmosƒ FX7.1 is an effect plugin.

Please unzip the downloaded files and keep a copy of it but erase any old versions existing in your HD.

(For the OSX version which comes with native Apple Silicon Support)

- ***CosmosƒFX71.component*** files : ideally should be copied to your system **/Library/Audio/Plug-Ins/Components** directory in your system harddisk. For example */Volumes/OSX-Main/Library/Audio/Plug-Ins/*

- ***Cosmosƒ*FX71*.vst3*** file : ideally should be copied to **/Library/Audio/Plug-Ins/VST3** directory in your hardisk. For example */Users/sinanbokesoy/Library/Audio/Plug-Ins/VST3*

You can install these files also to custom directories, however you will have to set this directory in your DAW so that it knows where to look for them. You can use custom directories to locate your preset banks.

(For the Windows version)

- Put the directory **CosmosfFX71PC** in **Program Files / CommonFiles / VST3** directory of your system harddisk.

!!Do not install just the .dll file but the whole directory into your VST3 directory.

For OSX : The app version you can keep anywhere you like.
For Windows : The app version has to be inside the downloaded folder, you can create a shortcut and move that anywhere you like.

CosmosFX71 uses the following directories : **Presets**, **impulses, MidiCtl** and **Seq** .

- For OSX these directories are created and files are copies at **Users/Shared/sonicLAB/CosmosFX**.

- And for Windows it can be found at **Documents/sonicLAB/CosmosFX**.

It is advised to keep your preset banks inside the Presets directory.

To run the Cosmos*f* FX7 plugin , you just need an audio input from the track you put the plugin.

some issues :

Clicks and Audio Interrupts: It is highly recommended to use a i7 Processor computer with Cosmos*f*. If your computer lacks of the necessary computing power, audio interrupts will occur and you will see a burst in CPU meter of your DAW. This means your computer cannot keep up calculating/filling the audio frame buffer in time and lags behind. In this case, you should decrease the event density or decrease meso and micro event duration scales; likewise the overlapping of discrete event will be less and the amount of audio rendering too.

Clicks -not because of lacking CPU power-, can happen when the audio output of each cycle is not zero padded or abruptly changing amplitude between the end and beginning of the new cycle. If you change the window type from “rectangle” to anything else except the “exp. attack”, the window will zero the amplitude at the start & end of each meso or micro event.

Clicks can happen when the sum amplitude of the overlapping micro or meso events is at digital overload hence clipping. To avoid this you can to alter the micro and meso amp gain sliders.

Clicks might happen due to DC offset in the signal chain. Sometimes the DC offset will be created by the chosen window functions such as “exp attack” or “exp decay”. A digital filter is never fast enough to handle an impulse like offset in the signal, so you will see that different combinations of windowing function in amplitude and filter envelope might generate audible clicks. This depends also on the audio waveform. (a sine vs. a sawtooth for instance) It is a trade off to introduce artificial methods to avoid these, which would smear the signal itself; hence an aesthetical choice.

Warning! : Due to its dynamic behaviour Cosmosƒ can generate unexpected audio levels depending on your operations, especially during the buffer feedback mode; so please keep your sound system output on a safe level.!

introducing the key shortcuts

You should grasp some essential information about the key shortcuts of Cosmosƒ *FX,* since they don't have a button/slider equivalent on the graphical user interface. Hence;

arrow keys >	by pressing any of them you can switch between the main screen and modulations screen of Cosmosƒ.
< **t** >	by pressing this key, you will switch the synchronization for the additive synthesis partials on/off. This switch is relevant only for the 'sample' mode of the additive synthesis section. The lower the frequency of the partial the longer it takes to play the total waveform, and the opposite for the higher frequency partials. When switched on, all the partials are forced to start together playing the sample waveform regarding their frequency setting. You will hear a chord. When switched off, they will start to swing like a wind-chime asynchronously.
< **Shift**>	When in *morphing* or on *surround editor* mode, pressing and holding the 'shift' key switches the mouse operation to where you can change the active ellipse radiuses to alter the distribution parameters. During this operation the ellipses won't change their position but just their radiuses.
< **p**>	Clean When pressed Cosmosƒ receives the host tempo value from the DAW and applies this to the MacroCellLength. Likewise the cycle lengths stay in sync with the host tempo changes. Pressing the key again deactivates it.
< **c**>	Cleans all the audio buffers of Cosmosƒ, sometimes it is useful to clean the buffers while handling complex situations also feedback level adjustments.
< **e**>	This key cleans the Object Sequencer tracks all at once. While you are recording and preparing a Cosmosƒ object, this key will simply clean your object.
< **z**>	Pressing on this key switches the app window size between factor x1 , x1.25 and x1.4. Do not use the resize handler of the window but use this key function instead to resize the app window.

how to read the manual :

Cosmos*f* is one of most advanced synthesisers and also a complex one , at least a different paradigm it does present in comparison to many other tools. Therefore it needs time to study, experience but is a very rewarding effort in return.

For starters, I suggest you to look in depth to all the tutorial videos found on the website (www.sonic-lab.com) We add in time more and more on different topics, and then looking into the manual will give a more detailed idea and hands-on resource.

There are of course topics which are not handled here in depth detail, such as stochastic functions, surround sound or the math of polar coordinates. You can find many resources on the internet on various levels.

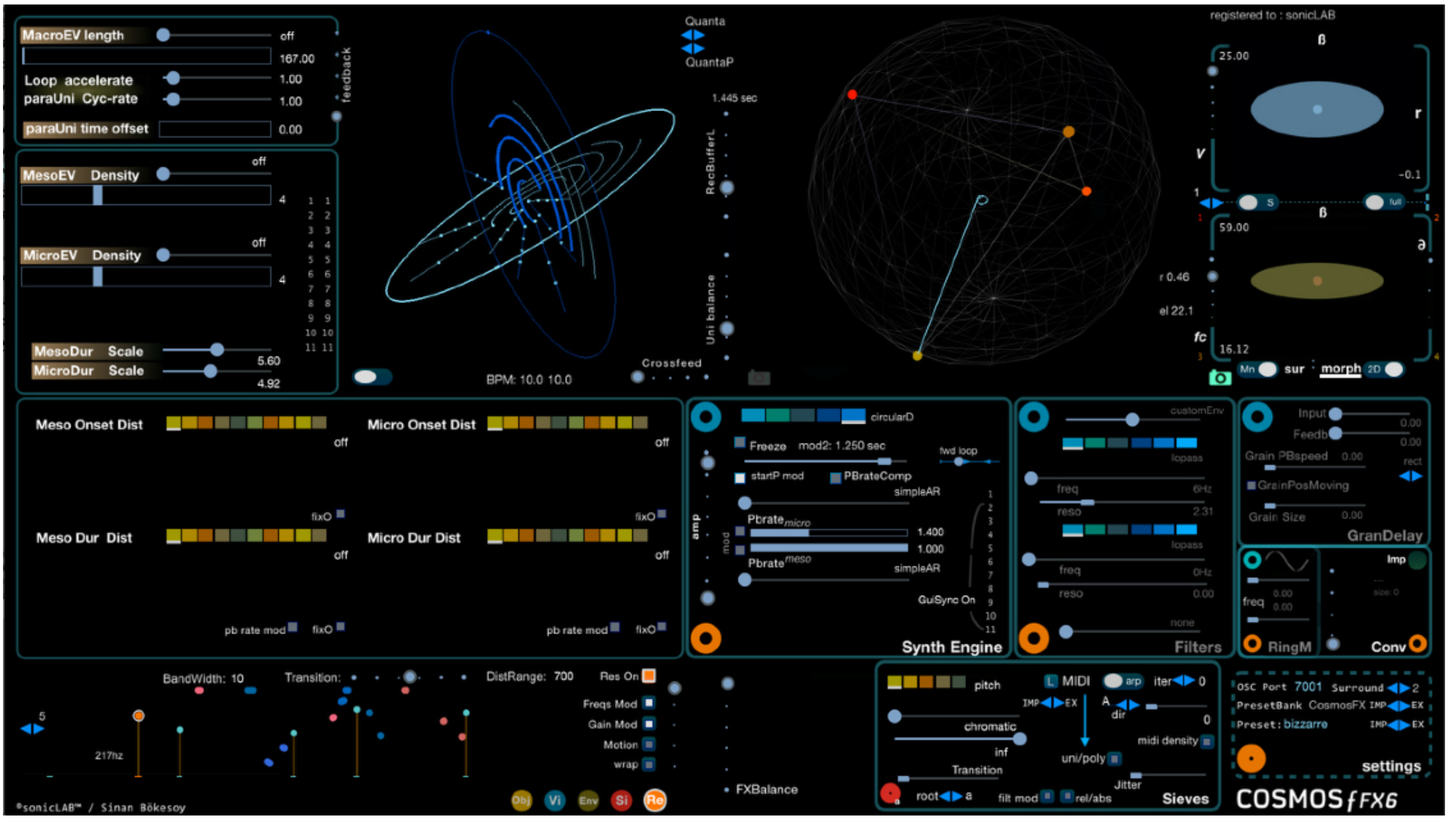
Every step on the structure of Cosmos*f* is driven by the language of mathematics mapped to sound synthesis. There is logic, transparent interaction and an enormous amount of potential to sonic emergence.

presets of Cosmos*f* :

Cosmos*f* presets are prepared keeping one perspective in mind : The input samples are chosen to be very simple in order to reflect the emergent behaviour and richness of the software. It is not our purpose here to add attractive and impressive sound samples and try to get around them, but choose very simple material to show you how Cosmos*f* becomes the computing engine preparing the impressive results which can’t be done by hand and by other software tools.

This is a stochastic synthesiser and has also a feedback routine of which level can be adjusted; btw this means things might probably not the same each time you play the presets and expect to behave the same way. As Heraclitus stated “You cannot enter the same river twice” !.

While this can open up very unexpected beautiful results at that moment, digital artifacts/clips/bad bits can harm easily the DSP chain. Therefore keep the “c” button of the keyboard handy which resets the buffers of Cosmos*f*.



What is Cosmosf?

Thanks again once more for purchasing Cosmos*f*, one of the most experimental and rewarding sound synthesis environment on the software synth world.

The roots of Cosmos*f* has been grounded during my Ph.D. research in computer music, which is about the creation of self evolving sonic structures starting with micro elements and organization towards the macro timbre where the sonic object gains its formal aspects. Such structures are subjected to be created by mathematical computing and algorithmic structure design.

There has been two predecessors of Cosmos*f* programmed by me during this period: the widely known *Stochos*, which has introduced efficient mappings of I. Xenakis's ideas on a modern computer music platform and the *Cosmos* application, which is more or less the attempt of expanding *Stochos* to operate on multiple time scales. Both were developed on MaxMSP platform and freely distributed.

The idea of integrating a bottom-up sonic organization into these top-down organizing event generation systems has been realized within Cosmos*f*, which is itself a complex event generation and modulation system but is also a emergent sonic structure thanks to its recursive mapping of its output back to its micro level organization. Constructing and programming of such an application was only possible for me on the C++ programming platform; and I am very excited for being able to preset it to you.

For the academically interested person, at the end of this manual there is a list of published articles following this research. But in this manual I will try to explain the operation of Cosmos*f* by avoiding any academic descriptions; however a certain experience in the field of sound synthesis is necessary. Cosmos*f* is a tool which I would define as beyond the line and it is why I believe that Cosmos*f* might be useful for composers of any music genre. It is truly an experience followed by experimental thinking and the intuitive exploration of new possibilities within digital audio.

There is still so much to put in Cosmos*ƒ*, and they will be showing up as the planned updates of the application. For instance in 2012, the V2 of the standalone version, has offered a genuine feature, the ‘parallel universe’, which replicates a Cosmos *cycle* with its event generation mechanism but with fully independent synthesis engine from the latter. V2.2 introduced a rich and intuitive morphing mechanism all with specially designed visualization support.

Finally the VST/AU version of Cosmos*ƒ* has arrived, which is probably on of the rare examples that a research product with such a rich outcome enters to the domain of professional music production platforms. And with the release of Cosmos*ƒ* Plugin V2, the software gains enourmous potential as you can independently define the synthesis structure of each micro and meso event and operate the powerful modulation tools on them. It is a wormhole to complex sonic structures.

Cosmos*ƒ* v*Saturn* includes the “sieves” engine to construct all the musical scales to be used on various levels of the synthesis engine. It also opens a new world to Midi performance possibilities.

Cosmos*ƒ* v*Saturn* supports surround sound audio with independent movement control of its meso-events in surround space by 2D/3D manipulation possibilities. The latest version Cosmos*ƒ* Saturn v2 is a high precision digital audio lab with impeccable sonic quality including high quality filters.

Cosmos*ƒ* Saturn v3 is the excellence reached after years of development. It includes dual macro events with overlapping ability, increased meso and micro event density and highest precision and stability for event generation mechanism.

And Cosmos*ƒ* v4 and Cosmos*ƒ* FX v4 is a total rewrite of the existing code, an enormous amount work to implement new features, extreme stability to carry on for the next developments. One of the most advanced sound synthesis tool of our times.

Cosmos*ƒ* Saturn5 and Cosmos*ƒ* FX5 introduces a truely modern and professional studio tools serving as an instrument, algorithmic soundscape renderer and a modern surround sound production tool for serious sound design tasks.

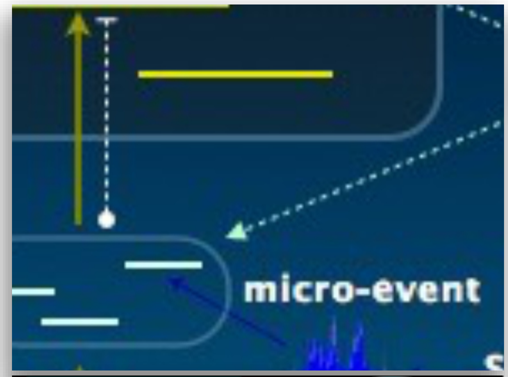
Cosmos*ƒ* Saturn6 and FX6 introduces the “Object” a 2150track parameter sequencer which records your gestures in various ways and plays them back in various configurations.

Cosmos*ƒ* FX7.1 is the most recent and stable version up to date (with native Apple Silicon support).

wish you a creative day,

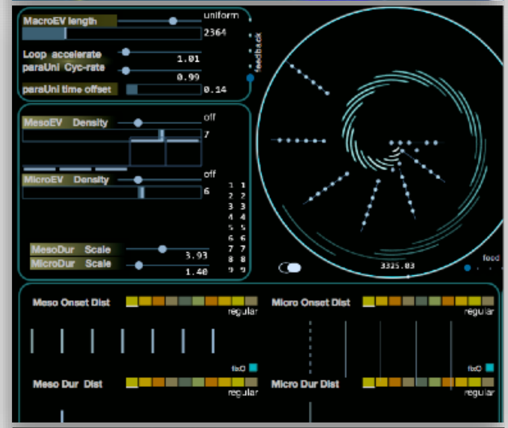
Dr. Sinan Bökesoy., October. 2022

chapters:

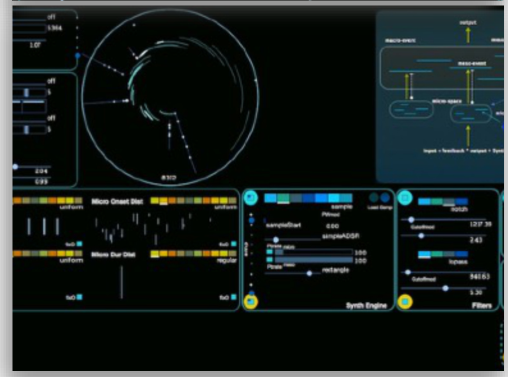


the Organization of the Manual

Structural facts and the event distribution process : Cosmos*f* is an advanced event distribution process where each event is assigned to sonic elements and its attributes. Therefore it is essential to understand the facts



The structure and controls of the main screen : The parameters concerning the event distribution process, the synth section assigning the sonic attributes for each event and some application settings are configured on the main screen along with some visualization tools.

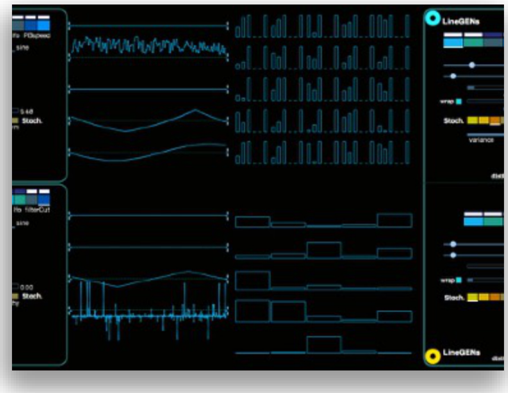


The Synth Engine of CosmosF: Cosmos*f* exhibits rich sonic potential with some familiar synthesis tools directed by rich modulation functions. Also the application has a recursive audio loop from its output to the micro event level input giving some unique emergent character to the sonic potential of the application.

The Morphing Mechanism: Cosmos*f* offers a novel preset morphing tool, maybe the first audio rate precision mechanism which handles hundreds of parameters in 2D or 3D interface.

Quantization settings: With the Plugin version of Cosmos*f*, users can assign independent quantization setting for each Cosmos*f* universe in order to create unique results and poly-rithmic structures.

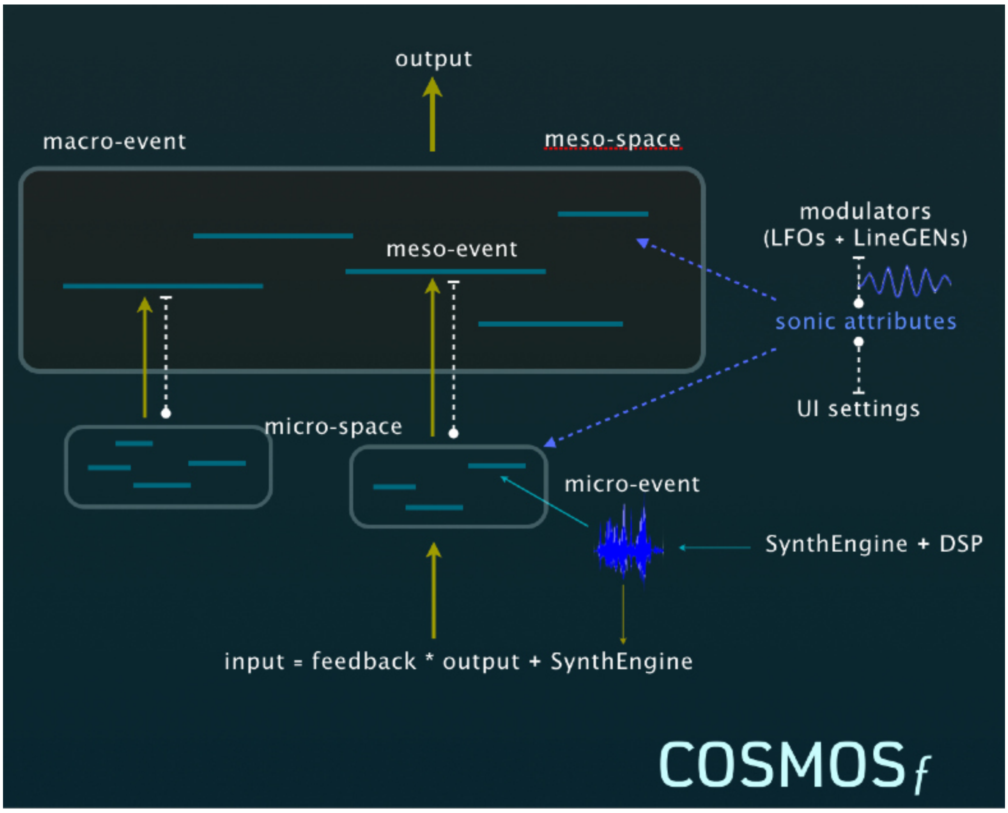
The Modulation sources of Cosmosf: Cosmos*f* exhibits excellent parameter modulation capabilities; driving the synthesis engine in real time within audio rate precision. The LFO's and LineGen's operate on meso and micro level and can be controlled with stochastic functions plus assigned deterministic values to modulate the parameters of present modulation waveforms.



"Sieves" engine in Cosmosf: Musical scales can be implemented to render various parts of the synthesis engine and a new world of midi performance possibilities are introduced.

Communicating with Cosmosf: User can control all the parameters of Cosmos*f* from external software/hardware by using a dedicated OSC(OpenSoundControl) protocol.

Structural facts & the stochastic event distribution



The top-down event organization in Cosmosf:

Music is about events; sonic components are organised in time as they arrive to our ear through the vibrations in the air. A note drawn on a score, or a midi event on the editor of a sequencer or a phrase of recorded sound in a track of your DAW, they all exhibit an event with specific start and stop time and certain evolution during its duration.

Xenakis has introduced the use of statistical mathematics for the organization of notes and sequences and he used this approach extensively in his works beginning in late 50's. He has used different types of stochastic functions to decide the distribution of musical components and configuration of the structure. His ingenious approach led to the roots of sonic granulation in computer music and some computer aided composition as named algorithmic music.

Cosmos*f* is an advanced stochastic event generator and organizer distributing sonic elements on multiple time scales along with complex modulation scheme. As you see on the figure above, we define the whole being of Cosmos*f* as a *macro sonic event*, having a certain duration titled as **MacroEV Length**.

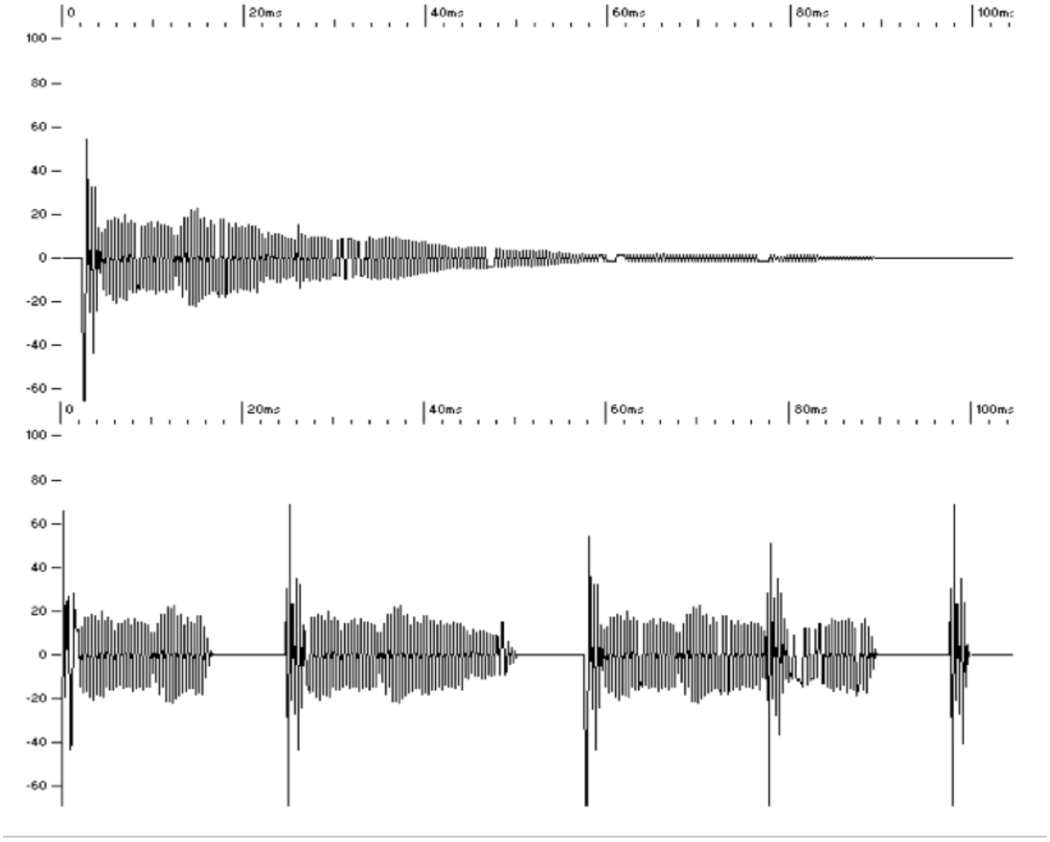
Meso-events of certain density are distributed inside this time space with their **onset time** and **duration** parameter calculated by stochastic functions. Also the **meso event density** can be calculated with stochastic functions.

Each meso event defines a **micro-space** in the Cosmos*f* event distribution system. In each micro-space, events of certain density are distributed with stochastic functions. They are called **micro-events**. The micro-events are the sonic elements in Cosmos*f*; they will carry the waveform information assigned by the **Synthesis-Engine** to them. The micro-events or meso-events can overlap each other.

The event distribution process in Cosmos*f* follows the hierarchy explained above whenever there is a new macro event to be filled with meso events; and then instantly they will be filled with micro events. It does calculate the necessary onset time and duration values with the given density values and assigned stochastic functions.

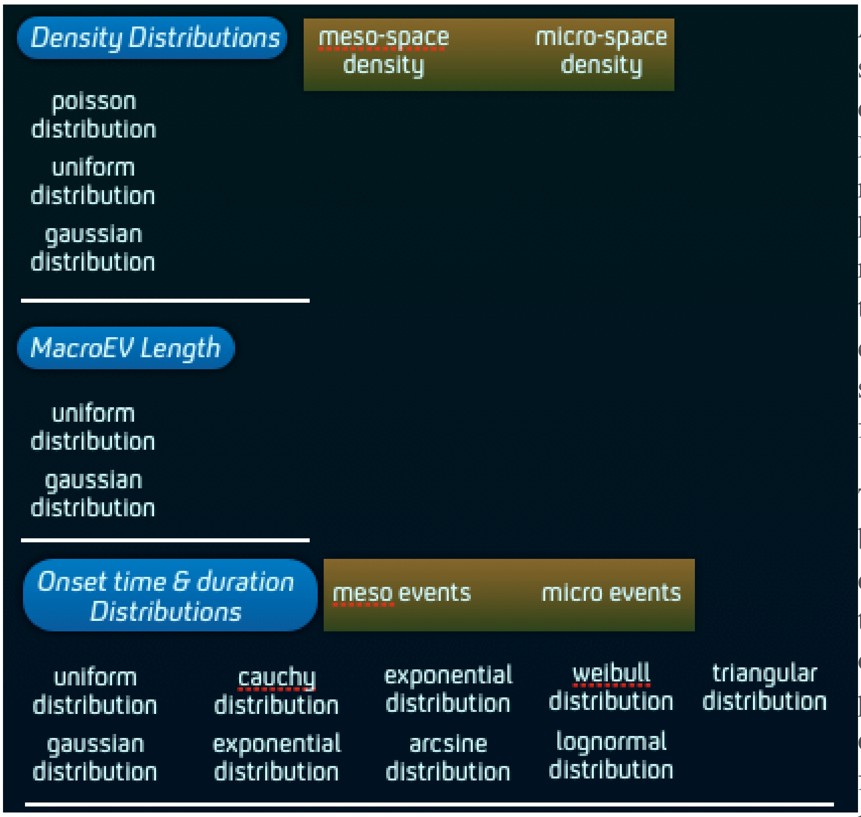
This self similar distribution structure augments the complexity of the distribution process proposed by Xenakis, while keeping the system integrity. Like in the nature, the organization happens on multiple scales; the local/macro density, the dynamic behaviour and therefore the perceived complexity.

We mentioned that the micro-events are carrying the sonic entity assigned by the Synthesis-Engine, but what does the meso-events do? Actually the micro-events accumulate each other and become a meso-event. In other words, a meso-event is itself a sonic-event consisting of micro-sonic events generated by Cosmos*f*. On the upper part of the figure below, you will see the waveform assigned to the micro-events. And the waveform below represents the meso-event waveform constructed with the accumulation of the micro events.



The meso-events accumulate in time to construct the macro-event, which is the output of Cosmos*f*. And here comes another unique feature of Cosmos*f*. The output can be fed back to the micro-level events. This opens the doors of self evolving emergent sonic character. By introducing a hierarchy of multiple time scales for the event distribution process; recursive audio feedback + modulation sources constructed with stochastic - deterministic functions, it is possible to form sonic creations that cannot be handled by traditional synthesis methods. The user intervenes with the system in real-time by inputting a sound source at the micro level and interacting with the system by controlling the parameters for the sonic event generation on different time scales and parameters for different type of synthesis/modulation generators.

Cosmos*f* uses probability distribution functions, also named **stochastic functions** to decide the density, onset time, and duration parameters of each meso and micro event. For instance the following table is a list of the stochastic functions operating in *Cosmosf*;



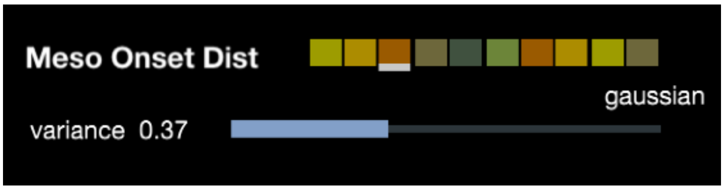
Density Distributions
meso-space density
micro-space density
poisson distribution
uniform distribution
gaussian distribution

MacroEV Length
uniform distribution
gaussian distribution

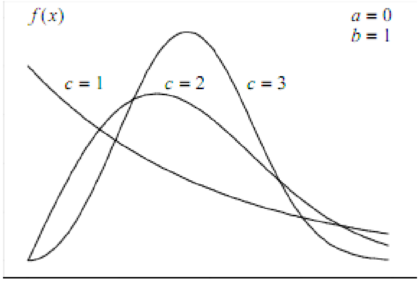
Onset time & duration Distributions
meso events
micro events
uniform distribution
cauchy distribution
exponential distribution
weibull distribution
triangular distribution
gaussian distribution
exponential distribution
arcsine distribution
lognormal distribution

A probability distribution function shows distinctive behaviour when distributing the values as its output. For example we want to distribute micro events in a micro-space. We know the micro-space length from the relevant meso-event duration. And now to obtain the duration of the micro-events with some density, we call a stochastic function, which will decide for us the micro-event duration.

The duration distribution should be between 0 - 'micro space duration'. We call this function the same amount of times as the micro space density. Most of the stochastic functions have parameters, a value of certain range to change the behaviour of the stochastic function. The reason that Cosmos*f* has many types of stochastic functions is to offer to you different types of 'distribution shape', which gives a distinctive character to the process being used for.

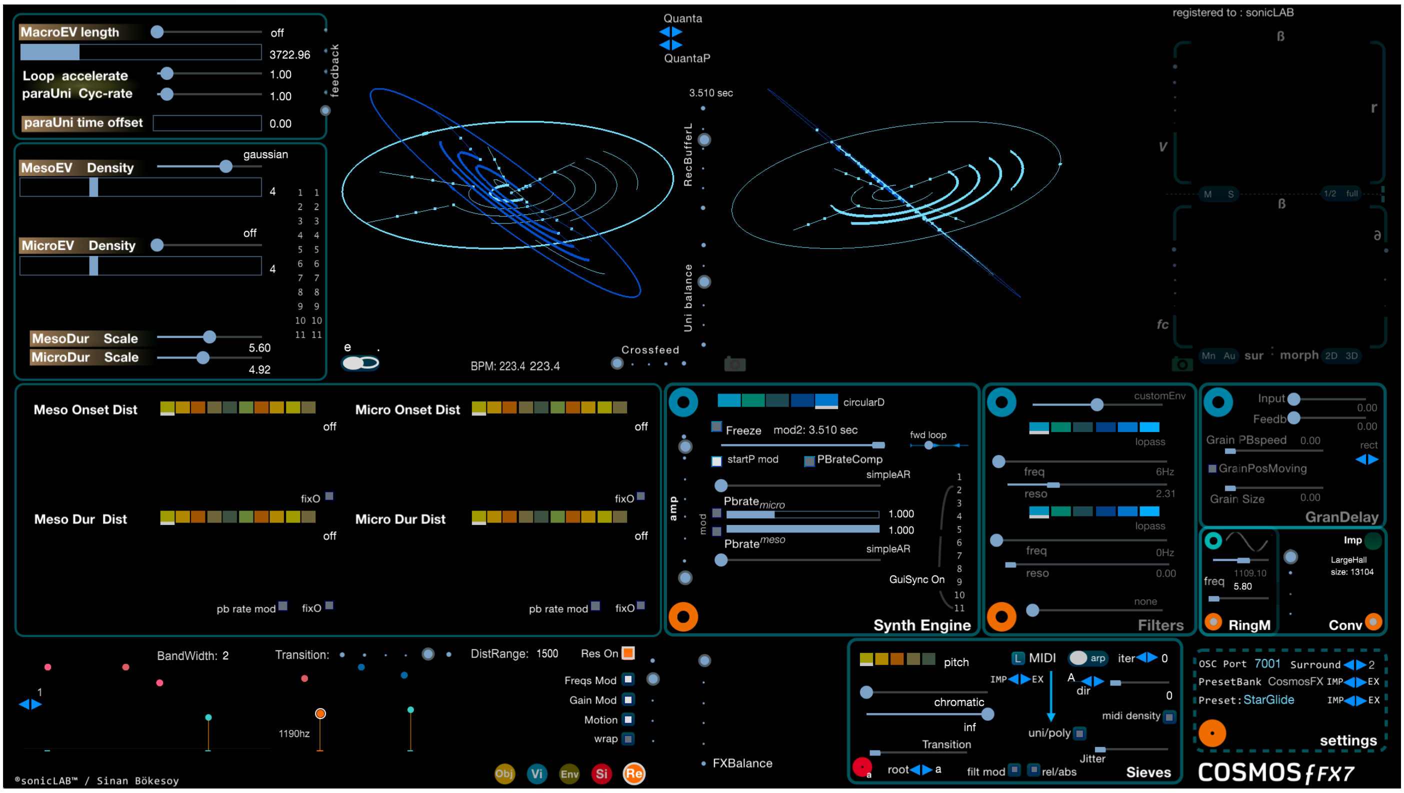


Meso Onset Dist
gaussian
variance 0.37



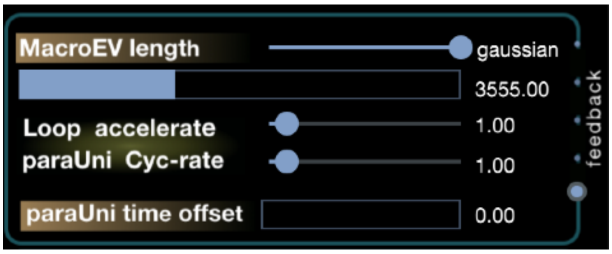
On the figure above the gaussian distribution function has been assigned to the Meso-Onset distribution. The parameter is set to .37. The 'distribution shape' of the Weibull distribution is given on the right with three different parameter settings (1,2,3). If you are interested to know in detail about stochastic functions, this link is a recommended reading; http://ftp.arl.mil/random/random.pdf

Cosmos*f* and the controls of the main screen



EVENT

The top-down event organization in Cosmos*f* is controlled within the panel on the top left side of the main screen.



The macro event length can be set with the slider under the **MacroEV length** label. The value is set in miliseconds. You can also use stochastic functions to define the MacroEV length. With either **uniform** random or **gaussian** random distribution functions, one can calculate the MacroEV length after each full cycle of Cosmos*f*. The type of the stochastic function can be set with the point slider next to the MacroEV length label.

The **Loop accelerate** slider can speed up or slow down the whole process of Cosmos*f* between 0.05-10 times. Please consider pressing the *'f'* button when changing the value on a slider, likewise a smooth acceleration or retardando can be achieved. (It is the event distribution time information which is being altered with this slider not the digital sampling rate of the system.)

The blue point is a vertical slider controlling the feedback audio gain of the audio loop going out from Cosmos*f* and entering back to the micro level event input to Cosmos*f*. *Please adjust to level carefully to avoid undesired distortion.*

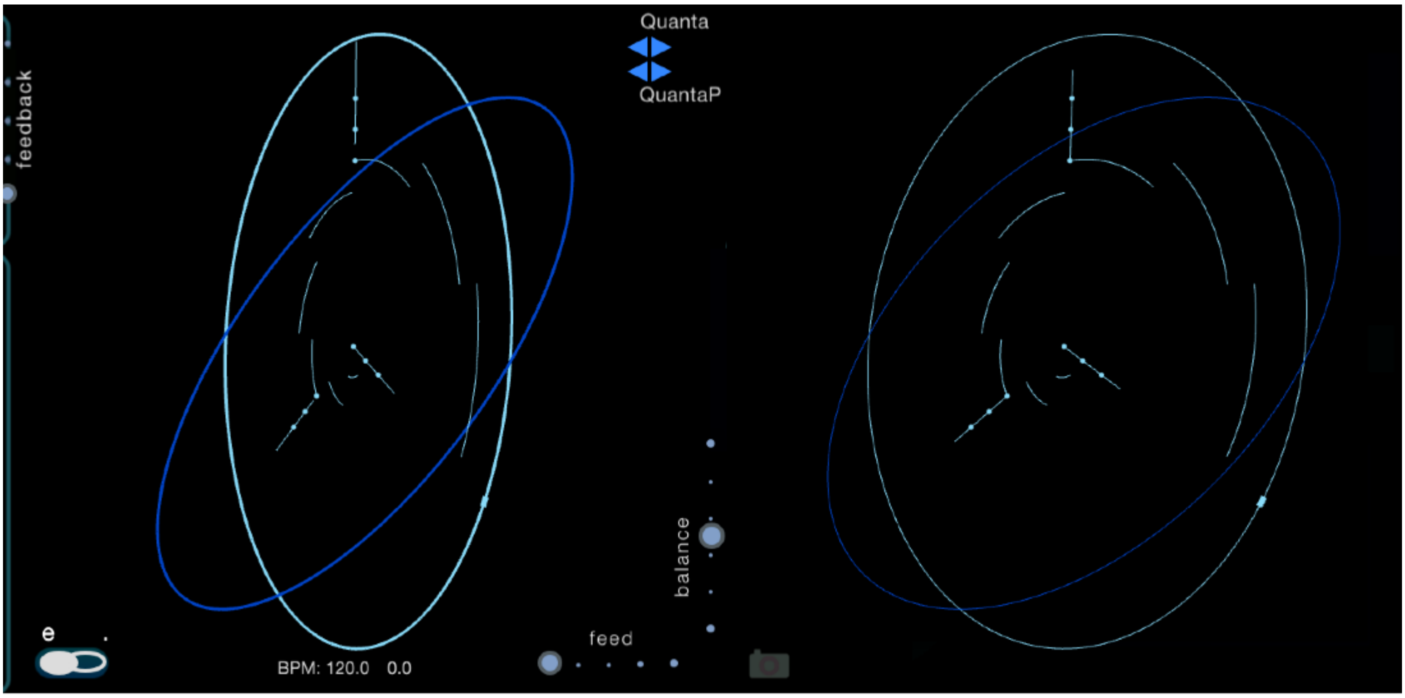
explaning the new "Parallel Universes" feature in Cosmos*f*.......

Cosmosf offers a unique feature, a powerful macro-sound design paradigm named "*parallel universes*". Cosmos*f* event generation mechanism consists of meso and micro level events distributed in one Cosmos*f* cycle, which can be regarded as the *macro sound* level. A "*parallel universe*" extension complements the original Cosmos*f* cycle with another cycle operating in parallel with the same event generation parameters -the same stochastic big-bang conditions- but with totally independent synthesis engine. Therefore with the *parallel universe*, we let initialize another parallel Cosmos*f* cycle with the same event generation paradigm, where the material of this universe can be chosen different.

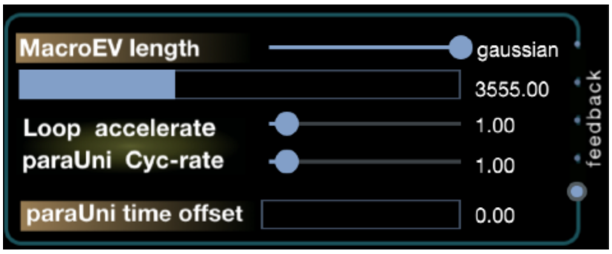
To activate the "*parallel universes*", click on the oval switch to bottom left of the primary cycle, which will convert into this . And you will see 2 cycles lying next to each other.

(in case the primary cycle's synth engine is not set to additive

When you click on this photo button between the universes, then immediately the primary universe parameters will be copied to the parallel universe.



There are 2 parameter sliders on the top-left part of the main screen: the **paraUni Cyc-rate slider** lets you change the parallel cycle rate relative to the primary cycle. For instance the time can be set to a different speed for each cycle. The **paraUni time offset slider** lets you set an onset offset between each cycle, which can be also seen as a phase shift. Therefore you can apply perfect continuous looping cycles by setting a proper phase shift.


MacroEV length gaussian 3555.00
Loop accelerate 1.00
paraUni Cyc-rate 1.00
paraUni time offset 0.00

As explained above, the "*parallel universe*" share the same event generation mechanism with the same parameter settings. But you can set the synthesis engine parameters individually by first clicking on the cycle area, of which synth parameters you want to edit. The cycle will be likewise highlighted. Then you can use the synthesis engine panels. If you click again on the same cycle, then it will be un-editable, and the other cycle becomes editable automatically.

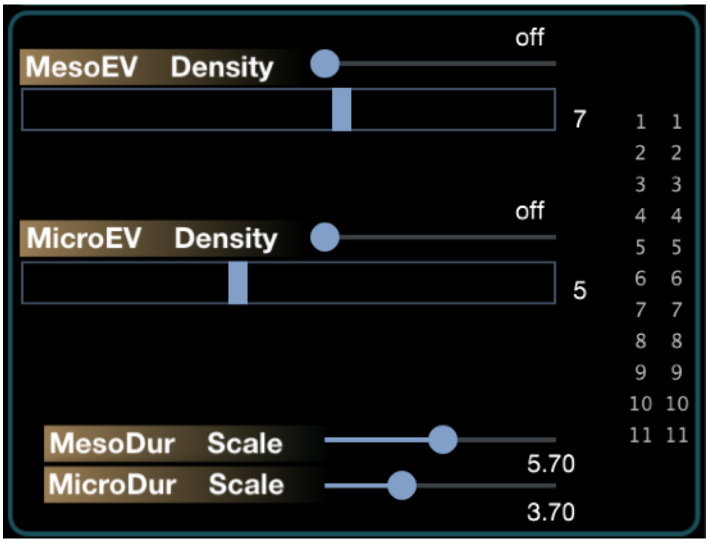
You will notice the two sliders positioned between the cycles. The vertical **balance slider** sets the amplitude balance between the parallel cycles. And the horizontal **feedback slider** sets the amount of signal which enters to the secondary cycle from the primary cycle to be processed with the synth engine of the secondary cycle.

The user interface on the main screen of Cosmos*f* shows the synthesis engine parameters of the active editable universe. All the parameters belonging to a universe will be taken in account when the preset morphing feature is being applied, which will be mentioned in the next section.

………back to the stochastic event generation mechanism

The main screen panel in the figure below has controls to define the event density distribution process for the meso and micro events. The meso event density in the meso space and the micro event density in the micro-space can have a value between 0 - 11 (Cosmos*f* Saturn v3)

Let's say you set 5 for the meso event density and 4 for the micro event density. The total amount of micro-events in the Cosmos*f* cycle is then 5*4 = 20. The density can be controlled with stochastic functions such as the **uniform, gaussian, poisson and binomial** distributions. The maximum micro-event density is 11*1 = 121, which will be distributed in a Cosmos*f* cycle with calculated onset and duration parameters.



MesoEV Density off 7

MicroEV Density off 5

MesoDur Scale 5.70

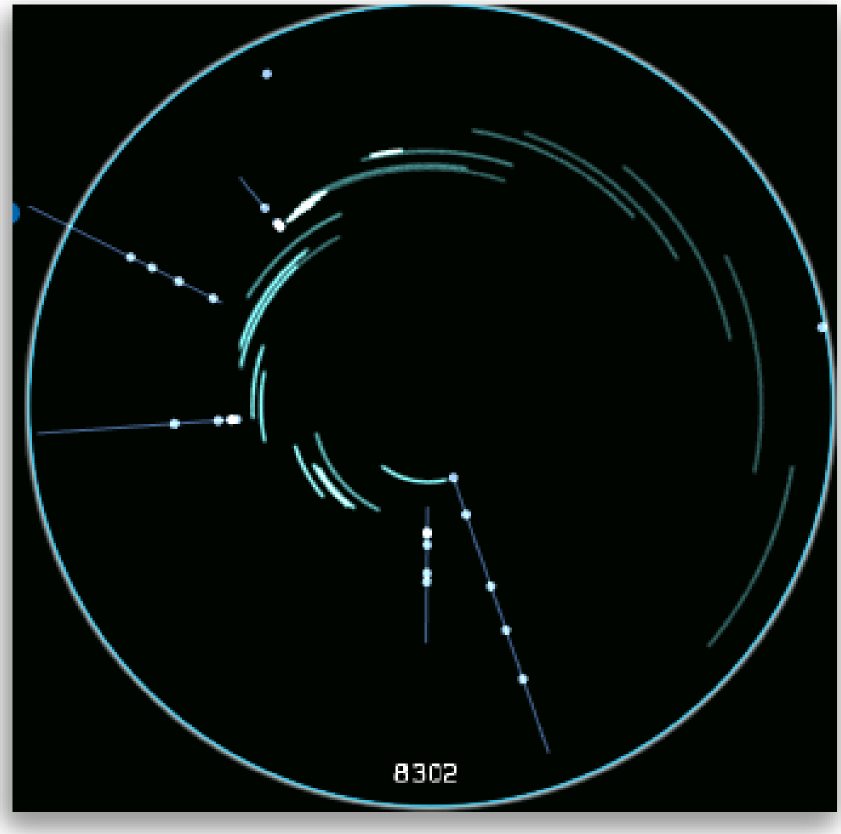
MicroDur Scale 3.70

With each new cycle a new **MesoEV density** will be assigned, and with each new meso-event onset, a new **MicroEV density** will be calculated. To set the maximum density amount for a distribution, use the sliders with the blue bar. Cosmos*f* shows the current calculated density with the unfilled blue bar on the same slider.

The meso and micro event durations can scaled by the user, which means their duration value will be multiplied with the value set on the sliders **MesoDur Scale** and **MicroDur Scale**. Scaling the value of the event durations do affect the overlapping between them. Scaling the value of the event durations do affect the overlapping between them. These can be set individually for each micro and meso event too.

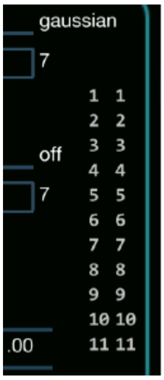
What happens when the meso event durations are scaled ? The actual cycle duration in Cosmos*f* is calculated by taking the duration of the last meso event distributed in the current Cosmos*f* cycle. This means it can be larger or smaller then the value set on the MacroEV length slider. The current calculated value is being shown on the circular Cosmos*f* event visualization area, on its bottom part.

However you can force Cosmos*f* to obey the value set on the MacroEV length slider but skipping the excess duration values. To do that you can press the *'l'* button and pressing again will return to the initial behaviour.



The Cosmos*f* cycle visualization gives you an instant look to how the event distribution process generates the meso and micro events in each cycle of Cosmos*f*. Around the circle you will see the small dot turning clockwise to show you the time. A full circle equals always the to cycle time as shown at the bottom part of the circle.

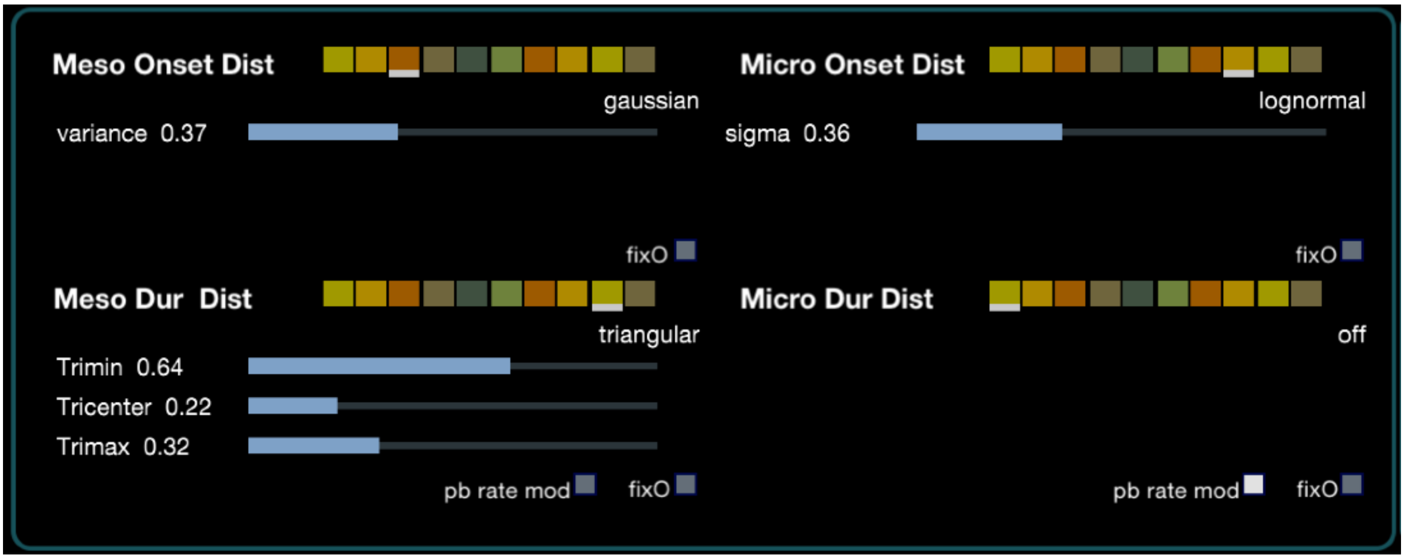
The straight lines on the circle define the onset position of meso-events distributed in the macro-event. The small circles found on each meso-event line represent the micro-events. When their onset time arrives they start to move on an arc around the center of the circle. the end of the arc is the end of the micro-end duration. The micro-events are positioned on the relevant meso-event line relatively to each other according to their onset time. The length of the meso-event line represents the meso-event duration. The arcs drawn on the screen represent each micro-event path with the absolute onset time as the start of the arc and the duration of the micro-event as the arc length. You can easily observe the overlapping events, their lifetime etc. information compact enough to grasp at one look.



Another feature is the possibility of **muting/unmuting** the meso events and the micro events with individual switches laid out on the event density control panel. The switches on the left column are dedicated to each meso event; when they are muted the button is in select state. The right column is dedicated to the switches muting the micro events. For instance if you click the 2nd switch, then every 2nd micro event in a micro-space is muted.

This feature is very handy in order to bring rhythmical silences in the flow of Cosmos*f* cycle, also to deconstruct / construct the macro object on the event level.

The bottom left part of the main screen is dedicated to the control set for the stochastic functions distributing the meso and micro event data. For instance each color button assigns a stochastic function for each distribution. You can vary their parameters with the slider showing up below the color bar area.



variance 0.37

sigma 0.36

Trimin 0.64

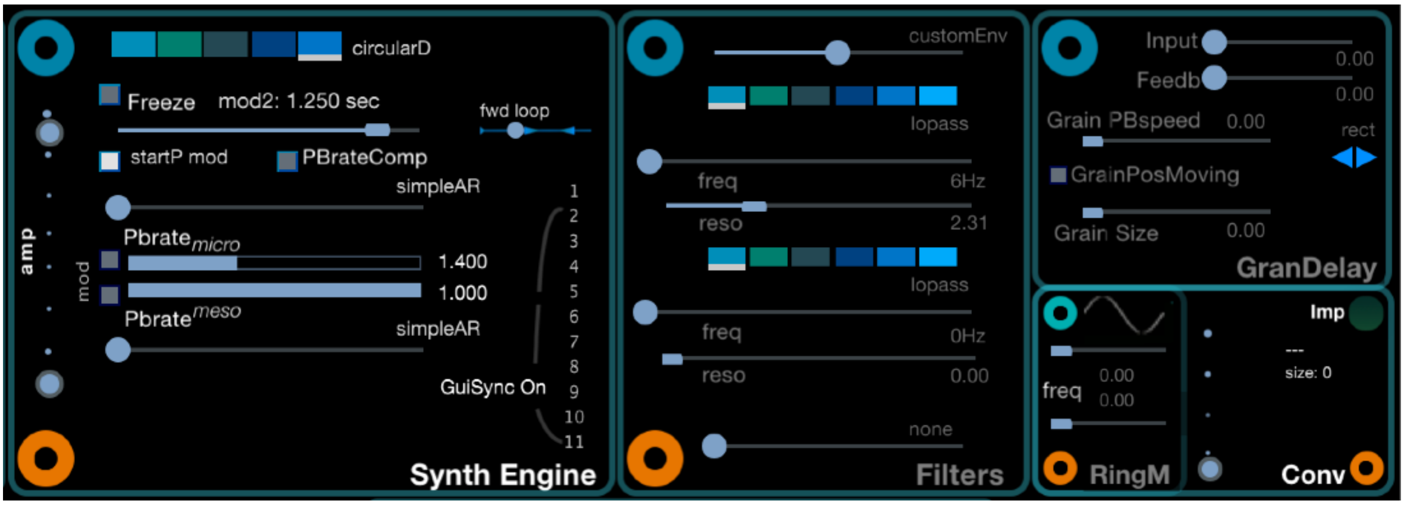
Tricenter 0.22

Trimax 0.32

One peculiar feature in Cosmos*f* is the ability to **freeze** the stochastic distribution for meso/micro event onset and duration distributions. The buttons labeled as **fixO** are responsible to activate/disactivate this behavior. Likewise when clicked, the distribution data remains in the memory and Cosmos*f* no more does any new distribution calculation for the frozen distribution. This serves as a compositional tool, such as repeating certain distribution processes becomes as stable perceived patterns, while keeping other elements calculated with stochastic functions again with each new event cycle.

With the new Cosmos*f*, you can map the duration values of the meso and micro events directly to playback rate values of these events. This is inverse proportional, likewise when the event is short the pitch will be higher. This can be activated by the **pb rate mode** switches on this panel.

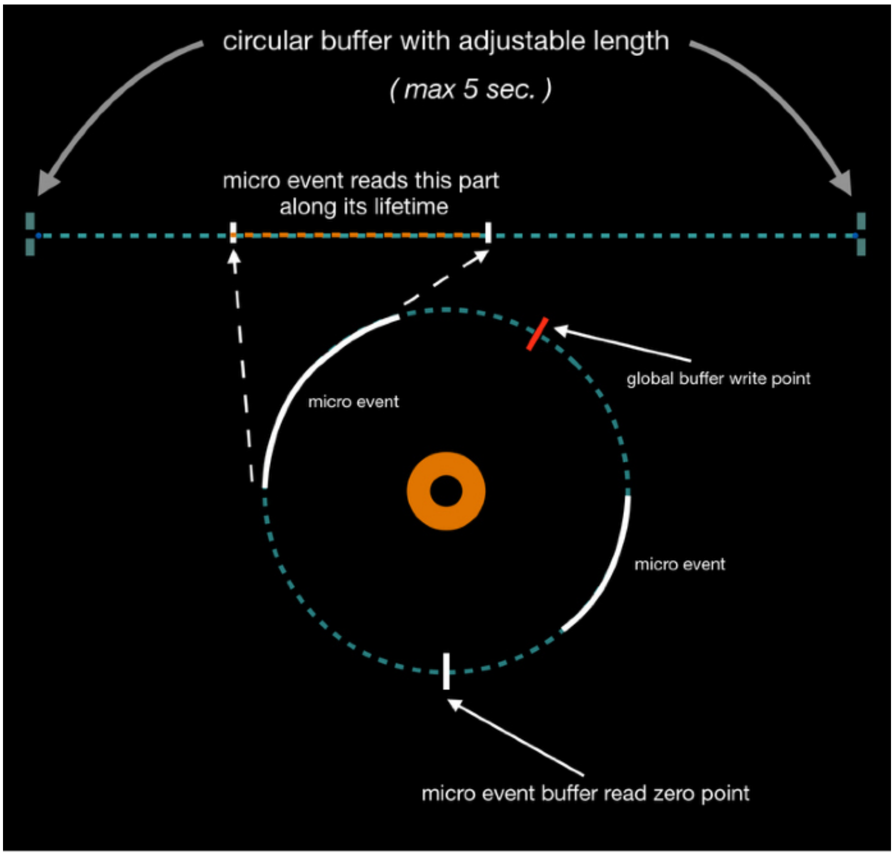
The synth engine of Cosmos*f* FX



The right bottom side of the main screen is dedicated to Synth Engine panel of Cosmos*f* FX. It has been mentioned that the Synth-Engine is responsible to assign the sonic material to the micro-events. It calculates

Cosmos*f*Fx synth engine is based on live complex delay buffer handling in various ways. On **Cosmos*f*Fx** "**s**" button is discarded, as the effect plugin becomes active when it has an audio input and starts/stops its process with DAW transport controls in parallel.

Like on every audio FX plugin, there is a **FXBalance** slider at the bottom of the screen, which you might want to adjust to balance the wet/dry content in the output.


circular buffer with adjustable length
(max 5 sec.)
micro event reads this part
along its lifetime
micro event
global buffer write point
micro event
micro event buffer read zero point

How is the live input being processed ?

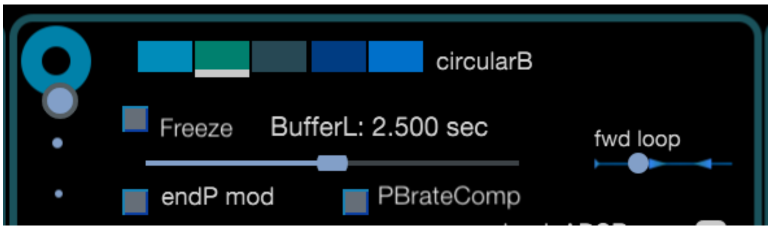
Cosmos*f* uses a continuous read/write buffer. A buffer of 5 seconds is recording the incoming audio to the plugin and when the write pointer reaches the end it restarts from the beginning of the buffer.

Reading from such circular buffer in various ways and various speed has interesting consequences, and Cosmos*f* uses this phenomena with all its dedicated modulation power.

How is the audio data created at the bottom level ?

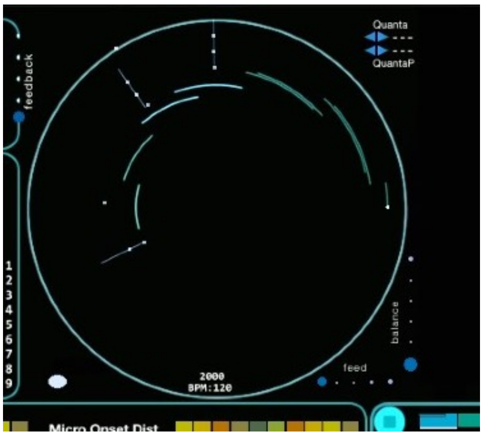
The micro events read during their life span from this buffer with assigned playback rate and start & end points as shown on the figure at the left side.

The micro events read during their life span from this buffer with assigned playback rate and start & end points as shown on the figure at the left side. The red line is the global buffer read/write point, of which speed is constant defined with the DAW sample-rate.



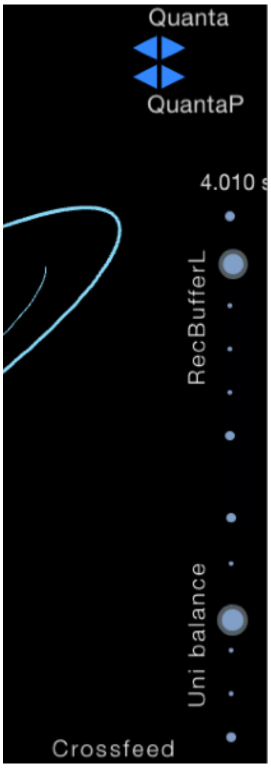
The synth engine of Cosmos*f* Fx v4 offers 5 different delay buffer handling methods which concerns the mapping of the incoming data as the micro event audio content.

All the modes, except the **directSmp** mode, offer following controls: *startP & endP mod* switch, *PBrateComp* switch, *BufferL slider* and *playback direction* slider.



One again, each of these arcs representing a micro event life span, are being filled with the live audio input fed into Cosmos*f* and each different mode handles this process in a different way offering the total handling of Delay Buffer Science !

- If the **directSmp mode** on the synthesis engine is selected, each micro event samples the incoming audio into its buffer and plays in from there synchronously. If the playback rate of the micro event faster then the reading from the buffer, then data will be missing (as the content cannot be coming from the future!) You are free to apply further synthesis engine tools on this micro event data content. (Filter, GranDelay etc.) And of course each micro event can have its dedicated parameters.



When the *startP & endP mod* switch is turned off, (so it is on endP mod.) each micro event starts reading the buffer from the zero point.

The end point until which the micro event reads from the recording buffer is defined with the *BufferL* slider. The maximum buffer size of this slider is defined with the *RecBufferL* slider as you see on the left figure.

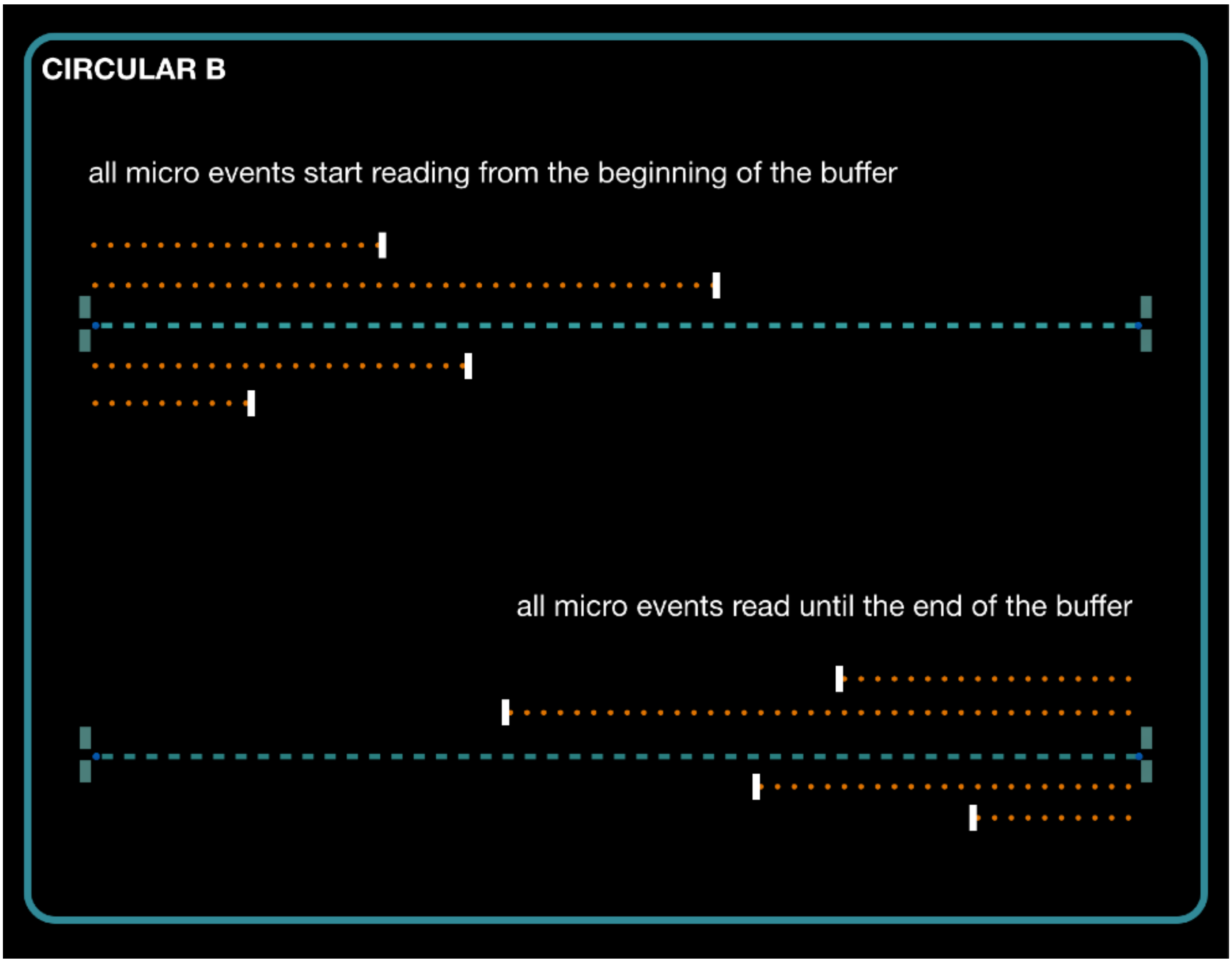
For example the *RecBufferL* is set to 4.010seconds and this will be the maximum range, which you would be able to use with the *BufferL* slider.

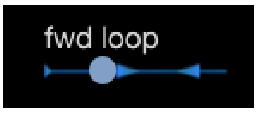
The end point of the micro event can be further modulated with the LineGen3 and LFO3 based on the value you set with the *BufferL* slider.

When the *startP & endP mod* switch is turned on, (so it is on startP mod.) each micro event starts reading the buffer with the *BufferL* slider, and you will notice that the slider end values are inverse. You can apply further modulation with the LineGen3 and LFO3 based on the value you set here.

The end point until which each micro event reads from the buffer is fixed with the RecBufferL slider value.

On the figure below this two types of behaviour of the *startP & endP mod* switch has been explained graphically as well. The red dots represent different micro events but they share the same starting and ending points depending on the setting of the mod switch used for this **Circular B** mode.


CIRCULAR B
all micro events start reading from the beginning of the buffer
all micro events read until the end of the buffer



On the upper right side of the Synthesis Engine panel, you can arrange the playback modes for each micro event. When it is in *one shot* mode, the buffer dedicated to the micro event will be played once and you will get silence if the content is shorter then micro event duration. So you can try to experiment with the other playback modes as well :

one shot, forward loop, fwd & rev random, reverse

Let's explain the PBrateComp (PlayBack rate compensate) switch : There are two dependencies which affect the playback rate of the micro events, hence their perceived pitch.

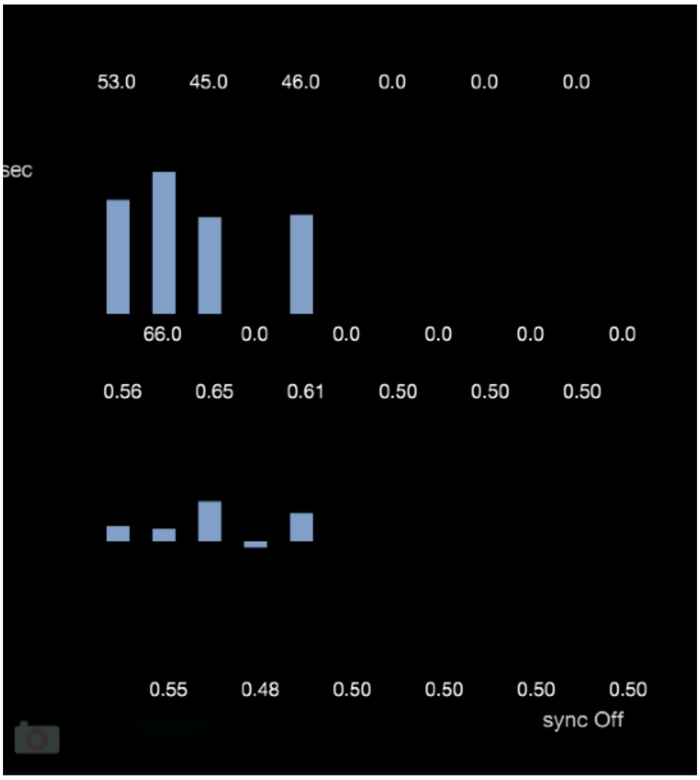
- The *PBrate slider* for micro/meso events. This is the usual slider to control the post pb rate directly.
- The buffer length to be read by each micro event, which is defined as the length between the start and end pointers. These pointers can be changing with modulation sources as explained above with the case of Circular B mode.

In the last case the micro event has to read the buffer length set between the start and stop pointers in the time frame set as the micro event duration. Assume that the micro event duration is two times longer then before, likewise the playback rate will be two times slower to fit the event duration to the same buffer chunk length. Quite a relativistic situation !

Therefore the *PBrate* is a dynamically changing one inversely proportional to the buffer length being read by each micro event. The *PBrateComp* switch tries to fix this by applying an inverse coefficient dynamically. Hence, the perceived playback rate will be re-fixed but the buffer content will be changing continuously as expected.

Finally the **Freeze** button below the *BufferL slider* lets you freeze the content of the record buffer and stops the overwriting process of Cosmos*f* FX. Therefore you will keep using the contents of the last recorded buffer.

If the **Additive mode** of the synthesis engine is selected, 12 partials are assigned for each micro event reading from the recording buffer at different playback rates and intensity values.



Again here, the *startP & endP mod* switch setting is important. When it is off;

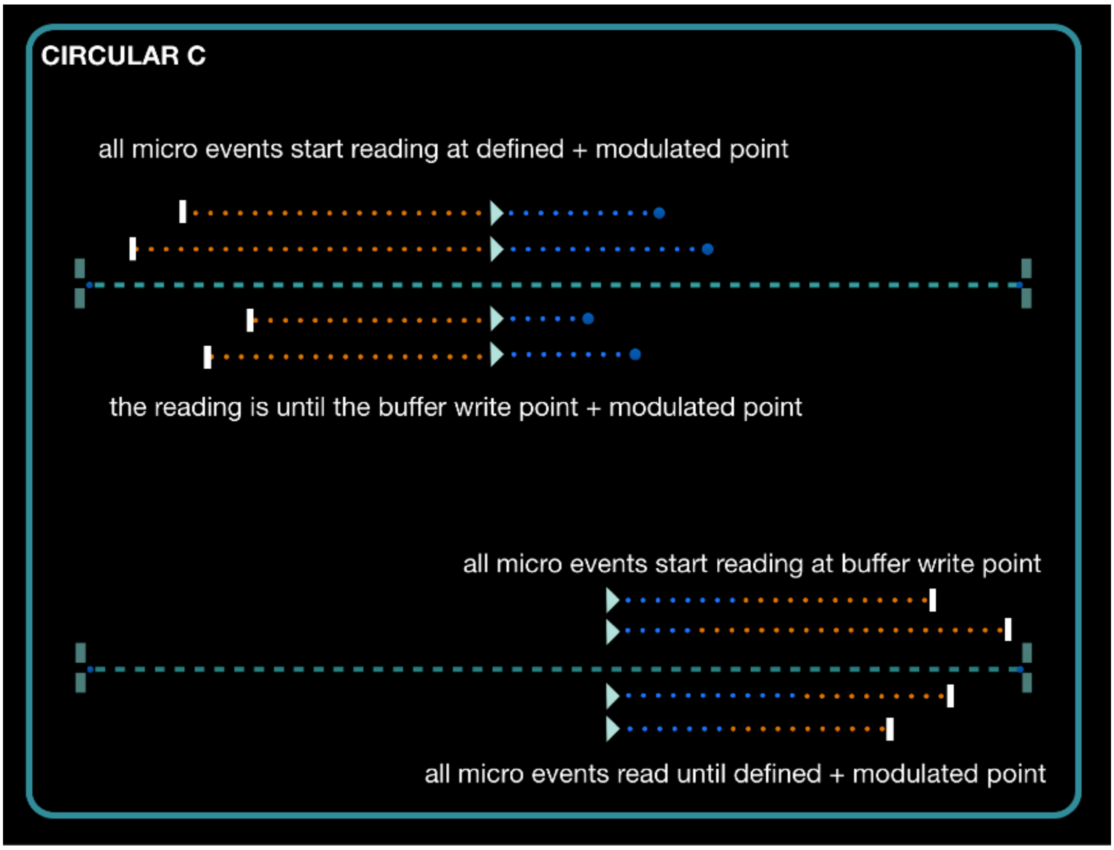
The read start pointer of the micro events move together with the buffer read/write pointer of the recording buffer.

The read stop pointer of the micro events move also together with the buffer read/write pointer of the recording buffer but the value set as the *bufferL slider* will be added to that. Also the modulations sources can modulate this added value set by the *bufferL slider*.

Therefore the buffer chunk read by the micro event is not a fixed recorded part of the buffer but a moving one along with the read/write pointer !.

Each of the 12 partials will share this same buffer chunk to play, however they can be given independent playback rates and amplitudes values. The figure above shows these values as vertical bars. The top row shows the amplitude values and the bottom row shows the playback rate coefficients for each partial.

If the **CircularC mode** on the synthesis engine is selected, each micro event samples the incoming audio into its buffer with unique start and end points defined for this mode. Let's quickly explain this:



all micro events start reading at defined + modulated point

the reading is until the buffer write point + modulated point

all micro events start reading at buffer write point

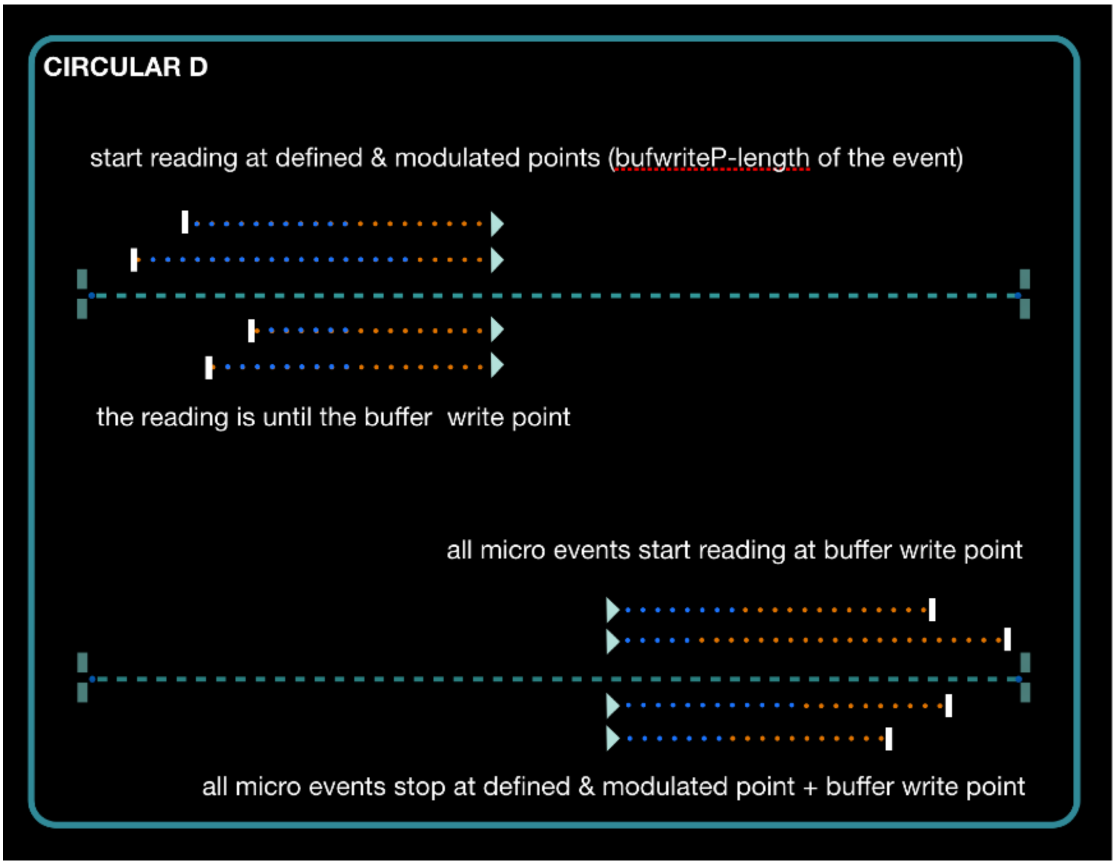
all micro events read until defined + modulated point

Assuming the *startP & endP mod* switch is off, the read start point is defined again with the *BufferL slider*. The read end point is a moving one together with the buffer read/write pointer of the record buffer and also a modulation source value can be added to that.

On the upper part of the figure above, you see the different starting points for each micro event as they can be set individually with the *bufferL slider*. The triangle shows the buffer read/stop pointer which does move and loop inside the record buffer continuously. The blue dotted lines are the added part by the modulations sources.

When the *startP & endP mod* switch is on, the behaviour is reverse. The buffer read start point for each micro event is now the buffer read/write pointer of the record buffer and also a modulation source value added. The read stop point is defined by the value of the *BufferL slider* respectively.

The last available mode is the **CircularD mode** on the synthesis engine. Let's quickly explain this:


CIRCULAR D
start reading at defined & modulated points (bufwriteP-length of the event)
the reading is until the buffer write point
all micro events start reading at buffer write point
all micro events stop at defined & modulated point + buffer write point

When the *startP & endP mod* switch is off, the reading start point for each micro event is calculated like following. The micro event duration will be subtracted from the moving buffer read/write pointer and along with an additional amount accordingly to the modulation source selected, which is shown with blue dot line at the upper part of the figure above. The read end point for the micro events is the moving buffer read/write point itself. The *BufferL slider* only sets a top limit for the start reading point.

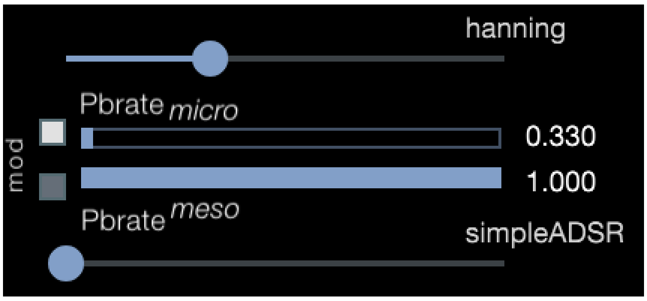
When the *startP & endP mod* switch is on, the reading start point for each micro event becomes the moving buffer read/write point itself. The reading end point is the *bufferL slider* point subtracted by the micro event duration and then modulation applied on and this final value is added to the moving buffer read/write point.

All these different mode operations are hard to remember and follow just by reading these lines, however you will get acquinted with them easily by trying the different modes and hearing their distinguished behaviour.

NOTE : All these parameters can be changed in real time but only the *RecBufferL slider* and *BufferL slider* value can be modulated within the *morphing engine*. *The BufferL slider* value can be set individually for each micro event as mentioned before.

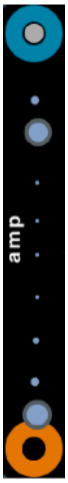
After assigning their sonic source to the micro-events, Cosmos*f* further calculates the waveform as following;

The **Pbrate micro** slider sets the playback rate of the sample or the pitch of the assigned synth oscillators. This Playback Rate concerns the final data playback rate which now is assigned to the micro events.



There are two horizontal **Pbrate** sliders, one for micro-event playback setting, one for meso-event playback rate setting located at the middle part of the Synth-Engine panel. The **button** switches located to the left of each slider activate the pitch modulation respectively coming from the LFO’s or LineGen modulation functions.

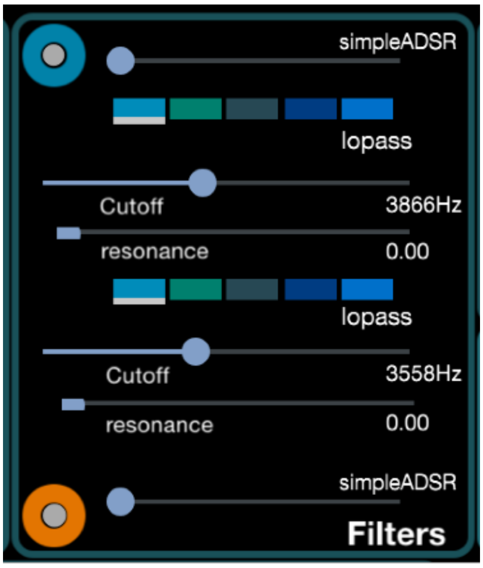
The point sliders above and below the PBrate sliders will set the windowing function applied to the calculated waveform by the Synthesis-Engine. These can be *simpleADSR*(the attack & release durations are fixed to %5 of the micro & meso event duration); simpleAR, *triangle, hanning, customEnv, blackmanHarris, exponential attack and decay.* window shapes.



Cosmos*f* then takes the value of **amp** vertical slider to set the micro/meso event amplitude intensity. There are two vertical sliders, one for micro-event amplitude settings, one for meso-event amplitude setting located at the left part of the Synth-Engine panel. To alter it you need to push the small blue circles up&down (0-1). When they are located closest to the bigger blue and orange circles, their setting is at maximum.

The blue and orange circles have button switches, which enable the amplitude modulation respectively to take effect from the LFO’s or LineGen modulation functions.

Beware that a digital clip has no mercy, though it can happen very fast and your ear won’t hear it, it can damage the following DSP chain of the application such as a bad seed.



simpleADSR

lopass

Cutoff 3866Hz

resonance 0.00

lopass

Cutoff 3558Hz

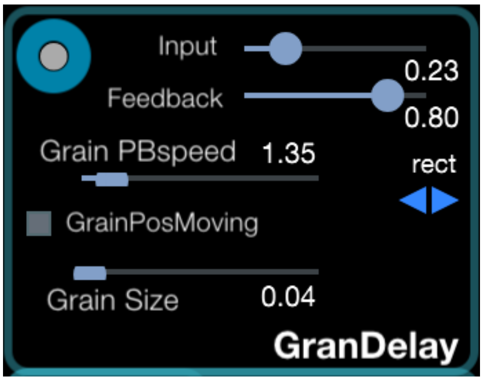
resonance 0.00

simpleADSR

As the next step, CosmosF calculates the filter processing on the waveform with the settings located on the Filters control panel. The upper part is dedicated to the micro-event filters, and bottom part to the meso-event filters. Again the big blue and orange circles have button switches to turn on the micro/meso event filters respectively,

There are four filter types available; *LoPass, HiPass, Notch, Peak, Formant and Comb Filter* types. Their **cutoff** and **resonance** values can be set with the respective sliders on the panel.

In the *Formant Filter* mode, you can continuously morph between formants by changing the cutoff slider. The formant positions are shown on the user interface when you turn on the formant filter mode. The resonance slider still helps you alter the formant filter quality. For best results, a pulse or sawtooth like waveform could be preferred to use with the *Formant Filter*.



Input 0.23

Feedback 0.80

Grain PBspeed 1.35

rect

☐ GrainPosMoving

Grain Size 0.04

The cutoff parameter can be modulated with LFO's and LineGen's. To enable the filter processing, you have to activate the button switch on the blue and yellow circles assigned to the micro-event and meso-event filter modulations.

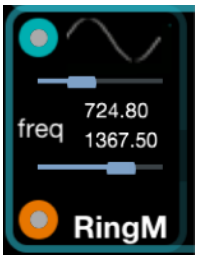
You can apply envelope functions to the micro and meso event filters by setting the sliders on top and bottom part of the *Filters* section. The available ones are; *simpleAR, triangle, hanning, customEnv, blackmanHarris, exponiental Attack and Decay.*

The next signal processing tool on Cosmos*f* synth-engine is the **GranDelay.** It is a granular delay taking its input from the micro-event audio data. It can be activated with the button switch on the blue circle, and is only operating on the micro-events.

The **Input** slider sets the amount of the signal input to the granular delay. It works like a *Wet/Dry* slider. Additionally there is also the feedback signal coming back from the output of the delay routine. The amount of the feedback is set by the **Feedback** slider.

The GranDelay fills its buffer with the audio data coming from the micro-event; the buffer size is set with the **Grain Size** slider as a ratio to the micro-event duration. The **Grain PBspeed** slider sets the loop playback speed of the audio data stored in the delay buffer. The **GrainPosMoving** button switch activates the movement of the delay buffer start point relative inside the micro-event. The grain position is moving from the beginning of the micro event to the end. The **Grain PBspeed** can be modulated by LFO's and LineGen's. To enable the GranDelay processing, you have to activate the button switch on the blue circle.

By using the extreme values on the **GranDelay** section, you can capture interesting digital audio artifacts, glitches, many creative processing results. You can also apply windowing function to each grain with the left-right buttons labeled as "*win*". Available windows are *rectange, triangle, cosine, blackman-harris* and *pulse*.

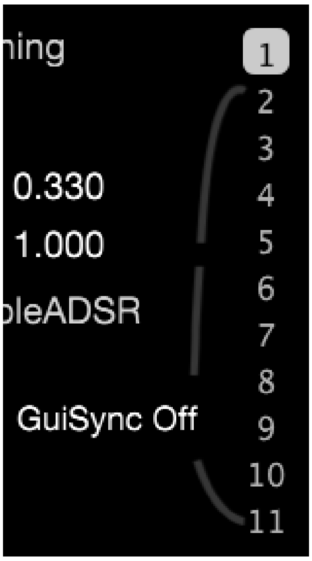


The last signal processing tool on Cosmos*f* synth-engine is the **RingM.** The Ring Modulation is a processing where a sinusoidal waveform with adjustable frequency will be multiplied with the incoming audio data. This gives a spectrum with all additions and substractions of the ring modulation frequency and all the partial frequencies in the incoming audio data. It is a classic processing method with origins of electronic analog equipment.

In Cosmos*f* the ring modulation can be applied on micro level and meso level independently. You can activate the processing on each level with the switch buttons on the panel. Then you can set the modulation frequency with the respective sliders for the micro and meso domain processing.

This ring modulation frequency for the micro and meso level can be modulated also with LFO and LineGen modulators. The first LFO's are responsable for modulating the RingM frequency, and the 6th LineGen for the micro and the 5th LineGen for the meso level operation is responsable for modulating the frequency.

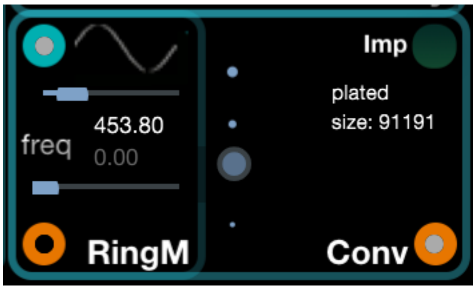
With the arrival of Cosmos*f* Plugin V2, a new potential of sonic construction has been introduced. Namely, you can edit the synthesis parameters of each micro and meso event independently. Each event can have its own synthesis engine, own sample data etc..existing Cosmos*f* users will instantly grasp the potential behind this.



Just click on the event index number of the micro or meso event you would like to edit, and the user interface will change itself according to the synth engine parameter settings of the selected event.

By turning the "**GuiSync**" switch on and off, you can choose whether the applied change on the user interface is applied on all the events (micro/meso) or not.

When you send parameter editing to Cosmos*f* remotely (DAW automation or OSC) then first you have select which event you would like to address and then send messages to its parameter space.

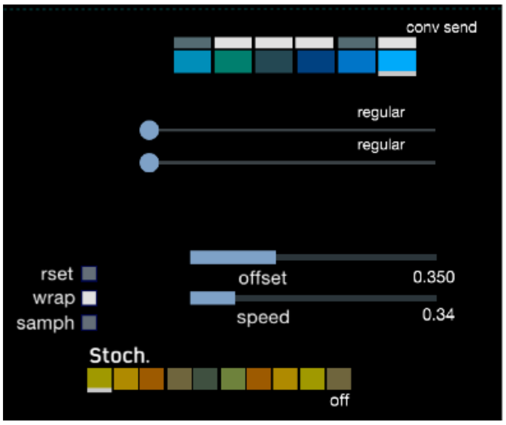


The section right to the RingModulation section is the convolution engine of Cosmos*f*. The convolution is a digital signal process where an impulse response is convolved with the audio signal and the audio signal gets the characteristics of the system which responses to the impulse. For instance if the impulse response is captured of a reverberant space then the audio signal is projected to this reverberant environment as a result of the convolution process. You can create impulse responses of very complex processes (if their nature is time invariant) and use here.

You can import impulse response by pressing the “Imp” button and there are some examples already in the “*Impulses*” directory of Cosmos*f*. At the moment mono and stereo impulse response files are accepted (*wav/aiff*) The impulse response file name and size appears on respectively below the button.

The longer the impulse response , the more CPU is needed to calculate the convolution in real time.

Cosmos*f* uses the meso event outputs and feeds the convolution engine with them. There is a switch button which activates the convolution on the orange circle as usual. Each meso event can have a signal send value, which is calculated with the meso event **LineGen6** modulator (*modulation screen of Cosmosf*)



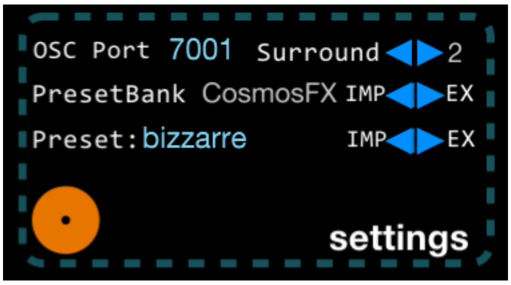
offset 0.350

speed 0.34

Likewise there can be a stochastic mix of convolution signal send of meso event audio, putting the elements at various degrees of processing.

You have to activate and set the parameters of the **LineGen6** for meso events to send meso event signals to the convolution engine.

The vertical slider in the convolution engine section is a master dry/wet parameter which balances the dry meso events output with the convolution engine process. All these settings will be saved along with the preset as usual.



OSC Port 7001 Surround 2

PresetBank CosmosFX IMP EX

Preset: bizzarre IMP EX

On the **settings** panel, you can do some organisational settings for Cosmos*f*. For instance, on top you can see the unique OSC receive port number of the current Cosmos*f* plugin instance. You can change it by clicking on it and editing it, however if you type a port number which is being used already by another task, then the text will appear dark blue.

A valid port number looks in light blue color.

Next to it, there is the Surround channels setting defining the output surround configuration of Cosmos*f*. This will be explained in detail in the *Surround sound* section.

Below this, on the **PresetBank** slot you can load a preset bank or save your current presets as a bank to disk. Next to the **PresetBank** slot, the **CurPreset** slot is located, where you can select a preset to import. Also you can export your edited presets to one of the 64 preset slots. To type a name for each preset just click below the name field of the preset.

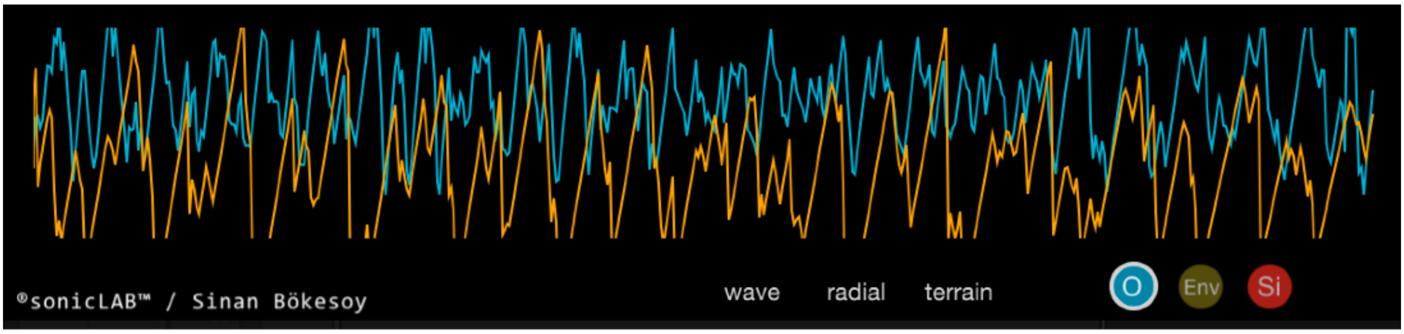
The next issue to talk about this panel is the *preset import/export* in Cosmos*f*. You can save the whole parameter space of Cosmos*f* into a preset slot if you press the **EX** label, where you will see all the presets listed on the upper part of the screen. Just click on a slot you wish to override, which also saves to the disk the total preset list. If you click on **EX** without selecting a slot the list disappears. If you press on the **IMP** label, you will be ready to import a preset from the preset list showing up on the upper part of the screen. If you decide not to do, just press again on the **IMP** label.

You can rename a preset when you click on its title and edit the text field as usual. You can to export it to a preset slot to keep the name change.

In the demo version of Cosmosf, preset saving is disabled.

the visualization options :

Please notice the visual interface panel at left bottom of the Cosmos*f* primary screen.

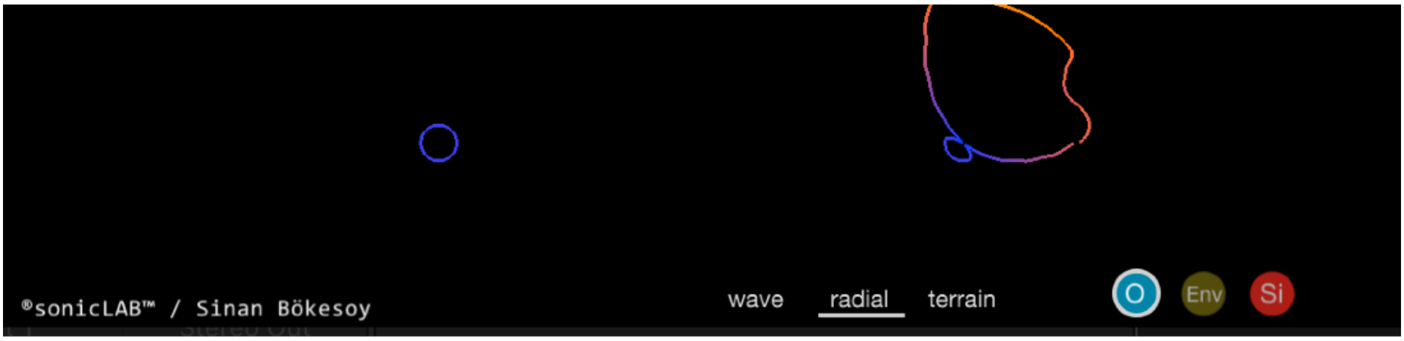


This panel represents three modes labeled with “**O**” , “**Env**” and “**Si**”. When you select “**O**”, the graph shows the micro event and meso event waveform in different formats. When you select the “***wave***” mode , the format is like an oscilloscope. The blue waveform graph is micro and orange one is for meso events.

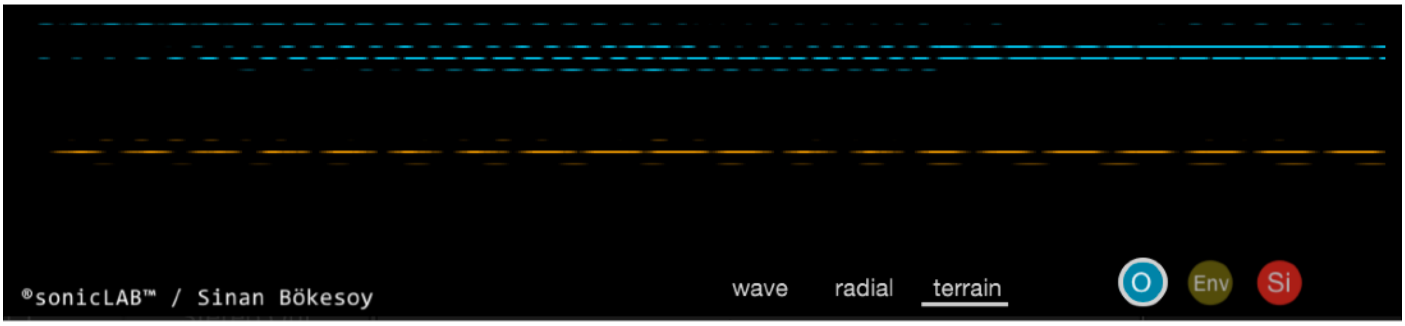
If you are in “GuiSync off” mode, then the graph shows the selected micro and meso event outputs, and not the sum of all the events.

If you select the “***radial***” mode , it shows you the RMS intensity scale of the meso event outputs altogether on a circular waveform. The meso events RMS output will be distributed on the circle and the color intensity shows the strength of the signal output. The right circle is dedicated to the parallel cycle meso events.

From a palette of the dark violet to yellow color output (weak — strong), you can have a glimpse on the Cosmos*f* output levels.

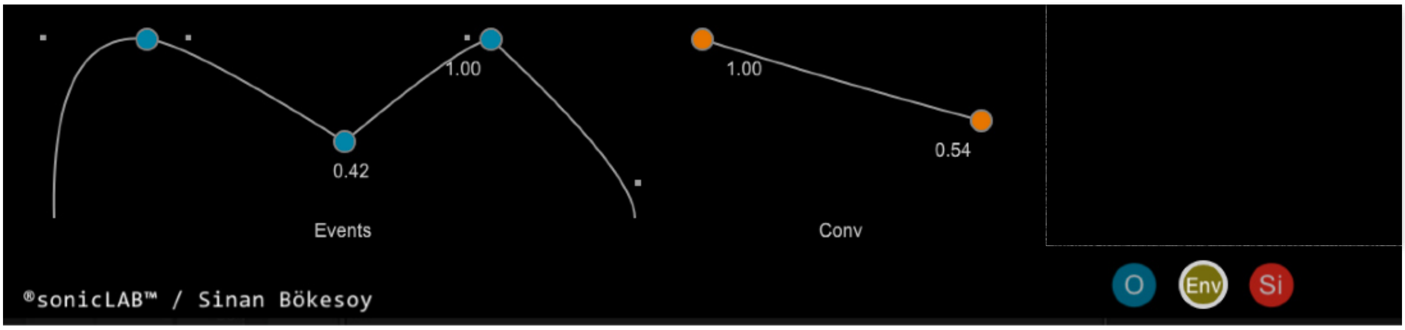


If you select the “***terrain***” mode , the graph will show you another variation of intensity projection. Here you will see each micro and meso event on single lines and the color brightness shows the local intensity of their output

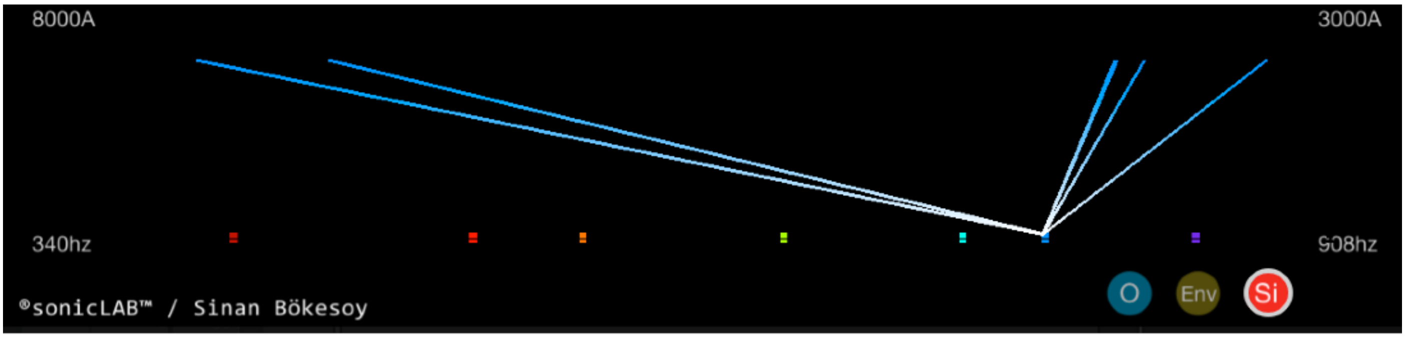


When you select the “**Env**” button on the panel, you will see the custom envelope editor of Cosmos*f*. The left envelope is a multi section bezier curve editor, which evidently resembles to an ADSR envelope of a classic synthesizer paradigm. You can shape this envelope as you wish and define the curvature of the lines by dragging the little bezier control rectangles on the editor. This custom envelope can be used for amplitude envelope and also filter cutoff envelope on micro and meso events.

Please note that if the attack or release time is set zero on the envelope, the audio might click if the first and last calculated sample of the micro or meso events are not zero. So consider leaving a minimum amount of attack / release time when using this envelope.

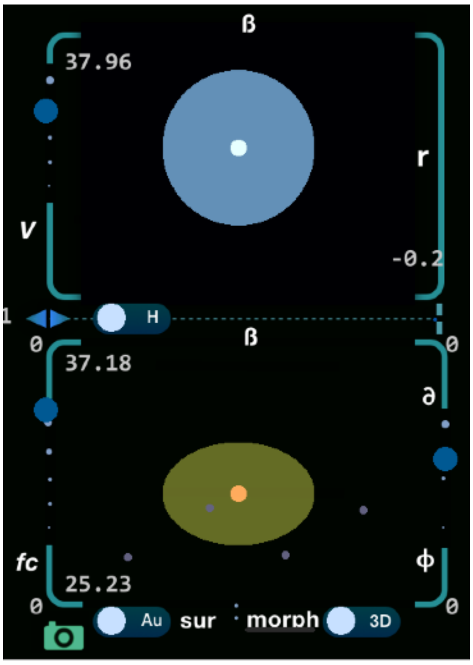


The envelope editor in the middle part is dedicated the envelope the impulse response used by the convolution engine. You can create various additional space shaping by using this envelope on the impulse response.

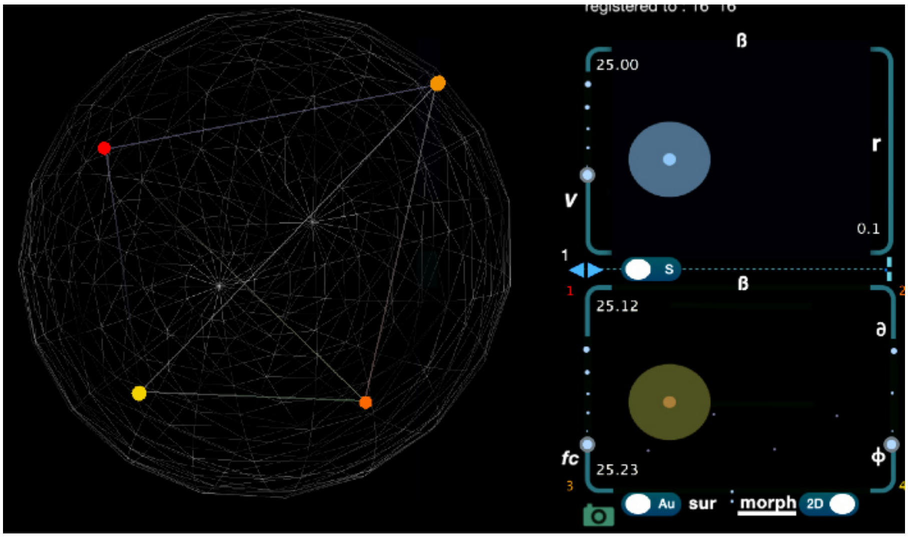


When you select the “**Si**” button on the panel, you will see the Sieve engine visualisation. The upper part is dedicated to the raw calculated values of the relevant Sieve input (pitch, amplitude, filter cutoff frequency, granular pitch or Ring Mod frequency)

The bottom part shows the discrete frequency values depending on the selected scale representing the Sieve output. So each active micro/meso event is shown as a line beginning at the upper part and ending at the sieved point / bottom part.



the morphing mechanism...



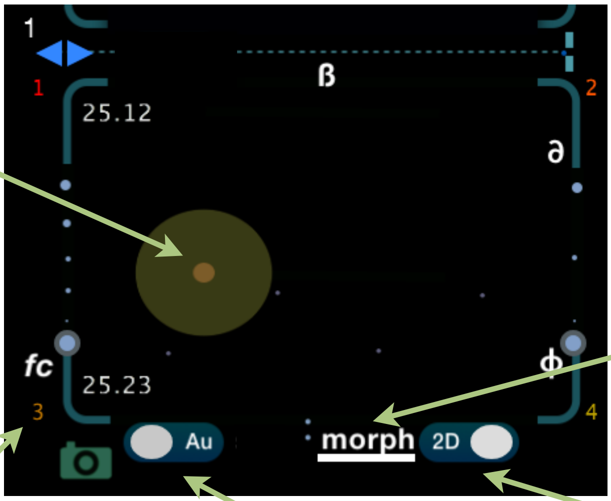
PRESET MORHPING

Cosmos*f* has many parameters and it is very difficult to introduce a gradual change to multiple parameters at once without the help of a higher level tool. Cosmos*f* introduces a unique **preset morphing** tool, which lets you morph between the 4 selected presets within various schemes and parameters. A sonic morphing process defines a metaphor of exploring the sonic shape and timbral evolution on intermediate levels between two or more sonic materials. There is no robust methodology to bring exact solutions in morphing processes however they offer a rich terrain of possibilities which could be handled within various approaches to bring interesting perceptual phenomena.

By manipulating the **morphing tool** the individual composants of Cosmos*f* will exhibit a process of change gradually in timbre space exhibiting a dynamic morphology. By communicating directly with the Cosmos*f* synthesis model at the level of a bottom-up sonic construction process, and observing the process of change of the individual composants, the result will deliver the necessary correspondence and smoothness in gradual transformations.

Cosmos*f* handles all the parameters of a *preset* as a vector in its parameter space. Morphing between presets means simply altering this vector by altering the distance from each preset and calculate the weighted parameter space as the current morphing state of the application.

Cosmos*f* has a 2D and a 3D morphing mechanism, where the 4 selected presets are defined as geometrical objects. On the 2D morphing mechanism, each preset is represented on the corner of the panel as below.



morphing pointer

morph activation switch

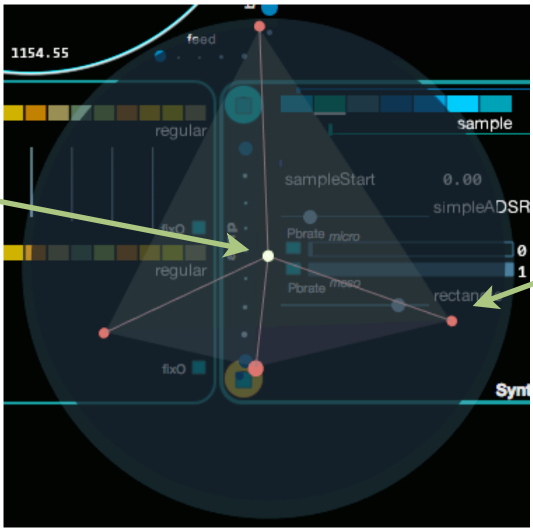
switch for 2D/3D morphing/ambisonic surround mechanism

selected preset number at the corner; you can click on the camera for snapshot of the current Cosmosƒ state.

switch for 2D manual or automatic distribution in 3D mode

The red ball, **"the mophing pointer"**, can be dragged with the mouse and positioned anywhere on the rectangle 2D surface. Cosmosƒ calculates the distance of the morphing pointer from each corner and applies an euclidian distance weightening operation to morph the current parameter space between the 4 selected presets for the 2D morphing mechanism.

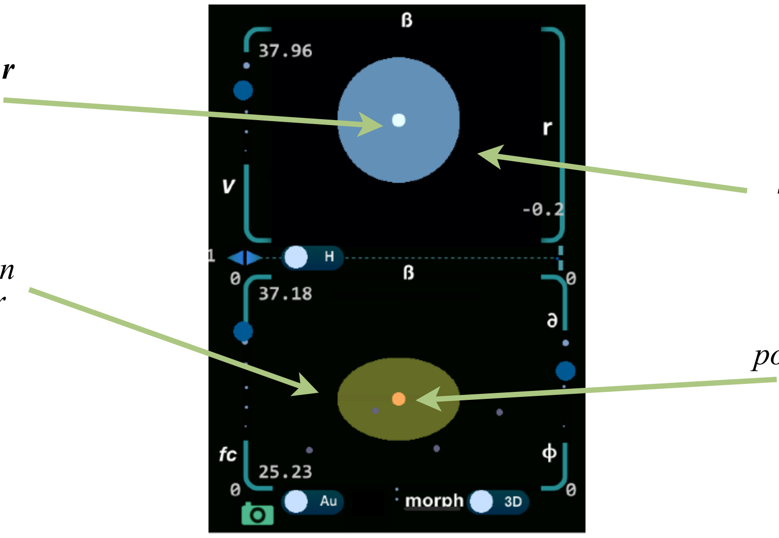
The **3D morphing** mode is a novel approach in preset morphing. It does represent each one of the 4 selected presets as a corner of a tetrahadron fitting inside a sphere in 3D space. You can also use the snapshot mode to assign instantly the current state of Cosmosƒ to any corner slot by pressing on them. Likewise the morphing space will be altered according to your preferences.



morphing pointer in 3D space

presets position in 3D space

To move the morphing pointer in 3D space Cosmos*f* let's you manipulate the ***phi*** and the ***theta*** angle as the 2 spherical coordinates and also the ***r*** angle. For this you have to use 2 control surfaces with their own morphing pointer. The ***theta*** angle is a common coordinate setting in these 2 control surfaces as shown on the figure below.



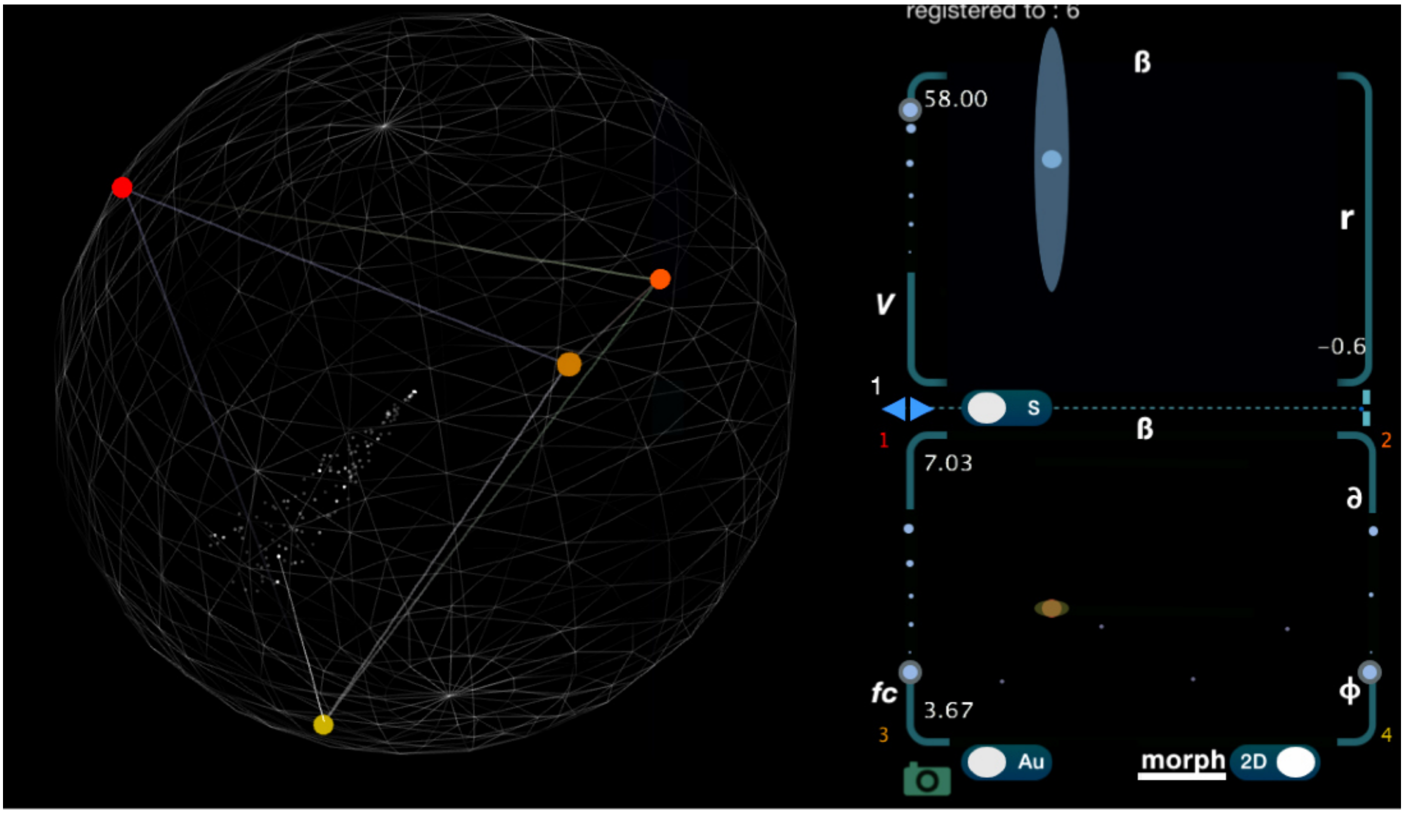
*pointer to set the **r** and **theta** angle*

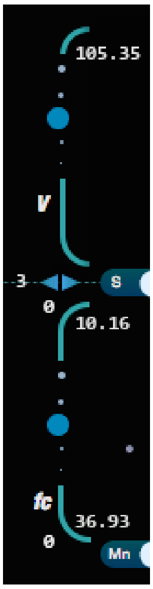
ellipse defining the stochastic distribution range for the pointer

ellipse defining the stochastic distribution range for the pointer

*pointer to set the **phi** and **theta** angle*

The ellipses define the random distribution range of the pointer for the relevant spherical coordinates. Their size can be manipulated by holding the '***r***' button and clicking on the morphing pointer of the relevant surface panel. The numbers indicate the distribution range for the respective coordinate axis. Below, two examples are shown using the stochastic distribution for the morphing pointers. On the left, a total random distribution in 3D space is achieved. As shown on the right, by limiting the distribution ellipse ranges, unique morphing spaces can be created.

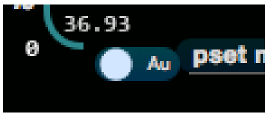




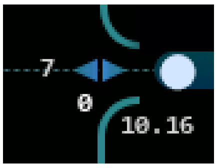
- The speed of the stochastic distribution on the morphing pointers are independent of the Cosmos*f* cell length settings. It can be controlled with the ***v*** slider, which is to the left of the upper control surface. The upmost position setting is the fastest.

- The movement of the morphing pointer can be filtered with linear interpolation. The degree of this can be set with the ***fc*** slider. At the top position there is no filtering applied. Trying different setting with the speed parameter, you will experience interesting transition of the morphing process.

Cosmos*f* offers also an automated morphing mode, where the morph pointer is being altered by mathematical functions.



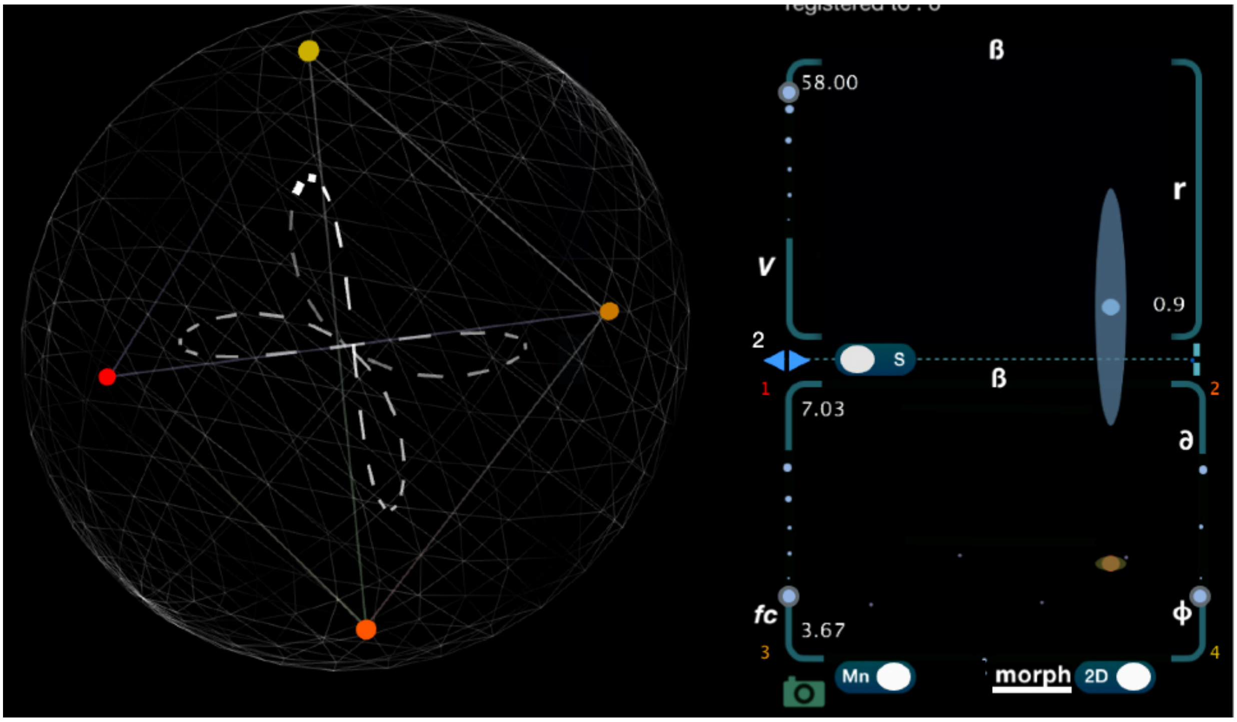
You can activate this mode with the switch shown on the figure, by choosing the “Au” position.



The type of the mathematical function controlling the morphing pointer can be set with these increment/decrement buttons.

The range (e.g the size) of the function graph can be set with morphing pointer of the upper control surface as such you would edit the ***r*** and ***theta*** coordinates. Some mathematical functions use both of them for parameter exchange.

Also the speed and the movement filter can be set with the relevant sliders when in the automatic morphing mode directly effecting the behaviour of the mathematical function.



All the morphing parameters can be saved along with a chosen preset destination. Also the morphing pointers, ellipses sizes and the speed & interpolation slider can be controlled remotely via OSC. However as soon as you activate the morphing process, the current preset settings are irrelevant since Cosmos*f* calculates its parameter space according to the distance of the morphing pointer from the selected 4 corner presets.

A quick try for the morphing feature would be having a look at the presets.The morphing mechanism turns Cosmos*f* into a super oscillator however it is highly sensitive:

- After loading any preset, you have to activate the preset morphing by clicking on the *morph activation switch.*

- You can assign any preset for the morphing process by clicking on the preset numbers around the panel.

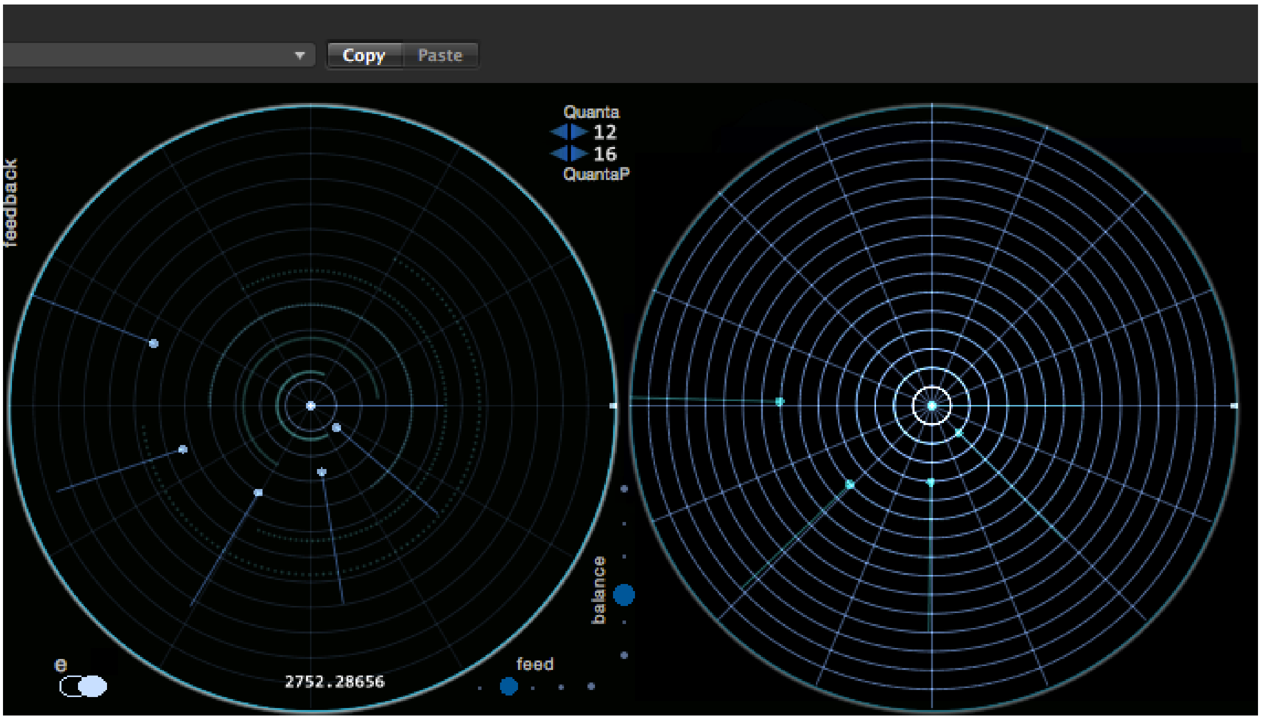
- All the morphing parameters are saved along with the preset and they can be also controlled by OSC messages.

- For smooth and gradual morphing states, respect the DSP section differences of the used presets in the morphing process. For instance, if one preset has *GranDelay* applied and the other has not, then there can be an abrupt change in the audio fortunately.

Quantization Feature on Cosmos*f* Cycles :

- Cosmos*f* can distribute the onset time and duration info for micro events and meso events in complex ways. However there can be times that you might want to quantize the onset time of each event, just like on a midi sequencer software. With the Plugin version of Cosmos*f*, users can assign independent quantization setting for each Cosmos*f* universe in order to create unique results and poly-rithmic structures.

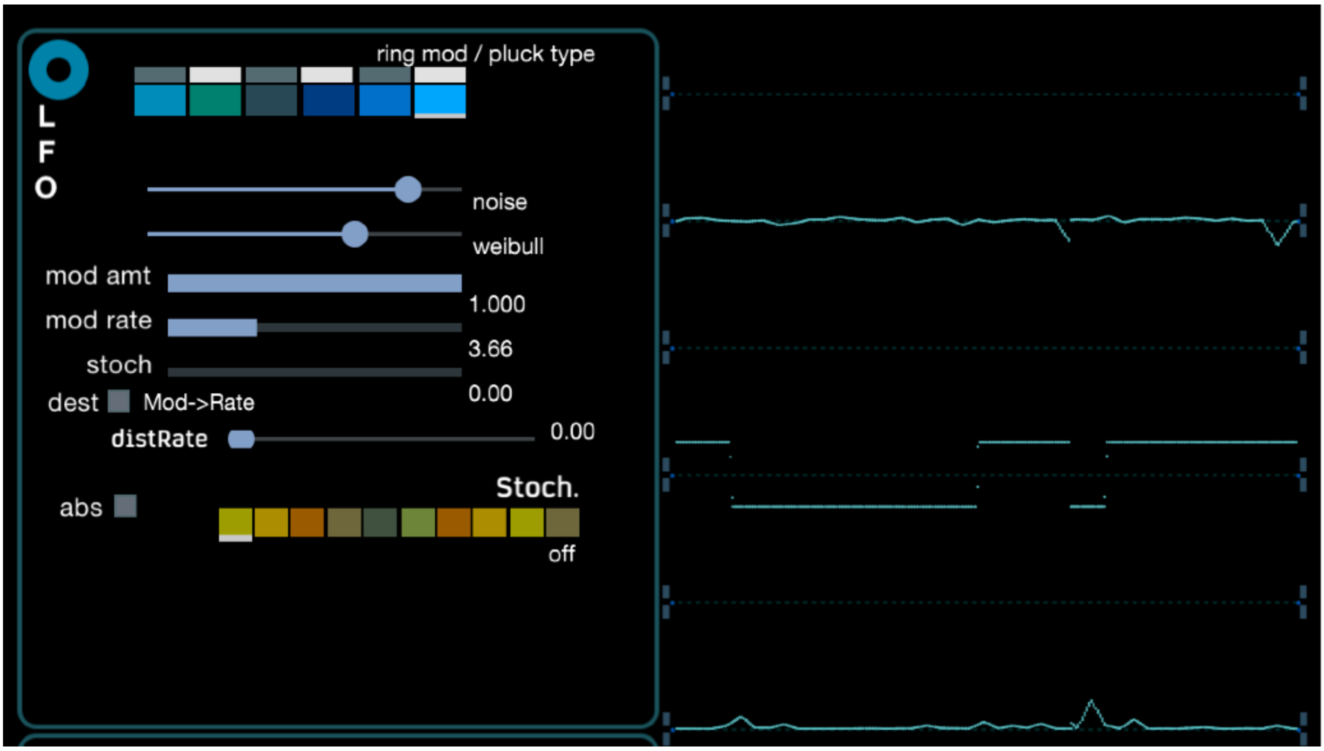
- As you see on the figure below, the **Quanta** parameter fields let you assign your quantization scheme for each universe. For short time, a spiderweb will appear to show you the 2D quantization of micro and meso events for the selected parameter setting.



- The assigned quantization parameter is for that selected meso event index and the micro events distributed in itself. Likewise you can give different quantization setting to each meso event and the micro space it is encapsulating.

- One phenomena you would witness during this process is, the overlapping of quantized events and their amplitude accumulation. Enough amount of events overlapped and sync-ed on the same onset time point, can create excessive amplitude bursts. The quantization is in audio rate resolution.

LFOs



Cosmos*f* has 6 LFO’s running on micro-event synthesis destinations and 5 LFO’s running on meso-event synthesis destinations. You can use the blue colored bars at the top part of the LFO panels to choose the LFO destination for the relative events. To activate the LFO you have to click on the relevant switch located at the top of each LFO destination bar. Also you have to activate the modulation enable switch for the relevant destination. The text bar at the bottom of the screen gives you instant information about the current enabled modulation destinations.

The LFO destinations for the micro-events are;

LFO Amp : Modulates the amplitude of all micro events simultaneously with the assigned modulation parameters. If activated, it does also modulate the ring modulation frequency for the micro events.

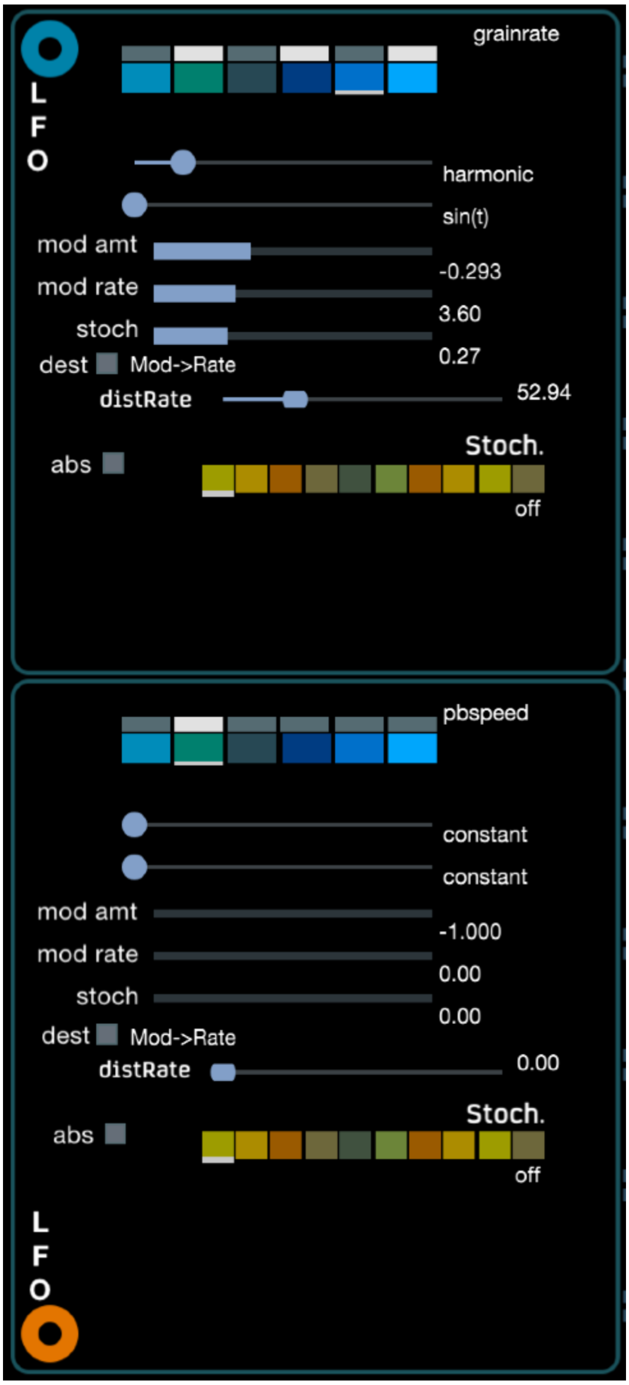
LFO PBspeed : Modulates the playback speed of all micro events simultaneously with the assigned modulation parameters.

LFO Buffermod: This LFO serves for two purposes. This LFO modulates the startP & endP points of all micro events simultaneously based on the BufferL slider value with the assigned modulation parameters.

LFO FilterCut : Modulates the filter cutoff frequency of all micro events simultaneously with the assigned modulation parameters.

LFO GrainRate : Modulates the granular delay buffer playback speed of all micro events simultaneously with the assigned modulation parameters.

LFO Ring Mod : Modulates simultaneously the ring mod frequency (when active).



grainrate
harmonic
sin(t)
mod amt -0.293
mod rate 3.60
stoch 0.27
dest Mod->Rate
distRate 52.94
abs
off

pbspeed
constant
constant
mod amt -1.000
mod rate 0.00
stoch 0.00
dest Mod->Rate
distRate 0.00
abs
off

The LFO destinations for the meso-events are;

LFOm Amp: Modulates the amplitude of all meso events simultaneously with the assigned modulation parameters.

LFOm PBspeed : Modulates the playback speed of all meso events simultaneously with the assigned modulation parameters.

LFOm Panning : Modulates the panning in the stereo field of all meso events simultaneously with the assigned modulation parameters. In Cosmos*f*, micro-events are mono 1ch audio carriers and the meso events are stereo 2 ch audio carriers. Therefore the output of Cosmos*f* becomes stereo.

LFOm FilterCut : Modulates the filter cutoff frequency of all meso events simultaneously with the assigned modulation parameters.

LFOm Ringmod : If activated, it does modulate the ring modulation frequency for the meso events.

LFOm Conv send : If activated, it does modulate the amount of the meso event audio signal sent to the convolution engine.

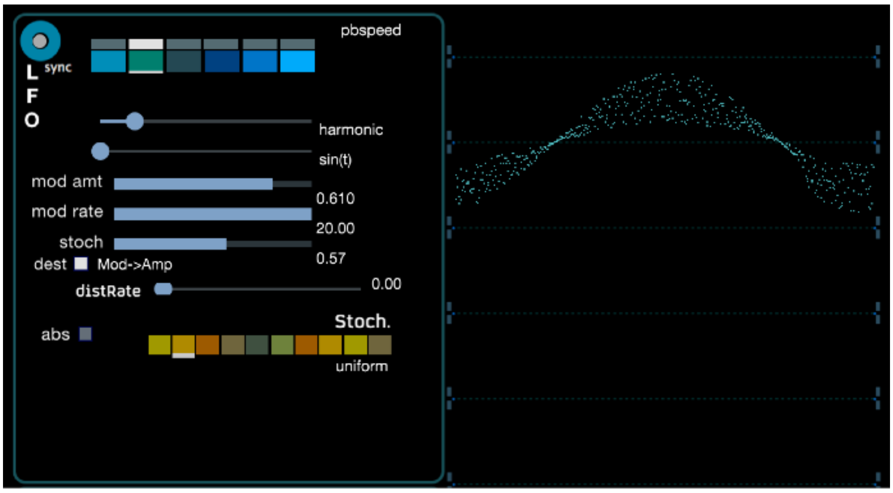
The construction of a LFO waveform is the same both for micro and meso level LFO's. First step is the selection of the category for the **mod waveform** with the point slider below the color bar. The available choices are;

constant, harmonic oscillators, ramps, square'ish waveforms, exponentials and *noise based waveforms.*

With the slider below attached, you will choose the type of waveform for each category.

The next slider is **mod amt** setting, which multiplies the waveform amplitude with the setting of this slider. It has a bipolar range. If you select *constant* as the LFO waveform, you can give an offset value to the LFO with the **mod amt** slider.

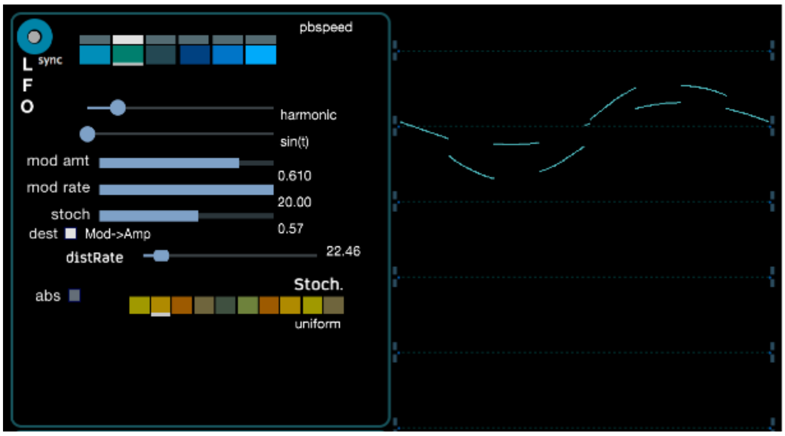
The next slider below is **mod rate** setting, which sets the speed of the LFO. You can set a stochastic function to modulate the modulation speed or the modulation amplitude. The switch for this is the blue button labeled **'dest'** on the left part of the LFO panel. The slider below the mod rate setting is the **stoch mod** setting, which sets the amount of modulation applied by the stochastic function to the LFO parameters. You can select the stochastic function and set its parameters just like the case with the event distribution process on the main screen.



harmonic
sin(t)
mod amt 0.610
mod rate 20.00
stoch 0.57
dest ☐ Mod->Amp
distRate 0.00
abs ☐
Stoch.
uniform

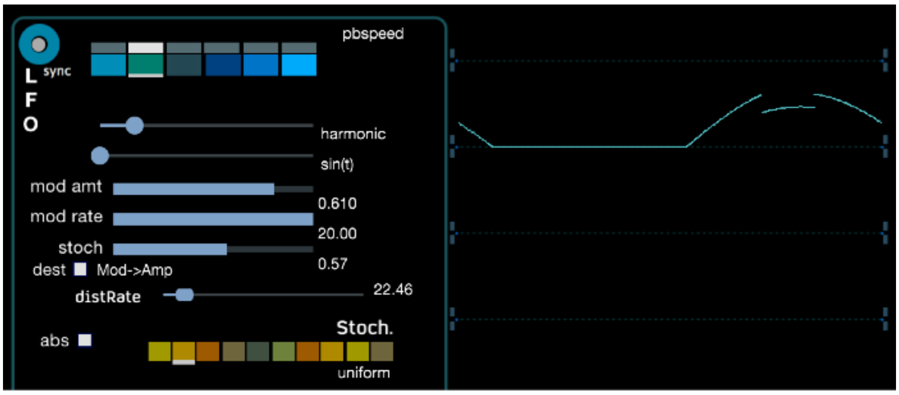
On the figure above, you see that the **LFO amp** is active and has the **sine** waveform assigned with **mod amt** setting .61 and **mod rate** value 20.00. The stochastic function is set to **uniform** distribution modulating the amplitude of the sine waveform with the **stoch mod** setting .57. You can watch each LFO waveform visualized on the right side of the panel.

The **distRate** slider is a useful setting, which slows down (downsampling) the stochastic function. The figure below is a good example, the only difference with the figure above is the **distRate** setting.



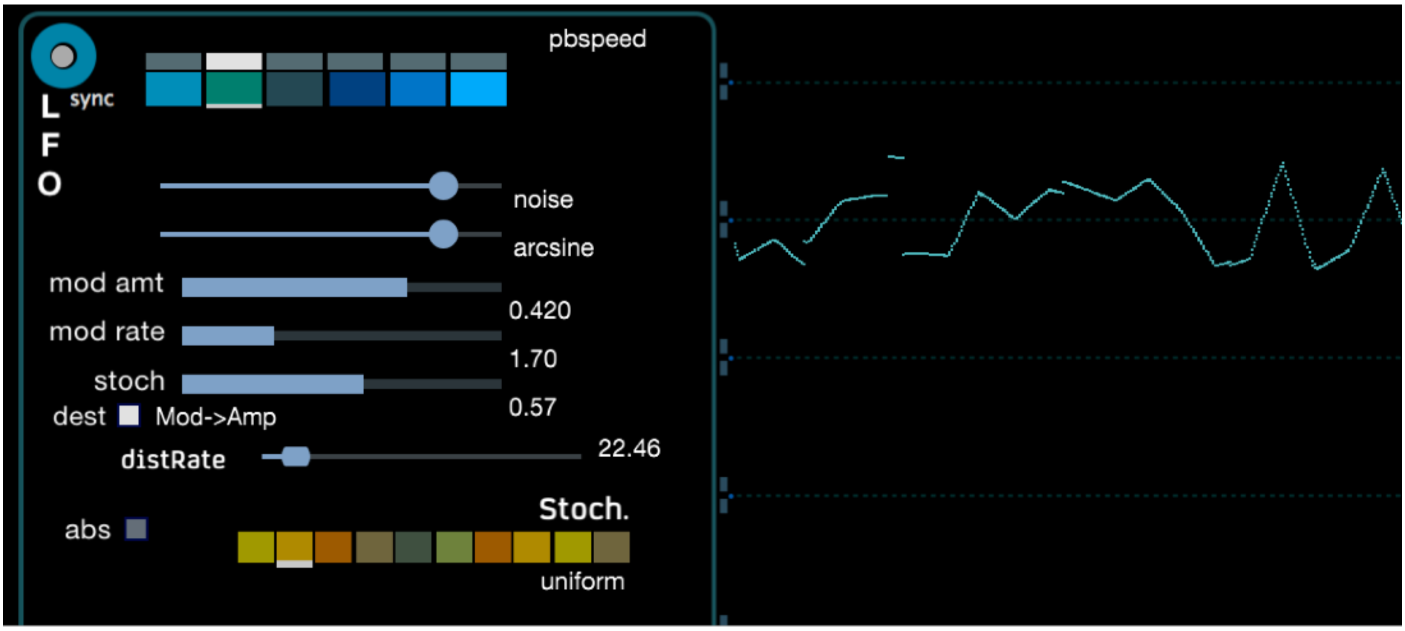
harmonic
sin(t)
mod amt 0.610
mod rate 20.00
stoch 0.57
dest ☐ Mod->Amp
distRate 22.46
abs ☐
Stoch.
uniform

The **abs** button located left to the stochastic function selection bar serves for multiplying the negative values of the waveform with -1; likewise all the negative values are mapped to the positive region. The figure below shows an example with all the settings remaining as above but the **abs** switch is turned on.



harmonic
sin(t)
mod amt 0.610
mod rate 20.00
stoch 0.57
dest ☐ Mod->Amp
distRate 22.46
abs ☐
Stoch.
uniform

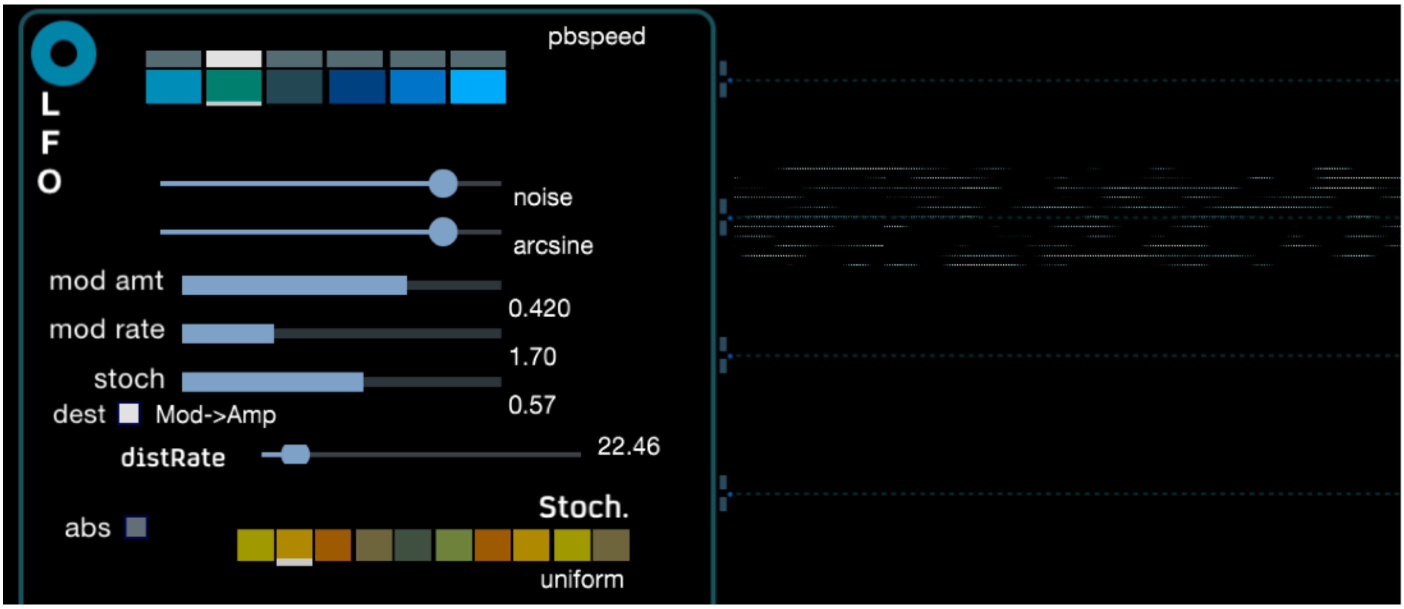
LFO sync mode : When the circle switch is turned on, the micro (or meso) events are synced, sharing the same LFO values. On the example below, the micro events share the playback speed modulation calculated as shown on the graph on the right side.



noise
arcsine
mod amt 0.420
mod rate 1.70
stoch 0.57
dest ☐ Mod->Amp
distRate 22.46
abs ☐
uniform

When the circle switch is turned off, the micro (or meso) events are not synced, so for each event an independent LFO is run but with the same settings. However if these settings involve stochastic modulations so each LFO will be affected in a different way as you would expected.

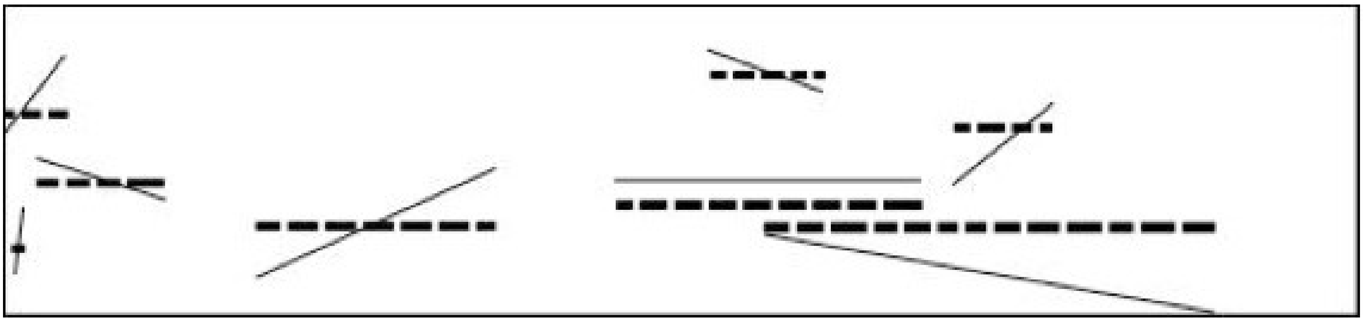
In order to visualize this better, we use the terrain view where each LFO value is show on a line and the color brightness projects the LFO value. Below you can see an example.



noise
arcsine
mod amt 0.420
mod rate 1.70
stoch 0.57
dest ☐ Mod->Amp
distRate 22.46
abs ☐
uniform

LineGENs

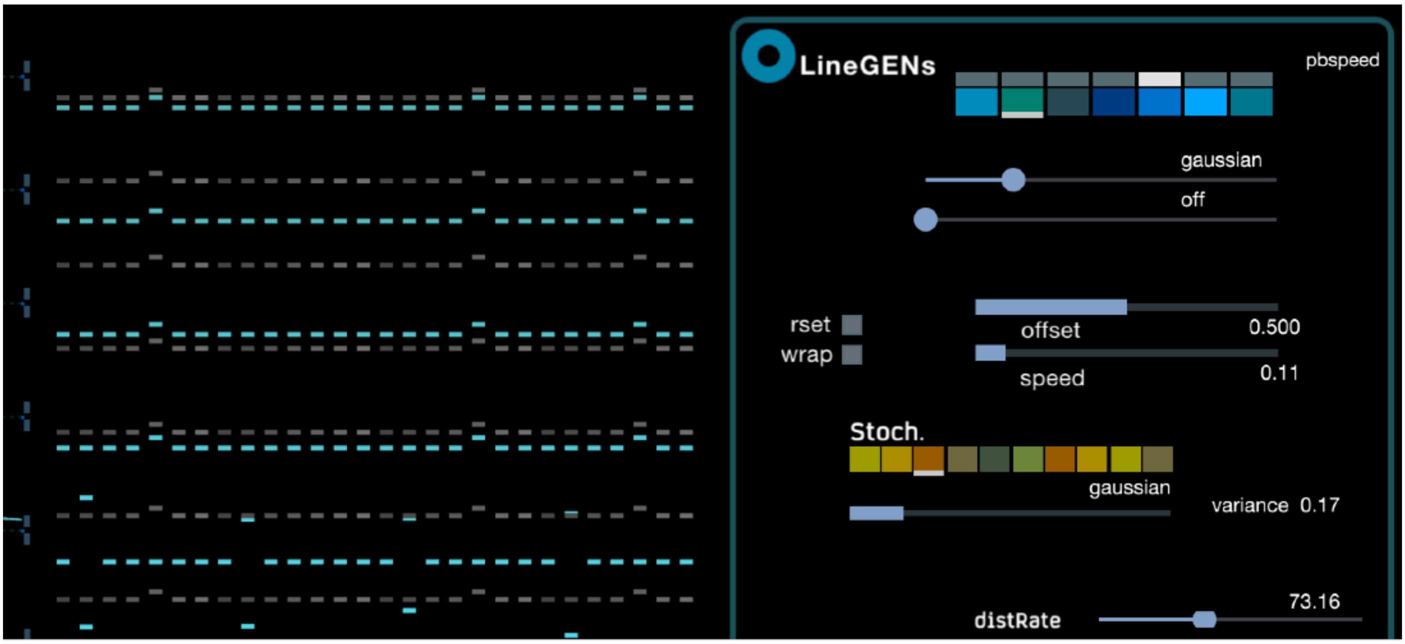
The concept of LineGenerators are reminiscent to the stochastic glissandi of I. Xenakis, which he constructed with triangular stochastic function and applied them on various instruments in his orchestrations. A basic graphical example is below, where you see different modulation curves/lines with specific departure and arrival point located on each event (dashed-lines). The departure point and the speed of this modulation was being calculated with stochastic functions to alter the pitch of the string instruments continuously while creating a rich glissandi perceived as a volume.



There is nothing stopping us to use such line generators for various synthesis parameters in Cosmos*f*. Each micro-event in Cosmos*f* can have 6 different LineGen's running simultaneously. And each meso-event can have up to 5 different LineGen's running simultaneously. The moving vertical bars on the left to the LineGENs panel visualize the current value of the line generators.

LineGENs

You can select on the blue color bar the LineGen that you would like to edit. The button switches above the color bar activate/deactivate the LineGen's. Please consider also to enable the modulation switch of your relevant synthesis parameter destination to make the LineGen active on that destination.



offset 0.500

speed 0.11

variance 0.17

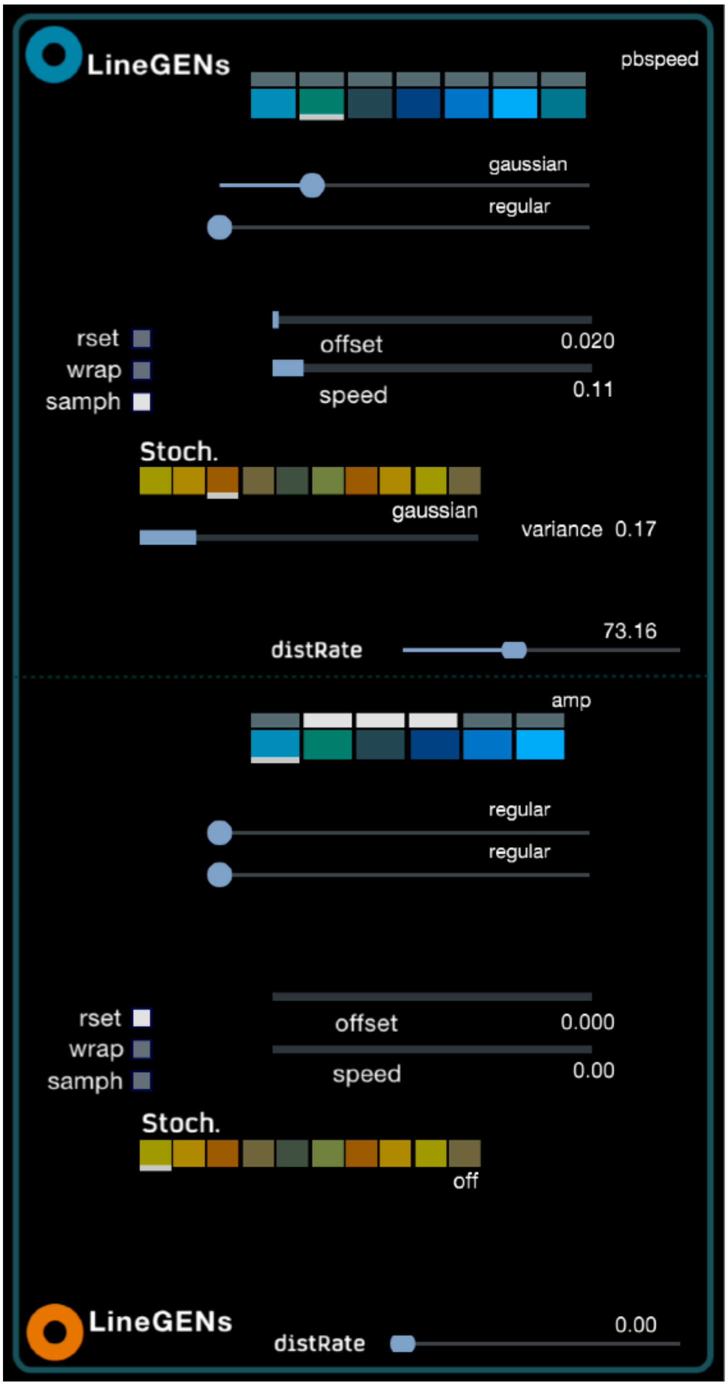
distRate 73.16

These are the following modulation destinations for the micro-event LineGen’s;

micro-event amplitude / additive yynthesis partial amp modulation, playback speed, Buffer pointer modulation, filter cutoff , granular delay grainsize, wave morph / additive synthesis partial freq modulation and ring modulation

These are the following modulation destinations for the meso-event LineGen’s;

micro-event amplitude, playback speed, panning, Filter Cutoff, and ring modulation and convolution send.



offset 0.020
speed 0.11
variance 0.17
distRate 73.16
offset 0.000
speed 0.00
distRate 0.00

A Line Generator is being constructed as following;

-It has an offset setting, where the line generator takes its departure point.

-It has a speed setting, which defines the increment speed/steepness of the line generator.

Both settings can be modulated by stochastic functions as in the case with LFO’s. The 2 point sliders below the LineGen selection color bar let you choose, which stochastic function you would like to assign to the offset and speed settings.

The relevant stochastic functions can be edited with the function selection color bar at the bottom part of the LineGen panel.

The modulation speed of the stochastic function can be altered (downsampling) with the **distRate** slider. Likewise you can apply the same type of stochastic modulation on a parameter with different speeds via the distRate slider on the LineGenerator and the LFO

The **wrap** button switch controls the behaviour of the line generators around its limit values, which are offset-1 or 0-offset. When the line generator reaches its upper or lower limit value it folds back like a mirror effect from its upper or lower limit, if the wrap button state is off. If the **wrap** state is on, then the line generators will wrap around its lower limit value creating a discontinuous jump from its upper limit to its lower limit value.

The **rset** button switch will activate the value resetting process of each LineGen at the beginning of an event. The LineGen value will start from the offset value and move on according the other settings.

The **samph** button switch will activate the classic sample/hold value generation of from a modulation source. Here with the beginning of each micro or meso event the instant value of the LineGen will be sampled and applied as a static value from the LineGen modulator during the event life span.

You can run both the LFO's and LineGen's with great speed and audio precision to obtain most interesting modulation effects on the synthesis parameters such as FM (Frequency Modulation), AM (Amplitude Modulation) , Filter modulation, PWM, interesting granular synthesis applications with the WaveStart Modulation etc...all with the intriguing combinations of stochastic modulation synthesis.

In order to clearly present the mechanism, the following list gives you the equations about how Cosmos*f* do calculate the modulations created by LineGens and LFO's on certain synthesis parameters.

For the calculation of **micro/meso event playback rate modulation**; minimum value = 0.00001;

```
PB slider setting * (LineGen value + LFO value) + PB slider setting
```

For the calculation of **micro/meso event amplitude modulation**; the value will be clamped between 0 - 1;

```
Amp slider setting * (LineGen value + LFO value) + Amp slider setting
```

For the calculation of **micro event Buffer pointer mod.**; the value will be clamped between 0 - 1;

```
BufferL slider setting * (LineGen value + LFO value) + BufferL slider setting
```

For the calculation of **micro/meso event filter cutoff modulation**; the value will be clamped between 0 - 1;

```
Filter Cutoff setting + (LineGen value + LFO value) * 220000
```

For the calculation of **micro/meso event ring mod frequency modulation**;

```
Ring mod setting * (LineGen value + LFO value) + Ring mod setting
```

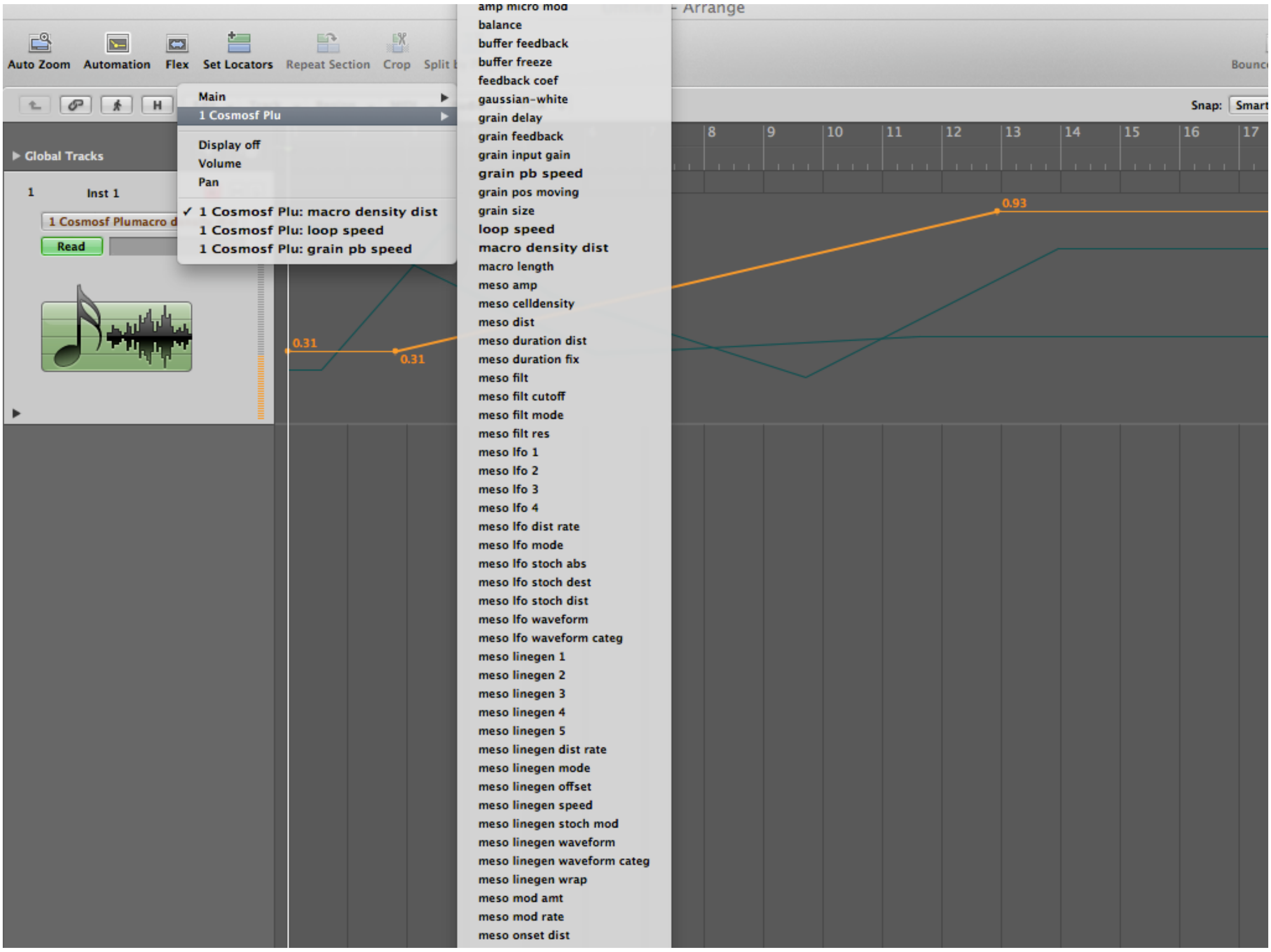
For the calculation of **meso event panning modulation**;

```
LineGenm value + LFOm value
```

automation of Cosmos*f* parameters in DAW

Cosmos*f* has many parameters which you are able to automate with the DAW, which would give you great possibilities to shape and control the outcome of Cosmos*f* in realtime and continuum on many levels.

Below you can see a Logic screen, and the parameter list of Cosmos*f* to be assigned on the relevant track. Different DAW applications have different methods for doing this. We have successfully tested on Logic, Ableton Live and Reaper.



Cosmos*f* Plugin has the ability of identify the synthesis engine for each micro or meso event independently.

To define the synthesis parameters for each unique micro or micro event, first you have to set the "**Eventindex**" on the automation window of your DAW. Then any synthesis engine parameter automation will be applied to the micro or meso event specified with this index number. This is also valid for the surround engine where you need to specify first the mesoevent number with **Eventindex** to control the spatialisation. If you set the **GuiSync** automation parameter, then all the event will synchronize with the relevant parameter change you do apply.

Also when automating the LFO and LineGen parameters, first the relevant LFO or LineGen should be activated with the automation by assigning for example *meso lfo 1 , micro lfo 3 etc.*

“Sieves” engine in Cosmos*f* FX :

I. Xenakis (1922-2001) has developed a system for creating integer-sequence generator called *sieves.* Xenakis did use sieves for the generation of pitch scales and rhythm sequences in his compositions. Sieves are calculated easily with the aid of computer using modular arithmetic. *Cosmosf* uses *sieves* to generate any pitch scale within a minimal user interface. These scales can be assigned to various synthesis parameters on Cosmos*f* such as;

	- PB rate	- Amplitude	**- Filter Cutoff**	- DelayBufferL	**- RingMode Freq**
- micro events	X	X	X	X	X
- meso events		X	X		X

Stochastic Output → **Sieves** → **scaled Output into synthesis engine**

As illustrated above, the *sieves* engine takes the raw Cosmos*f* values, coming from the UI, LFO’s or LineGen’s and filters them into the calculated musical scales by the sieves. One can think that the generated *sieve sequence* serves as gravitational field points in space where the particles passing nearby are directed towards the nearest field point. Cosmos*f* *sieves* create these special states of the sonic parameter space by altering the events emitted from Cosmos*f* engine.

Note about Cosmos*f* FX usage : All these are relative values for Cosmos*f* FX , which only processes incoming live audio input, which is the absolute reference. Cosmos*f* FX sieves engine will apply its processing with its relative parameter values. E.g. only if you play a A note as audio input, then everything fits with the existing setup of Cosmos*f* FX matching this absolute reference pitch.

Currently the following musical scales are defined and their intervals are calculated by the *sieves* engine in Cosmos*f*. They are the common western musical scales and also the exotic ones from middle east and asia. Among them the *maqam scale* has micro-tonal adjustments.

chromatic
major
minor
pentatonic
dorian
phrygian
maqam
in-sen

→

6
12
24
48
96

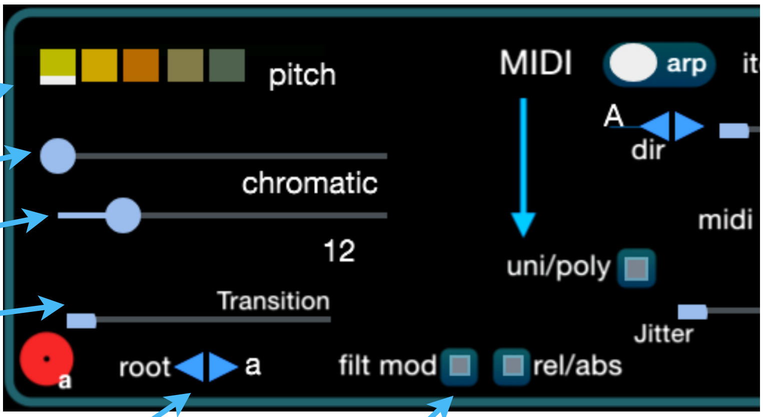
Western music divides a scale into 12 semitones, establishing a geometric series of frequency relationship. The *sieves* engine presents you the possibility to set these divisions from 6 - 96 offering the micro-tonal scale relationships.

The *sieves* engine also lets you design your custom scale with arbitrary interval values between scale components and the intonation of each scale tone, offering in real time operation. There are two custom scales available and all the scales can be assigned to relevant synthesis engine parameter outputs in order to scale them.

sieve parameters and the user interface :

The parameters of the *sieves* engine and its user interface is the *sieves* panel on the main screen of Cosmos*f*.

The bar on the topleft lets you select the synthesis parameter on which the selected scale is being applied. Each scale has a division setting between 6 - 96.



synthesis parameter to be scaled

scale type

scale division

transition ratio of between scaled or unscaled output

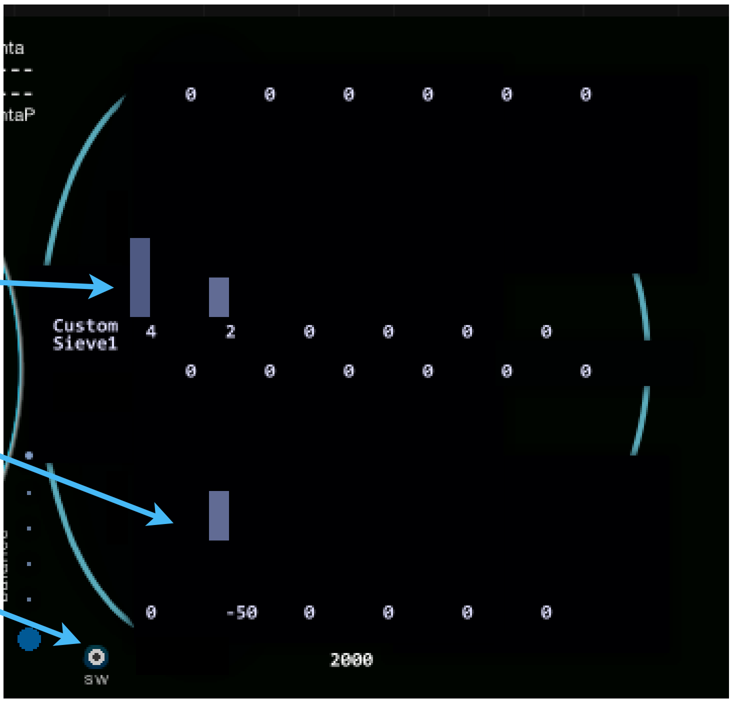
scale root tone

pre/post modulation scaling

The transition ratio between scaled values and raw stochastic values of Cosmos*f* is a powerful parameter which you can use to morph the scales to unsieved raw Cosmos*f* outputs continuously.

The pre/post modulation switch lets you apply the scales before or after LFO/LineGen modulations. As a summary; for each synthesis parameter mentioned on the topleft bar, you can choose a scale, its division, its transition ratio and the pre/post modulation switch.

The *sieves* engine lets you design your custom scales as well. For this, you have to select the *custom1* or *custom2* on the *scale type* slider. The custom scale user interface will show up in the place of the additive synthesis engine with similar layout. You can define two custom scales for each preset of Cosmos*f*.



the interval value between the sequence elements of the sieve, hence the custom scale

the intonation of the sequence element (-100/100 cents)

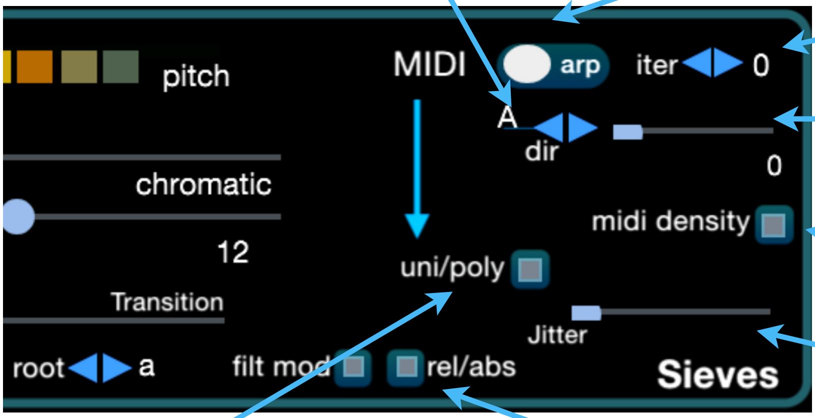
switch between additive synthesis / custom sieve UI

Important : When using the *morphing* engine of Cosmos*f*, the *sieve* parameters will morph as well between the selected morphing presets. However the *midi* parameters will not. All the *sieve* engine parameters act the same on both universes of Cosmos*f*.

performance with MIDI :

How can a stochastic synthesiser with such experimental outcome can be performed with MIDI? This question has been answered within the *sieves* engine, and now users can perform on the keyboard (still the most common way to interact with a soft synth) setting the micro event pitches according to the selected current scale.

There are two MIDI response modes ; *Uni* and *Poly* ; as unison and polyphonic. In the *Uni* mode, the user can play only one note at a time on the keyboard, and the PB rate setting for the micro events will follow the pressed key value, e.g. middle *A* on the keyboard will map to PB Rate setting “1”.



direction of arpeggio

arpeggio on/off

octave iterations in the direction of arpeggio

octave division of the arpeggio

density sw to modulate the micro event density by midi chords

stochastic modulation on the arpeggio intervals

switch to activate polyphonic midi chord mapping to micro ev.

midi pitch to micro event PB rate mapping behaviour

Important : If on a Cosmos*f* preset by default each micro event has different settings as *PB Rate,* then the played midi key can apply a *relative* change on each micro event *PB Rate* or an *absolute* same value setting regarding the *rel/abs* button state, which can be found on the bottom of the *sieves* panel.

If the MIDI response mode is set to *poly,* you can perform chords on the keyboard and each key of the chord will be mapped to one micro event *PB Rate*. For instance, if the current micro event density is 3 and you play a C major chord, then the C will be mapped to the 1st micro event, the E to the 2nd micro event, and the G to the third. You cannot perform more keys then the micro event density value.

However you can set the micro event density to be modulated by the note quantity of your MIDI chords. For example, if you perform a maj7 chord, the micro event density will change to 4 as there are 4 notes in this chord. This behaviour can be turned on/off by the *density* button on the *sieves* panel.

Important : When in *poly* mode, the *res/abs* switch will be automatically turned on, because by sending midi keynotes to Cosmos*f* you specify what *PB rate* value to each micro event will be assigned. Therefore the previous PB rate relations will be discarded and they accept the absolute values coming from MIDI.

performance in arpeggio mode :

The logic behind the arpeggiator is, that it iterates the micro event pitch values in one octave intervals higher or lower on the next meso event. As an example; if your micro event density is 3 and meso event density is 2, then performing an A minor chord on the keyboard will assign the note frequencies A,C,E to 1st, 2nd and 3rd micro events of the 1st meso event. The second meso event will have A1,C1,E1 (one octave higher if the direction is set to *U*) on its micro events.

- The direction of the arpeggios can be set as *D = down, A = Alternate, U = Up*.

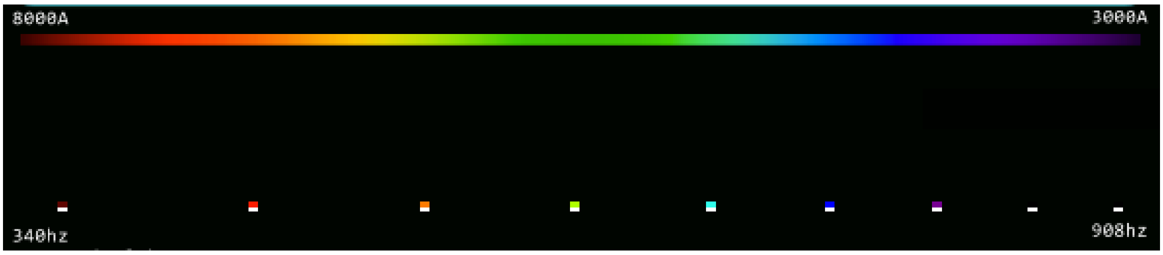
- If the meso event density is higher than the arpeggio iteration, then the arpeggio will return to original micro event values before the end of the macro event cycle. By playing with the density values of meso and micro events (still they can be modulated with stochastic functions) very interesting rhythmic sequences can be created.

- The octave division slider for the arpeggiator acts as following; when it is set to *1* the arpeggiator iteration will be one octave (it depends on the selected tonal scale what that means as frequency ratio). When you adjust the slider, you alter the interval as divisions of this one octave in continuum.

- You can also apply stochastic modulation on the iteration interval by setting its amount with the jitter slider named "*J*" on the down right of the *sieves* panel.

visualisation of the *sieves* engine :

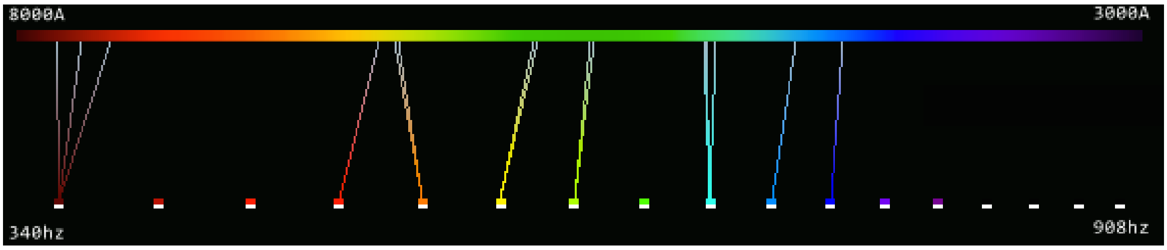
For the visualization of the *sieves* filtering and mapping process, the idea of converting the sound frequencies into light wave lengths has been applied. Below you see a spectrum bar of the visible light on the figure, ranging from ultra violett into infra red, it represents the wavelengths approx. 3000anstrom to 8000angstrom.



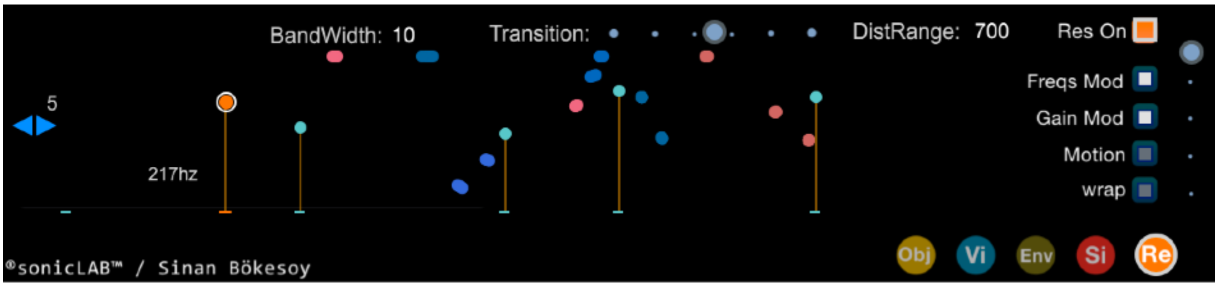
The sonic frequencies can be mapped easily into this range, and one can obtain a range between 340Hz-908Hz. Of course this becomes a very narrow frequency range for defining a pitch range, but what we do is mapping the appropriate octave divisions of these frequencies (in proper tonal scales octave ratios) back into this range of the light spectrum.

The dots at the bottom of the figure represent the selected tonal scale and its elements which fit between 340Hz-908Hz.

The incoming Cosmos*f* data into the *sieves* engine is being pointed on the top bar with the mapping process and the scaled output of the *sieves* on the bottom of the visualization panel. Both values are connected with a line, and the playful visuals of the lines represent clearly the filtering and mapping process of the *sieves* engine.



“Resonator” filters in CosmosFX :



BandWidth: 10

DistRange: 700

Cosmos*f* FX6 introduces Resonator filter section, where one can define the frequency, bandwidth and the gain of up to 6 resonator filters to be applied on the meso level of Cosmos*f*. Hence the Resonator filters come before the meso level filter section of the synthesis engine.

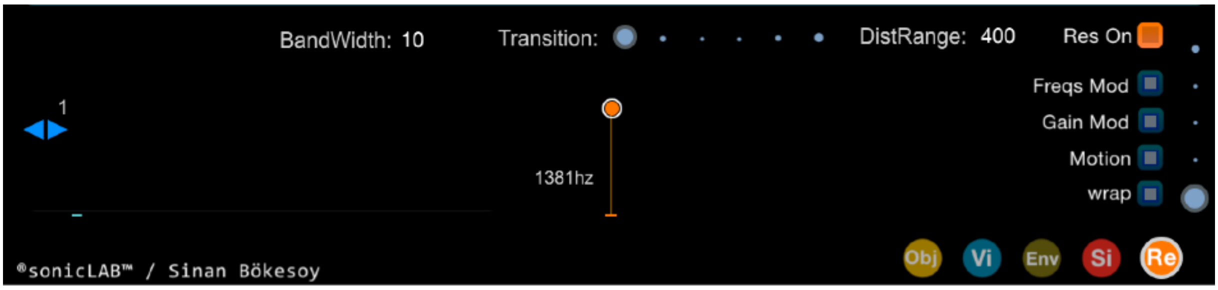
The particular behavior of Cosmos*f* resonators are the stochastic modulation on their frequency and gain parameters. The user interface of the Resonator section is straight forward:

- One can activate the interface display by pressing the “RE” button.

- The “ResOn” button at the top right turns on the Resonators.

- The vertical slider on the right is the dry/wet FX balance of the Resonator section.

- The middle field area is where one positions the resonator filters on the frequency/gain axis.



BandWidth: 10

DistRange: 400

The arrow buttons on the left let you select which resonator is active for you to modify on the interface. There can be 6 resonators at once. They won’t show up if their gain value is zero.

- The modulation function on the Resonators is a gaussian distribution function. In order to activate the modulation, you have to click on the “Freqs Mod” and the “Gain Mod” buttons on the right side. Each resonator will have its own gaussian distribution function.

- The next step would be define the distribution range. You can type the value on the text field next to the “DistRange” label. This value is in hertz eventually. The distribution range is between 0 - 10000Hz, and for the gain values this corresponds to 0.0 - 1.0.

- The distribution for the resonator are discrete and calculated with every new meso event. In order to make the transition continuous, you can use the Transition speed slider. When it is on the left, the value is zero and the distribution is discrete. In the middle the value is 1, which means the transition takes as long as the meso event durations. On the far right, the slider value is 2, meaning that the transition is two times faster.

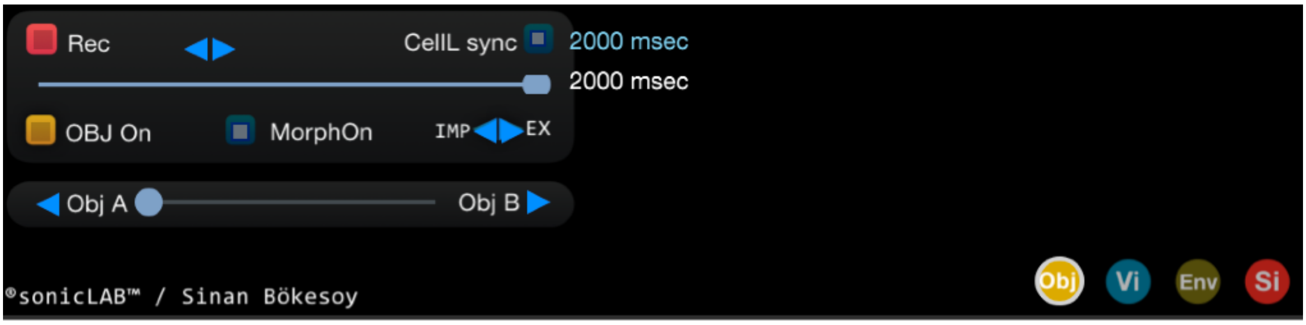
- When you click on the “Motion” button, all the resonators will share the same current gaussian distribution value, meaning that the resonator motion is a parallel motion between them.

- When you click on the “wrap” button, you do apply a bouncing at the limits of the frequency and the gain values. Hence distribution values greater 10000Hz and smaller then 0hz will be bounced back, and the same for the gain values.

- And as the last, you can change the bandwidth value of the resonators by typing it directly on the text field. The minimum value is 2 and represents the smallest bandwidth, sharpest resonator.

One can assign midi controller to all of the buttons and sliders of the Resonator filters section. You can also record the sliders “FxBalance” and the “TransitionSpeed” with the “Object” sequencer.

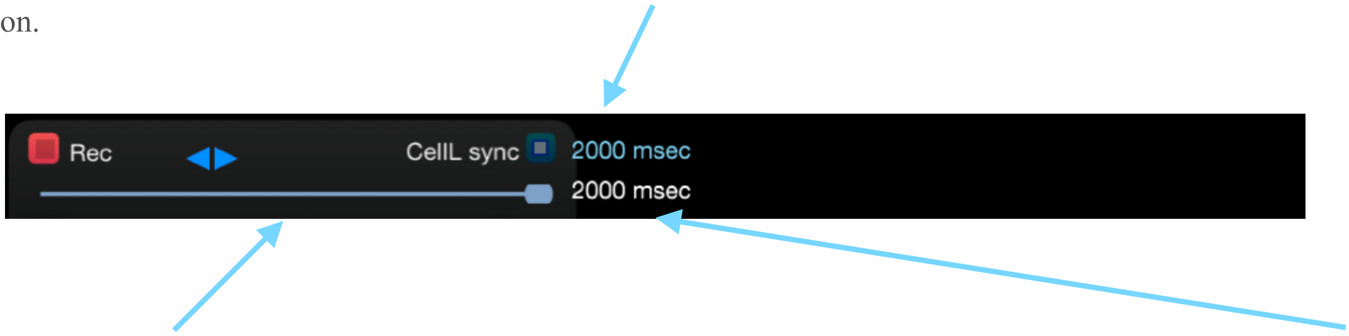
introducing the “Object” in Cosmos*f*



Sound as an ‘object’ formed by organized elements, is exhibited in continuous motion and a process of change. Cosmos*f* introduces various tools to generate the elements of the sound, sets them in motion between stochastic/ deterministic and ordered/disordered frontiers.

In order to add a new level of sonic design, the “Object” in Cosmos*f* lets you drive a gigantic number (**2150**) of gesture recording tracks set for each of Cosmos*f* parameter. All these tracks form a sequencer which you can drive in various ways.

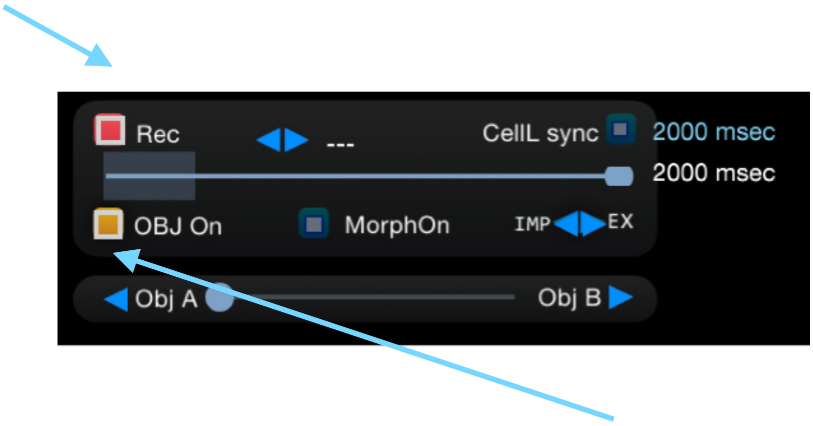
First you set a time cell of the Object duration, hence the length of the recording in msec. by tapping on this text button.



This slider will slow / speed up the Object motion. According its setting, the actual length of the object is displayed here.

For example if the slider is at half value, then the actual will be two times longer then your time cell setting.

Now you can start to record some gestures to the Object sequencer. For this you do activate the recording mode, either by clicking this Rec button or the ‘**r**’ key on your computer keyboard.



When you activate the Recording mode, it automatically turns on the Object mode as well. Then if Cosmos*f* is running, a rising bar indicates the relative time state of the sequencer which indeed is always in looping mode.

While in recording mode, you can move the sliders or click on the button switches of the Cosmos*f* interface, to record these changes into relevant tracks of the Object sequencer. When you touch any parameter on the interface, the relevant sequencer track gets activated and will be listening to your gestures on that parameter control.

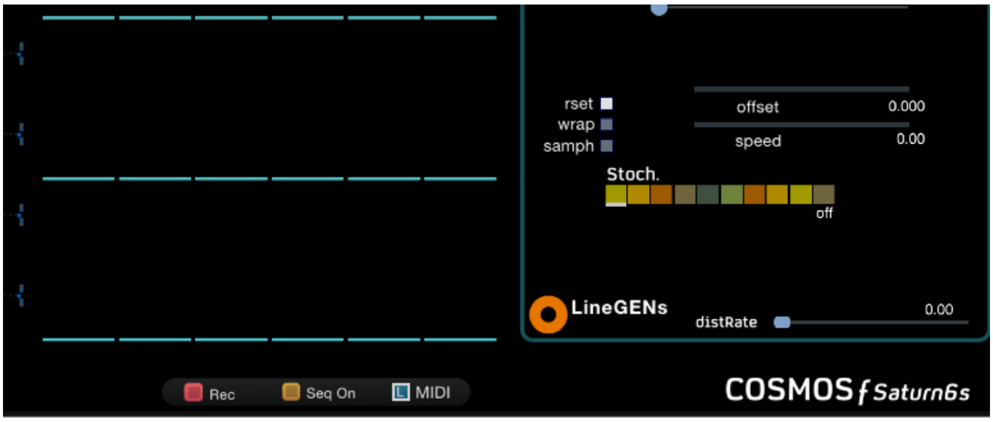
Since this is a looping recording, when you are done with your gesture performance, either click on the Rec button or press the '**r**' key on the keyboard again to disactivate recording. Otherwise each new loop in the recording mode will override the previous recorded values.

As soon as you deactivate the recording mode, the Object sequencer track switches to playback mode and plays your latest track recording on the relevant parameter control. Even when you enter to recording mode again, the previously recorded tracks will be in playback mode unless you touch the relevant parameter on the Cosmos*f* interface.

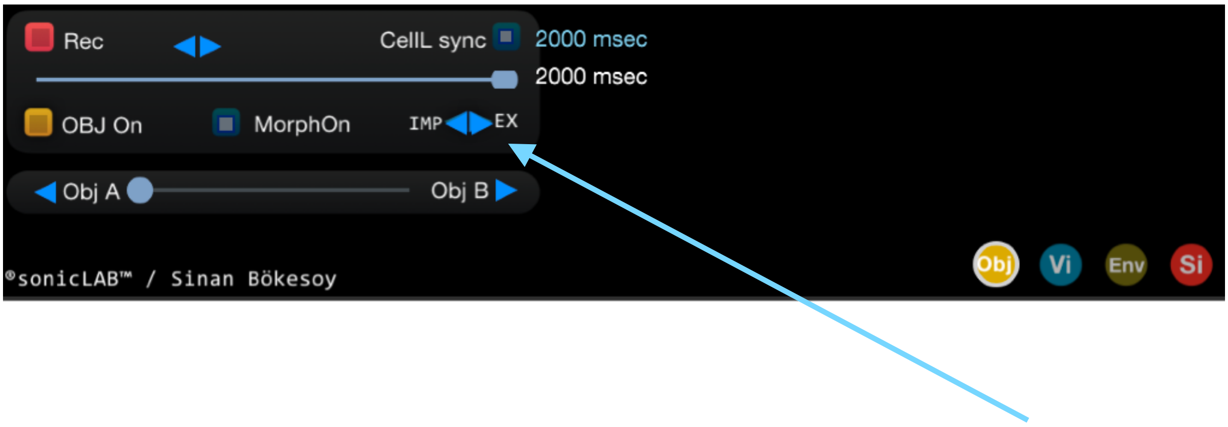
You can record multiple gestures with your Midi controller at the same time, and each gesture will be recorded to its relevant track of the Object sequencer.

Click on the '**e**' key on the keyboard to erase the current Object sequencer as a fresh restart.

Click on the Obj On/Off button to disactivate the Object sequencer playback.



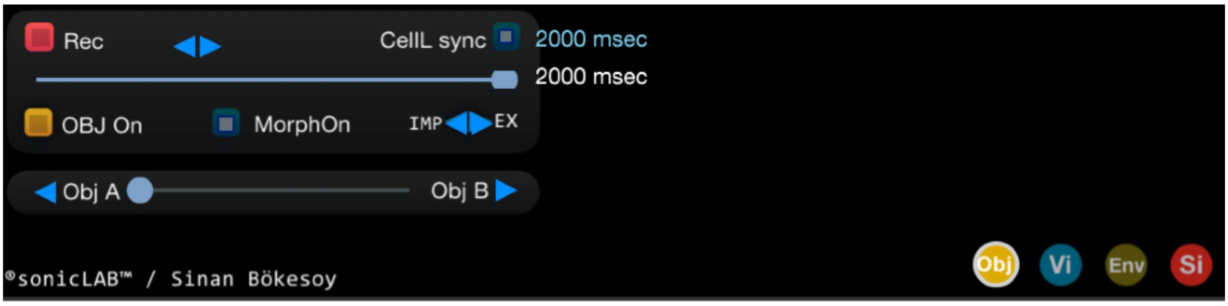
For your convenience, the Rec and Obj On/Off button is also located on the Cosmos*f* Modulation screen. You can use these buttons in the same manner to control different modes of the Object sequencer.



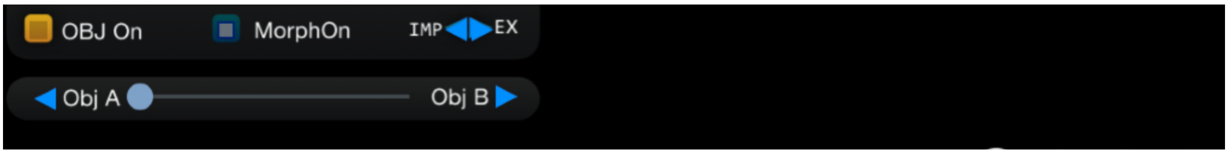
You can export your recorded Object sequence to the harddisk of your computer with EX button and import an existing one from the harddisk with the IMP button.

You can save the state of the Object sequencer along with its sequence data filename to each preset of Cosmos*f*. Once you recall that preset, the relevant settings will be done and the sequence file will be loaded to the Object sequencer.

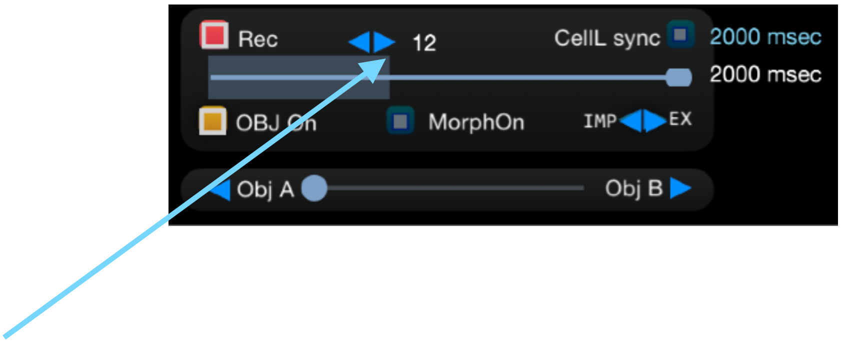
Other parameters on the Object panel are;



Morphing between two Object sequences is possible. You can load two independent sequences to locations **ObjA** and **ObjB** and then activate morphing by clicking on the **MorphOn** button.



The morph slider between ObjA and ObjB will set the degree of morphing.



You can even quantize a running Object sequence by setting the quantize value the same manner as on a DAW with this button. The time bar will also behave in stepwise motion accordingly.

You can set the Object length the same as the macro cell length of Cosmos*f* by activating the **CellL sync** switch. Thus they will synchronize easily.

The last subject is what you can record to Object sequencer and not:

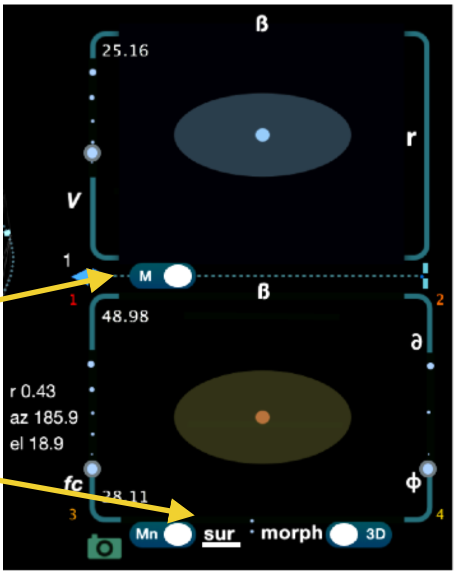
Since the preset morphing mechanism also addresses all the parameters of Cosmos*ƒ*, you cannot use preset morphing 2D slider gestures to record all its actions to the Object Sequencer but you can record the morph speed and morph interpolation slider gestures.

As Cosmos*ƒ* lets you edit each micro event and meso event synth parameters individually, so can as well record their individual parameter changes to the Object sequencer. Use the **GuiSync** button to apply the process of change to all events at the same time and record to their tracks respectively.

When you want to record the parameters of the parallel universe cycle events, then activate the parallel universe interface first by clicking on its cycle, and then you can start recording all its parameters with your gestures.

“surround sound” in Cosmos*f* :

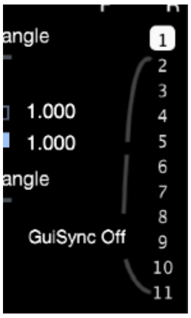
Cosmosf is able to spatialize sonic events for surround sound production by localizing its meso events in planar and hemispheric space coordinates. Hence, Cosmos*f* offers 2 channels stereo, 4 channels 1st order 2D ambisonic, 6 channels planar hexagonal localization mode on a 2D circular surface and 16 channels 3rd order ambisonic spatialization inside a 3D full sphere.



important note *: The surround engine distributions are mapped equally to both universes of Cosmosf. Hence their meso events share the same surround spatialisation.*

To adjust the surround sound parameters, we use the same interface of the morphing engine panel. When you click on the ***sur*** label, you do activate the surround distribution editor. When you click on the button **M/S**, you can switch between surround and morphing engine visuals on the same interface.

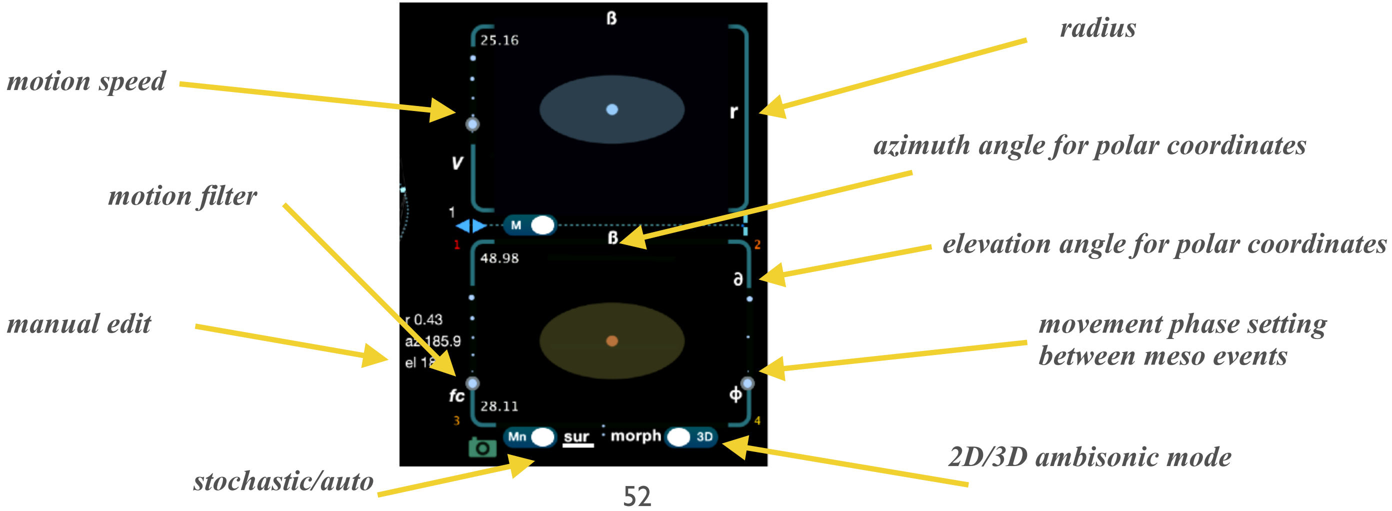
Each meso event of Cosmos*f* can have independent settings to localise itself in surround sound space. You can use the GUI sync switch to turn on for editing all the meso event localisation settings simultaneously or and turn GUI sync off to edit them individually. Cosmos*f* offers stochastic spatial movement or mathematically defined curve movements to choose from.



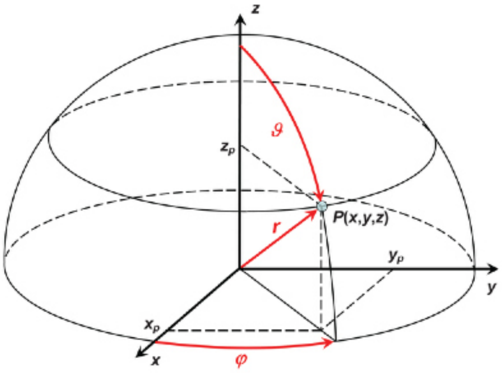
Remember that micro events together form up the meso events, so in fact it is this organisation of micro events inside a meso event which you do localise together with the surround sound engine. Cosmos*f* uses the *constant power panning* strategy to balance the gain of the meso events in the surround space for the stereo and hexagonal distributions. For 4ch and 16chns the 1st order and third order *ambisonic* encodings are used.

important note : Cosmos*f* ambisonic output is an encoded one with **ACN ordering SN3D formatting**, in compliance with the latest industrial usage. You can use enter this output directly to a decoder for any conversion purposes. channel & loudspeaker configurations and for binaural audio decoding. There are many free or paid audio plugin solutions for this type of conversions available on the market.

If you would like to manually set the spatial position of the meso event or add stochastic motion, then put the switch in ***MAN*** mode. If you want to move the meso events automatically with curve functions then put the switch in ***AUTO*** mode.



The 6ch mode will drive the motion on a circular plane, the 4ch mode with a 1st order ambisonic and the 16ch mode with a third order ambisonic distribution encoder. For each surround mode, Cosmos*f* will show you the speaker positions on the interface. The order of the loudspeakers are pointed with blinking lights in corresponding order.

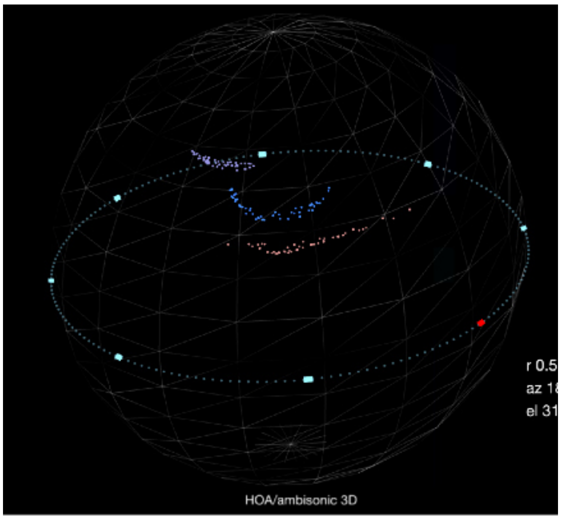


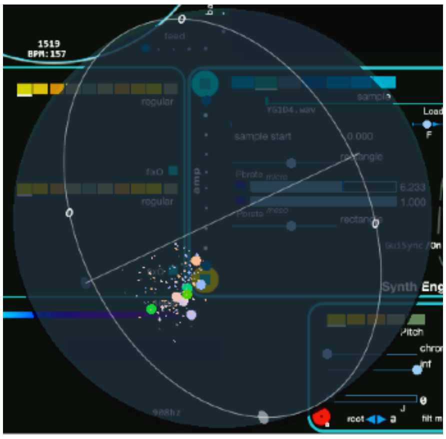
To manually move the meso events in space, you have to get familiar with the polar coordinate system, which is the editing mode in Cosmos*f* morphing and surround sound engine. The polar coordinate system uses a ***radius*** value as the distance from the origin and the ***azimuth*** angle to define any point in a circle. It does use additionally the ***elevation*** angle to define a point inside a spheric volume as the height.

Like in the morphing engine of Cosmos*f*, you will adjust the position and the radius of the blue and green ellipses on the control panel in order to manipulate the position distribution of the meso events in surround space. The ellipse radiuses define the angles and radius distribution of the polar coordinates which define the position of meso events.

Hence, when you are in *stochastic* mode, you can change the ellipse sizes (press and release the ‘r’ button on the keyboard and drag the mouse on the relevant ellipse) to define the range of distribution for the stochastic surround mechanism.

When in *auto* mode, the ellipse positions are relevant for the curve function parameters which move the meso events in a deterministic way in the surround space.





On these figures you see different stochastic distributions of meso events in the surround sound space. You can manipulate this by just adjusting the ellipse position and sizes on the control panel.

To move quickly each meso event to the virtual loudspeaker positions, just give the maximum ***r*** (radius) value in order to scale the position up to the surface of the sphere, and adjust the ***azimuth*** value, e.g. 0,45,90,135 degrees.

For both stochastic and auto mode, you can type in the radius, azimuth and elevation values manually by clicking on the text labels as well.

One can define the speed of the motion with the slider labeled ***v*** and filter the abrupt changes in the movement with the filter slider labeled ***fc.***

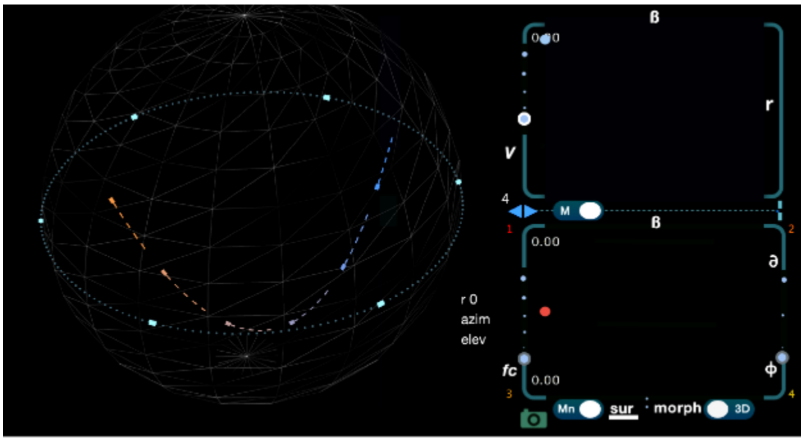
On the right side of the bottom panel, the slider labeled ***phi*** lets you set a phase difference between meso events so that they can line up in sequential order.

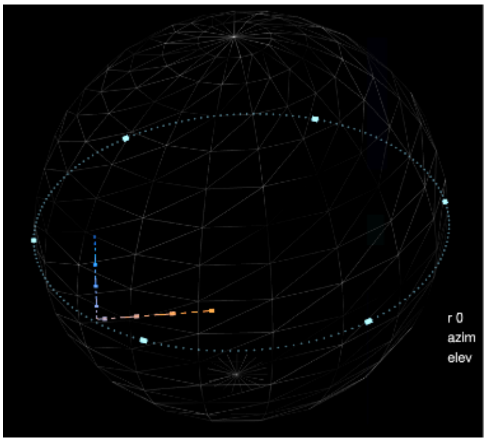
How to use the ambisonic rendered surround sound ? : *Cosmosf* Saturn6S does 1st order or 3rd order Ambisonic encoding. This means in order to play this encoded output directly through your loudspeaker setup or decode a stereo binaural audio down-mix, you have to apply third party ambisonic decoder plugin directly after Cosmos*f* Saturn6S on your DAW track. There are a number of free / paid ambisonic sound plugins which deliver extensive tasks for decoding and various format conversions.

Why Cosmos*f* does offer ambisonic encoding and not also decoding ? : Encoded sound material can be used as production material of VR and 360 video soundtracks, or relevant game sound. Such applications require headtracking which rotates the ambisonic sound world according the head orientation of the listener. Cosmos*f* delivers the required standard encoded output compatible with these applications. So when you design your soundscape and render it as a concrete material, you can rotate it according to the head-tracking on these media applications.

If Cosmos*f* would deliver the output as the decoded material then you could play it directly through your loudspeakers, however it would not be available for headtracking. As stated above, there are a rich amount of third party plugins for decoding process.

On these figures below you see automated distributions with the available math functions. You can manipulate this by just adjusting the parameters for the polar coordinates as well.

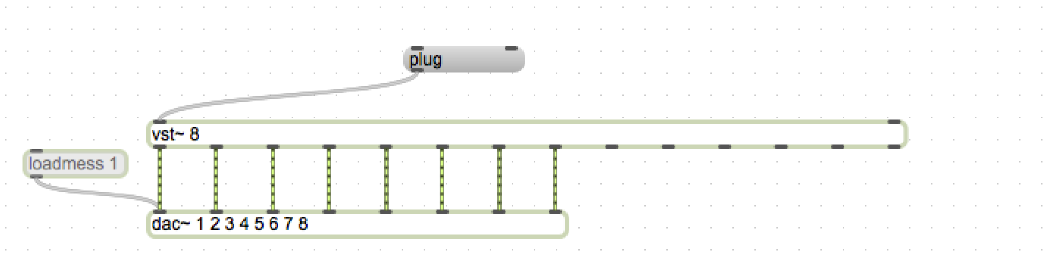




Useful hints : When the radius is set so that the events are going through the loudspeaker positions, then some discontinuity might happen in sound due to the fact that the loudness finds a singularity right at the loudspeaker position. So please consider applying the radius scale just a bit less, to hear the difference.

You will see that when the azimuth is set to zero, then the event spatialization is right in the center of loudspeaker 1 and 2. This is just identical to the real world case.

- *MaxMSP* : You can use the *vst* object of MaxMSP to run the Cosmos*f* plugin. However the channel number arguments of the *vst* objects is of no use since MaxMSP responds to Cosmos*f* with the maximum channel number eight, no matter what you set as the argument of the *vst* object. The solution is simple. You can set the argument as 6channels for the *vst* object and then decide on Cosmos*f* user interface how many channels you would like to use for the output. The *vst* object will use the corresponding audio outputs depending on the configuration of Cosmos*f*.



- *Nuendo (Steinberg)* : The newest update of Nuendo supports ambisonic bus natively. And Cosmos*f* Saturn6.1S auto-configures itself to the 3rd order ambisonic bus. In order to get sound from Cosmos*f*, you have configure your audio output in Nuendo as 3rd order ambisonic, which is 16channels. Then regarding the output mode of Cosmos*f* it can use the first 2 channels (stereo) or the full 16channels (3rd order ambisonic). Remember that the ambisonic output of Cosmos*f* is encoded, so you have to decode it on the same bus with another plugin to here it in binaural format or directly feed your loudspeakers.

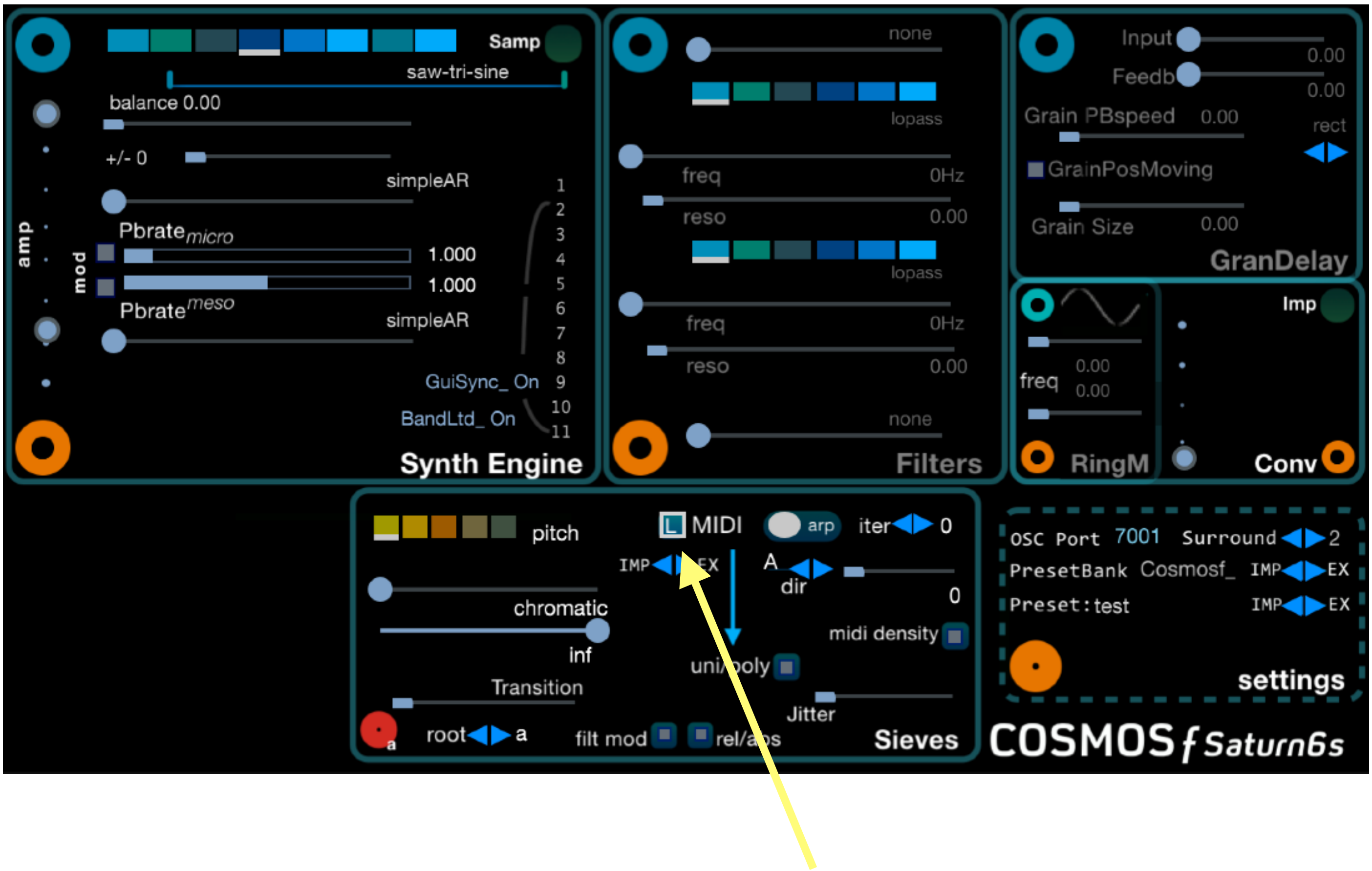
For other DAW systems, the logic is similar to Logic, whenever they update or offer ambisonic bus support such as the Nuendo, we will update Cosmos*f* Saturn as well.

communicating with Cosmos*f* :

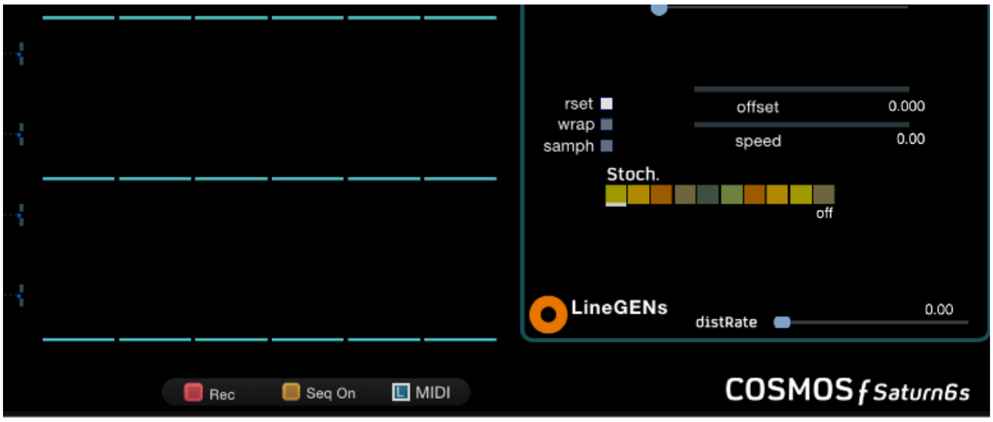
There are two possibilities to communicate with Cosmos*f*. By sending standard **Midi** controller messages or dedicated **OSC** messages.

Let's start with **MIDI controller messages**;

Cosmos*f* has an immense amount of parameters and there is no default layout for which controller message addresses which parameter. However you can do your assignments in seconds following these actions;



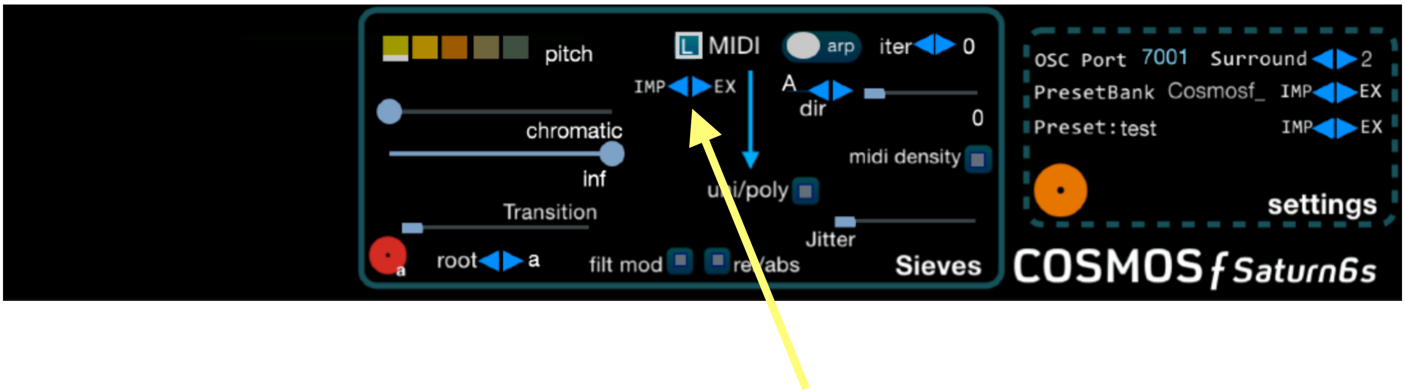
1. First activate the **Midi Learn** mode of Cosmos*f* by pressing on this button or pressing the "L" key on your computer keyboard. The same mechanism you will find also on the Cosmos*f* modulation screen, so that you don't have to switch between screens for this.



2. Second, move any slider or press any button on Cosmos*f* interface. This action will teach the relevant parameters which you want to assign to the Midi controller which you will use in the next step.

3. Third, move a slider or press on the controller button which generates some midi controller message on your Midi controller. This midi controller number will be assigned to the Cosmos*f* parameter which you have selected on the second step.

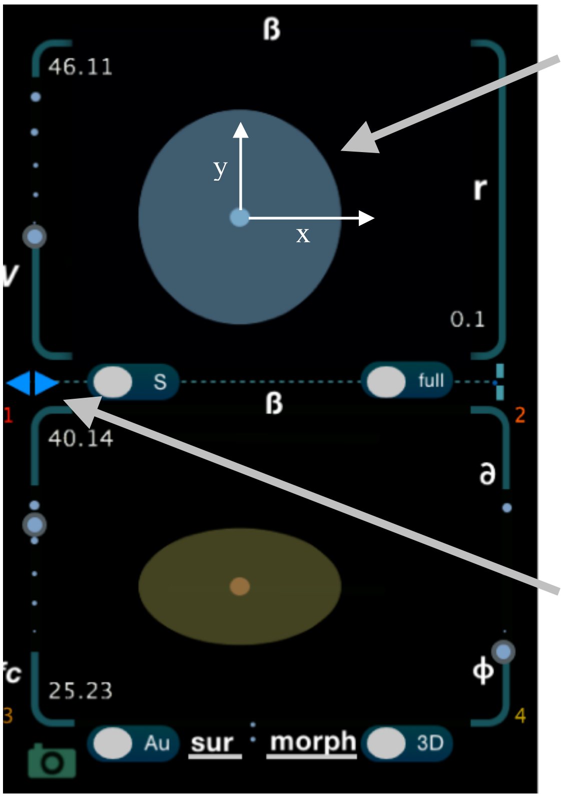
4. You will notice that the "L" mode (Midi learn mode) of Cosmos*f* will be deactivated as soon as you move the Midi controller, which means that Cosmos*f* has done the assignment successfully.



Once you have made your assignments and have created a Midi controller layout for Cosmos*f*, you can save this setup on your computer harddisk by clicking the "**EX**" button and giving a file name. You can import it later by using the "**IMP**" button. The "**default**" preset (no. 87) has a default MidiSeq loading with.

For your convenience, this setup will be saved also automatically with your Cosmos*f* preset, if you save your preset along with its latest state. The preset, when recalled, will look for this MidiCtl file and load it automatically for you along with the preset.

Of course you can load later different MidiCtl layouts to your working preset.



Note : When assigning the preset morph / surround sound 2D sliders to your Midi Controller, you have to assign the **x** and **y** movement of the slider to different controller messages. In order to assign the y movement of a 2D slider, keep pressing the **Cntrl** key on your keyboard and then move the slider.

When doing this for assigning the distribution ranges on the same 2D sliders, press the "**Shift**" key and then hold the "**Cntrl**" key together to assign the **y** size of the ellipse. If you do this without the "**Cntrl**" key then the **x** size of the ellipse will be assigned.

As a last note, you can assign the Object sequencer speed and the sequence morph sliders to Midi controllers respectively. You will not be able to control the decrement - increment switches on the Cosmos*f* interface.

The other possibility to communicate with Cosmosƒ Saturn6S is using **OSC messages**.

(OSC support is not available on the Cosmosƒ Saturn6 but only on 6S.)

Open Sound Control (OSC) is a content format for messaging among computers, sound synthesisers, and other multimedia devices that are optimized for modern networking technology. By using the benefits of the modern networking technology, it became an alternative to the MIDI standard as delivering accuracy and flexibility.

Cosmosƒ lets you control its parameter space through OSC protocol by considering the following scheme;

- Cosmosƒ tries to choose its default port address but if it is already occupied by other software then you can click on the OSC port label and type another one. If it is still not valid, it will appear in dark blue.
- All the OSC messages you want to send to Cosmosƒ should start with "/Cosmosf" address field. Then the arguments will follow according to the following scheme. Note that the value range of the argument and the type of the value eg. integer / floating point /or string;
- You have to specify which event you would like to edit to change its synthesis parameters. This you can do with the "*/eventindex*" tag as shown below. Then, every parameter change will be applied to this event.

/Cosmosf/eventindex	1-11 (int)		
/Cosmosf/macroEVL	0-10000 (float)		
/CellLMod	0-2	(int)	
/FxBal	0-1	(float)	"only in CosmosƒFX"
/CircFreeze	0-1	(float)	"only in CosmosƒFX"
/DelayBehav	bool		"only in CosmosƒFX"
/PitchCompens	bool		"only in CosmosƒFX"
/LoopAcc	0-10	(float)	
/CellFeedback	0-8	(float)	
/paraCycleShift	0-1	(float)	
/paraCycleRatio	0-1	(float)	
/paraUniverse	0-1	(integer)	
/paraCrossfade	0-1	(float)	
/paraFeedbackM	0-1	(float)	
/mesoDensity	0-11	(integer)	
/mesoDensityMod	0-3	(integer)	
/mesoCellScale	0-11	(float)	
/mesoOnsetDist	0-8	(integer)	

OSC - Synth Engine

for accessing the primary cycle synthesis parameters

/Cosmosf/microAmpGain	0-1	(float)
/mesoAmpGain	0-1	(float)
/microAmpMod	0-1	(integer)
/mesoAmpMod	0-1	(integer)
/microPBMod	0-1	(integer)
/mesoPBMod	0-1	(integer)
/BufferL	0-1	(float)
/PBdirection	0-2	(integer)

/microWindow	0-4	(integer)
/mesoWindow	0-4	(integer)
/microFWindow	0-6	(integer)
/mesoFWindow	0-6	(integer)
/microPBrate	0-1	(float)
/mesoPBrate	0-1	(float)
/engType	0-6	(integer)

/Cosmosf/microCutOff	0-15000	(float)
/mesoCutOff	0-15000	(float)
/microFiltMod	0-1	(integer)
/mesoFiltMod	0-1	(integer)
/microFiltType	0-3	(integer)
/mesoFiltType	0-3	(integer)
/microFiltRes	0-10	(float)
/mesoFiltRes	0-10	(float)

/Cosmosf/GrnMod	0-1	(integer)
/GrnPBSpeed	0-10	(float)
/GrnSize	0-1	(float)
/GrnInput	0-1	(float)
/GrnFeedback	0-1	(float)
/GrnPosMov	0-1	(integer)

/Cosmosf/RingMod	0-1	(integer)
/RingModFreq	0-2000	(float)
/RingModM	0-1	(integer)
/RingModFreqM	0-2000	(float)

/Cosmosf/mesOnsetFreeze	bool
/Cosmosf/mesDurFreeze	bool
/Cosmosf/micOnsetFreeze	bool
/Cosmosf/micDurFreeze	bool

/Cosmosf/MidiArpJit	0-100	(integer)
/MidiArpFract	0-1	(float)
/SieveTransition	0-100	(int)
/SieveScale	0-9	(int)
/SieveDiv	0-10	(int)
/SievePreMod	0-1	(mod)

/Cosmosf/Running	bool
/Cosmosf/GuiSync	bool
/Cosmosf/AddSync	bool
/Cosmosf/mSnapshot1	bool
/Cosmosf/mSnapshot2	bool
/Cosmosf/mSnapshot3	bool
/Cosmosf/mSnapshot4	bool

for micro events		**for meso events**	
/Cosmosf/LFOampAct	bool		
/Cosmosf/LFOspeedAct	bool	/Cosmosf/LFOSpeedActM	bool
/Cosmosf/LFOFiltAct	bool	/Cosmosf/LFOFiltActM	bool
/Cosmosf/LFOgrainAct	bool	/Cosmosf/LFOPanActM	bool
/Cosmosf/LFOSmpStAct	bool	/Cosmosf/LFORingActM	bool
/Cosmosf/LFORingAct	bool		

/Cosmosf/Line1Act	bool	/Cosmosf/Line1ActM	bool
/Cosmosf/Line2Act	bool	/Cosmosf/Line2ActM	bool
/Cosmosf/Line3Act	bool	/Cosmosf/Line3ActM	bool
/Cosmosf/Line4Act	bool	/Cosmosf/Line4ActM	bool
/Cosmosf/Line5Act	bool	/Cosmosf/Line5ActM	bool
/Cosmosf/Line6Act	bool	/Cosmosf/Line6ActM	bool
/Cosmosf/Line7Act	bool		

/Cosmosf/MorphTheta	0-1	(float)
/MorphPhi	0-1	(float)
/morphR	0-1	(float)
/morphspeed	0-2000	(float)
/morphfilter	0-1	(float)
/MorphFunc	0-10	(int)
/MorphellipseXr	0-1	(float)
/MorphellipseYr	0-1	(float)
/MorphellipseY2r	0-1	(float)

---

now some examples; for instance composition of the following OSC messages;

/Cosmosf/MorphTheta 0.5	(sets the 3D morph theta angle to 180deg.)
/Cosmosf/LFOampAct	(sets the micro event LFOamp switch on/off)

published articles

BOKESOY, S. ; **PAPE**, G. « Stochos: Software for Real-Time Synthesis of Stochastic Music» in : *Computer Music Journal* 27(3) MIT Press 2003. - pp. 33-43.

BOKESOY, S. « The Cosmos model : An event generation system for synthesising emergent sonic structures » in : *Proceedings of the International Computer Music Conference*, Barcelone, pp. 259-262, 2005.

BOKESOY, S., « Feedback Implementation within a Complex Event Generation System for Synthesising Sonic Structures » in : *Proc. of Digital Audio Effects (DAFx'06)*, Montreal, Canada, pp. 199-203., 2006.

BOKESOY, S., « Synthesis of a Macro Sound Structure within a Self-Organising System » in : *Proc. of Digital Audio Effects (DAFx'07)*, Bordeaux, France, September 10-15, 2007.

BOKESOY, S., « Presenting CosmosF as a Case Study of Audio Application Design in Openframeworks» in : *Proceedings of the International Computer Music Conference*, Ljubljana, September 9-14, 2012.

BOKESOY, S., «The development of a morphing tool in Cosmos*f* » in : *Journees Informatique Musicale*, Paris, May 12-15, 2013.