

**Fundamental 4 is designed&programmed by Sinan Bökesoy.**

all rights reserved  
sonicPlanet LTD 2025  
[www.sonic-lab.com](http://www.sonic-lab.com)

## installation of Fundamental 4

Fundamental4 comes as an instrument plugin and as a standalone app.

*(For the OSX version)*

Please open the downloaded *dmg* file. You will find the plugin and standalone versions.

- **Fundamental4.component** file : ideally should be copied to your system **/Library/Audio/Plug-Ins/Components** directory in your system hard-disk. For example */Volumes/OSX-Main/Library/Audio/Plug-Ins/*
- **Fundamental4.vst3** file : ideally should be copied to **/Library/Audio/Plug-Ins/Components/VST3** directory in your hard-disk.

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 also find the manual of Fundamental4 in that install image.

*(For the Windows version)*

- Unzip the download file.
- Put the directory **Fundamental4** in **Program Files / Common Files / VST3** directory of your system hard disk.
- You will also find the manual of Fundamental4 inside the folder.

*!!On Windows, do not install just the .vst3 file but the whole directory into your VST3 Plugins directory.*

*For OSX :* The standalone app version you can keep anywhere you like.

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

Fundamental uses the following directories : **PresetBanks**, **mydata** and **scl** .

- On OSX, these directories are created and relevant files are copied at **Users/Shared/sonicLAB/Fundamental4**.
- And for Windows it can be found at **Documents/sonicLAB/Fundamental4**.
- 
- When you need a fresh reinstall of the preset bank and app data; just erase these directories and relaunch the app. They will be recreated with default content.

*It is advised to keep your preset banks inside their original location, the PresetBanks directory.*

- Fundamental registered version works with iLOK licensing and can be authorized on the iLOK USB key , iLOK Cloud or the host computer.
- If you don't have an iLOK account, you can get it for free at [www.ilok.com](http://www.ilok.com)
- You are responsible to carry on by moving your license to different computers when you need. We don't deliver tutorials on how to use the iLOK environment.

**It is recommended to remove the previous versions of Fundamental from your systems plugins folder while running Fundamental4.**



## **some important issues :**

- . Fundamental uses sample level precision needed to control in real time all these parameters. Internal operations are normalized for 96kHz. Likewise recommended sampling rate to use for best sound quality is 96kHz.
- . Around 3800Hz there will be a transition to mathematical sine wave from the vintage sine wave. The explanation you can find at the Appendix section.
- . The new features on Fundamental4 such as the Oscillator Cloning, brings a considerable CPU needs. Therefore we advise to use Fundamental3 inside a DAW for better multi-core CPU handling rather than the standalone version while using these demanding features.
- . The suggested buffer size to run Fundamental4 is 512. Larger buffer sizes will effect the sequencer timing precision, as it is a sample accurate sequencer.

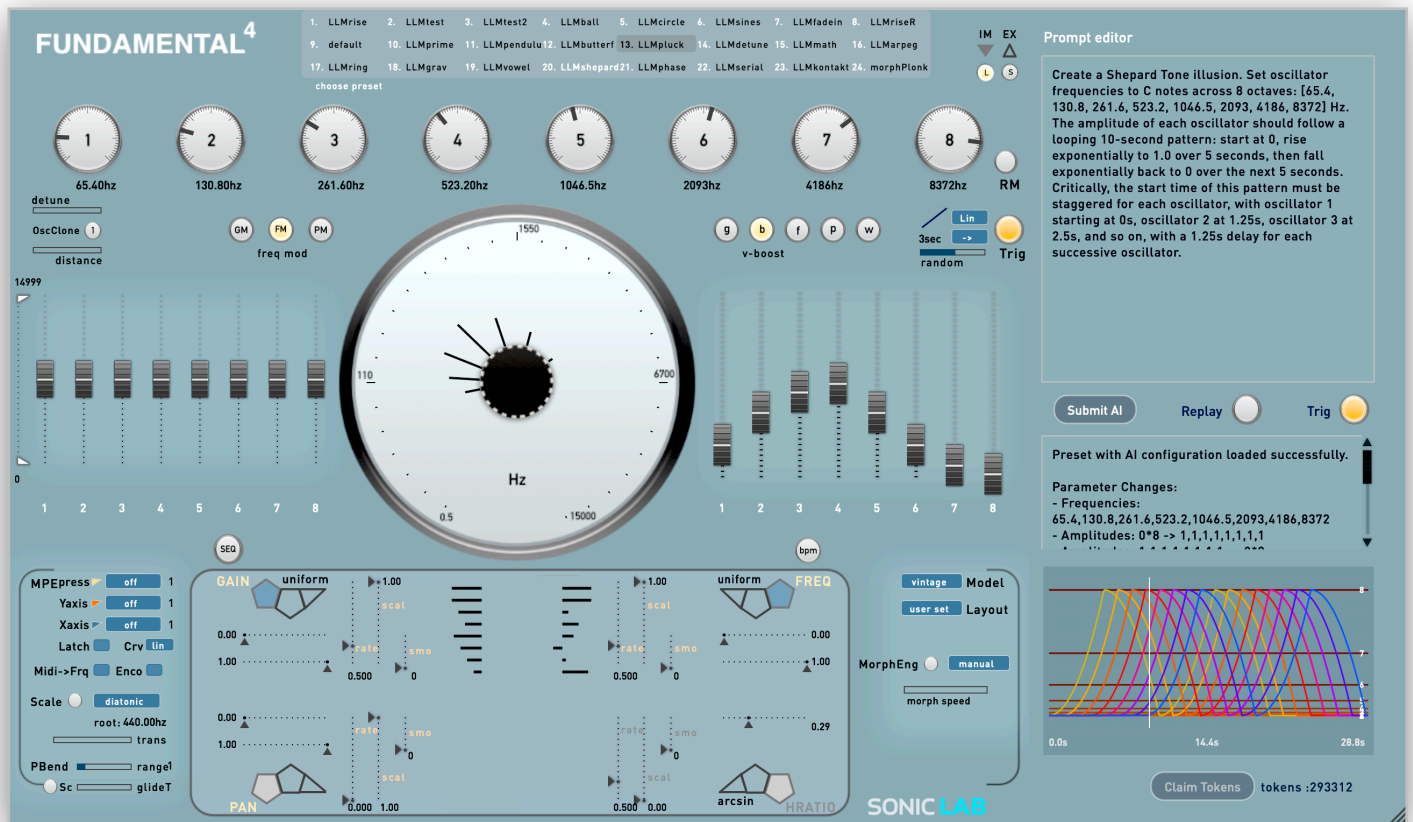
## **The key short-cuts**

“**l**” / “**L**” : Locks the osc gain sliders together, so that you can move all of them together.

“**v**” / “**V**” : When the sequencer is selected, this switches between the note grid and frequency trace display modes.

“**m**” / “**M**” : turns the morphing state on / off.

“**p**” / “**P**” : Switches between the main frequency wheel display of Fundamental and the morphing scene display.



## What is Fundamental - Our Philosophy

sonicLAB's commitment to audio development is to combine art, science, and craftsmanship in software form. We believe that **Fundamental** connects the past, present, and future of sound design. This significant role is key to understanding its design and purpose.

To appreciate this, let's briefly touch upon the history of electronic music. Beginning in the 1950s, the medium evolved in the hands of many different pioneers, across various centers, and in multiple directions. The tools available to composers were limited by the technology of the time, a factor that heavily influenced the musical genres that emerged from these efforts.

The history of the field is rich with interdisciplinary paths, proving that there can be no single authority, methodology, or technology that serves all the needs of electronic music production. Much of this technology was a transformation of equipment from other disciplines. In response, composers and researchers created their own labs to incorporate a multidisciplinary approach, adapting these tools to produce new sonic universes.

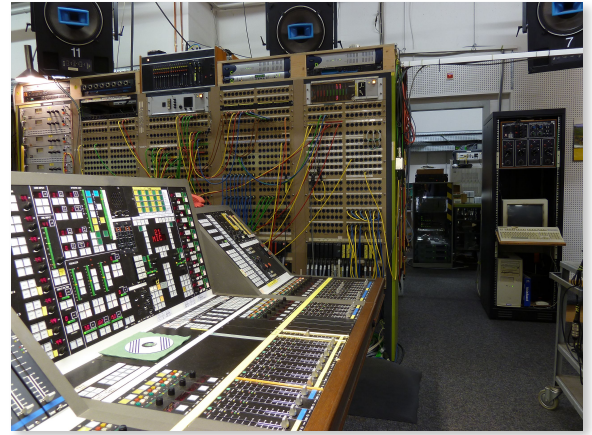
For instance, under the direction of **Pierre Schaeffer**, the famous **GRM** (Groupe de Recherches Musicales) at Radio France pioneered the techniques of *musique concrète*. This involved manipulating recorded sounds with **tape machines** and analog processors, meticulously combining and recomposing them into new works through a laborious workflow.

Magnetic tapes and audio mixers were the analog equivalents of the basic tools found in today's DAWs. Many influential composers of the era, including **Karlheinz Stockhausen** and **Iannis Xenakis**, visited the GRM before moving on to forge their own distinct paths.



Concurrently, the **Westdeutscher Rundfunk (WDR)** established its studio for electronic music in Cologne, initially led by **Herbert Eimert**. The studio's approach, particularly after **Karlheinz Stockhausen** joined as a composer and later became director, was entirely different.

It repurposed scientific equipment from the station's testing and calibration department, including sine-wave generators, noise generators, band-pass filters, audio mixers, and oscilloscopes. The purpose was to use pure sine waves to artificially construct unique sonic spectra, forming the foundational elements for Stockhausen's compositions.



Karlheinz Stockhausen, one of Olivier Messiaen's most prominent students, was a leading figure in electronic serial music. He extended the serialist approach beyond organizing just pitch and duration to meticulously structure the very fabric of the sound: its **timbre**. Using the test equipment available at the WDR, he layered precise combinations of sine waves to create complex, artificial timbres. In his landmark work, *Studie II* (1954), he constructed "tone mixtures" from clusters of five sine waves. In *Studie II*, he has calculated the frequencies with a simple formula : 25 multiplied by the square root of 5, to create a 81 tone scale.

100	=	100	→	100 Hz
$100 \times 25\sqrt{5}$	=	106.649494220837	→	107 Hz
$106.649494220837 \times 25\sqrt{5}$	=	113.74114617560345	→	114 Hz
$113.74114617560345 \times 25\sqrt{5}$	=	121.30435711726397	→	121 Hz
$121.30435711726397 \times 25\sqrt{5}$	=	129.37048333339991	→	129 Hz
$129.37048333339991 \times 25\sqrt{5}$	=	137.97296614612284	→	138 Hz

Unsurprisingly, using laboratory test equipment for composition came with significant limitations. For instance, the sine wave generators could only produce **integer frequencies** (e.g., 100 Hz, not 100.5 Hz). This meant Stockhausen's precisely calculated scale ratios had to be rounded, which altered the subtle beating patterns between the layered sine waves.

Furthermore, this equipment had **no built-in modulation capabilities**. The amplitude envelope of each sound, for example, wasn't automated but was instead shaped manually by Stockhausen during the recording process. In time, other composers adapted these tools for different musical approaches, and Stockhausen himself moved beyond strict serialism in his later electronic works, incorporating a wider array of tools and techniques.



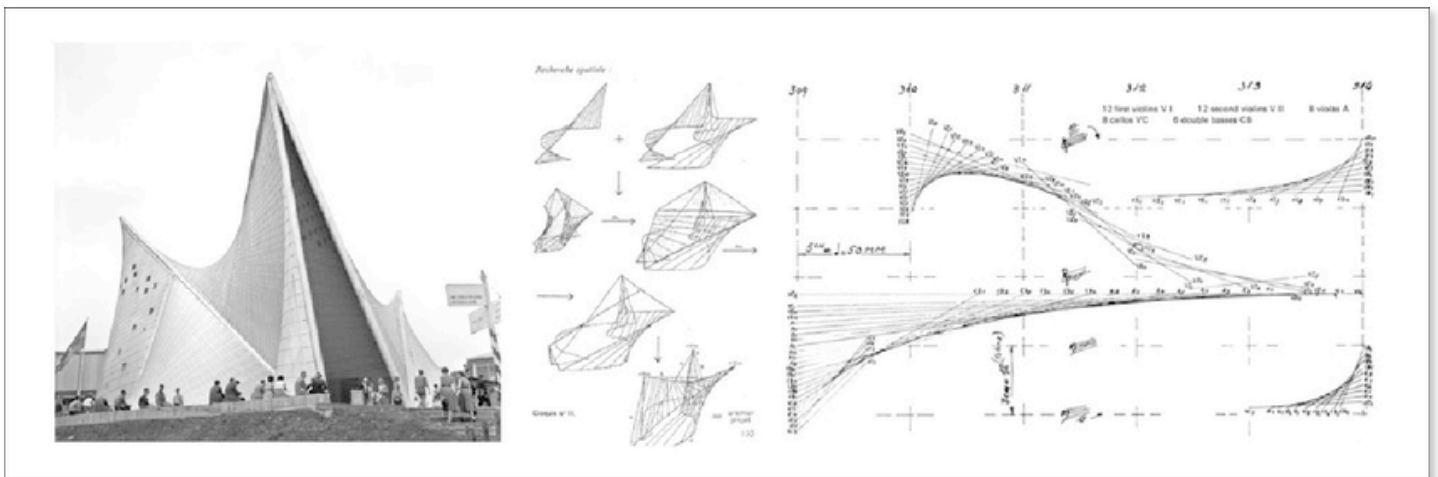
Coincidentally, this same era saw the first experiments in computer-generated sound at **Bell Labs**. It was here that **Max Mathews**, along with his colleague, composer **Jean-Claude Risset**, conceived the first programming languages for audio, laying the foundations of computer music.



Although this was the dawn of digital computing, these developments gave a major boost to the scientific study of sound. Two decades later, **IRCAM** (Institut de Recherche et Coordination Acoustique/Musique) in Paris would become a primary center for these efforts. IRCAM fostered collaboration by regularly inviting legendary American computer music scientists to work with contemporary composers, leading to crucial developments in spectral and real-time processing. Jean-Claude Risset also served as the head of IRCAM's computer department during this pivotal time.

A true pioneer of **algorithmic composition**, Iannis Xenakis forged a remarkable synthesis of mathematics, acoustics, architecture, and music. Beginning in the late 1950s, he developed **stochastic music**, using mathematical probability distributions to generate musical events.

He would perform complex calculations by hand and map the results to nearly every musical parameter: from notation and instrumentation to the gestures that create "sound masses" from layered parts. In doing so, he sought to invoke the forces of nature, capturing its complexity and the dynamic relationship between order and chaos. This groundbreaking, multidisciplinary approach fueled a lifetime of work that he later extended into the digital realm with custom hardware and software.

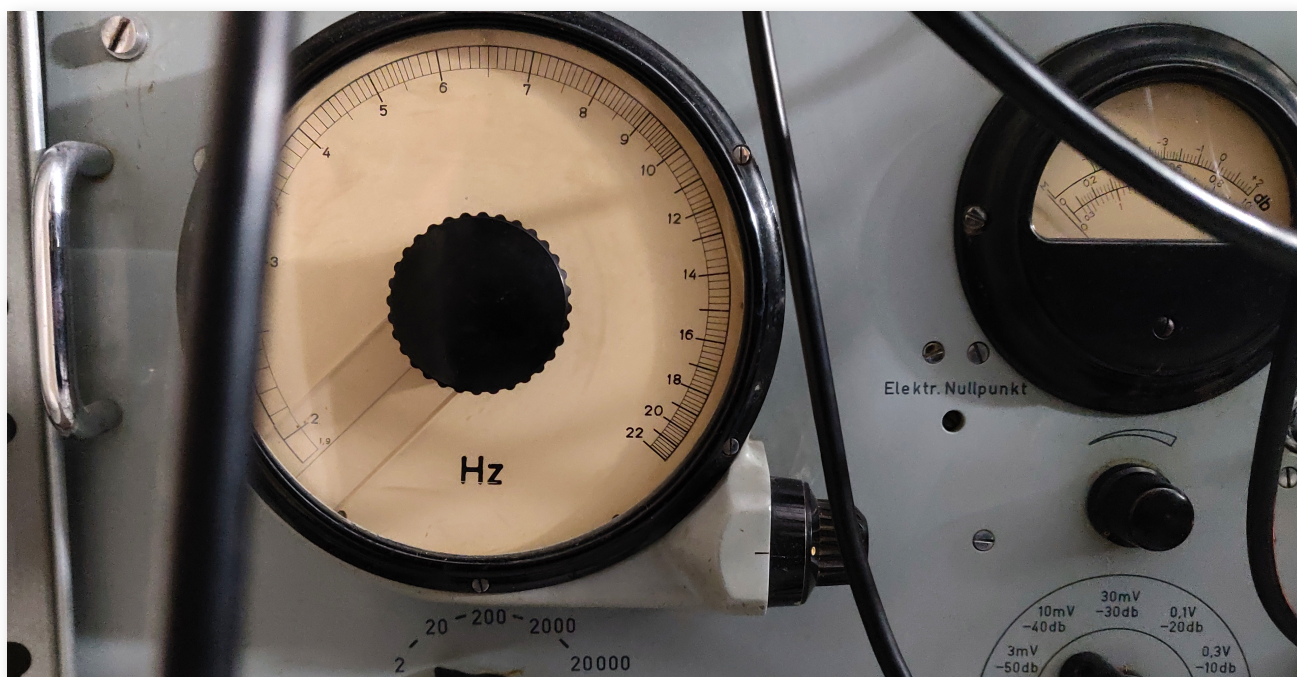




## Back to the question : *What is Fundamental ?*

**Fundamental** is a software synthesizer that unites these different roots of electronic music history. Its wave engine is a faithful sonic recreation of a vintage Rohde & Schwarz vacuum-tube oscillator—the same model that was widely used at the WDR studio.

However, **Fundamental** leaves the limitations of the original device behind. It introduces complex modulation possibilities, allowing you to continuously control parameters like frequency, gain, and stereo panning in ways that were impossible on the original hardware. This combination of authentic vintage tone and deep, modern control is what makes **Fundamental** a truly new kind of instrument.



The sound of vintage test equipment isn't technically pure—and that's its strength. The vacuum-tube gain stage adds subtle warmth, while frequencies below 50 Hz distort into something far more complex than a simple sine wave. These imperfections are the signature of a rich, vintage sound. **Fundamental** provides eight instances of this virtualized tone generator, complete with all the artifacts that give the original hardware its character.

The seeds for this project were planted during a visit to the studio of the musician **Hainbach** in 2020. After creating our sonicLAB VOLBot and PaSSBot instruments, it was time for a unique oscillator. The goal was to do more than just recreate the sound of a test tone generator; it was to transform it into a playable musical instrument by applying sonicLAB's expertise in generative design. A piece of test equipment is merely a tone generator; but **Fundamental** is an instrument to be performed and compose with.

sonicLAB incorporates Xenakis's methodology too and has successfully implemented it in its previous products of sonic creation. The perspective derived from Xenakis follows that these tools are not designed just to deliver static results but in contrary calculated and animated in continuous motion which is not possible to do by hand. Now imagine the results of implementing the signature modulation scheme on 8 parallel instances of a vintage sine wave generator in continuum.

For **Fundamental**, sonicLAB developed a unique poly-morphing wavetable oscillator algorithm to faithfully recreate the vintage tone with infinitesimal steps between possible gain and frequency combinations. (Please see the Appendix section for details.) Additionally, you can choose different tuning scales or import custom ones to quantize the pitch of these oscillators.

Four different continuous stochastic modulations (**GENs**), stochastic Attack-Release envelopes, and tuning scales (also reminiscent of sonicLAB's PaSSBot) can drive each voice and put it in constant motion. The sonicLAB design provides a dynamic structure that will take you to unheard sonic territories.

Furthermore, sonicLAB implemented direct interaction with MPE controllers and provides a user interface that combines past and present trends for fast and intuitive results.

**Fundamental 2** adds an audio-rate parameter morphing tool derived and adapted from sonicLAB's *Cosmosf Saturn*. It also now includes very useful direct standard MIDI mappings for sequencing purposes. It may also be the first synthesizer to feature a Morse code translator embedded as a synthesis modulation tool. Combined with **Fundamental's** complex sine wave engine, it can create very interesting generative patterns.

**Fundamental 2** also introduces a stylized ring modulator, designed with a focus on musical balance rather than pure mathematical representation.

**Fundamental 3** incorporates a stochastic sequencer in the sonicLAB style (a simplified version of the event generation system of *Cosmosf*) and oscillator cloning, which multiplies each voice to create even greater sonic richness.

### **Fundamental 4** *An important step in audio software design.*

When **Fundamental** came out in 2020, it has been announced that it feels like something Stockhausen, Xenakis and HAL 9000 might have dreamt up after a bender. Now with **Fundamental4**, the AI component ( HAL 9000 ) has been introduced.

In a pioneering move, sonicLAB has embedded an AI Large Language Model (LLM) into **Fundamental4**, allowing for control over synthesis parameters through natural language.

Simply describe the setup and its control over time you want to achieve in the prompt input field. By using precise, natural language prompts, the AI LLM can translate and unlock the ability to design sounds and configurations that go beyond manual control. It can create setups so precise they would be impractical to build by hand or with detailed DAW automation.

### ***What is the purpose of the AI and why Fundamental ?***

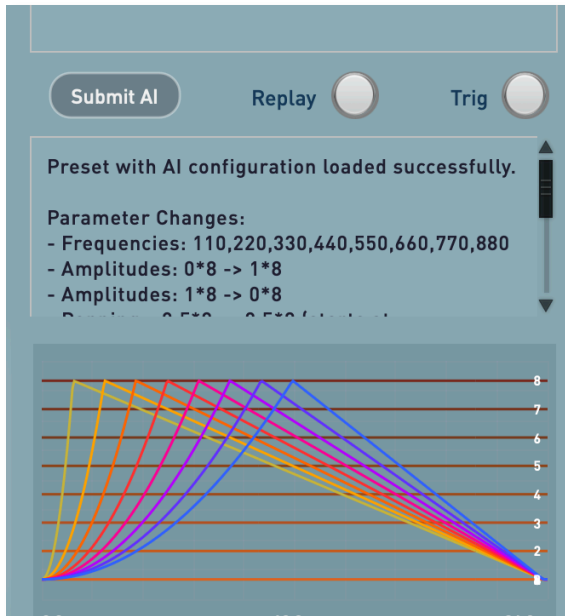
The integration of an AI Large Language Model (LLM) into **Fundamental** represents a new paradigm in sound design, moving beyond traditional knobs and sliders to an alternative powerful method of control: natural language.

**It shifts your role from a parameter programmer to a system designer.** Instead of thinking "what value should this knob have?", you're thinking "what makes the structure of the sonic universe?". This is a fundamentally creative and high-level approach to sound design.

It offers bridging the gap between creative intent and technical execution, allowing you to shape sound parallel of thought.

## What is an LLM in this Context?

At its core, an LLM is a generative AI trained to understand the patterns, context, and nuances of human language and respond likewise. For **Fundamental**, it's not a chatbot; it's a **specialized translator**. It is designed to interpret your creative instrument setups, expressed in plain English, and convert them into the precise mathematical instructions that the Fundamental synthesis engine can understand.



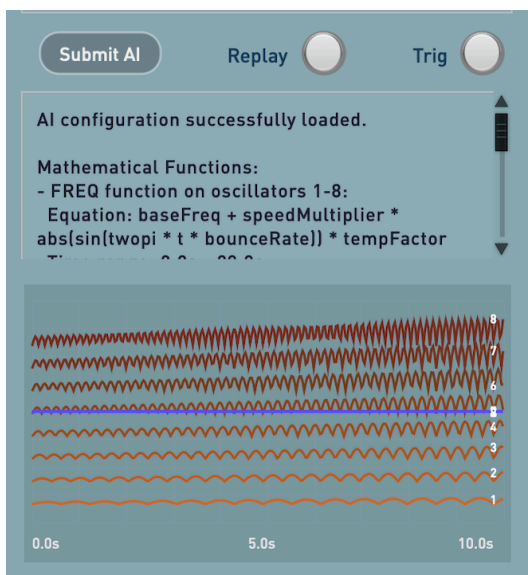
A descriptive prompt like “*Set all oscillator frequencies to multiples of 110hz. Give each oscillator gain an exponential curve rise which gets slower for each higher indexed oscillator.*” is analyzed for its intent and translated into a specific set of values for frequencies, gain levels, and timings.

**Fundamental** operates exclusively with sine waves, however their creative potential goes far beyond simple tones. The true magic happens in the listening experience, where combinations of these pure waveforms create emergent psychoacoustic phenomena—like shimmering **beating frequencies**—that have been used as compositional tools since the dawn of electronic music.

These effects are born from incredibly precise and delicate relationships between frequencies and their modulation over time. This is why the LLM is a game-changer. While manual control or mouse control or even detailed DAW automation can be too coarse to reliably shape these interactions, the LLM can define and modulate multiple parameters with sample-accurate precision in timing and values. This provides an unprecedented level of intentional control, transforming these subtle physical phenomena into a powerful and deliberate part of your sound design palette.

And there is more to this with using a Generative AI interface. For example this prompt :

*Each oscillator is a gaseous molecule moving inside a cube and changing directions and movement by hitting its internal walls. The amplitude of the oscillator is maximum at the cube center and minimum at the walls during this movement. The frequency is changing with the speed of the movement by adding to the initial frequency value. Change the speed of the movements by simulating a temperature change from cold to hot in the cube for 20 seconds.*



Such approach allows you to control the synthesizer based on imagination, physics, and metaphor rather than just numbers. It creates an incredibly intuitive bridge between a conceptual idea and a sonic result.

By setting a system in motion and observing the outcome, you create a partnership with the AI. This often leads to unexpected and inspiring sounds that you might not have intentionally created. It truly leverages the "generative" aspect of generative AI, making it a tool for genuine exploration and creation, not just for executing simple commands.



## A New Frontier in Sound Design

The goal of this integration is to open creative workflows that are difficult, tedious, or even impossible with a traditional interface.

- **Complexity Made Simple:** Instead of manually adjusting numerous controls one by one, you can execute complex, multi-parameter changes with a single phrase. You can focus on the desired sonic result rather than the mechanics of achieving it.
- **Generative Exploration:** The LLM can generate intricate relationships between parameters that you might not have discovered manually. It excels at creating the phenomena that would be prohibitively time-consuming to program by hand, turning the AI into a powerful partner for sonic exploration and "happy accidents."
- **Hard-wired to the synthesis engine :** The LLM in **Fundamental** is not an abstract layer of control; it is **hard-wired** directly to the synthesis engine. This is a core design philosophy: the results of your prompts are direct translations, not loose interpretations. The AI operates as an integrated sonic assistant, guided by your descriptions.

Mastering this new method of communication takes practice. Learning how to formulate effective prompts is key to unlocking its full potential, allowing you to move from simple commands to designing complex sounds with ease and in incredibly short times.

For this reason, we have carefully designed a bank of presets that utilize the LLM. Each preset demonstrates a specific use case and can be used as a reference for how to control **Fundamental's** parameters with your own prompts.

---

Ultimately, the purpose of **Fundamental** has never been simply to recreate a nostalgic experience by emulating vintage gear. Rather, it is to build a bridge to the past—honoring the spirit of hands-on experimentation and compositional discovery from that era. Leveraging sonicLAB's expertise in unique instrument design, **Fundamental** uses today's technological advances to calculate and render generative soundscapes that would be impossible to create with traditional tools.

I want to salute all the composers and researchers who established these foundational roots for us. I feel fortunate to live in a time of such technological progress, which allows us to develop, create, and produce so freely. It is with great happiness that we deliver this electronic music tool to you.

Dr. Sinan Bökesoy, August 2025.

## A Pay to Go system with No Subscription

### Why the AI Feature is a Paid Service

High-quality AI is computationally expensive. The Large Language Models used in modern applications require massive amounts of GPU hardware and dedicated server infrastructure. Because of this, AI API services are not free and charge based on the amount of data processed (tokens). To cover these ongoing costs, many software companies that integrate these services use a subscription model.

### The Issue with Subscriptions with Musical Production Dynamics

A typical subscription model requires a fixed monthly payment. This often comes with daily usage limits and, more importantly, you continue to pay each month even if you don't use the service at all. We believe there is a fairer way.

### Our Solution: A Fair Pay-As-You-Go System

While it was more complex to develop, **Fundamental 4** uses a **pay-as-you-go** system. You purchase a specific amount of tokens for a one-time price. An account will be created automatically for you, and tokens are only deducted when you make an AI request. Your remaining tokens **never expire**.

This model is ideal for a creative tool. You may have moments of intense AI use for a specific project, followed by periods where you don't need it. With our system, you only pay for what you actually use. **Fundamental 4** remains fully functional with all its core features, with or without the LLM.

### Pricing Details

We aim for fair and transparent pricing. For example, our basic package provides **500,000 tokens** for approximately **\$3**, which is equivalent to 150-200 typical prompt requests in **Fundamental 4**. (Note: Prices in other currencies may fluctuate slightly due to exchange rates.) This price covers our direct costs for using a state-of-the-art AI service, as well as transaction and operational fees.

### Activation and handling of the User Token account :

The process for purchasing and activating tokens is straightforward.

1. **Purchase:** You can purchase token packages for **Fundamental 4** on the sonicLAB online shop.
2. **Activate:** After your purchase, you will receive an email with instructions for activating the tokens within the application.
3. **Use:** Once activated, your token balance will be instantly available in the app.

### Replenishing Your Tokens

When your token balance reaches zero, the account is considered depleted. At that point, you can purchase a new token package and activate it directly within the app to continue using the AI feature.

**Important:** It is strongly recommended that you wait until your current token balance is fully used before activating a new token package.

**Important :** Needless to say, you need an internet connection to use the LLM component on Fundamental 4.

Also as it relies on 3rd party services , the operation can slow down or less responsive than usual at times which we cannot predict.

## Token Consumption on Fundamental 4

The AI feature in **Fundamental 4** connects to a powerful, and paid cloud service to process your prompts. This is because we use a high-quality Large Language Model (LLM) with approximately 400 billion parameters to ensure sophisticated and consistent results.

Access to this AI service is billed based on **tokens**, which are small units of text (roughly 4 characters per token). The cost for each prompt is calculated from several factors:

- **Input Tokens:** The text of your prompt, plus any system instructions.
- **Output Tokens:** The detailed response generated by the AI.
- **Cache Tokens:** Data read from or written to memory to optimize responses.

Each of these has a different cost, with **output tokens** typically being the most expensive.

Each user prompt submission and the retrieval of the response will consume tokens from the user token account. The remaining tokens will be shown and updated at the bottom right corner.



### **Important notes and advices on Token consumption:**

#### *The First Prompt is Key*

Your **first prompt** in a session typically has the highest token cost. This is because it includes not only your typed text but also a larger **system prompt** that establishes the context for the AI. This system prompt is then cached by the service for the remainder of your session.

#### *How Caching Saves Tokens*


For all **subsequent prompts** within the same session, the API reads the system prompt from this cache instead of reprocessing it. This significantly reduces the number of input tokens sent for each new request, resulting in a lower cost.

#### *Recommendation* 💡

To minimize token consumption, it is best to **keep the application open** for the duration of your sound design session. Restarting the application clears the cache, meaning each new "first" prompt will be charged at the higher initial rate.

Restarting the app and submitting many first prompts will maximize the token consumption.

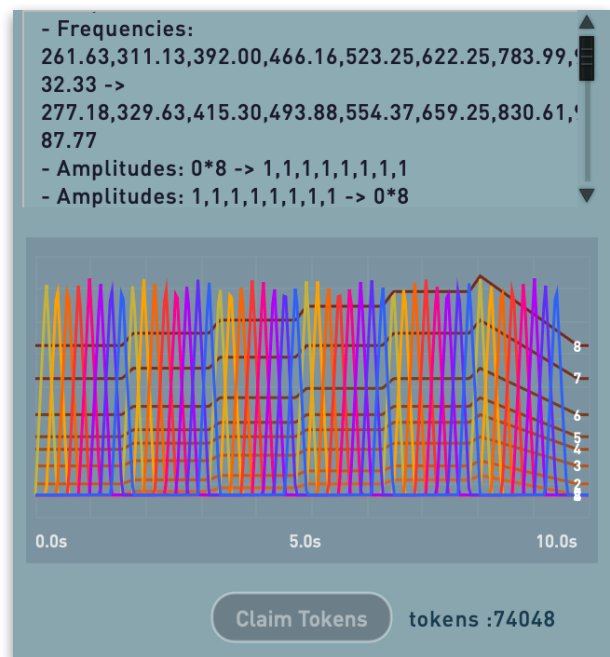
The total API cost is a combination of **Input**, **Cache Read/Write**, and **Output** tokens, each with a different price set by the AI service provider.

Because these token types have different costs, the consumption figure displayed in the app is a **nominal value** that represents their combined total. After each prompt, the API returns a detailed breakdown, and **Fundamental 4** uses the specific cost of each token type to calculate this single, representative value for you. 

Since **output tokens** are the most expensive, prompts that generate a long and detailed response from the AI will result in the highest token consumption. For example this prompt

*“ Set frequencies to the notes of a C minor 7th chord across two octaves beginning with 261.63 Hz. Create an ascending arpeggio by activating the amplitude of each oscillator from 0 to 1, one after another. Each note should have a duration of 0.2 seconds. After the last note plays, the sequence should immediately repeat from the beginning but by added a semitone interval to each note frequency. Let this run for 10 seconds. ”*

will include a sequence of events in its response, to compose its ever changing arpeggio phrase as instructed on the prompt.



When you save a preset in **Fundamental 4**, all AI-related information is stored along with it. This includes your original prompt, the full AI response, and the translated performance data.

This means you don't need to resubmit your prompt every time you load the preset; you can simply press the **Trig** button to activate the saved performance immediately. 

## chapters:

Pages

<b>Integrating and running the Fundamental in your workspace</b>	<b>15</b>
<b>Structure and the controls of the interface</b>	<b>19</b>
<b>Ring Modulation on Fundamental</b>	<b>23</b>
<b>Fundamental GEN modulation generators</b>	<b>28</b>
<b>Stochastic Sequencer</b>	<b>32</b>
<b>The morphing engine</b>	<b>37</b>
<b>Performing the Fundamental</b>	<b>41</b>
<b>LLM interface on Fundamental4</b>	<b>45</b>
<b>Controlling Fundamental4 with Prompts</b>	<b>46</b>
<b>Current Restrictions on LLM</b>	<b>53</b>
<b>Tuning scales on Fundamental</b>	<b>54</b>
<b>High precision Midi control of Fundamental</b>	<b>55</b>
<b>Automation in DAW with Fundamental</b>	<b>59</b>
<b>Appendix</b>	<b>60</b>

## Integrating and running the Fundamental4 in your workspace

**Fundamental4** is an instrument AU/VST3 plugin, and also a standalone app.

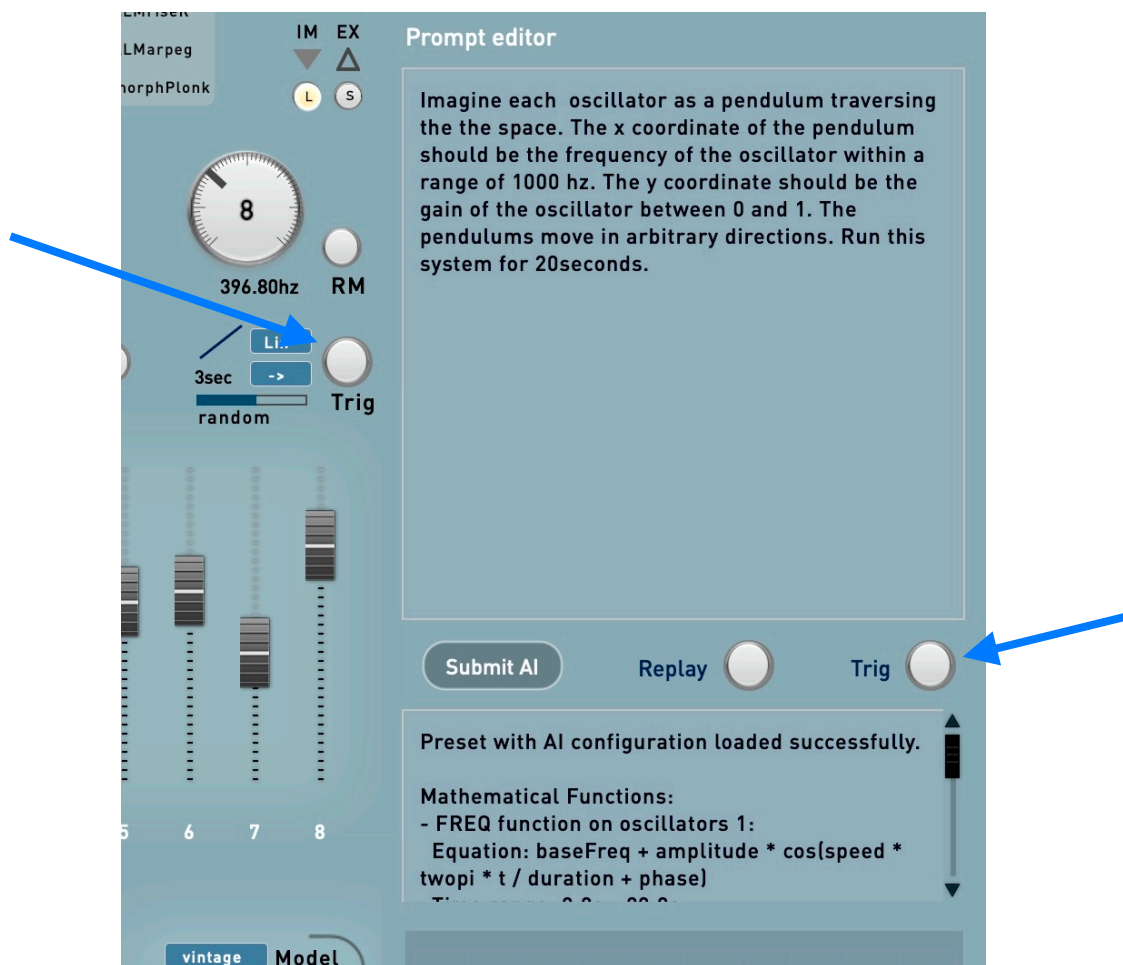
**On OSX :** When you launch Fundamental first time, it will create a directory at `/Users/Shared/sonicLAB/Fundamental4` and copy all its necessary files to operate inside this folder.

**On Windows :** As above, the relevant files are copied to `Documents/sonicLAB/Fundamental4` folder.

Fundamental will load the default preset. If you are using the plugin version, all the settings and parameter changes which you apply on the Fundamental will be saved along with your DAW project and also recalled back when you open that project.

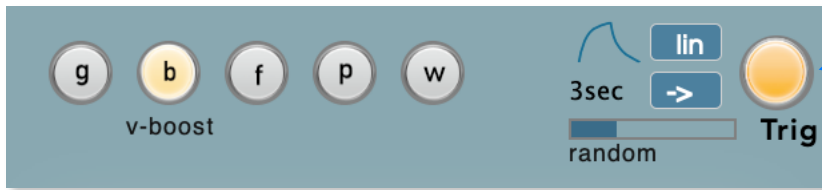
### Triggering Fundamental4 :

- When a preset is loaded , Fundamental4 expects a trigger to run the output. There are two ways of triggering Fundamental4. One is the Trig button to just run the synthesis engine and turn on the audio output. The other ( new in Fundamental4 ) is to run the modulation data created by the LLM (Large Language Model) of the AI assistant which also triggers the synthesis engine and turns on the audio.



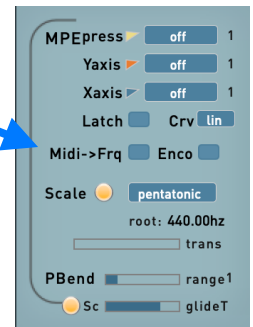
Alternately, you can use the Midi key C5 to trigger the synthesis engine, and C#5 to trigger both the LLM and the synthesis engine together. For the latter, you need to have a prompt response held in memory either sent by the AI or loaded from a preset.

## AUDIO TRIG



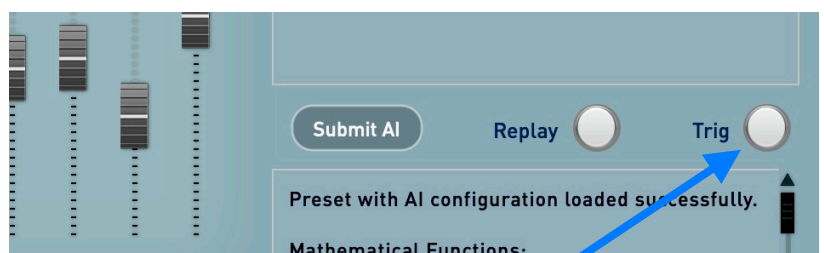
Triggers just the audio output without LLM, you can use also the MIDI C5 note.

- You can also trigger the audio output with a **C5** Midi key ( 72 ) ( **Midi->Freq switch must be off**). A note on triggers it on, and a note off triggers it off.
- For the same purpose, you can also use the *TrigAR* plugin automation on your DAW to define the on / off instances along the timeline of the track.
- Triggering Fundamental happens through a AttackRelease envelope , which has by default a “gate” state, meaning also as pass thru.
- If the **Midi2Freq switch is on**, you can perform each Fundamental oscillator with midi note on input on respective midi channel. This will also turn the Trig state on/off.



Triggers both the LLM modulation and the audio output , you can use also the MIDI C#5 note.

**Important :** You can save the Trig state along with the preset, so that when recalled it will keep the last state. Likewise you can navigate between presets and create transitions without acting on the Trig button again and again.



## LLM TRIG

Triggers both the LLM modulation and the audio output , you can use also the MIDI C#5 note.

- For the same purpose, you can also use the *TrigPerform* plugin automation on your DAW to define the on / off instances along the timeline of the track.

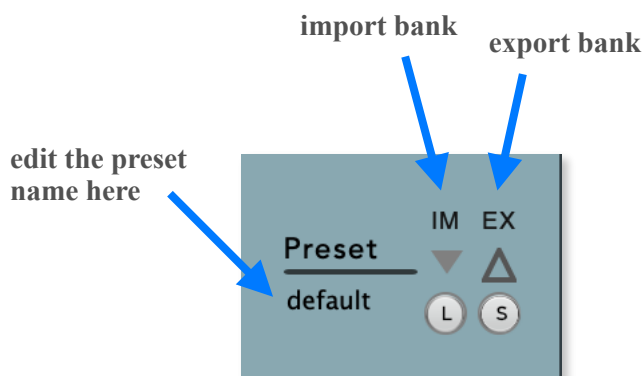
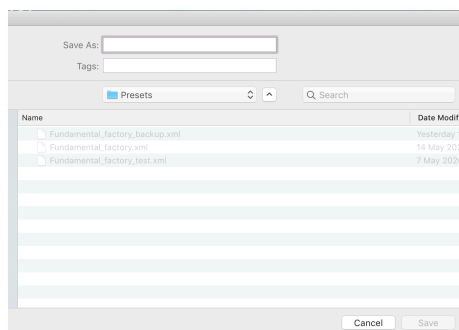
**Important :** If there is no LLM data ( modulation data created through user prompt interpretation ) the LLM Trig button won't perform anything. You have to either submit a prompt to the AI or load a preset which contain an LLM data as well.



**Presets :** Fundamental4 offers banks with 24 preset slots. You can load from these slots (use double click ) , save onto them, import and export other banks from hard disk. Fundamental comes with a number of preset banks. To load and save presets press the respective button as shown below and the current work bank will open as 24 slots.



For both operations, hover your mouse pointer and press on the desired preset slot. When saving on a slot, a confirmation dialogue will open. The Load button will remain lit and the preset loading window will remain floating unless you turn the load button back off. This will give you the possibility to navigate between your presets quickly during a live performance.



You can reach also a copy of Fundamental preset banks prepared by various artists at [www.sonic-lab.com/Fundamental/PresetBanks](http://www.sonic-lab.com/Fundamental/PresetBanks)

You can export the current work bank to disk and also import anytime another bank from disk.

### How Saving Presets Works

- **Saving is destructive:** When you save a preset to a bank slot, it permanently **overwrites** the slot's previous content. Also changes made on original Fundamental4 banks should be exported with a different bank name, otherwise these custom changes will be lost.
- **Changes are persistent:** Therefore first make a copy of an existing bank, and do your changes there. When you reopen the application, your recent changes to that custom bank will still be there.

### ! Important: Best Practices for Managing Banks

To protect your factory presets and your own creations, we strongly recommend this workflow:

1. **Back Up First:** Before you start editing, make a copy of the factory bank file. You can find your banks in the **Fundamental** user directory.
2. **Work on a Copy:** Always load and work on a *copy* of an original bank.
3. **Export Your Work:** When you're done creating sounds, use the **Export Bank** function to save your work as a new bank file with a unique name (e.g., "My\_Sounds.bank").

Following these steps ensures your presets are always safer.

**Alternative preset selection :** When not in **Midi->Freq** mode , you can use the **midi notes** 21-37 (A0-C2) to select a preset between slot 1-16.

### **How to Retrieve the Original Bank Presets**

- You can simply find all the preset banks on this link. [www.sonic-lab.com/Fundamental/PresetBanks](http://www.sonic-lab.com/Fundamental/PresetBanks)
- Also deleting the Fundamental4 work directory from your hard-disk and relaunching the app will recreate the default preset bank data, which you can continue to use.

### **New in Fundamental4**

- Fundamental4 does come with an LLM prompt interpreter which uses an AI API to create synthesis modulation data based on the user prompt content. Hence, this mechanism contains the following components.

*User prompt :* A text content being sent to the AI API.

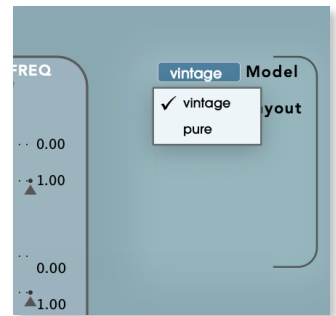
*Response from the AI :* A timeline structured data which will be interpreted by the synthesis engine which creates a human readable score description and graphical representation of the modulation data.

When you save a preset with this content, it will be reloaded as it is back when you select the same preset. Likewise you don't need to submit again the prompt to the AI API to receive and create the same data.

## Structure and the controls on the user interface

At the heart of **Fundamental** are eight independent voices of synthesis. Each voice can draw from one of two sound sources, giving you a choice between vintage character and digital purity.

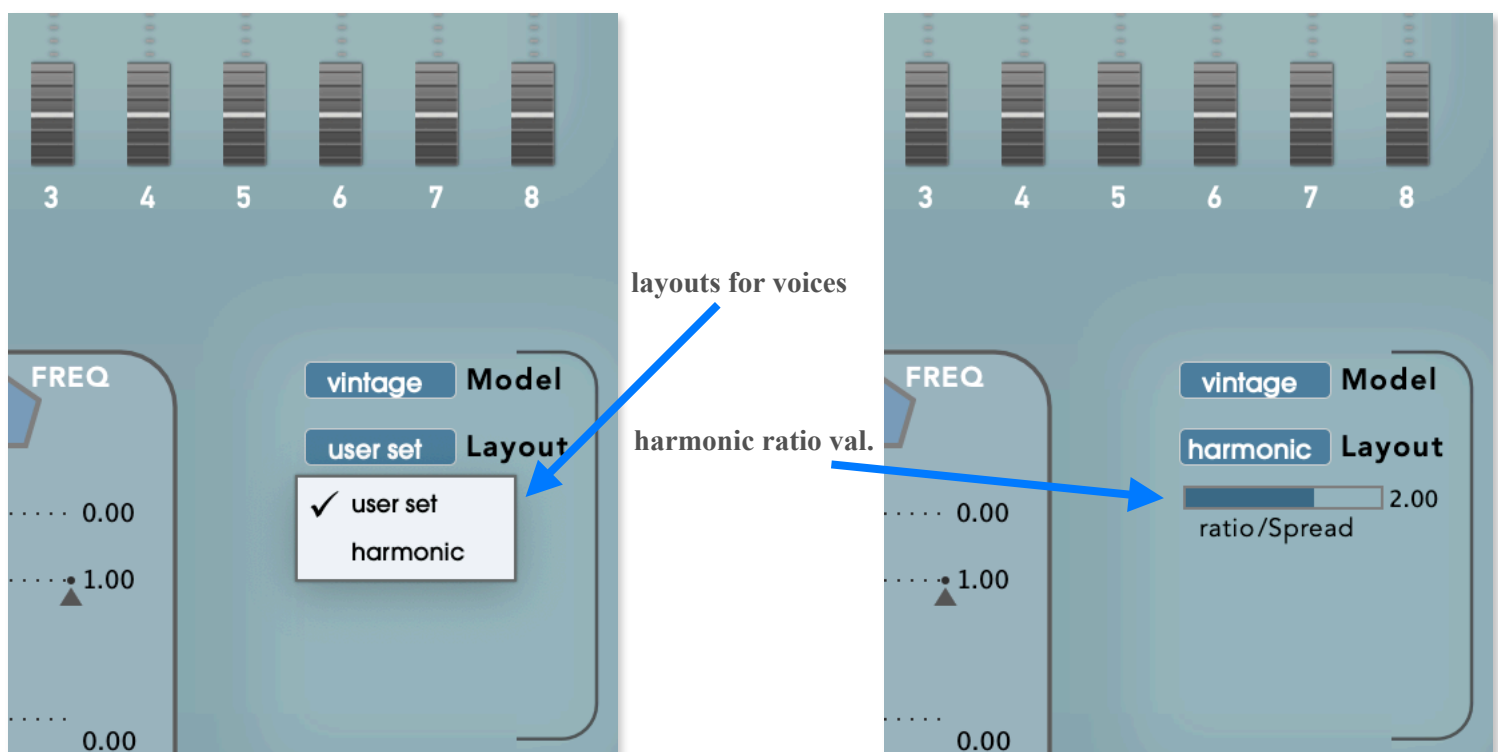
The vintage mode uses high-quality wavetables derived from a classic Rohde & Schwarz vacuum-tube generator, capturing all of its signature warmth and rich sonic detail. For absolute clarity and precision, you can instantly switch any voice to a mathematically perfect sine wave, ideal for clean sub-basses or complex sound design.



The organizational layout of voices consist of 2 modes: User set layout, harmonic distribution layout.

**User set layout** : Each voice operates independently with relevant parameter values assigned to it.

**harmonic distributed layout** : The first voice behaves as a **fundamental** tone, and the others are harmonics of this fundamental distributed with frequencies set with harmonic ratio value ( between 0 - 3). The harmonic ratio can be set also with Midi CC31 message.



The harmonic ratio val. ( ratio / Spread ) defines the frequency relation between the fundamental and the next harmonic.  $1.\text{harmonic} / \text{fundamental} = 2.\text{harmonic} / 1.\text{harmonic} = \dots = \text{ratio} / \text{Spread value}$ .

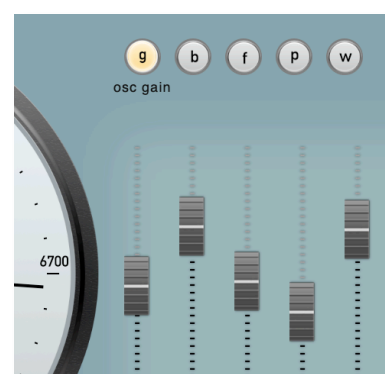
By default it is set to 2.0 ( octave relation ) , you can also apply to it a stochastic modulator ( H.Ratio ) which will change it for every harmonic individually.



Above, you can see a layout of the voices where the fundamental is set to 220Hz and the harmonics are automatically calculated and set with a ratio of 2.0 in this case. Whenever you change the fundamental frequency, the harmonics will adapt to this change.

You can edit the frequency of each oscillator in three ways:

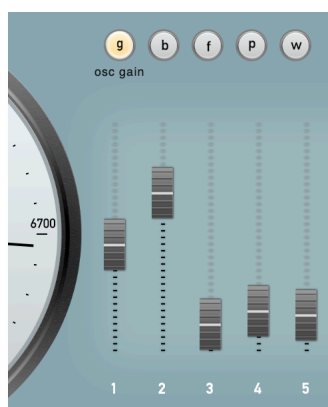
- **Coarse Adjustment:** Click and drag the large rotary dial for quick, sweeping changes.
- **Precise Entry:** Double-click the frequency display above the dial and type in an exact value in Hz.
- **Fine-Tuning:** Use the smaller **Fine Tune** slider for small, precise adjustments to the current frequency.



You can control the level and character of each voice in two ways:

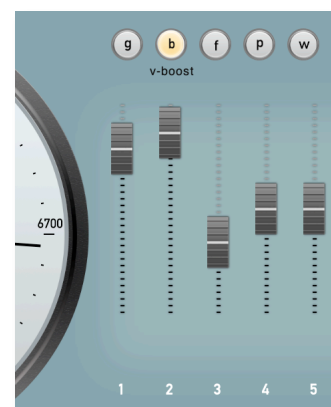
**Gain Slider (Character) :** This slider does more than just control volume; it models the original vacuum-tube hardware. Increasing the Gain not only makes the sound louder but also enriches its timbre with warmth and saturation, just like driving a real tube circuit. Use this to shape the core character of your sound.

**V-Boost Slider (Volume) :** The V-Boost slider provides a clean volume boost of up to 9 dB for its voice. After setting the desired character with the Gain slider, use V-Boost to precisely balance the final levels of all eight voices in your patch.



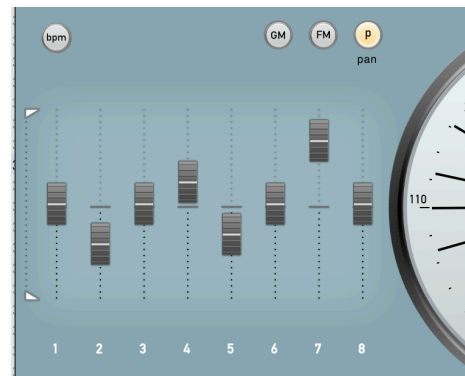
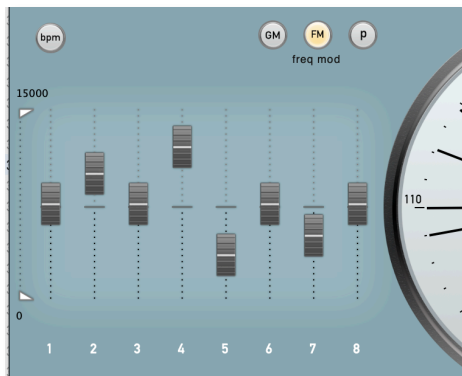
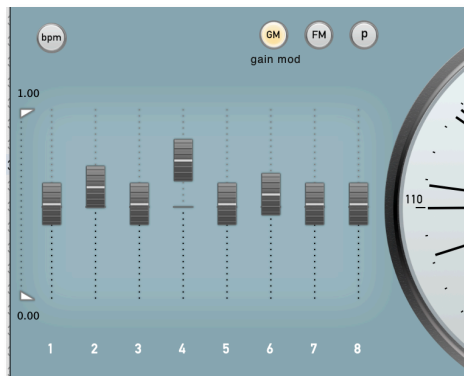
You reach these setups by selecting the appropriate buttons above the sliders.

When you double click on the sliders, they will move to their default position.



The left bank of the sliders are dedicated to modulation depth assignment for each voices gain modulation, frequency modulation and panning modulation. When they are on middle position, the modulation amount becomes zero.

These faders apply bipolar depth to the modulation depending on their position relative to the middle position. For the pan fader set, the upper position is right and bottom position is left oriented.



Technically, the stochastic modulator output value of a voice is multiplied with the relevant modulation depth fader value belonging to that voice.

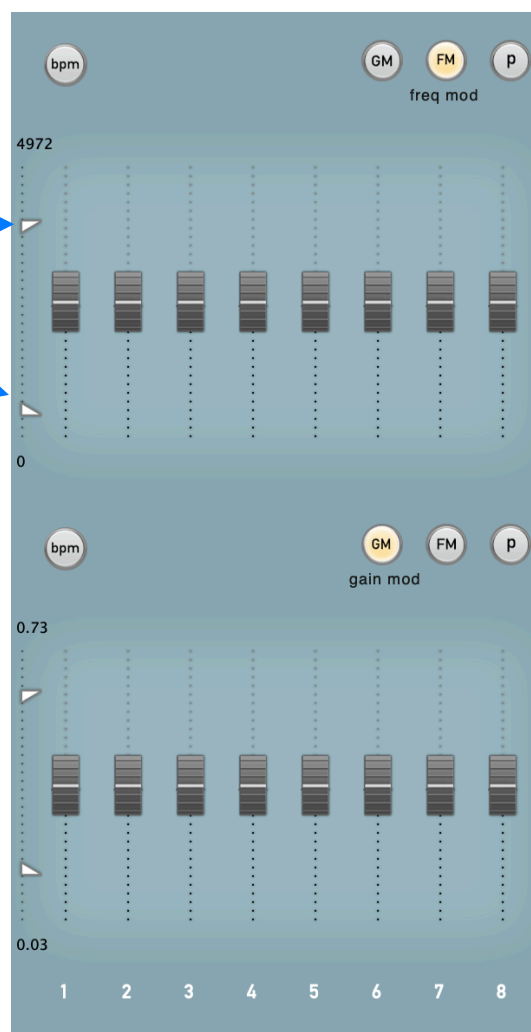
### High & Low Barriers :

One can limit the frequency and gain output values ( output of set value + modulation applied to it ) between Low and High barriers.

The values for the barriers can be set with the left most slider of the left bank. If you are on **FM** modulation interface, then the barrier slider is for the frequency barriers.

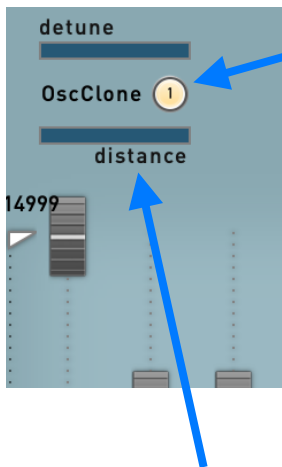
If you are on GM modulation interface, then the barrier slider is for the gain barriers.

Technically any value exceeding the low and high settings will be bounced back inside the barriers range.



## Oscillator Cloning :

Fundamental offers oscillator cloning feature which multiplies of the active number of oscillators.



When turned on, each oscillator has a clone and in total there will be 16 oscillators in charge ( an additional 8 ).

The clone oscillators are real oscillators ( not a trick of DSP chorusing ). They share the same frequency, gain and panning setting. The detuning slider setting will determine the amount of frequency detuning between the clones and the master oscillator.

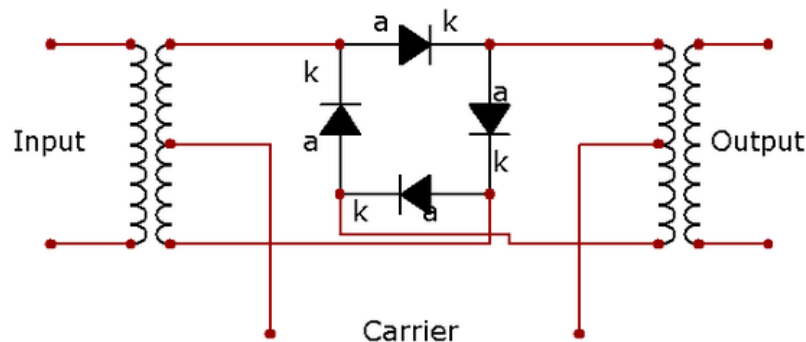
Also the clone oscillators have their unique GEN's running for pitch, gain and panning modulations. The control parameters and stochastic function selection is the same but they will have their unique calculated output.

The distance slider determines the degree of variation of the clone GEN's in comparison to the master oscillator GEN. When the distance is zero, then also the master GEN's output are applied to the clone oscillators. When the distance setting is max, the clone GEN's have their unique calculated output values applied to the clone oscillators.

## The Ring Modulator on Fundamental

Some background : In electronics, the **ring modulation** is a signal processing function, in which two frequencies are combined to generate an output signal. One signal, called the **carrier** is typically a sine wave or another simple waveform and the other is called the **modulator** signal and can be more complicated serving as an input signal.

The name derives from the fact that the analog circuit of diodes originally used to implement this technique takes the shape of the ring: a diode ring.



Ring modulation in a way modulates the amplitude of the signal, both modulator and carrier signal is being multiplied on the same time scale. If the modulator consists of a sine wave of frequency  $f_2$  and the carrier is a sine wave of frequency  $f_1$ , then mathematically it can be shown that the output signal consists two sine waves with frequencies  $f_1 + f_2$  and  $f_1 - f_2$ . These two output frequencies are known as sidebands. If one of the input signals has significant overtones, the output will sound quite different, since each harmonic of this input signal will generate its own pair of sidebands that do not have to be harmonically related.

The ring modulator has been used in electronic music as earliest experiments for synthesizing new tones and also as an effects unit. Namely Werner Meyer-Eppler has used extensively the earliest musical instrument utilizing a ring modulator ( *Melochord* 1947 built by Harald Bode ) in his early days of Bonn Uni. electronic music studio.

Meyer-Eppler's student Karlheinz Stockhausen, a prominent figure in 20th century music, has used the ring modulation in many of his compositions starting with *Gesang der Junglinge* , 1956. He has experimented with ring modulation both for synthetic sound generation and also processing live acoustic instruments directly with it.

Ring modulation has also captured great interest in sci-fi movie sound design as major component such as on *Forbidden Planet* 1956, and *Doctor Who*, 1963.

Harald Bode designed the Bode Ring Modulator in 1961 and also the Bode Frequency Shifter ( reducing one side band from the output signal ) in 1964 as commercially available products for sound synthesis and then later has been licenced to companies like Moog , Buchla etc. One of the very famous analog synthesizers having the ring modulation as a distinctive technique is the Yamaha CS80.



For the simplest scenario, on Fundamental one has to turn on the **RM** switch located above the Trig switch to activate the ring modulation. The 8th oscillator of Fundamental will act as the carrier signal generator and all the other 7 oscillators will become the modulator signal generators.



The figure above shows the gain settings of the 1st osc ( a modulator signal ) and also the 8th osc ( the carrier signal ).

For the ring modulation, the carrier oscillator has to have a gain setting other than zero. The frequency values of the carrier signal and modulator signals can be set with the frequency dials.

The circular display of Fundamental shows the carrier signal frequency ( red line ) and the produced side band frequencies ( with yellow/orange color lines ). For the case above we use one modulator signal ( osc1 ) and the carrier signal ( osc8 ) will produce the side band signals (  $244 \text{ hz} + 690 \text{ hz}$  ) and (  $244 \text{ hz} - 690 \text{ hz}$  ).

Note that the carrier signal is not directly in the output but only shown as visual reference on this circular display panel.

Each **RM modulator** oscillator has its own gain / frequency / boost and panning setting. And the gain / frequency and pan values can be also modulated individually with the powerful GEN stochastic modulators of the Fundamental.

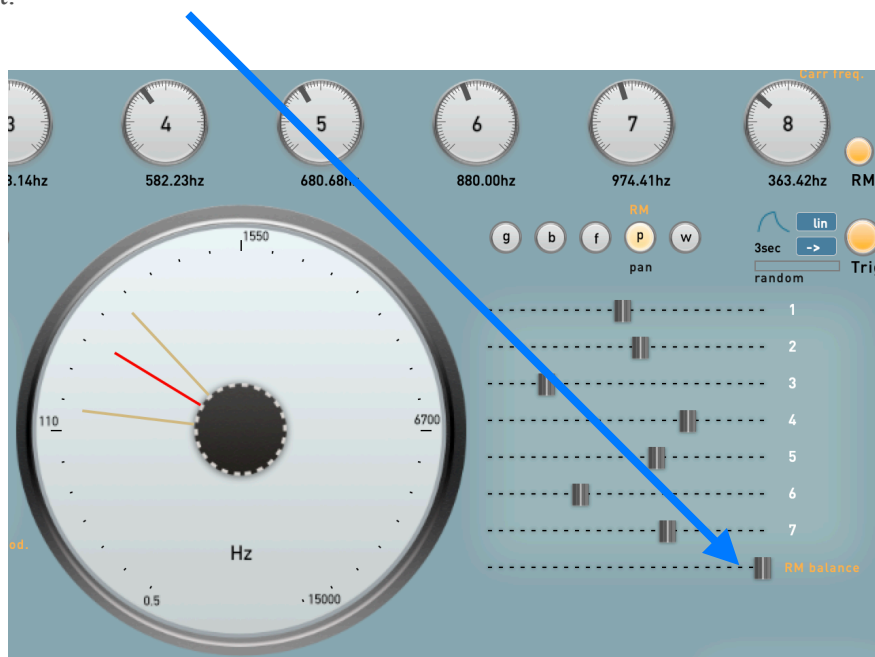
The carrier oscillator ( 8th osc of Fundamental ) has its own gain / frequency / boost setting. Panning is irrelevant to this oscillator. The gain and frequency value of this oscillator can be also modulated individually with the powerful GEN stochastic modulators of the Fundamental.

The carrier oscillator will not be affected by the envelope playback modes, hence its gain setting will remain constant.



You will see helper labels which show the functions dedicated to the carrier osc on the user interface.

As stated above, in Ring Modulation (RM) mode, the **Pan** slider for **Oscillator 8** gets a new job. Instead of panning, it acts as an **RM Balance** control. Moving the slider lets you continuously morph the sound from completely dry (unmodulated) to fully wet (100% ring-modulated), giving you precise control over the intensity of the effect.



When you use **Harmonic Distribution** mode together with **Ring Modulation (RM)**, the first seven oscillators form a harmonic series based on your settings. However, **Oscillator 8** (the RM carrier) remains independent, allowing you to set its frequency freely. For instance, you could have a harmonic series based on 20 Hz for Oscillators 1-7 while the carrier is set to 360.26 Hz.



The Ring Modulation feature is designed to integrate seamlessly into the existing layout of **Fundamental**. Beyond the main **RM switch**, no extra sliders or buttons have been added, keeping the interface clean and familiar.

Importantly, all of **Fundamental's** advanced control features—including **MIDI**, **MPE**, and the **morphing engine**—remain fully functional when Ring Modulation is active. You can use all your performance and modulation tools without compromise.

### about the sonicLAB implementation of RingModulation

At sonicLAB, we are dedicated to achieving the highest audio quality, so our implementation of Ring Modulation (RM) in **Fundamental** was a careful and considered process.

As you know, **Fundamental** offers two distinct oscillator modes: the harmonically rich **Vintage** mode and the clean **Pure Sine** mode. Ring modulating the **Vintage** oscillators is a unique challenge, as their complex harmonics generate a dense spectrum of sidebands.

Our goal was to develop a custom RM algorithm that preserves the beautiful "grit" of these interactions while delivering a pristine, high-definition sound without digital artifacts. This required significant development and does demand more computational power, especially as the algorithm dynamically recalculates signal balances to ensure the **RM Balance** control is perfectly smooth.

We encourage you to experiment with Ring Modulation on both oscillator modes to discover which character best suits your music.

A new preset bank called "**Intro2RM**" has been packaged with Fundamental dedicated to the use of RM. Therefore checking its presets will be a good practice as well.

## The AR ( Attack Release ) envelope on Fundamental

Fundamental offers you another gain modulation mechanism as attack release envelope being mapped to the gain sliders. One can also say that these are oscillator gain level mixer modulations.

AttackRelease envelopes are applied on the current settings of the gain sliders of each voice and there are several options offered :

-> gate, ->= stop at sustain, ->| full cycle , <-> looping

That said, the **gate** mode is a "no" envelope. For the other options, the current voice gain setting becomes the Sustain point of the envelope. When you activate the Trig button, an attack envelope from zero level up to the sustain level will be applied for the defined duration, and the same action from sustain level to release point level.

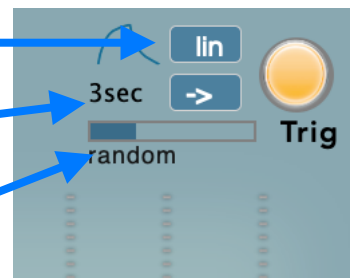
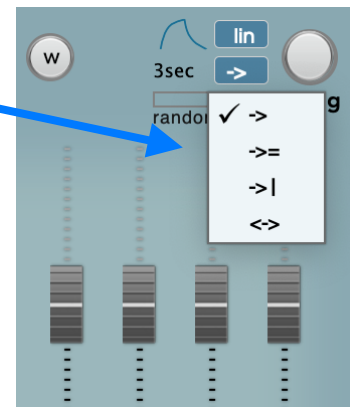
Depending on the randomness setting, the faders will move up to the sustain point and back to release point at different times.

The available envelope shapes are linear, exponential and logarithmic.

choose the envelope shape

type in the envelope duration

apply randomness to the duration for each voice

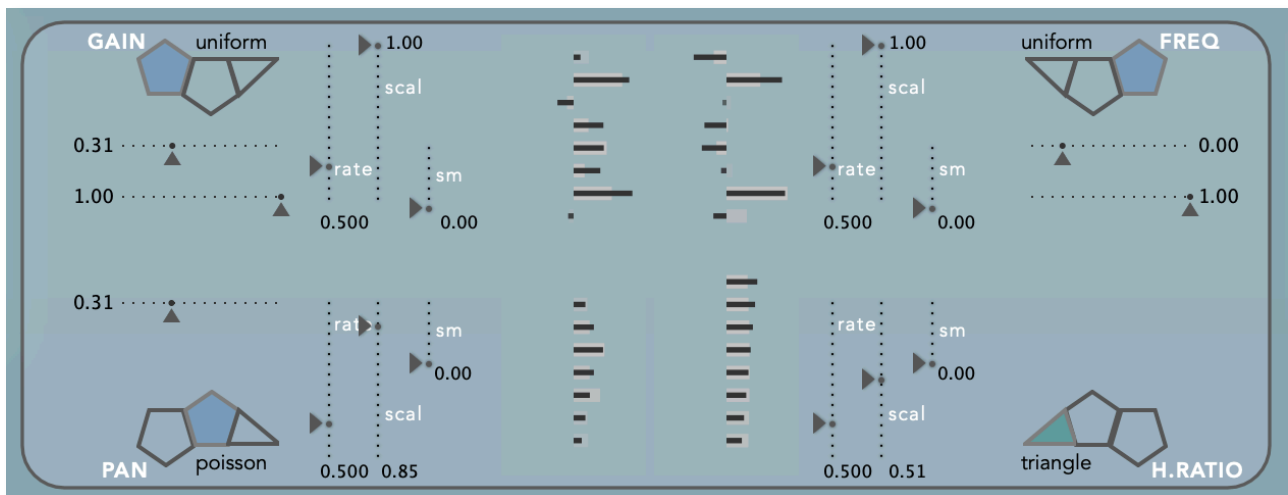


The envelope curves are : **lin** (linear), **exp** ( exponential ) , **log** ( logarithmic ) , **Aexp** ( very fast attack and exponential when decay ) and **ERnd** ( randomly alternating exponential fast / slow attack envelopes ). **ERnd** envelope curvature is also changing randomly at each envelope start.

- I. **->= mode** :The first **Trig** event starts the Attack phase, which rises to the Sustain level and holds there indefinitely. A second **Trig** event is required to start the Release phase.
- II. **->| mode** : Pressing **Trig** activates a complete Attack and Release cycle. As soon as the Attack phase reaches the Sustain level, the Release phase begins automatically.
- III. **<-> mode** : When triggered, the envelope continuously loops through its Attack and Release phases. You can use the **Random** slider to randomize the Attack and Release times for each new loop cycle.

All these actions can be followed with the automated move of the gain sliders.

## Fundamental GEN modulation generators



**Fundamental** features four continuous modulation generators per voice, known as **GENs**. Each **GEN** offers rich stochastic capabilities and is hardwired to a specific, labeled destination: **GAIN**, **FREQ** (Frequency), **PAN**, and **H.RATIO** (Harmonic Ratio).

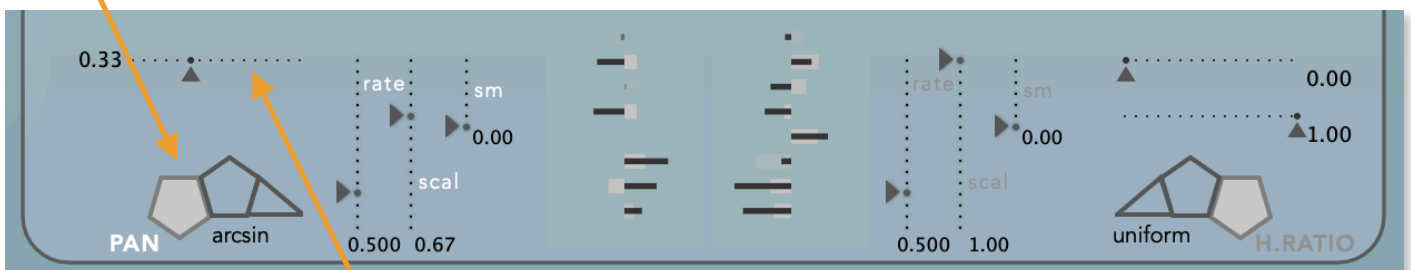
Each GEN offers modulation sources with continuous probabilistic distributions, discrete probabilistic distributions and continuous standard waveform functions as listed below. \*

uniform  
gaussian  
chauchy  
exponential  
weibull  
arcsine  
lognormal  
chisquare  
gamma

poisson  
bernoulli  
binomial  
pascal  
geometric

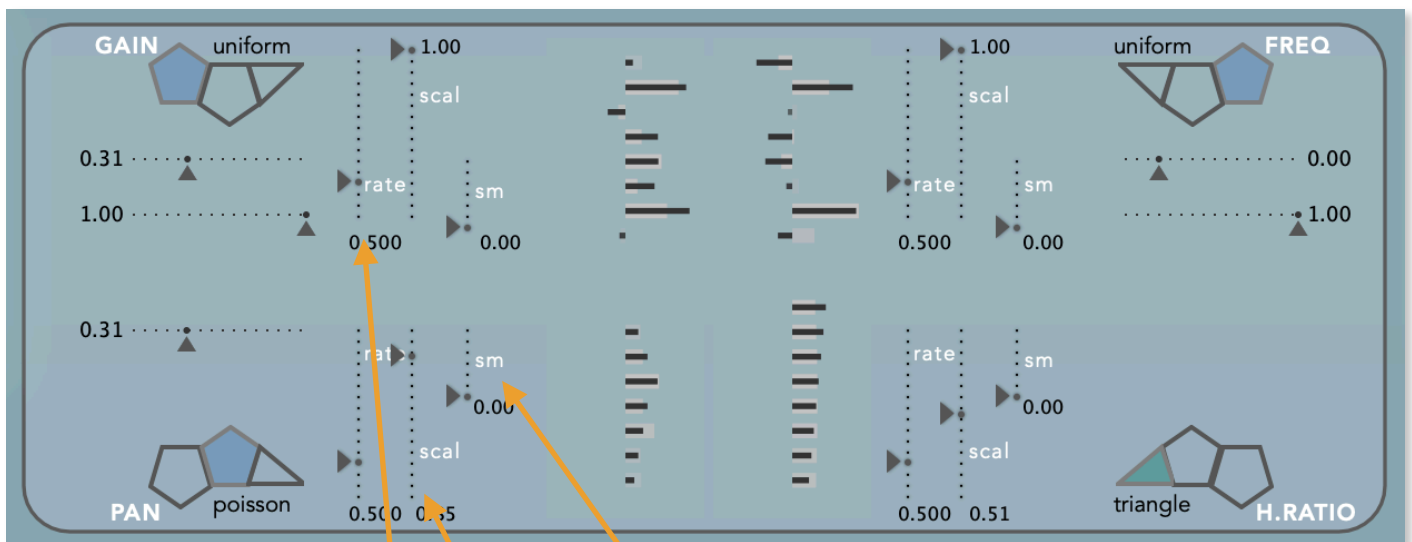
triangle  
sine  
sawtooth  
pulse  
exponent  
morse

Each GEN has relevant button switches to select the desired function category as listed above.



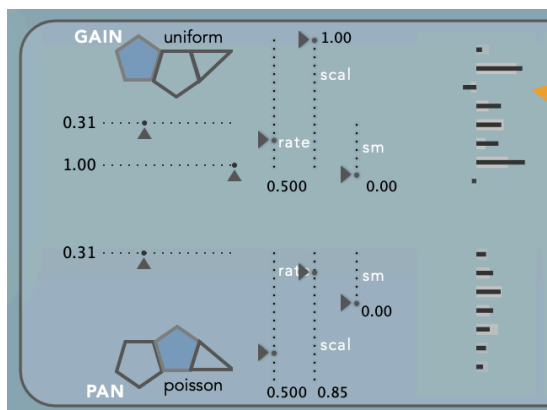
When you select a function, its corresponding parameter sliders will automatically appear. The number of sliders shown depends on the function—some have one or two, while others have none at all. We encourage you to experiment with them!

\* What are stochastic distributions? The ending page of this section answers this question.



Each **GEN** generator has three main rotary sliders to shape its output:

- **Rate** 🕒 Sets the speed (frequency) of the generator in Hertz (Hz). When **BPM Lock** mode is active, this control switches to musical note durations (e.g., 1/4, 1/8, 1/16).
- **Scaling** ⬆️⬆️ Controls the overall amplitude or intensity of the modulation signal. This acts as a depth control for the effect.
- **Smoothing** 🌊 Softens abrupt changes in the generator's output, adding a glide or "slew" effect. Higher values result in smoother, less jagged modulation.



The value display of each GEN represents the unique value calculated for each oscillator. The center point represents value 0.

### The Morse Code translator as a modulation source :

If lab test equipment can be a tone generator, then a Morse code generator can be a rhythmic pattern generator. 📡

Building on that idea, **Fundamental** includes a full Morse code translator. It converts text into corresponding dot-and-dash patterns, which can then be used to modulate the amplitude, frequency, and other parameters of the oscillators.

Because each of the eight oscillators can be assigned a different Morse code pattern, you can create incredibly rich and complex **polyrhythmic** textures. Combine this with **Fundamental's stochastic modulations** to easily explore evolving rhythmic landscapes that go far beyond traditional Morse code.

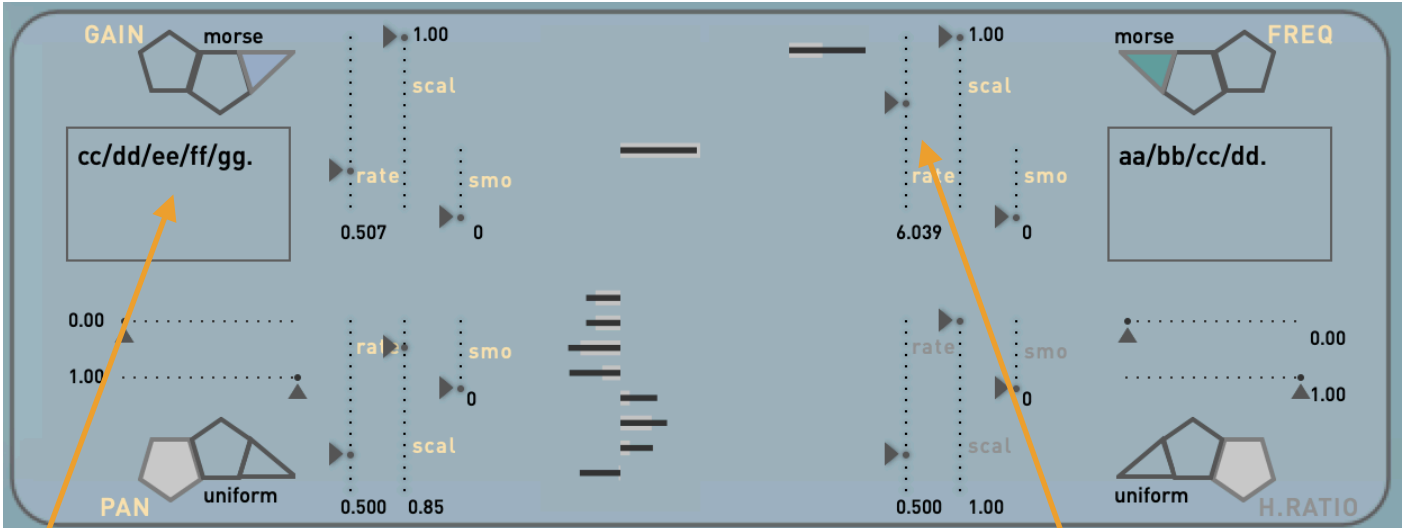


This pattern consists of one type of short / long signal combinations, such as note lengths.

A ·-	J ·---	S ...	1 ·----
B ----	K ---	T -	2 ·----
C ----	L ·---	U ·-	3 ----
D ---	M --	V ---	4 ----
E ·	N --	W ·--	5 ----
F ----	O ---	X ----	6 ----
G ---	P ----	Y ----	7 ----
H ----	Q ----	Z ----	8 ----
I ·	R ·-	0 ----	9 ----

To define a Morse code pattern for an oscillator, you will again use combinations of letters.

As each oscillator has four dedicated modulation generators (**GENs**), the text for the Morse code translator is typed directly into these **GEN** input fields. The specific commands and syntax for this are explained below.



When you select **Morse** as a modulation source on a **GEN**, a text field appears. You can assign patterns to sequential oscillators by separating text with a forward slash (/). A period (.) must be used to end the entire sequence.

For example, typing **cc/dd/ee.** will modulate Oscillator 1 with the pattern for "cc", Oscillator 2 with "dd", and Oscillator 3 with "ee".

The standard **GEN** parameters—**Rate**, **Scaling**, and **Smoothing**—affect the Morse code generator just as they do with any other modulation source.

For example, if you are using Morse code to modulate frequency, **Rate** controls the speed of the dot/dash pattern, while **Scaling** determines the depth or range of the frequency modulation.

Any text you enter for the Morse code generator is saved along with your preset. You are not limited to short phrases; feel free to experiment with much longer sequences to create complex rhythmic patterns. Please note

By assigning different Morse patterns to multiple oscillators at once, you can easily create rich **polyrhythmic** combinations.

When you combine these rhythms with **Fundamental's** other features—like quantizing the output to musical scales, limiting modulation ranges, and adding **stochastic modulations**—the simple Morse code generator transforms into a true **generative musical tool**.



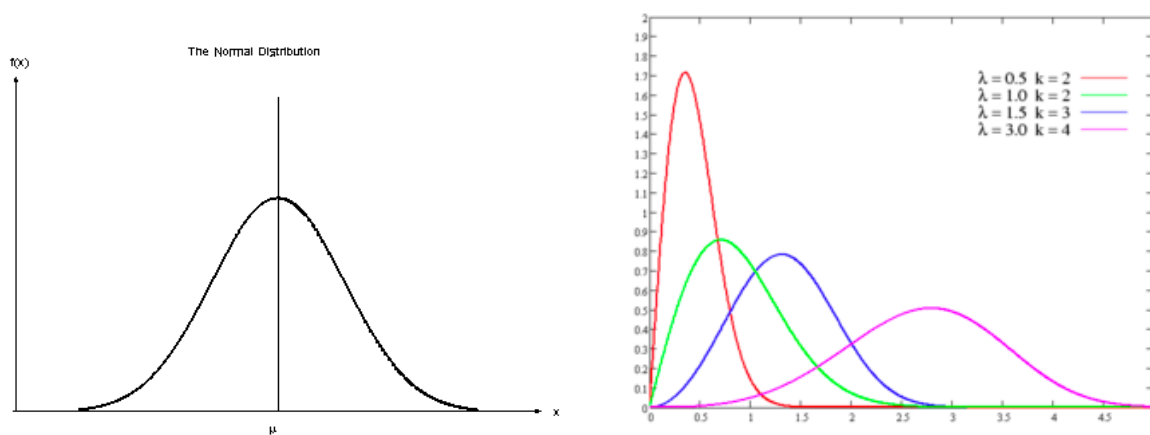
## What is a stochastic function / probabilistic distribution ?

Stochastic processes are widely used as mathematical models of systems and phenomena that appear to vary in a random manner. It is a field of statistics, and the term random **function** is also used to refer to a **stochastic** or random process, because a **stochastic** process can also be interpreted as a random element in a **function** space. A type of random function can be modeled with a probability distribution function showing distinctive behavior when distributing the values as its output.

If you flip a coin, both sides have the same probability of showing up, so this can be modeled with a uniform random distribution. For example, if we have a light bulb which goes on and off randomly due to a cable connection failure / electric spark, this is not a uniform random distribution but can be modeled with a poisson distribution. The movement of a mass of molecules in gaseous spaces can be modeled with brownian motion ( a random walk ).

Most of the stochastic functions have parameters, a value of certain range to change the behavior of the stochastic function. The reason that the **Fundamental** offers many types of stochastic functions is to present you different types of ‘distribution shapes’, which give a distinctive character to the process being used for.

Below, you can see the gaussian / normal distribution and the Weibull distribution shapes ( which varies according to its parameters ). As you can see, with the normal distribution of events, an x event has a higher possibility to happen towards the centers of the bell shaped curve.



Music is a distribution of events in organized manner and stochastic functions let us control the randomness between order and total disorder. This is applied heavily in 20th century contemporary music and computer music by the composer Iannis Xenakis \*.

sonicLAB products offer similar mechanisms and inspiration in form of modern tools to today's composer / sound designers. We can assign these distribution characteristics to various musical parameters and sound synthesis parameters at various levels. The GEN modulation sources are themselves LFO's which modulate various parameters of the sonicLAB PaSSBot LFO with a rich palette of stochastic distributions.

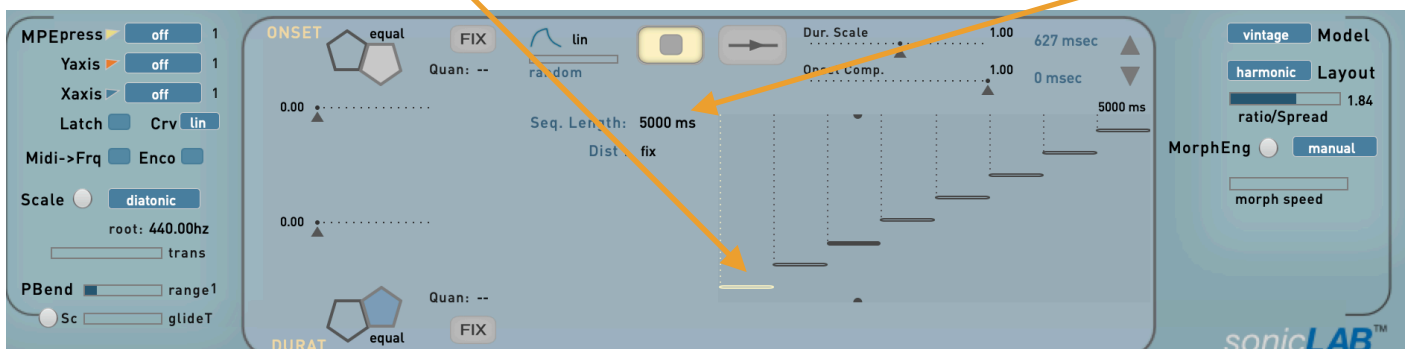
## The Stochastic Sequencer on Fundamental



Fundamental3 incorporates a stochastic sequencer. What does a sequencer do in general ? Basically it defines discrete note events with specific onset times and durations on a timeline. And this event timeline is being performed with a certain speed / tempo according to the start/stop actions of the sequencer.

Fundamental3 sequencer defines a note event for each of its oscillators. The sequencer activates the oscillator with the onset of its event (gate on ) and performs it along the duration of that event, and at the end it gates the oscillator off.

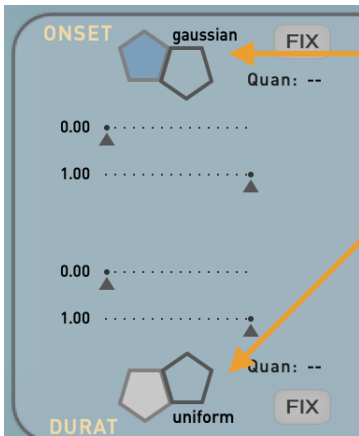
On the example case below the 8 events are distributed equally inside a sequence length of 5000ms.



That said, the event visuals show horizontally the onset and duration of each event and positioned from bottom to top beginning with Oscillator 1 note event until Oscillator 8. The vertical positioning of these events on this visual here has nothing to do with their frequency values. The frequency values are edited on the oscillators pane.

The sequence length can be edited directly next to the **Seq. Length** label. It can be also changed stochastically with each turn on the sequence by choosing a distribution next to the **Dist** label.

Also the standard Play/Stop button and play forward / forward-backward buttons are obvious performing modes.



Just like on the GEN modulation controls, the Onset and Duration values for the note events can be calculated with stochastic functions. For the case on the left, the onset distribution is being calculated with a gaussian function and the durations are distributed with a unifom random function.

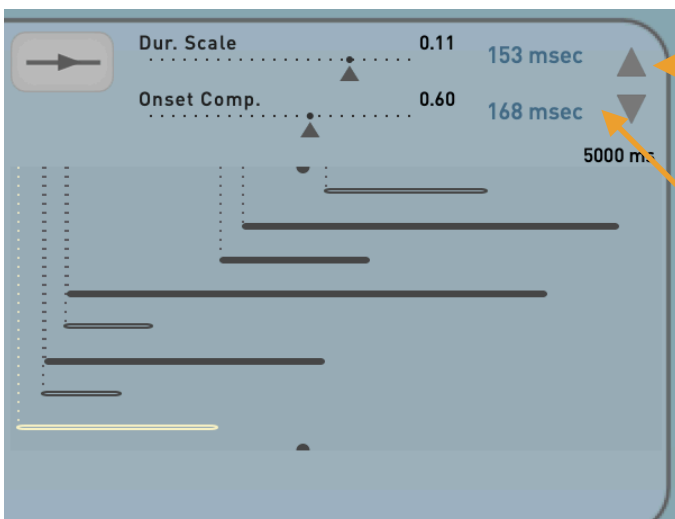
The distribution function parameters can be set directly with their sliders.



When the sequencer is running, a time pointer scrolls from left to the right on the sequencer events visual area. The intersecting events with the time pointer are blacked out emphasizing that they are currently performing.

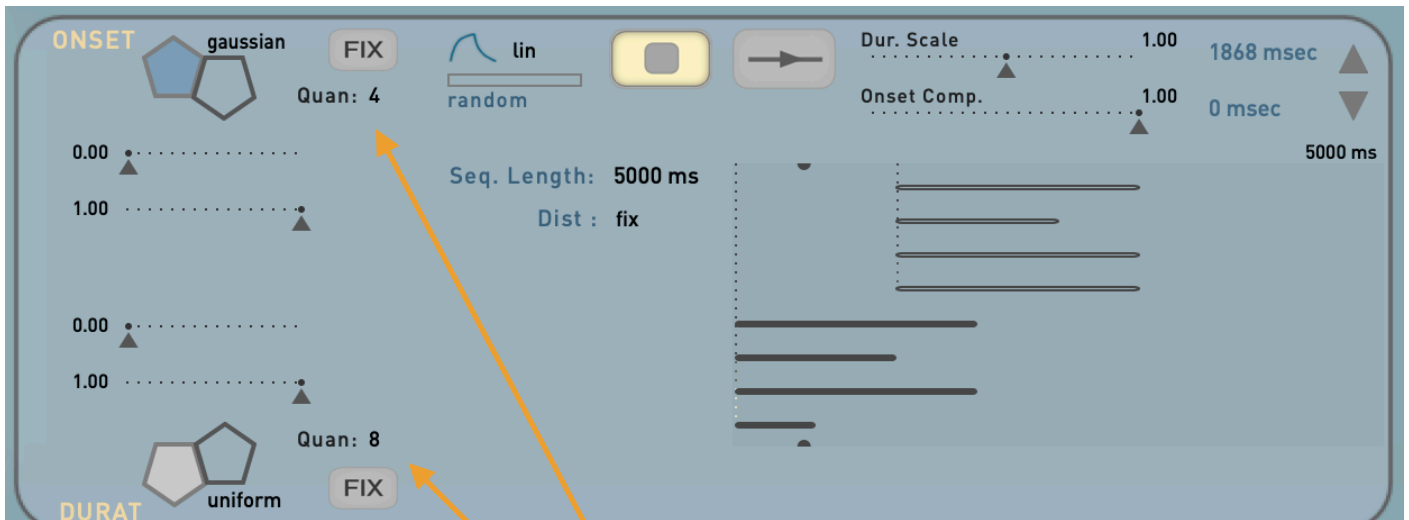
One can further manipulate the onset and duration of the events with **Dur. Scale** and **Onset Comp.** sliders.

The Dur. Scale multiplies all the duration values with its value ( between 0 - 2 ) and the Onset Comp. slider multiplies the onset values with its value by shrinking or expanding the onset distributions in time.



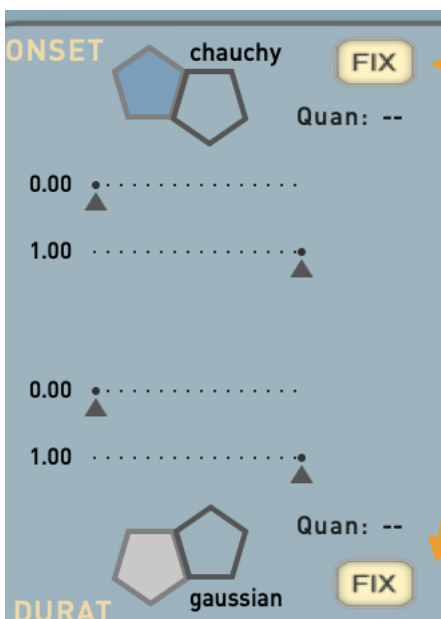
You can even assign a custom value for each events onset and duration setting. Just select the event you wish to edit with the up / down arrows.

The selected event will be highlighted and you can type in the onset time and duration value in milliseconds.



One can use the event quantizers for quantizing the onset and duration values after they have been distributed stochastically / edited manually.

The quantize values are specified just like on any DAW, as proportional values of a bar length. For the case above, you see that the onset durations are quantized with quarter note values and the durations are quantized with 8th notes. As expected, the events are beginning together on their quantized time values creating some chords.

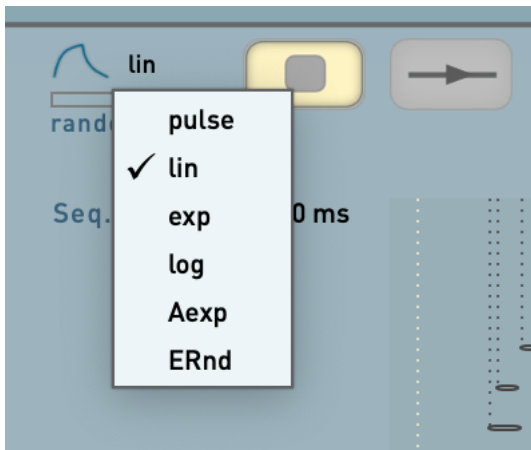


You can use the **FIX** switches to freeze the onset and duration values on their last distributed states. This action will hold their values until you deactivate the **FIX** switch.

When you save a preset, the last calculated onset time and duration values for each event will be saved as well. Thus, if the **FIX** switches were turned on when you save a preset; the next time you open the same preset, the sequence event layout will appear the same.

You can still alter the **Dur.Scale** and **Onset Comp.** sliders even when the **FIX** mode is active.

To create a totally custom sequence , just activate both **FIX** switches and use the custom onset / duration value editing tool for each event as explained above.



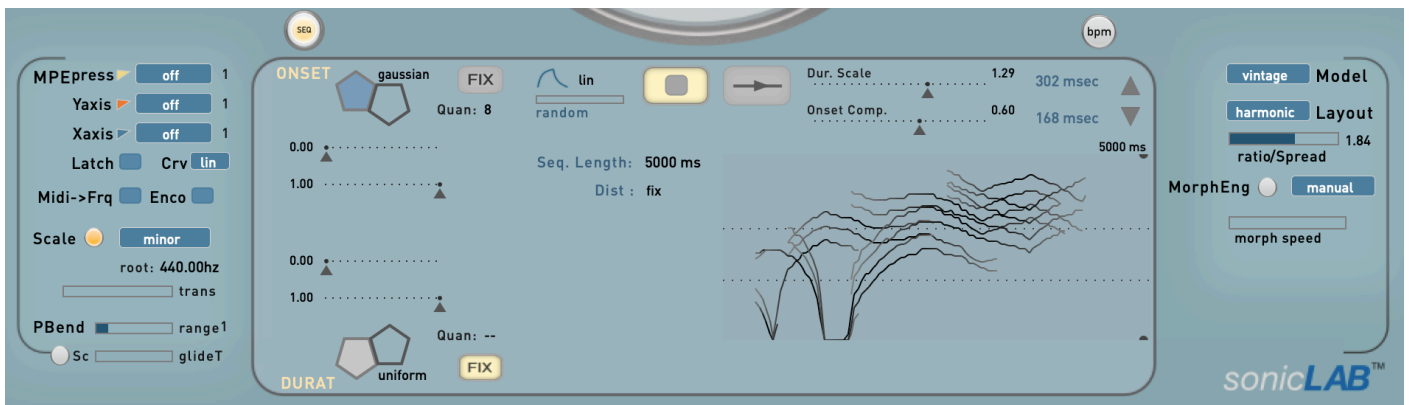
There is an additional envelope generator on Fundamental3 specific to the sequencer events applied to each oscillator. This is independent from the oscillator gain level mixer envelopes.

Fundamental3 sequencer actually mutes the oscillator when there is no note event for that oscillator and gates it back on when there is an event. And the type of the envelope applies to this note gate and creates the expected amplitude change.

You can select different types of amp. envelopes from the list and they will be applied on each event of the sequencer.

The length of each envelope is proportional to the duration of the event. And with the random slider this can be randomly changed as well.

Again, Fundamental3 incorporates two envelopes systems, one on the sequencer events and one on the oscillator gain level mixer, which do run together and will create complex combinations at will.



You can reach an alternative Sequencer visual by pressing the key “v” on your keyboard, which shows now the oscillator frequencies and the realtime change of them according the GEN modulation applied to the oscillator frequencies.

On the example case above, you can see the staircase change of the frequencies, because the tuning scale of Fundamental is active ( the minor scale is selected. )

Also the gain modulation of the oscillators are applied as color changes on the sequencer visuals. The soft passages are shown with light grey color and the high volume passages are shown full black.

## How to use the sequencer in a DAW

As you can clearly see, the sequencer has a Play/Stop switch button of which state can be saved along with the preset.

When the sequencer is not performing, Fundamental3 runs the oscillators according to its gain mixer levels and envelope settings. You might not want this to happen, when you want to run Fundamental3 sequence only on your defined parts of your tracks.

Therefore it is best to use the Trig On/Off button of Fundamental3 to turn the application on when you would like to enter the sequence part and turn it off when leaving the sequence part. You can do this easily by using the Midi Note C5 / 72 ( C4 on some DAWs) which turns on the Trig switch with a note on and turns it off with a note off instance.

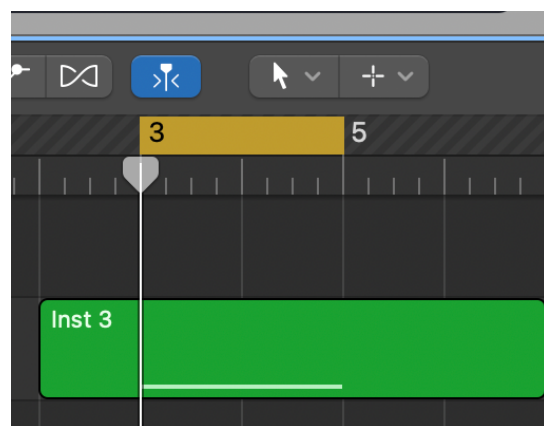
When the Trig turns on, the Fundamental3 sequencer will reset its position, which is convenient, and the same when you use the Play button of the Sequencer.

Eventually if the sequencer is in Play mode, the Trig button will turn on and off the application at the moments you would like it to perform.

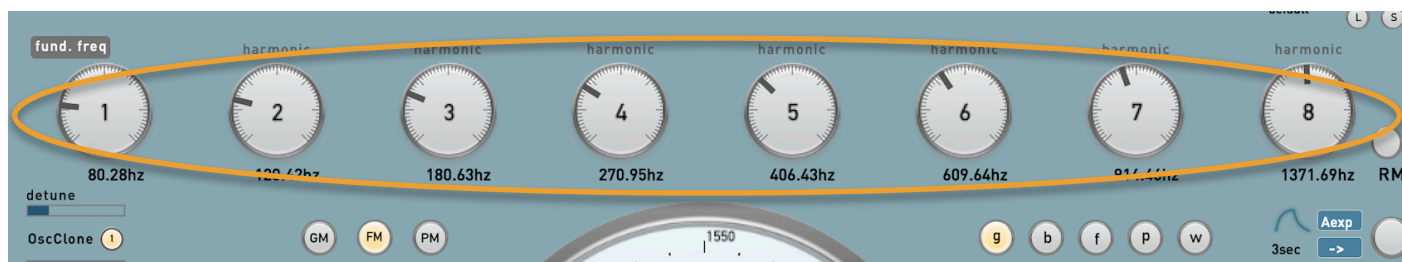
As example case, on the right, the Trig switch will be turned on on bar 3 and turned off on bar 5. This will perform the sequencer on the Fundamental3 between these moments. You can loop that region of course. Each note on event will trigger an envelope reset for all the oscillators in the sequence.

If you would like to play and stop the sequencer remotely on the track you can use the Midi Note D4 message.

As said, the moment it stops, Fundamental3 runs the all the oscillators in drone mode with the gain mixer level settings and its envelopes.

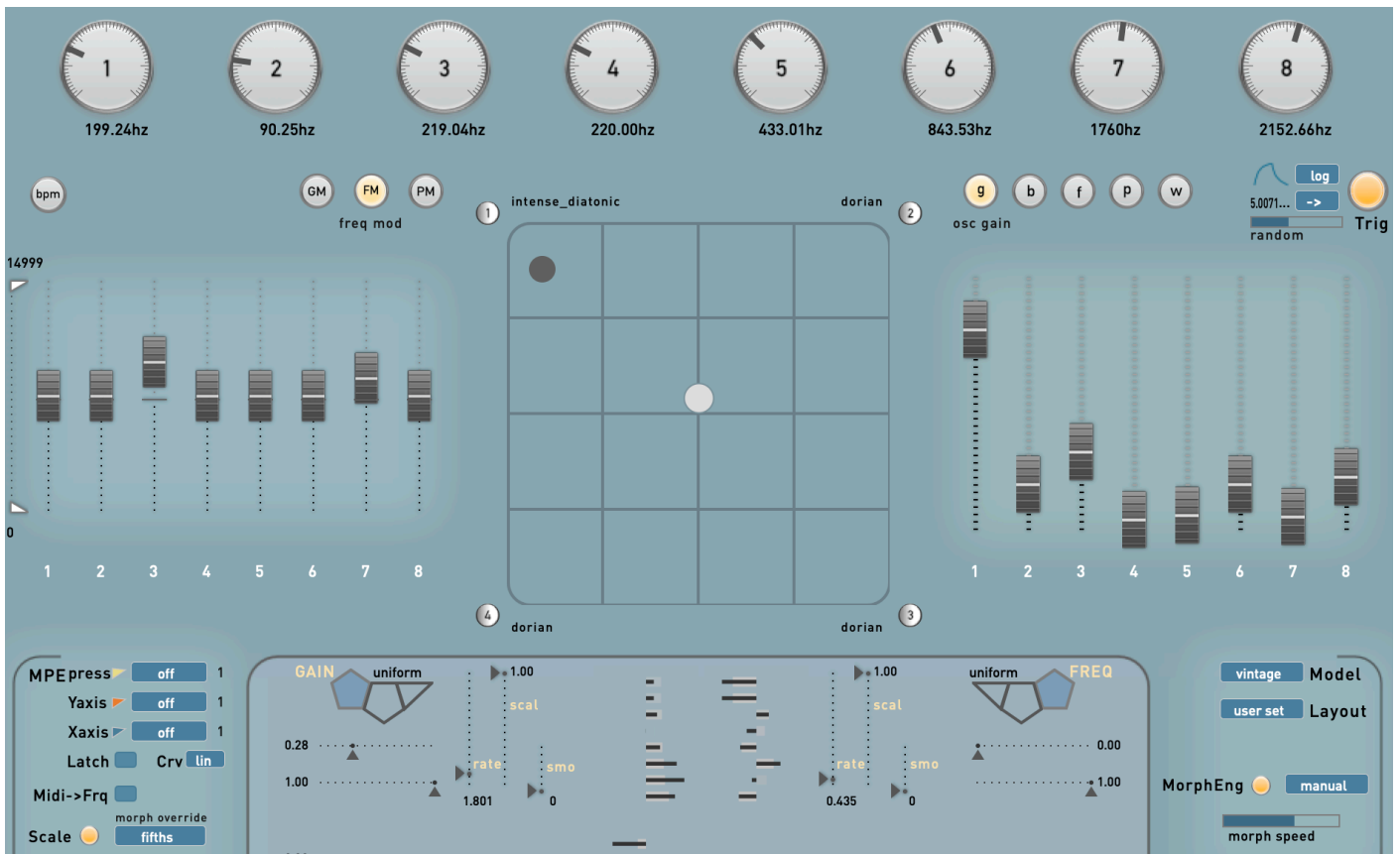


*Pay attention to not keeping the mouse pointer on any oscillator freq. knobs when performing on the track, otherwise they will capture the midi C4 / D4 as their frequency.*



Alternately, you can also use plugin automation for these Trig and Play/Stop switches, however using midi notes is more practical.

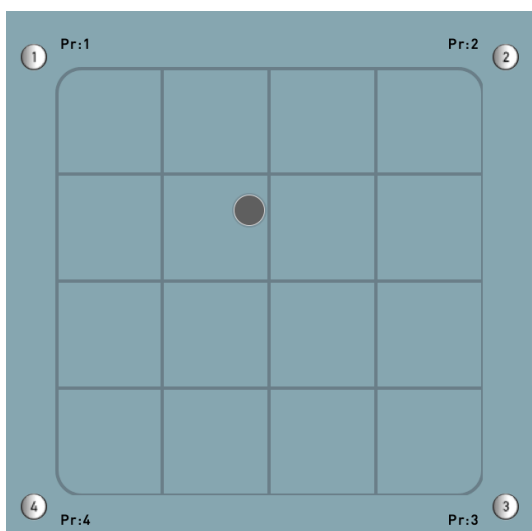
## The Morphing Engine on Fundamental



The morphing engine has the purpose of gradually morphing the parameters between 4 unique Fundamental instances. It has been derived from the sonicLAB Cosmof Saturn, which has an audio rate parameter space morphing engine, highest precision smooth passages along with visual projection.

These 4 instances can be custom assigned variations of the current work preset of Fundamental, or can be also chosen among the presets of the loaded preset bank.

On the example shot below the preset parameters instances are chosen from Preset1, 2, 3, and 4 as you can see them at the corners of the morph panel. The small filled circle inside the panel area represents the morph state, which you can drag with mouse or sending CC messages respectively.



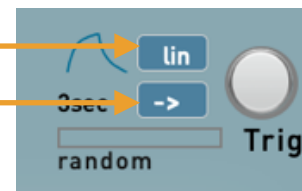
The distance of this circle from the corners defines how much each instance contributes to the current morph state. The closer the circle to an instance corner, to more that instance contributes to the morph state.

Morphing works best on continuously altered parameters where one can follow a gradual smooth change. *Osc pitch, gain, freq mod.* are example parameters. However changing the *envelope type* introduces abrupt change in audio for example. Therefore some Fundamental parameters have been excluded from the morphing engine. Below we explain each of them.

A Fundamental instance on the 3rd slot. Each slot represents a unique Fundamental parameter space.



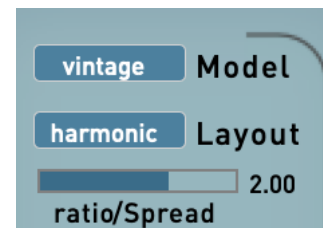
Some parameters on the envelope section such as (envelope type, and the playmode ) are excluded from the morphing engine, as their change introduce abrupt jumps on the audio process.



All MPE / midi parameters are excluded as well from the morphing engine parameter space.



The sine oscillator model and the oscillator layout parameters states are excluded as well because there happens no gradual change on sonic quality when you change them, they are structural high level parameters.



As you would expect. all MorphEng parameters are excluded from the process.

Likewise you have now an idea about the logic behind this organizational decision.



*What happens when you change an excluded parameters, for example the envelope type ?*

This change will be effective on all the 4 instances used by the morphing engine. For example when you load a preset to an instance slot , that preset envelope type will be effective on the other instances as well. We will give more examples about that.

## Operating the Morph engine

To activate the morph engine , click on its button and automatically the morph panel visual will pop up. If you haven't assigned any instances to the morph slots before, the application will load the Preset1, 2, 3 and 4 to the slots respectively.

You can switch between the morph panel visual and the Fundamental circle by using the **left and right arrow keys** on your computer keyboards. Alternately you can use also the 'p' key on your keyboard.

There are two types of morph operation : *manual or stochastic*. The stochastic type has additional parameters like distribution speed and distribution range.

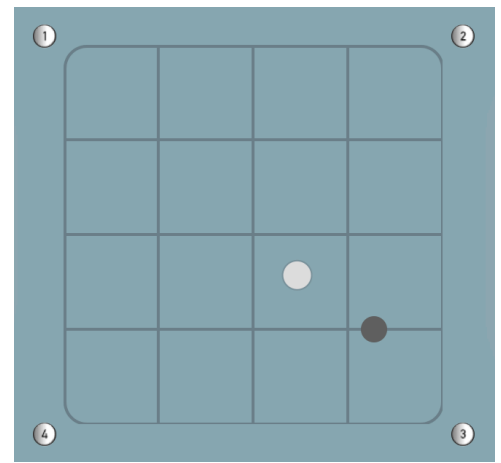




In both cases of morph operation, if the morph speed is not at maximum you will see a white circle following the lead dark circle.

Now the morph engine is smoothing out the abrupt changes on the dark circle position which you apply. At max morph speed, the change is identical but it can be quite slow at low speeds according to your needs. Hence in fact, it is the white circle what you listen to.

When the *stochastic morph mode* is selected, a random change will be applied to both x and y positions for the dark circle. Its speed is defined by the *dist speed* slider and the randomness range is defined with the *dist range* slider.



## Assigning new instances to the Morph Engine slots

When the Morph mode is turned on, all the user interface control is overridden by the morph engine ( except the parameters which are excluded from the morph process as listed above )

When you change the location of the morph circle, the current morph state will be calculated and this parameter state will be projected on the user interface elements. ( you will see them moving / changing by themselves.

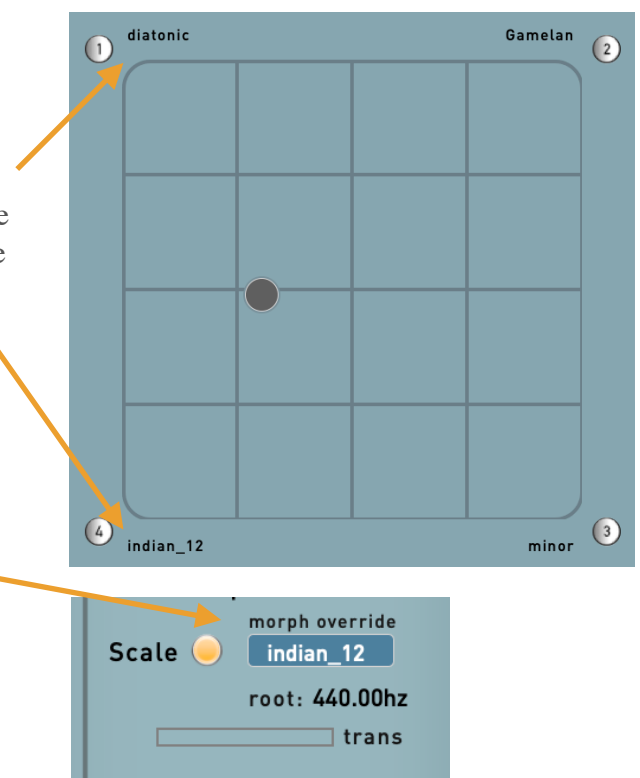
When you turn the Morph mode off, you will gain access to the UI controls back again. You can turn the morph engine on and off quickly by using the “m” key on your keyboard. Turning on the morph engine automatically switches to the relevant user interface.

## Morphing between tuning scales

When you turn on the tuning scale while the Morph Engine is on, then the engine will continuously morph between the tuning scale settings stored on each morph slot. These settings will be shown next to each slot automatically.

For a smooth change, we suggest that you keep the morph speed less than the maximum level.

Also you will see a notification that the preset tuning scale setting will overridden by the morph engine.

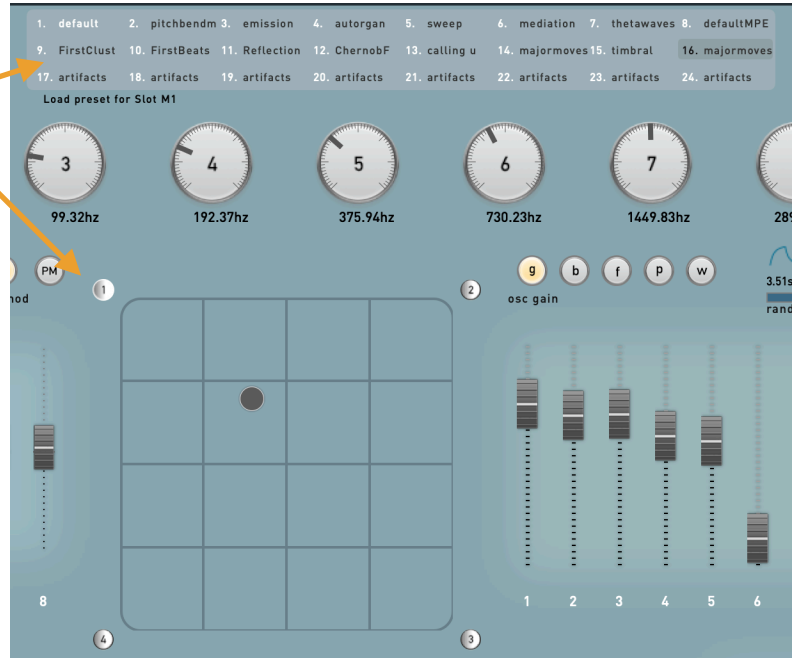


## - load a preset to a morph slot

To import a preset to a morph slot, first turn off the morph engine. Then click on one of the slot buttons and the preset bank window will pop up. Your selection will load the continuous parameters of the preset to be integrated in the morph process.

If you wish to cancel the load process just click again on the slot button.

If you wish to load a preset from different bank, first import the bank with the “import bank button” and then click on the morph slot to load the preset from the new bank.



## - export the current state to a morph slot

You can assign the current parameter state of Fundamental to a morph slot. This can be your departure preset with altered parameters or the current morph state which you would like to assign to a specific morph slot.

To do that, **control+click on the morph slot** and this will save the current state to that slot to be integrated in the morph process.

For exporting the current state to a morph slot, you don't need to turn off the morph engine. (with the exception of editing morse code for different slots ) The idea is that you can change the slot content on the fly and have a continuous experience.

When your morph setup is using tuning scales and turn off the morph engine, then the effective tuning scale will be your presets last set tuning scale as indicated on its menu item. This can be different from the calculated morph engine scale, no surprises !

## - Saving your preset with morph settings

All the morph slot contents and the morph engine settings will be saved along with your preset and can be imported back. Just use the usual “preset save” button to export your work preset to a preset slot on the last imported preset bank.

The sequencer parameters on the sequencer pane are not included to the morphing parameters.

## Performing the Fundamental

We have designed the **Fundamental** as a different instrument than the standard direct to midi key mapped instruments.

First of all, to achieve the deep sonic offerings of the Fundamental and possessing the frequency resolution of it, performing the Fundamental just with standard Midi scales is not enough. Otherwise it would easily become an organ like sounding instrument playing usual chord structures, which we don't prefer to happen.

Fundamental offers full MPE support and also can receive standard MIDI note and defined MIDI CC messages. Below we will explain these cases.

### Fundamental and MPE input

Fundamental makes full use of MPE controllers within following structure :

Each voice is hardwired to a defined MIDI note;

1. Voice : Midi Note **57** ( **A3** )
2. Voice : Midi Note **59** ( **B3** )
3. Voice : Midi Note **60** ( **C4** )
4. Voice : Midi Note **62** ( **D4** )
5. Voice : Midi Note **64** ( **E4** )
6. Voice : Midi Note **65** ( **F4** )
7. Voice : Midi Note **67** ( **G4** )
8. Voice : Midi Note **69** ( **A4** )

Midi Note **C5** controls the **Trig** switch, note on turns it on, and a note off turns it off. *Pay attention to not keeping the mouse pointer on any oscillator freq. knobs, otherwise they will capture the midi C5 as their frequency.*

The well known gestures of an MPE controller is as following:

- Pressure gesture on a midi key ( after a midi note on event )
- Glide along the Y axis of the pressed key.
- Glide along the X axis of the pressed key ( bending )

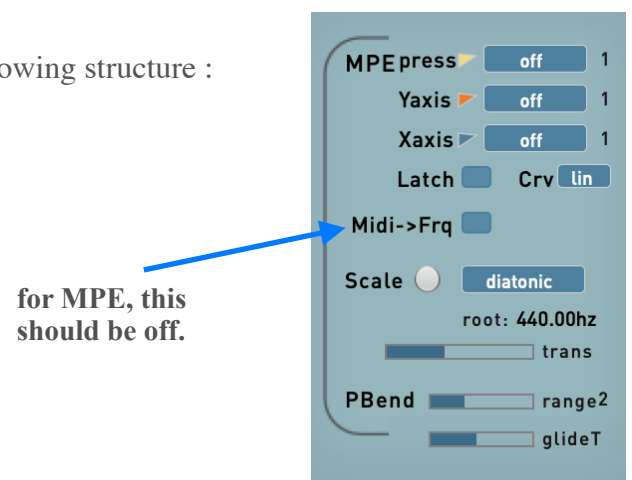
You can map these gestures to desired parameters acting on each voice of the Fundamental. The voice number is related to the pressed midi key as shown on the table above.

The available gesture mapping to parameter listing is likewise :

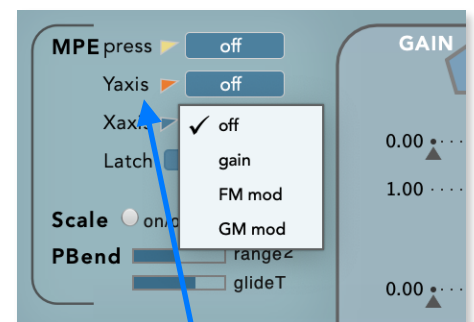
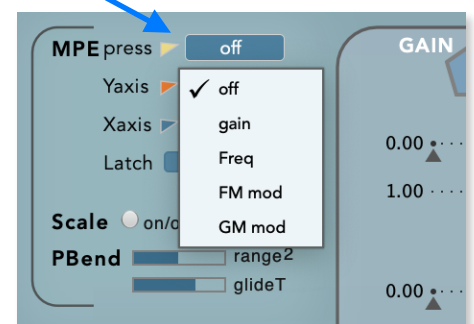
**MPE pressure** : off, gain, Freq, FM Mod, GM mod

**MPE Y axis** : off, gain, FM Mod, GM mod

**MPE X axis** : off, freq, FM Mod



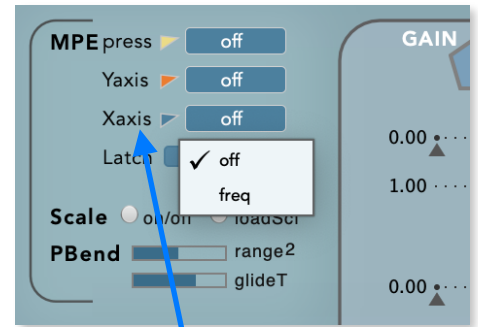
MPE pressure



MPE Y axis mapping

**Attention :** The MPEx is hard wired as a polyphonic pitch bender. When it is set to off, both this and the standard midi pitchbend effect will be set off.

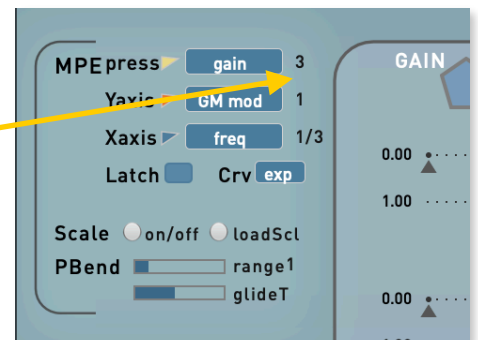
Just note the color coding used for each gesture ; **yellow**, **orange** and **blue**. The modulation which these gestures apply will be shown in real time below the left and right bank sliders depending their mapping.



**MPE X axis mapping**

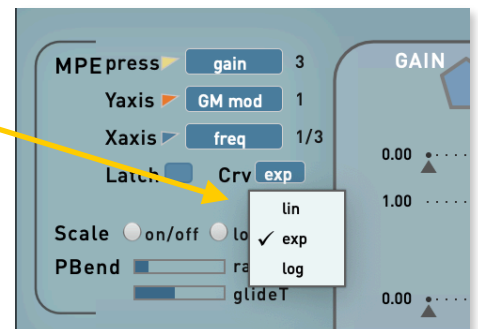
For each MPE component a scale factor can be set by using the MIDI controllers 28, 29 and 30 ( value 0 -> 127 triggers ). This will help you scale your performance input easily.

possible scale factors are : 0, 1, 2, 3, 1/3, 1/2.



One can also map the performance input with curve functions ( like the velocity curves on standard midi keyboards ) You can choose between linear, exponential and logarithmic curves.

The curve is being applied on Pressure and MPE Y component.



The **Latch** mode :

This mode provides a sustain state during the performance. When it is on, the last state of MPE pressure values will be kept even if the midi keys are off. How do we turn on / off the Latch mode ?

The Latch mode can be activated :

- by clicking on the **Latch button** on the user interface.
- by pressing the midi key 53 ( F3 ). Note On turns is on, and note off turns it off.
- with a midi controller message ( 64 ) which is in general used for the sustain pedal control.

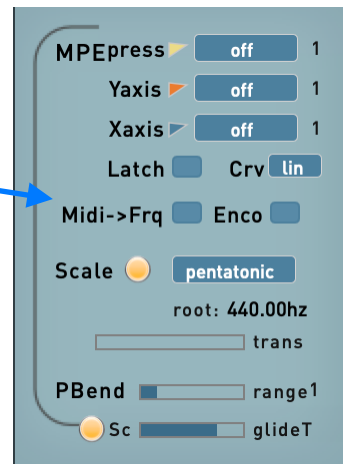
## Attention !

For using MPE and the Latch mode, the **Midi->Frq** switch should be off. Otherwise direct midi note-on input will take over.

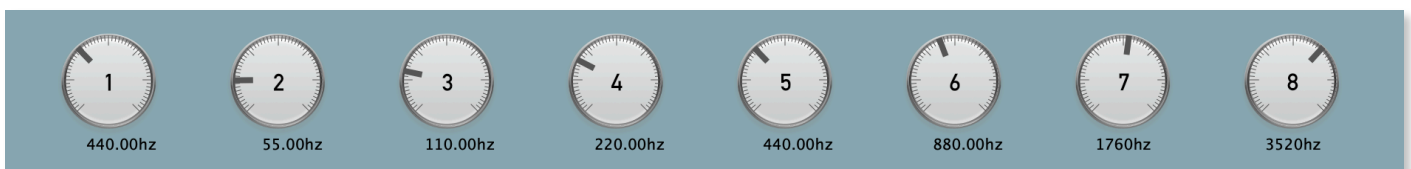
## Fundamental and Midi->Freq mode

Besides the MPE mode, Fundamental offers a new mode, where one can address each Fundamental oscillator pitch with Midi Note On messages on unique midi channels.

- Turn on this mode with its Midi->Freq switch, and this will bypass the MPE actions. Then for each oscillator, send midi note on messages to define the oscillator pitch on the relevant midi channel. ( **use midi channels 1-8 to map oscillators 1-8** )
- This is excellent to address the midi sequences mapped directly to selected Fundamental oscillators.
- Midi note-off will not be translated to gate off, just the midi note-on will be interpreted as gate on. However if at a current state, there are no midi note events being played, then the Trig switch will be turned off.
- If the Fundamental Trig switch is not on, then the next midi on message in the Midi->Freq mode will turn on the Trig switch.
- As expected, the received note-on pitch values will be filtered through the Fundamental tuning scale quantizer.



Here is another method to assign midi input diatonic scale notes to the voice frequencies.



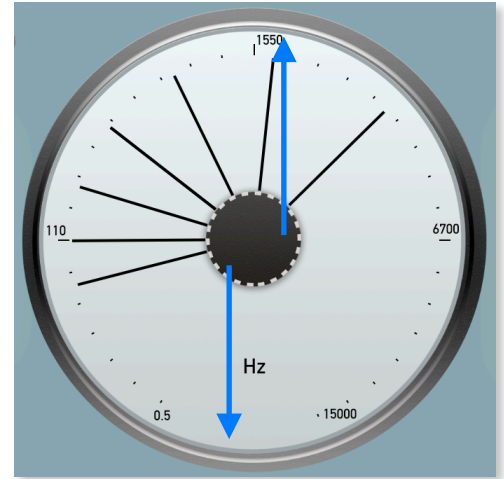
You need to go with your mouse pointer on top of any voice freq rotary slider, you will see that it will be highlighted. Then press any midi key on your keyboard and this will be converted to frequency and assigned to that voice. ( This action can be replicated on the iOS version by touching the rotary slider and pressing a midi key.

## Further performance gestures..

Another performance gesture one could apply on Fundamental is the **gliding all the voice pitches together** like a pitch bend.

You do that by dragging your mouse on the Fundamental circle. From the center towards the upper edge or lower edge for a negative glide or positive glide. When you drag inside the circle the offset bend value will be shown as corresponding number.

Then when you click inside the circle, the frequency values will go back to original settings.

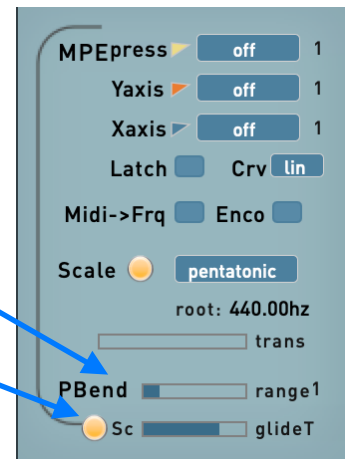


You can also use **Midi pitch bend** for the same action, and likewise record your gesture on a midi track.

The portamento glide acts like a frequency shifter on all oscillator frequencies. However, if Fundamental is on harmonic mode, the fundamental frequency will follow the glide and the harmonics will be locked to it.

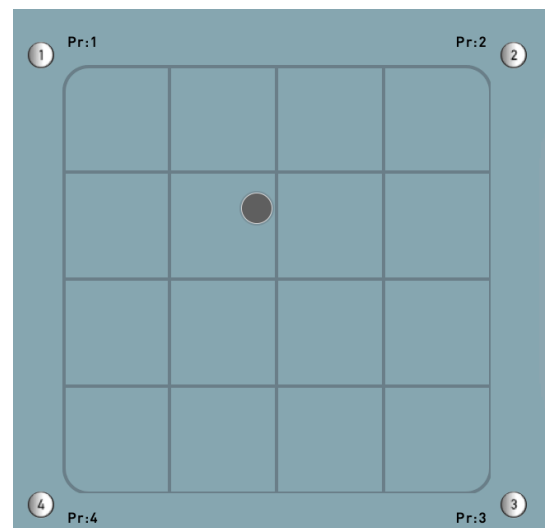
Of course one can assign the maximum glide range and the glide speed with these sliders.

When the **Sc** button is on, the glide will happen between the tune scaled oscillator frequency changes. By changing the **glideT** setting you can achieve very long glissandi.




There are some hard coded Midi Controller Messages addressing parameters of Fundamental :

- **Midi CC 14** : X position of the Morph Circle.
- **Midi CC 15** : Y position of the Morph Circle.
- **Midi CC 13** : sets the scale transition slider value.
- **Midi CC 31** : harmonic ratio slider value.
- **Midi CC 7 (Volume)** : controls oscillator final volume. Send on channels 1 - 8 to address the respective oscillator.
- **Midi CC 10 (Pan)** : controls oscillator final pan. Send on channels 1 - 8 to address the respective oscillator pan.



## LLM Interface on Fundamental4

Fundamental4 offers an easy to use LLM interface where you can create your prompt, submit it, perform its response and save it as part of the preset data. 

This is an easy to use text editor panel where you can type in your prompts.

Pressing the **Submit AI** button sends your prompt over the network to the cloud AI service.

The response panel indicates the status of your request. Once a successful response is received, the detailed results will be displayed in this panel.


**Trig:** This button applies the sequence of parameter changes received from the AI. It acts as a toggle: press it once to start the performance, and press it again to stop. The Midi Note C#5 will trigger this button and the synthesis engine Trig button together. ( Note on will turn it on, note off will turn it off. )

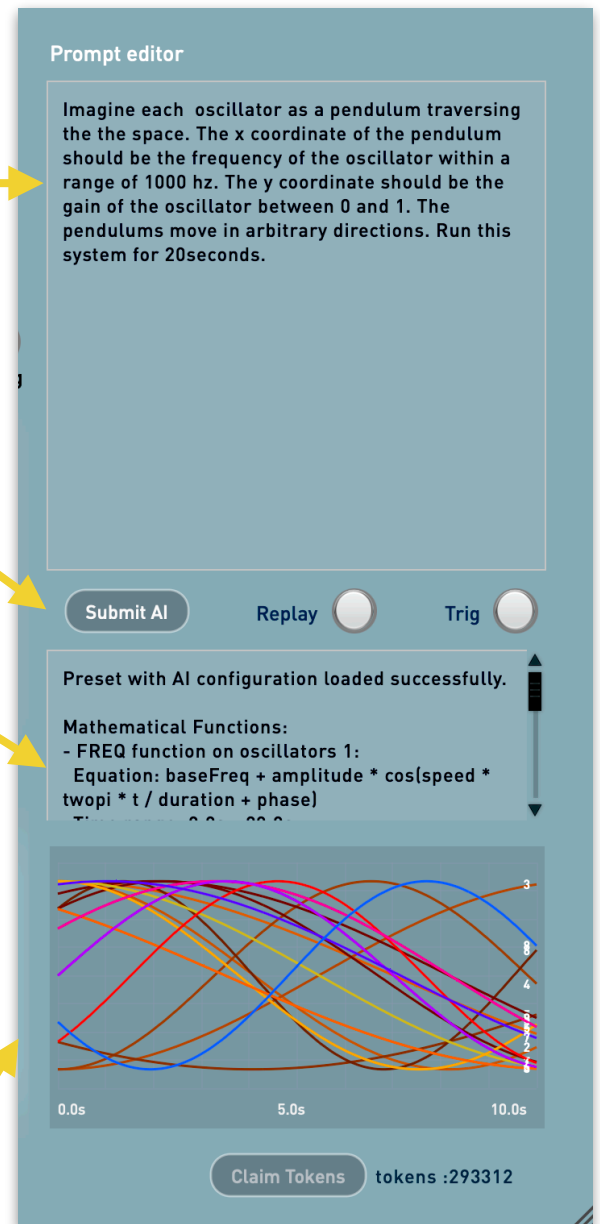
**Replay:** Instantly restarts the performance from the beginning. Midi Note B5 will also trigger the Replay mode ( a note on event will reset the time and replay the LLM )

The bottom panel provides a visualization of the AI's response, showing the translated parameter changes as they occur over time. This visual feedback is important, as it helps you connect what you see on the screen with the sound you are hearing.

For the performance of LLM through MIDI, send a note on on key C#5 and then anytime you can start the replay with a note on on B5.

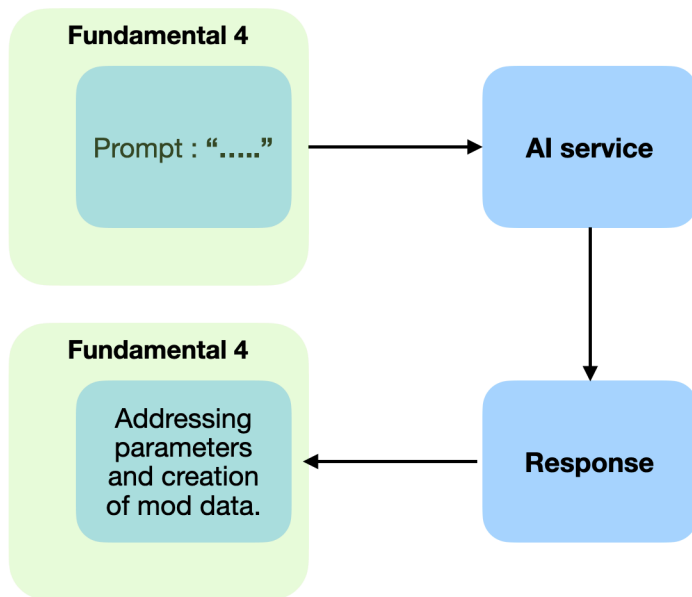
When you save a preset in **Fundamental 4**, all AI-related information is stored along with it. This includes your original prompt, the full AI response, and the translated performance data.

This means you don't need to resubmit your prompt every time you load the preset; you can simply press the **Trig** button to activate the saved performance immediately. 





## Controlling Fundamental 4 with Prompts



Using a natural language model to communicate with AI and getting its response translated to Fundamental4 synthesis engine for controlling parameters needs a structure, protocol, a proper conduct.

We have prepared a collection of important example prompts for you, covering both direct instructions and abstract models. These examples are taken directly from the full bank of presets dedicated to the LLM features in **Fundamental 4**.

**Important :** Using a natural language model to communicate with AI and getting its response translated to Fundamental4 synthesis engine may face unexpected results. You may not always get what you expect at first try. Tiny changes in wording sometimes makes lots of difference.

In all cases, the AI expects to have clear instructions , how it should start , which values are set at the start for the oscillators etc. Therefore studying the delivered bank presets are important for a reference.

The bridge between the prompt response and the Fundamental4 synthesis engine has been structured with following features :

- **Oscillator Gain and Frequency :** You can address any oscillator gain and frequency with your prompt and define their values directly or indirectly with abstract models.

For example consider this prompt : *Set the eight oscillators to the first eight partials of a 65Hz fundamental. Start all at max amplitude.*

This will configure the oscillators as shown in the screenshot below. Note that while you only specified the 65 Hz fundamental directly, the AI calculated and applied the correct frequencies for all eight partials based on your request.



An important detail is that when a prompt modifies an oscillator's gain, it controls the **Boost** slider, not the **Gain** slider.

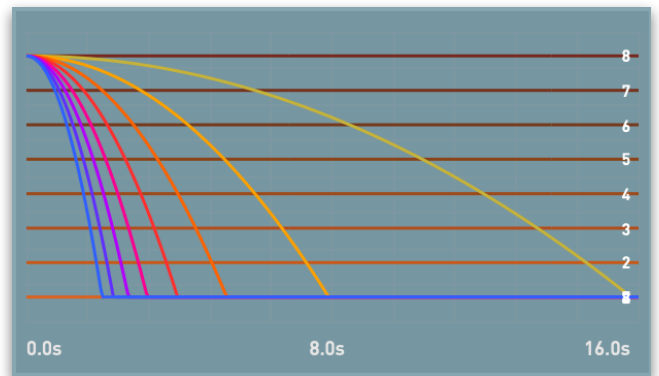
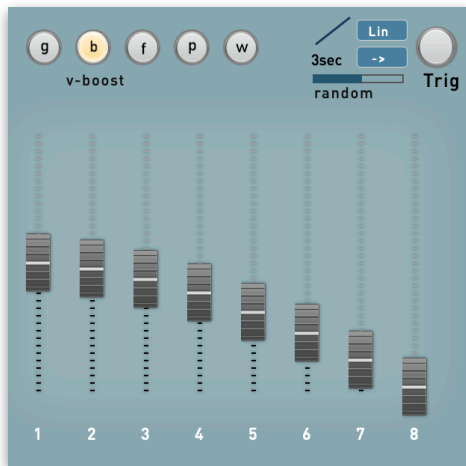
This is an intentional design choice. It leaves the main **Gain** slider free for you to make manual adjustments or to be modulated by other sources, like envelopes, completely independent of the AI's performance.



- **Setting Oscillator Gain and Frequency** : You can address any oscillator gain and frequency with your prompt and define their values directly or indirectly with abstract models.

And for the Gain part of the prompt, consider this : *Start all at max amplitude. Apply an exponential amplitude decay where the decay time is inversely proportional to the harmonic number. The fundamental's amplitude should decay to zero over 16 seconds, the second harmonic over 8 seconds, the third over 5.3s, the fourth over 4s, and so on.*

This will configure the boost sliders of the oscillators as shown in the screenshot below. Again note the complex relationship instructed on the prompt, the AI is expected to create this relation for the modulation of the Boost sliders over timeline.

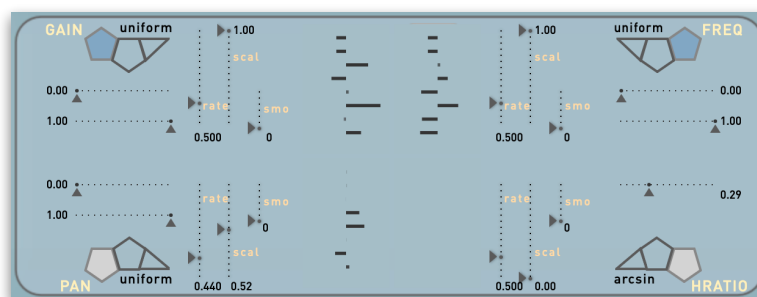


The Boost slider modulations has been drawn over the timeline and the AI has successfully described the exponential decay for each oscillator amplitude.

- **Frequency Modulation** : You can control the frequency modulation in **Fundamental 4** using prompt instructions.

As a reminder, **Fundamental** has four **GEN** modulators per voice, one of which is dedicated to frequency. The **GENs** can use a variety of stochastic and deterministic functions, and have **Rate** (speed) and **Scale** (depth) controls.

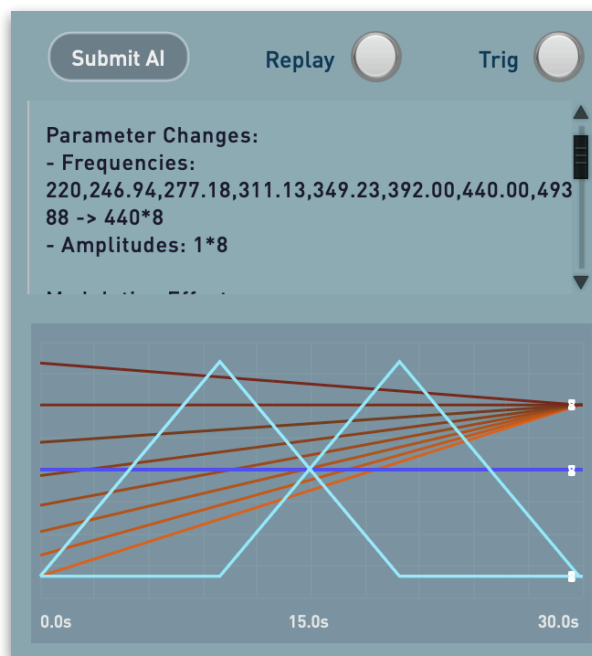
Within your prompt, you can specify the exact function you want to use for frequency modulation (referencing it by name), as well as its desired **Rate** and **Scale** values.



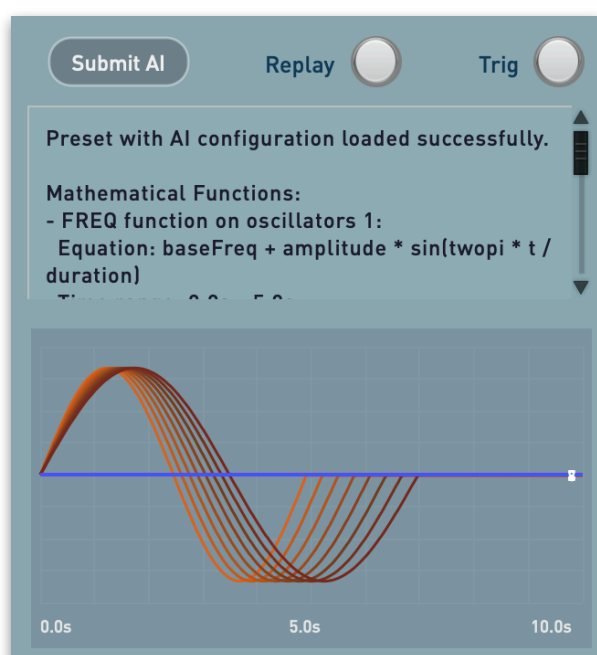
Consider this prompt : *Set all frequencies with a minor third interval apart beginning with 220hz. Set all frequency mod depth values to zero and mod type uniform and move the frequency modulation depth of even number indexed oscillators to maximum in 10 sec. Then bring the same oscillators frequency mod depth to 0 in another 10sec. During this 10 sec. bring the odd numbered oscillators frequency modulation depth to maximum. And finally bring these to zero as well in another 10 seconds. Converge the oscillator frequencies to 440hz during this total process.*

This prompt not only defines the initial oscillator frequency setup but also specifies the function for the GEN (e.g., "uniform distribution") which serves for the frequency modulation on Fundamental.

More powerfully, it instructs the AI to create a timed automation sequence for the modulation depth sliders of specific oscillators. The AI correctly interprets these layered commands, and once it responds, you can see the resulting complex automation visualized on the timeline.



Now please check this prompt : *set all oscillators to 50hz and change the oscillator frequencies with a sine wave within a range of -50 and +50 hz. The first oscillator should do this in 5 sec. and the second osc in 5.3sec and so on...*



- **Gain Modulation** : You can control the Gain modulation on each oscillator in **Fundamental 4** using prompt instructions.

As a reminder, **Fundamental** has a **GEN** modulator per voice, which is dedicated to gain. This Gen can use a variety of stochastic and deterministic functions, and have **Rate** (speed) and **Scale** (depth) control.

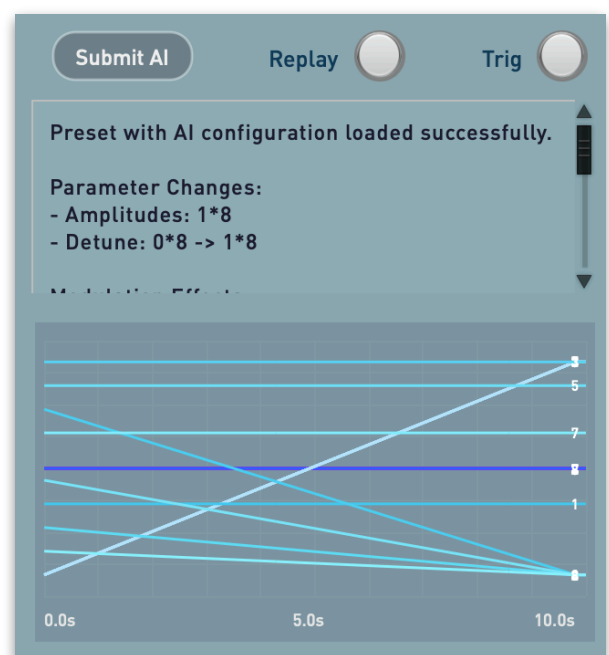
Within your prompt, you can specify the exact function you want to use for gain modulation (referencing it by name), as well as its desired **Rate** and **Scale** value.

Consider this prompt : *Set gaussian random amplitude modulation to all oscillators with a depth value randomly chosen between 0 and 1 for each oscillator. Then bring the amplitude modulation depth to 0 for even numbered oscillators and increase the modulation speed from 1 to 20 in 10 seconds.*

This prompt defines the behavior of the amplitude modulation GEN (e.g., "gaussian distribution") on Fundamental. It also instructs to calculate a random value for the modulation depth slider setting for each oscillator.

Finally, it creates a complex 10-second automation: the modulation depth for all **even-numbered** oscillators is reduced to zero, while the overall modulation **speed (Rate)** is simultaneously increased from 1 to 20 Hz.

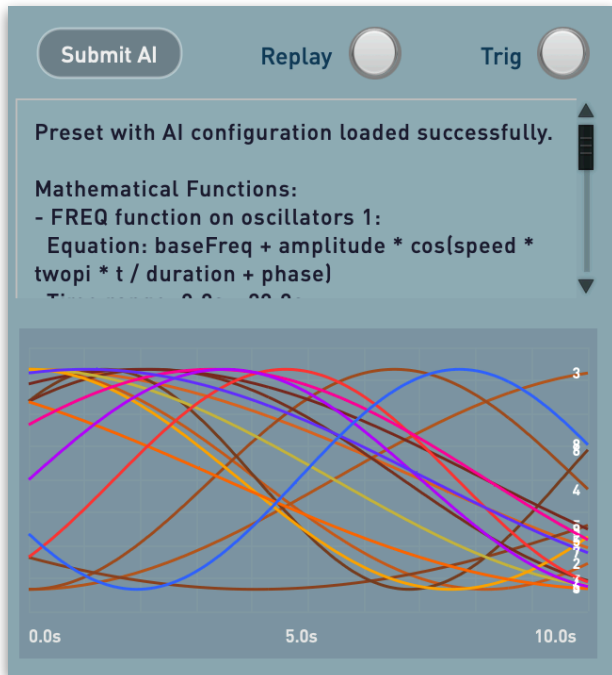
You can see on the modulation data display the change on the depth settings, how the even numbered ones are converging to a depth value of 0.



Now the example will be a description of a simulation scene via prompt where we will use the simulation output to modulate the frequency and gain of each oscillator.

Consider this prompt : *Imagine each oscillator as a pendulum traversing the the space. The x coordinate of the pendulum should be the frequency of the oscillator within a range of 1000 hz. The y coordinate should be the gain of the oscillator between 0 and 1. The pendulums move in arbitrary directions. Run this system for 20seconds.*

A pendulum is a weight suspended from a pivot that can swing freely, creating a smooth, natural arc. This prompt instructs the AI to create a **physical model** where each of the eight oscillators behaves like an independent pendulum, using its position to control sound parameters over a 20-second duration.



Specifically, for each oscillator:

- The pendulum's **horizontal (X) position** is mapped to its **frequency** across a 1000 Hz range.
- The pendulum's **vertical (Y) position** is mapped to its **gain** between 0 (silence) and 1 (full volume).

The instruction for the pendulums to move in 'arbitrary directions' results in a complex, organic, and continuously evolving modulation of both pitch and amplitude.

On the left, you can see the response panel depicting the mathematical equation which the AI has generated for the pendulum physics. The modulation data has been nicely drawn on the panel below.

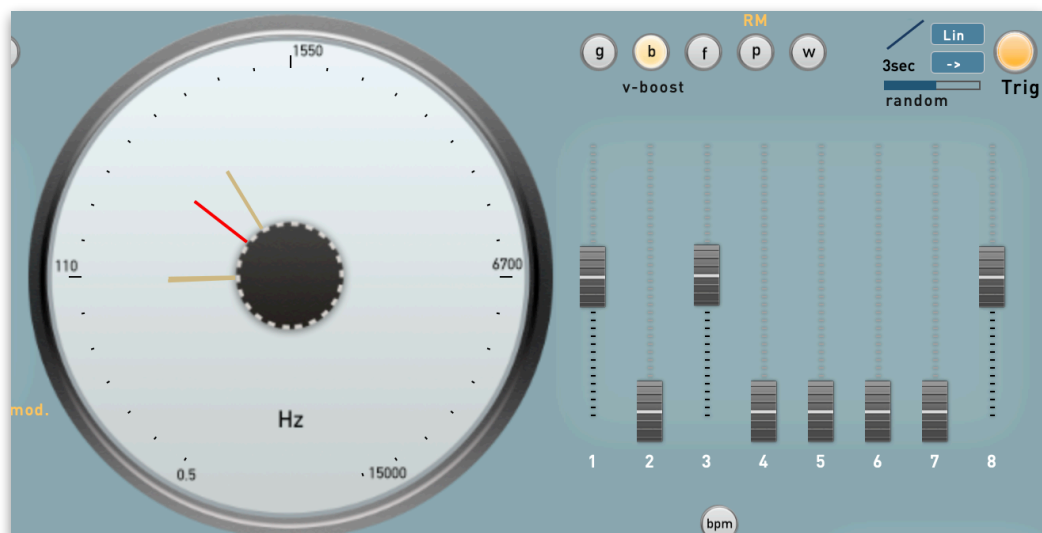
- **Ring Modulation** : You can control the Ring modulation on **Fundamental 4** using prompt instructions.

As a reminder, in **Fundamental's Ring Modulation (RM)** mode, the first seven oscillators modulate the eighth (carrier) oscillator.

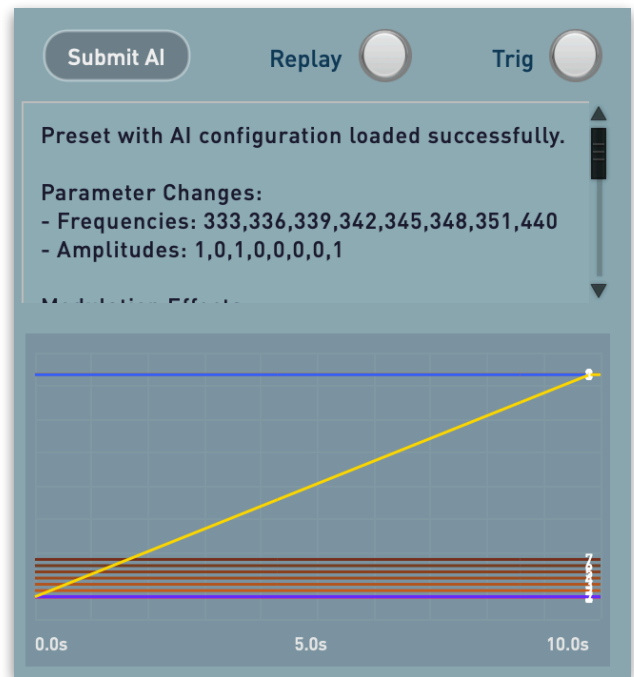
For detailed instructions on how to use this feature, please refer to the main Ring Modulation section of this manual.

Consider this prompt : *Apply ring mod with carrier frequency 440hz and all modulator frequencies 330hz plus their index number\*3. Set all modulator oscillators gain to 0 except the first and third. And also set the carrier oscillator level to the same. Change the ring mod balance from 0 to max in 10sec.*

This prompt describes the AI the frequency settings of each oscillator and the gain settings for the modulator and carrier oscillators. As you see on the Fundamental panel below, the oscillator gains are set respectively.



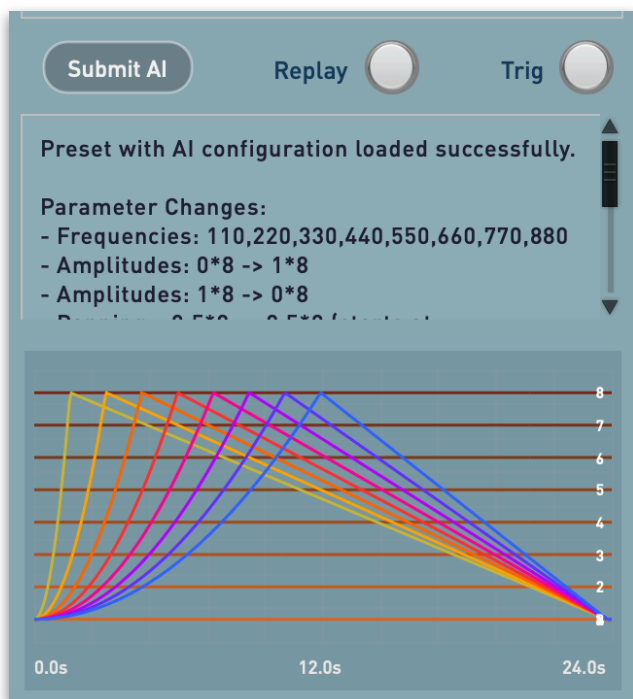
You can see the ring modulator balance change as the yellow ascending line which takes 10sec. At zero, one hears only modulator oscillators and at max, one hears the carrier oscillator and the sidebands created ( no modulator oscillators )



- **Pan Modulation** : You can control the pan position of each oscillator of **Fundamental 4** using prompt instructions.

As a reminder, **Fundamental** has four **GEN** modulators per voice, one of which is dedicated to Pan. Besides you can address directly the pan position slider settings with your prompt instructions.

Consider this prompt : *Set all oscillator frequencies to multiples of 110hz. Give each oscillator gain an exponential curve rise which gets slower for each higher indexed oscillator. Once an oscillator reaches the max gain value, it should start to fade off linearly to zero gain value. The last oscillator should have its full rise in 12 sec. For panning: When an oscillator reaches its max gain start changing its pan position from left to right in 12 sec.*

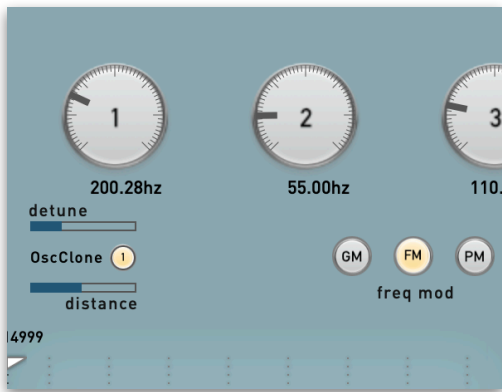


While this prompt primarily defines the oscillator gain behavior, its final instruction for panning demonstrates a more advanced concept. It creates a **conditional link**, triggering the start of each oscillator's pan movement only when that oscillator reaches a specific gain state.

In essence, the AI is not just creating two separate automations; it is generating the pan modulation as a direct consequence of the gain modulation's behavior.

- **Osc Detune control** : You can control the oscillator detune parameters by using prompt instructions.

As a reminder, **Fundamental** has an oscillator cloning feature, and each clone can be detuned with specific parameters. These can be addressed easily by your prompt input.



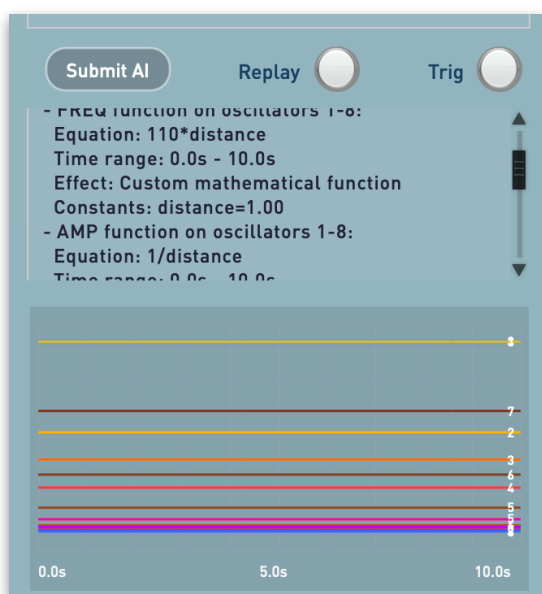
Consider this prompt : *Increase the detuning from 0 to strong detune during this 10 seconds.*

This will increase the level of detuning for 10 seconds.

## **Emergent or Abstract Prompting :**

We have seen the cases where we can address the structural component directly within the prompt and define their change over the timeline. We can introduce conditional links checking the behavior of interlinked components. This is already an emergent property , however there is more to that in using a generative AI tool embedded within Fundamental4.

**Prompt :** *Set the oscillator frequencies based the distance of the planets from the sun in our solar system. The first oscillator should have the frequency 110. Also apply similar relation to the oscillator amplitudes where the first oscillator should have the maximum level.*



**Physical Modeling Prompts:** By defining a physical environment in your prompt, you are not describing parameter values directly; you are describing a virtual **physical system** with its own objects (molecules), environment (a cube), and rules of physics (bouncing off walls, temperature affecting speed). The AI's job is to **simulate** this system and translate the results into sound.

**Systemic or Emergent Prompting:** It focuses on the fact that you are defining a **system** and allowing complex, unpredictable behavior to **emerge** from its simple rules. You set the initial conditions, but the final sonic result is generated by the simulation itself.

**Metaphorical or Abstract Prompting:** It highlights your use of a powerful **metaphor** ("gaseous molecule") to control the sound. You are describing an abstract concept and mapping its properties (position, speed) to concrete synthesis parameters (gain, frequency), which is a highly intuitive and creative way to work.



## Current Restrictions on LLM:

**Important:** When an AI performance is active, the LLM takes **exclusive control** over the parameters it is modulating. ⚠️

It will override all other control sources for those parameters, including any manual adjustments you make to the corresponding UI elements.

Currently, the LLM does not directly control the following features in **Fundamental 4**. This is an intentional design choice, as these are considered "macro-level" tools that operate separately from the LLM's real-time performance engine.

### **Morphing Panel**

The LLM generates its own performance based on a unique timeline. This is currently incompatible with the preset-based timeline used by the four-corner morphing system.

### **Stochastic Sequencer**

This feature, inspired by sonicLAB's *Cosmosf*, is a self-contained generative engine. For now, it is not controllable by the LLM, though we plan to introduce different kinds of LLM-based sequencing in the future.

### **Envelope Triggers**

Since you can also describe envelope shapes directly within an LLM prompt, the built-in envelope trigger system is kept separate to avoid conflicts between the two methods.

### **Tuning Scale**

You can define incredibly complex and precise pitch relationships within an LLM prompt. Therefore, the Tuning Scale remains a separate, optional macro control. You can still apply a scale on top of an LLM performance, but the AI does not control the scale selection, giving you maximum flexibility.

## Tuning scales on Fundamental

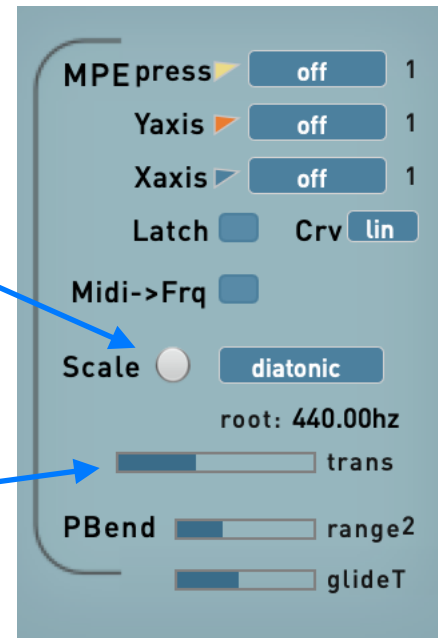
Scala format tuning scales can be imported to Fundamental and applied as pitch quantizers. There are already some example useful scales coming within the Fundamental package. The existing scala files are located at sonicLAB/Fundamental3/scl/ directory , and you can put new ones there and the list will update accordingly.

In order to make the tuning scale active, turn it on with its button. Also you need to choose a scale. A pop-up menu will let you choose from the existing scales of Fundamental. When you save a preset, these settings will be saved along with.

The reference frequency ( called root on the app but the scale can go both upwards and downwards between (2hz and 14999hz) based on that reference value ) can be typed directly on the interface.

By default, this is 440hz and your custom setting will be saved along with each preset.

Fundamental introduces a **scale transition** slider where you can gradually go from a defined scale quantization to non-scale state. Left end position of the slider is full scale state and the right end position of the slider is a no-scale state.



Technically the scales act like a pitch quantizers so the frequency calculated with the user setting + the freq. modulation being applied.

You can find in depth information and other examples of scala tune files on this link.

[http://www.huygens-fokker.org/scala/scl\\_format.html](http://www.huygens-fokker.org/scala/scl_format.html)

Scala files are merely text files describing a custom scale in its specific format.

## High precision Midi control on Fundamental with Rotary encoders

Fundamental incorporates special handling of encoder style Rotary Midi CC knobs, and therefore it can map the incremental motion of this type of knob to high resolution oscillator frequency and gain values.

A Rotary encoder is an electro-mechanical device that converts the angular position or motion of its shaft to analog or digital output signals. Rotary encoders are used in a wide range of applications that require monitoring or control of industrial systems including robotics, computer input devices (mouse / trackball ), many types of rotating platforms such as wind turbines etc.

There are two main types of rotary encoder: absolute and incremental. The output of an absolute encoder indicates the current shaft position, making it an angle transducer. For instance it gives an absolute value based on its angular position. ( a value of CC 0-127 for example in MIDI case. )

The output of an incremental encoder provides information about the motion of the shaft, which typically is processed elsewhere into information such as position, speed and distance. Therefore it produces a certain value immediately when moved clockwise or anti-clockwise by stepwise interpretation of the movement.

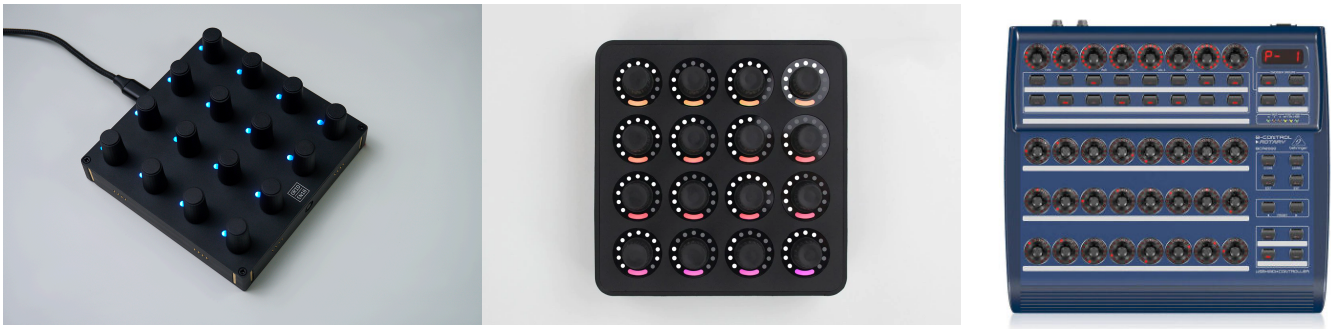
Rotary encoder equipped MIDI controllers exist since a while on the market. For example, DAW controllers and desktop digital mixing consoles have incorporated this type of encoders well for modulating parameters such as track level, panning etc. The photo on the right shows a Mackie DAW controller with 32 Rotary Encoders, and each knob has a led display around it showing the absolute value of its dedicated parameter destination.

The advantage of these endless rotary knobs is that they act relative to the current state ( or last recalled preset ) of the parameter which they do control. The motion of the knob creates a relative value based on the stepwise incremental / decremental motion of the knob and this adds up to the last recalled state of the parameter.

This is a convenient non-destructive editing method, which is not possible with an absolute standard 0-127 Midi CC controller.



Today, on the market one can find several rotary encoder style midi controller units not just dedicated as DAW controller but for general purpose usage. To name a few the Intech EN16, Midi Fighter Twister, Behringer BCR2000 etc.



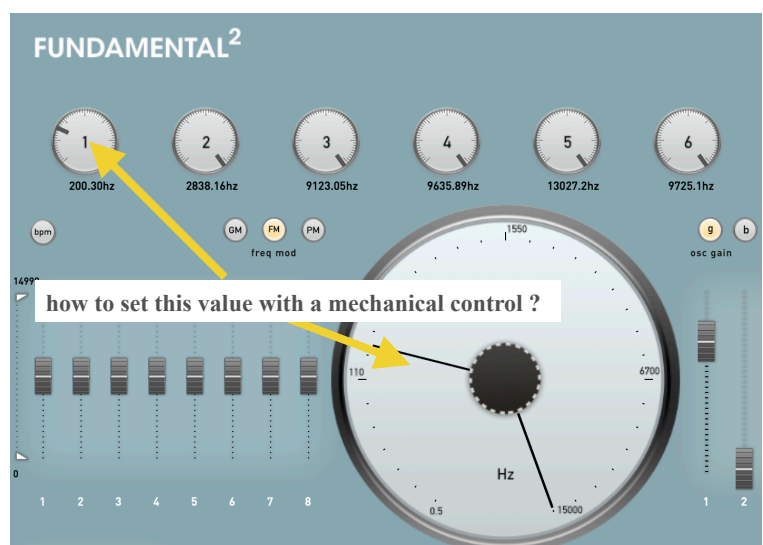
\* We are not affiliated business-wise with any Midi Controller device producer. There are different units in price / quality on the market.

### Why choose a Rotary Encoder control for Fundamental ?

Fundamental is a semi-generative synthesizer of which sound source is oscillators which model a Rohde&Schwartz test tone sine wave generator.

The sonic quality of the oscillator varies according its frequency and also the gain settings.

On its user interface one can set the frequency of an oscillator with a rotary dial knob, or type in the exact value as text or assign a midi controller on the DAW to control this parameter.



Standard Midi continuous controller data ranges between 0-127 integer values ( 7bit resolution) There is no way to address the frequency value range of the Fundamental oscillator ( 1hz - 15000hz ) with this range of modulation data.

Standard Midi controller access is destructive. Imagine a Fundamental preset; all the frequency and gain settings of each oscillators matters to achieve its beautiful interference between sine waves and the emergent sonic quality of the generative engine. But when you touch a midi controller, its current absolute value will jump the current setting on the Fundamental oscillator, which is not what we ideally prefer.

It is not feasible the invent a custom communication method and build a dedicated hardware for this, and yet we need to still use Midi and existing hardware to achieve this.

## What did sonicLAB achieve ?

- Therefore sonicLAB has decided to create a high resolution frequency mapper which utilizes the data coming from on an incremental endless Rotary encoder knob. And we succeeded reaching the entire frequency range of the Fundamental3 oscillators with one knob motion and with a **1hz** resolution. !
- There remained another problem, since each product on the market can send different Midi CC messages based on the direction of the knob motion. For example one product sends Midi CC 64 when the knob is being turned clockwise on each step, and Midi CC63 when anti-clockwise. Another product sends Midi CC127 and Midi CC0 for the same motion directions.
- What sonicLAB has achieved is a behavior agnostic to these values, it adapts itself with a simple twist. You just need to move the knob shortly in both directions and that's all, the message of the knob is captured.

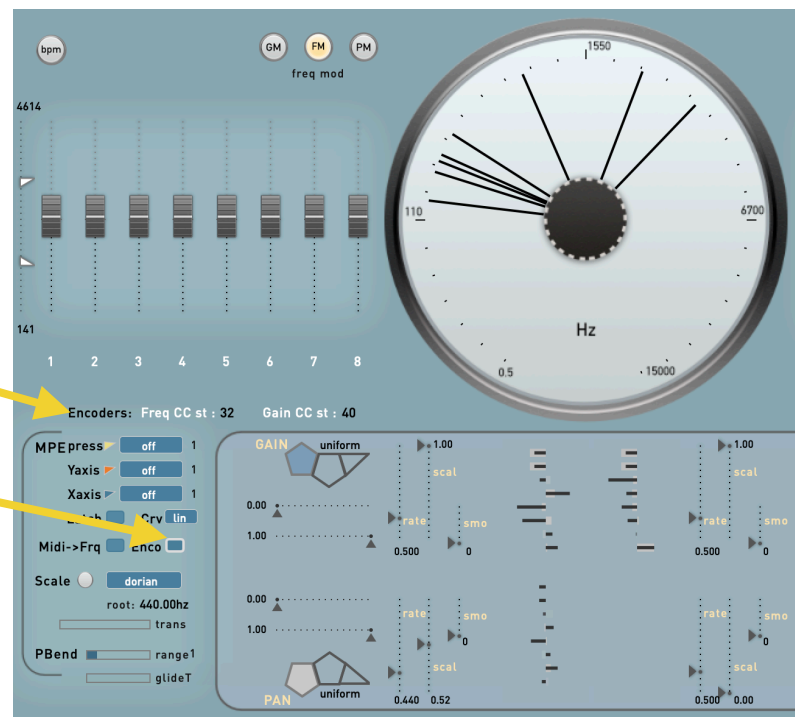
## implementation on Fundamental desktop and iOS versions

You have purchased a rotary encoder Midi controller and connected it to your computer / iPad.

Be sure that Fundamental receives it either from the DAW channel or your controller is selected as the input device on the standalone Fundamental.

click on the **Enco** button and see that the Encoders CC numbers appear on the interface.

As mentioned before you can control the frequency and the gain values of the 8 oscillators on the Fundamental with these Rotary encoders. Ideally you will need 16 of them to access all these parameters instantly.



You can specify the starting CC number for the frequency parameter control and gain parameter control on the interface.

**Encoders: Freq CC st : 32    Gain CC st : 40**

For example the default Midi CC start number for the frequency control is 32. Therefore a Midi CC32 will address the frequency of the Oscillator1 and Midi CC33 for the freq. of the Oscillator 2. When we set the start as 32, the Midi CC's 32-40 are allocated automatically for the frequency control of the oscillators.

The default Midi CC number for the gain control starts at 40. Likewise the Midi CC40 controls the gain of the Oscillator1.

You can customize these starting numbers the way it fits to your controller setup or you can choose to change the Midi CC number of your Rotary encoders respectively.

## Remarks on the use of Rotary Encoder control on Fundamental

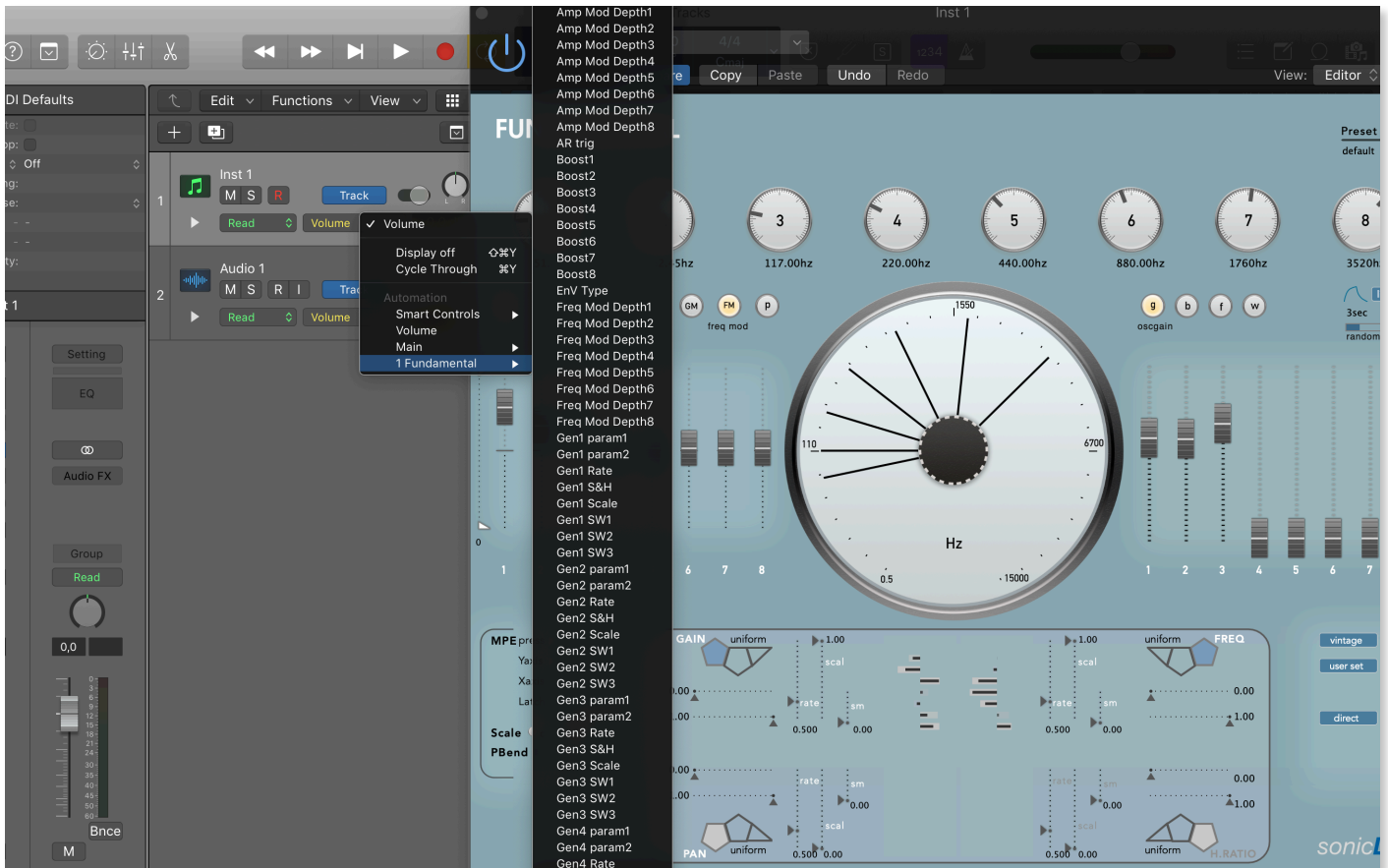
- Each step ( incremental or decremental ) of a Rotary Encoder movement will achieve a **1hz** relative change on the Oscillator frequency and an amount of **0.01** change on its gain level. This kind of frequency resolution is never achievable with a standard CC0-127 control.
- We advice to avoid rapid jumps on the Rotary Encoder movement. This implementation is meant to focus on fine movements and fine adjustments of the destination parameter. sonicLAB has implemented adaptive **motion -> freq scale mapping** and higher rotation speeds also maps to faster step intervals in an intelligent way. However the Rotary knobs mechanically won't comply to same results when rotated fast. This depends on the hardware quality and there is not much to do on the software side to fill these gaps.
- A useful analogy would be the use of mechanical mouse / trackballs during 90's ( before touch pads ) where one could not precisely know the advance of the mouse pointer when moved fast. Slow movements were always more predictable. Therefore when you rotate the Rotary knobs fast, the jump on the destination value will vary.
- sonicLAB implementation aims to adapt to any Rotary encoder whatever CC value it sends for its incremental and decremental change. However there is the logical restriction that the CC value of an incremental change ( a clockwise rotation step ) has to be higher then the value sent when there is an decremental change ( a anti-clockwise rotation step ) . When this condition is fulfilled, any Rotary Encoder will work within our implementation.
- First test the standalone version of Fundamental3 with your Rotary encoder controller. When confirming that it works, you can test it in your DAW. Every DAW has different midi port input / output handlings , and we cannot know and test each DAW. This is why that it is important your test first outside a DAW with the standalone version.
- First time you touch a Rotary encoder knob to control a parameter on Fundamental, apply a little twist back and forth on the knob so that the CC values for the incremental and decremental motion becomes evident to the software.



## Automation in DAW with Fundamental

Most Digital Audio Workstations (DAWs) allow you to record and edit automation for plugin parameters, and **every parameter in Fundamental** can be automated this way.

The screenshot below shows an example of **Fundamental's** parameters being automated on a track in Logic Pro.



It is also possible to assign a MIDI controller to every available parameter to extend your live interaction possibilities.

## APPENDIX

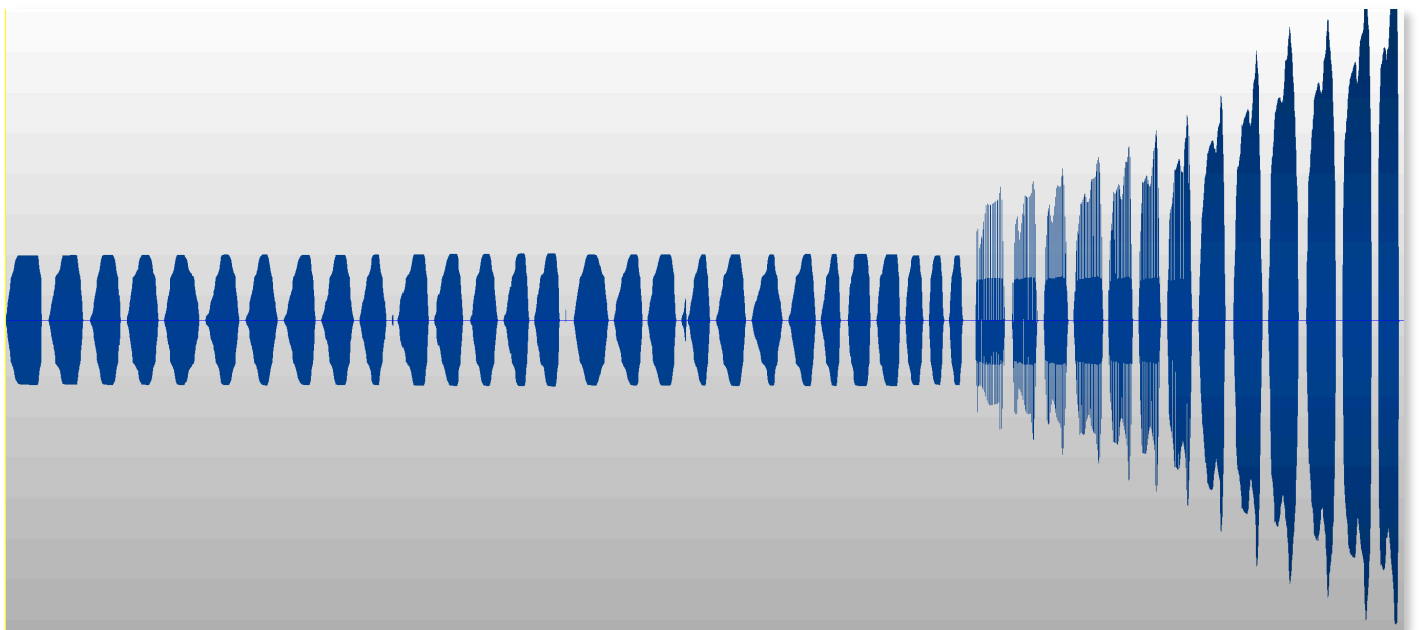
The core sound source for **Fundamental** is a vintage Rohde & Schwarz Sine Wave Generator.

As its name suggests, the original hardware generates sine waves, which are shaped by its **Frequency** and **Gain** controls.

However, the interaction between these simple controls produces a rich timbral palette that goes far beyond pure sine waves. Depending on their settings, the unit can create a range of complex waveforms with a character unique to this vintage test generator.

This **non-linear behavior** and its unique tonal palette—not just a pure sine wave output—is exactly what we were interested in.

Our process therefore began with meticulously recording the hardware across many different frequency ranges, continuously sweeping the **Gain** setting for each one to capture its full character.

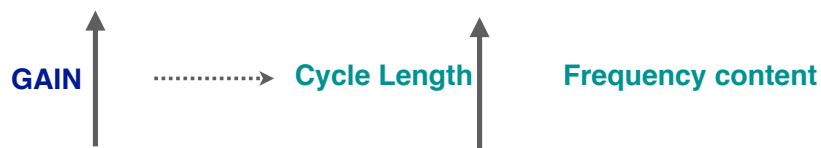
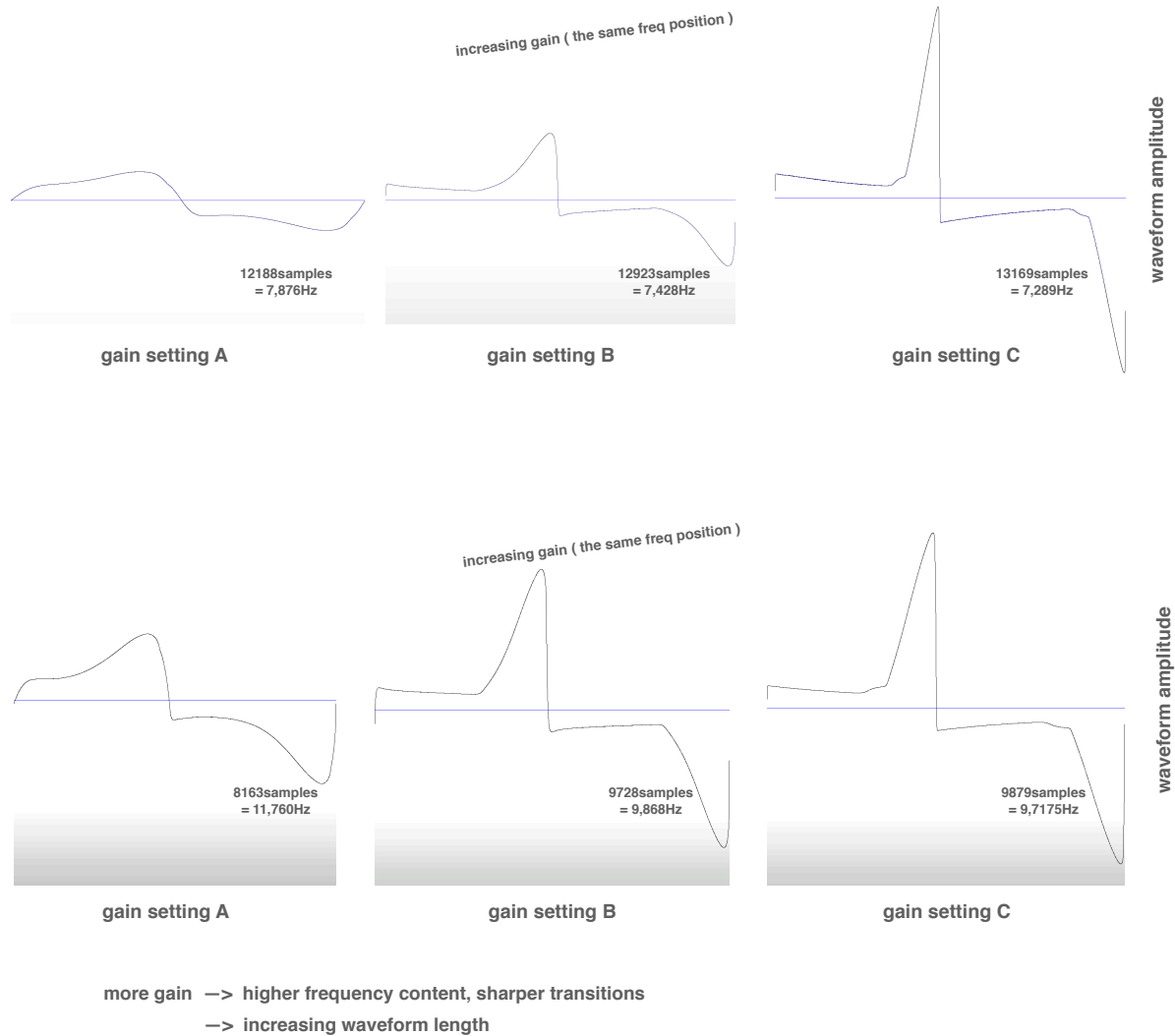


Through these simple, gestural actions on the hardware controls, we captured **thousands of unique waveforms**, each with its own distinct frequency and gain character.

The next step was to use custom software for deep analysis and **data mining** of these recordings—a task at which computers excel.

The figures below show how the vintage generator's output waveform changes dramatically based on its control settings.

This is especially apparent **below 50 Hz**, where the unit generates a range of complex waveforms rather than a pure sine wave.



Using custom software developed by sonicLAB, we analyzed all of these recordings to create, categorize, and store the corresponding wavetables.

The result is **hundreds of meticulously labeled wavetables**, all preserved at their full resolution **without any data compression** to ensure maximum audio fidelity.

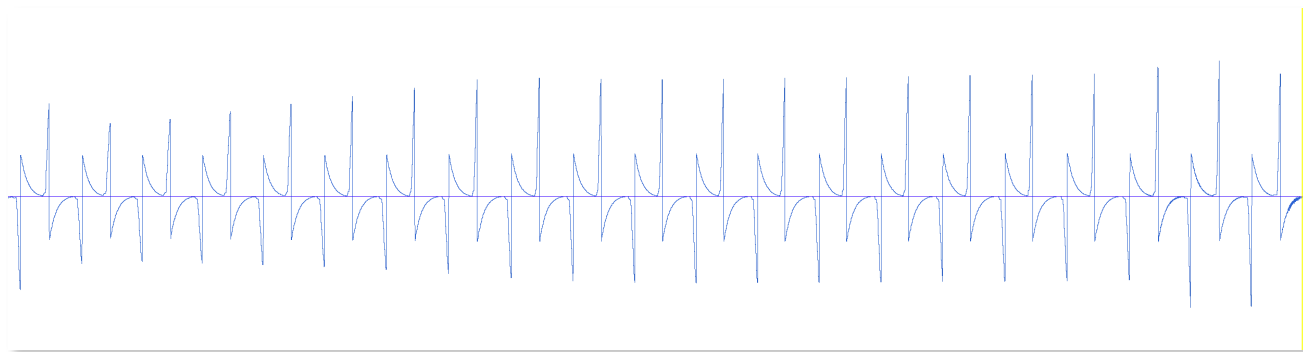
The resulting **wavetable sizes** vary dramatically—from over 57,000 samples down to just 17—depending on the **Frequency** setting of the original hardware.

The **Gain** setting has a dual effect: it not only alters the **wave shape**, but it also influences the **wavetable size**, even when the frequency is held constant. Capturing all of these complex interdependencies was a significant undertaking.

## Polymorphing WaveTable Synthesis developed by sonicLAB

Generic wavetable synthesis typically requires all waveforms to share the same fixed size, usually a power of two (e.g., 2048 samples).

While this approach is highly efficient for memory and CPU usage, the resulting data loss was unacceptable to us. It would have compromised the unique timbral quality of the vintage tone generator, particularly in its **lower registers** where the character is most complex.



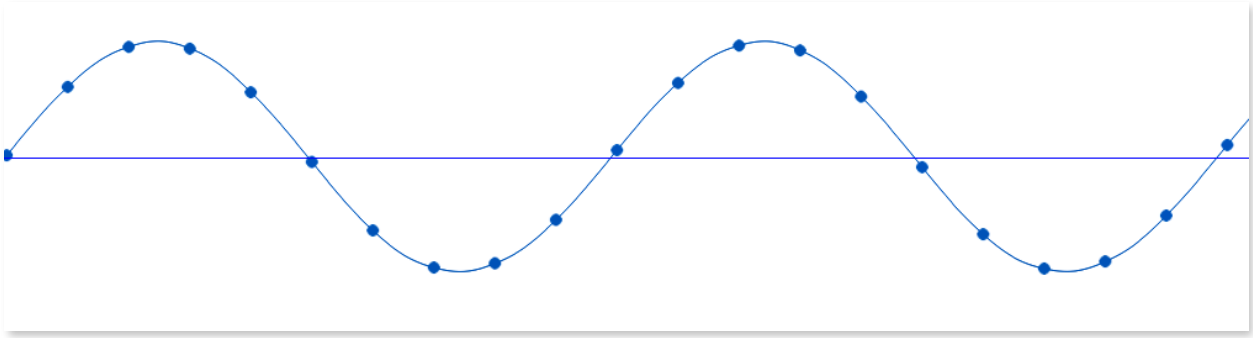
**Fundamental's** purpose is to generate any possible frequency and gain combination in a perfectly smooth continuum, achieving a resolution that surpasses the original analog hardware.

To do this, we developed a custom **polymorphing wavetable synthesis** algorithm. This complex C++ engine takes into account all the data captured from the vintage unit—with **no compromises** for optimization—and dynamically calculates the precise output waveform for any given frequency and gain value.

You can sweep the **Frequency** and **Gain** controls with sample-accurate precision, resulting in perfectly smooth, glitch-free transitions. In the background, a network of four oscillators works in real-time to render each individual sample. This continuous rendering preserves the rich tonal palette of the vintage hardware while responding flawlessly not just to your manual control, but also to complex, high-speed modulation from the **GENs** and other sources.

The algorithm has a specific crossover point. Based on empirical tests, we chose **3800 Hz** as the threshold where **Fundamental** smoothly morphs from the sampled wavetables to a perfectly computed sine wave. This is because sampling the original hardware at very high frequencies yields too little data to faithfully represent the waveform.

As the figure below illustrates, trying to reconstruct a high-frequency wave from limited sample data leads to digital artifacts. Problems such as rounding errors and timing quantization can distort the sound and cause **aliasing**. Our approach avoids these common pitfalls of the digital world.



**Fundamental** therefore offers the best of both worlds: the faithful, complex character of the vintage tone generator from **1 Hz to 3800 Hz**, and the pristine clarity of a pure mathematical sine wave for higher frequencies.

Dr. Sinan Bökesoy, May2020