NoiseProgrammer

USER GUIDE MANUAL

1. Purpose of the software

The purpose of this software is to create, record, share/use noise samples generated by user-implemented algorithm. Or just to have fun with it. If you are harsh noise artist, you can use this program for some recordings or live shows.

In opposite to many (often commercial) DAWs, where user is able to create any sound sample desired just by pushing several buttons, this program allows you to "implement" your sounds - the sound is generated on the fly by pure math only; this means that despite sound might look pretty random, it still is predictable, and offers waveform control that many programs doesn't even mention.

This software stands under CC licence, and it's okay for any non-commercial use (I won't support music-for-money things); free to redistribute, at any form desired.

2. Installation

This software doesn't need any installation. After unzipping, run .exe file and program should run on any machine, although sound might keep freezing on single core CPUs with clock slower than 1.5GHz.

For optimal performance, at least dual core CPU is required, 128MB of RAM, and GeForce 210 256MB or equivalent GPU. No expensive soundcard is required, as the sound is generated in the real-time.

Configuration file

This program even has its own config file, located under *<exe-directory>/data/config.cfg*

You can change the file on your own. If something goes wrong, remove the file, and the program will restore default settings.

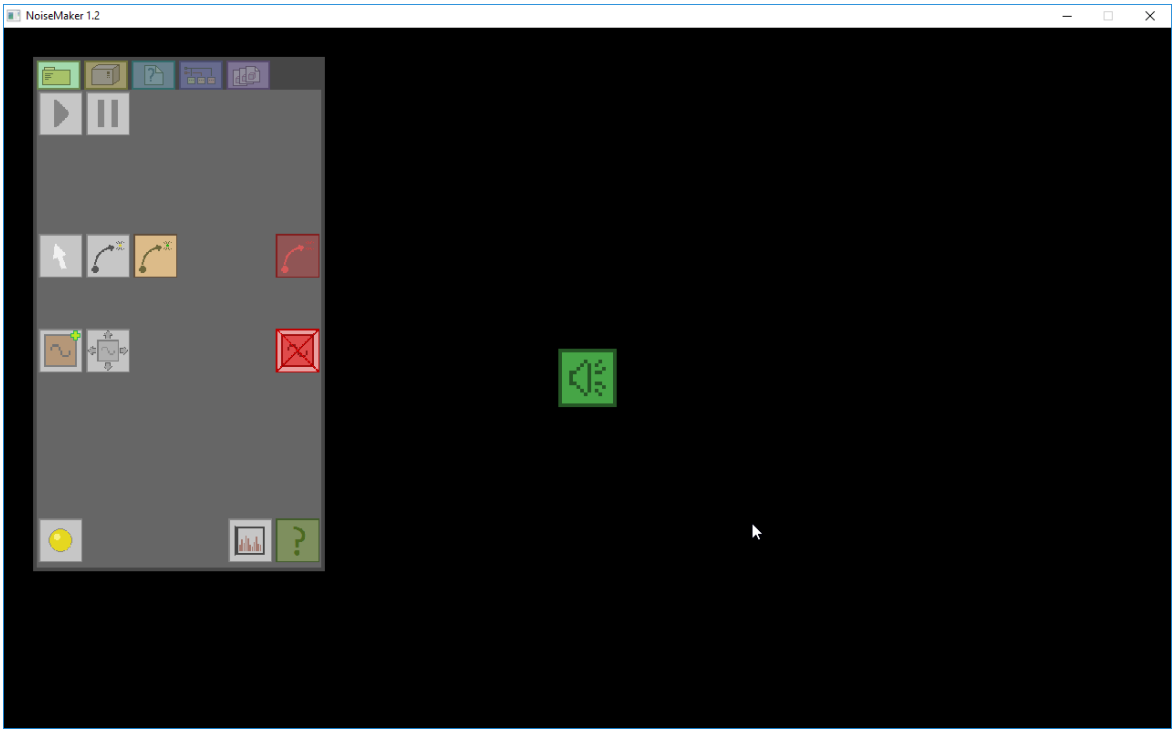
Content and explaination:

- *resolution.X* - screen width, in pixels (default 1024)
- *resolution.Y* - screen height, in pixels (default 768)
- *resolution.FullScreen* - 0 for window mode, 1 for fullscreen (default 0)
- *audio.Driver* - modify if you have troubles with sound (e.g. audio is not playing); 0 for OpenAL driver, 1 for BASS driver (default 0)
- *audio.Buffer* - buffer length in bytes. Decrease for lower latency, increase if the sound keeps stopping (default 4096)
- *audio.BufferQueue* - buffer queue size, in buffers. Decrease for lower latency, increase if the sound keeps stopping (default 4)

Note that BASS driver will use buffer of total amount of *audio.Buffer* * *audio.BufferQueue*. The division matters only for OpenAL, and it's usually better to increase *audio.BufferQueue* value only (if the sound stops). Anyway, OpenAL is considered faster than BASS because of its low-level design. You may notice high CPU usage - don't worry, it's for the sake of high performance.

3. First steps, hello noise world

Launched the program already? Let's take a quick look at the UI:



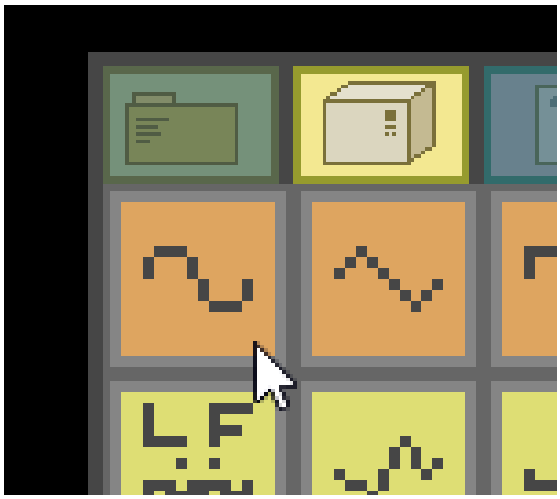
This is how your working panel looks like. On the left side you have a menu with different tabs; general options, like play, pause or record sound, tool selection etc.; the block library which will allow you to select blocks for your algorithm; parameter value/input control tabs, where you can set your parameters, or link them to read output of another block (yay!), and finally import/export tab, where you can load or save your hard work, and even share it with others - but this last thing must be done manually.

The green annoying symbol with a speaker on itself is the output node. It always appears after new project is started, and provides linking to your speaker system.

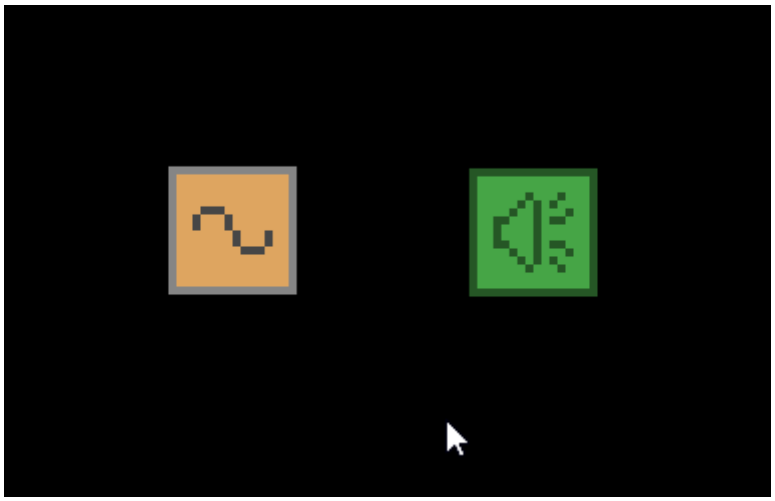
Remember one thing, if you are confused at the moment, you can press H key (if not done yet) and point the mouse over some fancy-looking components. In the right bottom corner the hint will appear. Unfortunately, I don't think about translating them to other languages, so english is required (otherwise you wouldn't read it, right?)

Generating your sound

In todays world, nothing can function without energy, no factories, no extraction, no light in your fridge without power plants. That's how it work here too - the first thing to bring your algorithm to life is to set up such a power plant, which is called a Generator. Switch to Library tab.



There are different kinds of Generators: sine, triangle, square, noise... let's try the simplest one, the Sine Generator - as you can assume, it generates the sine waveform with a given frequency and amplitude (volume). Basic setting for this one is *400Hz* and *0.5* (half-max) volume. Left click on the sine-like symbol (just under General tab) and put it somewhere near the green Output Node on the screen - the position and distance doesn't really matter, it's not a game; exact placement is always up to you. After all, right click to disable adding.



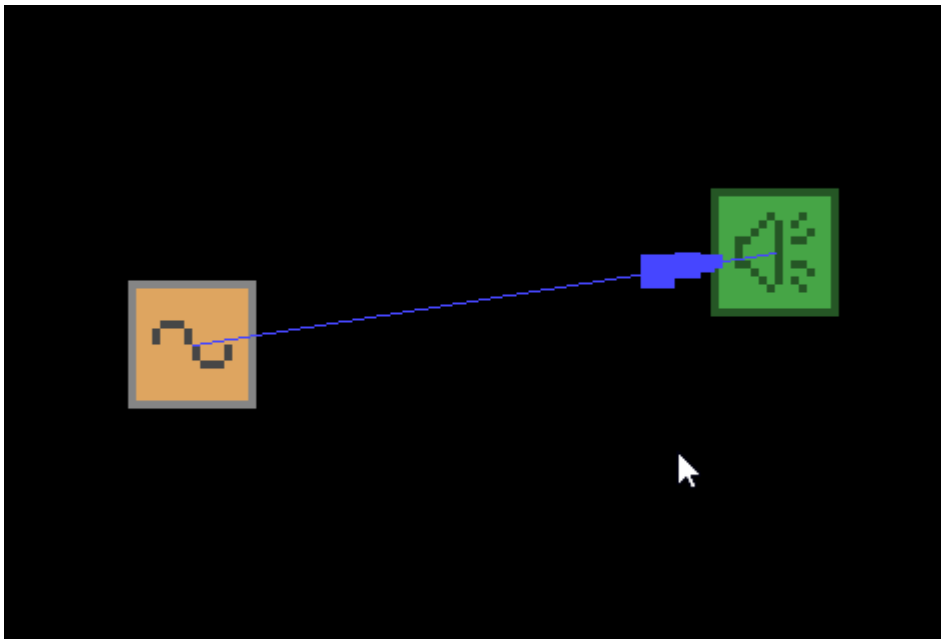
First step done. You can add any block in the same, simple drag&drop way. Or multiple copies of the same block, where every single one will work on its own. You can spam whole map with generators, clicking here and there until your computer stops. Switch to General tab and press the Play button.

That's right, you hear nothing. Simply adding the generator won't work. You have to link it to the output first.

Linking, how it works?

A link is a flow control object; an arrow, which indicates that data is passed from source block (like the Generator) to destination block, and sometimes further, but never backwards. Linking is simply telling the program that you want the data from one block output as the other block input; think of it as of a teacher, that wants to learn kids about gardening. The knowledge is the data, the teacher is a source, and kids are destination. Link for this knowledge would be from teacher to kids (source => destination). But what if teacher knows nothing about gardening? That's right, he should go to the library and take a book. The book here might be considered as another data, and library as a generator (let's skip all the book writing, printing and distribution process, please...) - so the full track would consist of three nodes and two links, like library => teacher => kids.

To start marking link, make sure you selected Linking tool in the General tab (keyboard shortcut L), point over your source Generator block, and drag it over the destination green Output Node. Releasing button over blank screen results in no link.

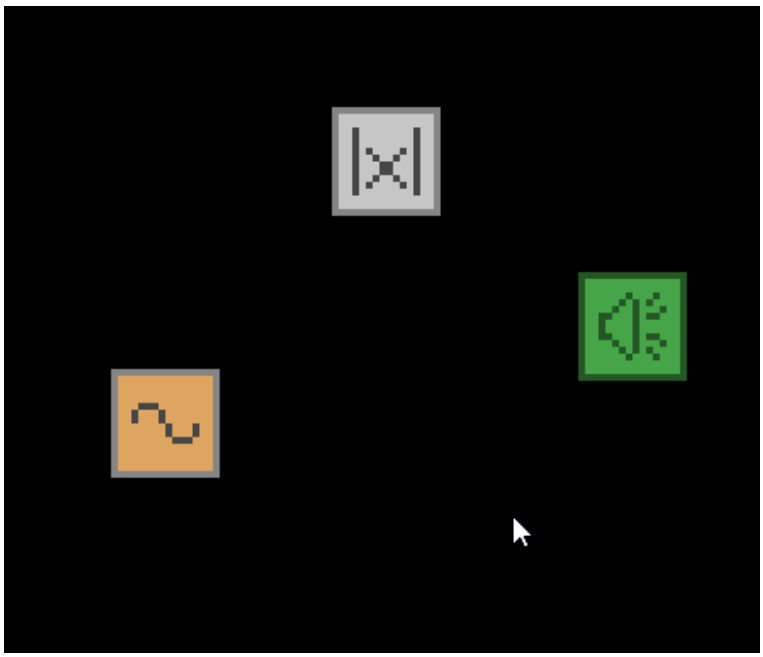


Now generator is successfully linked. Press the Play button once again (or Return key on keyboard). Magic, isn't it? Pure 400Hz wave. When you finish listening, press the Pause button or the Return key again, get over the blue "arrow" and press backspace; the link was removed - you can also remove links by selecting the Remove link tool and left clicking on or near the arrow. Don't worry, you can set this link once again after that.

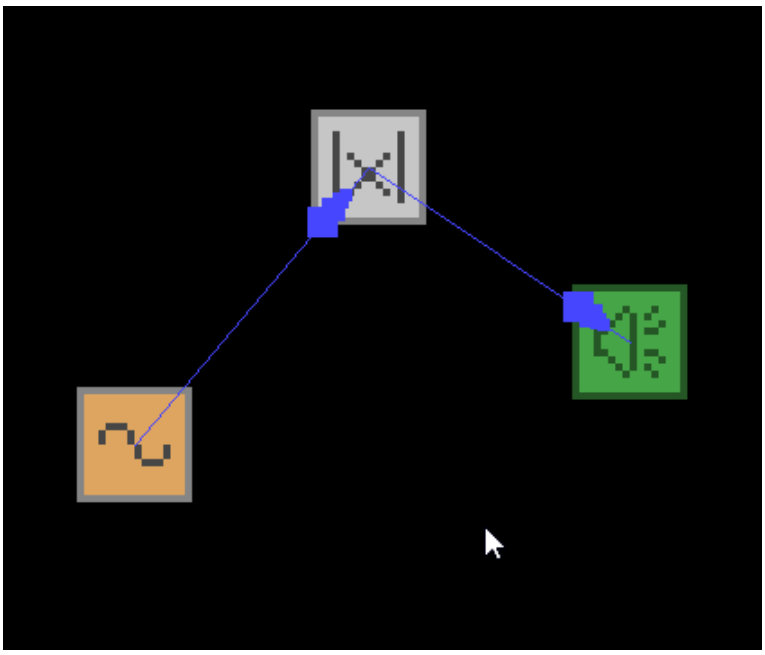
No more math!

Uh, I've got bad news... the algorithms are all over the math and logic, so if you don't like it at this point... anyway, dragging and dropping, and linking fancy blocks with weird images can still be considered as math!

You might have noticed that different blocks have different colors. This has been done to group them in the particular way. Thus, the orange blocks are wave/constant generators, yellow are noise generators, blue blocks are joining gates, and gray ones are math processors. Finally, we have comparators in light brown/dark yellow, stereo operations in dark gray, and noise effects in red color. Let's start with Absolute (the light gray block with |x| symbol) - it calculates an absolute value of the input, and passes it on its output, like most processing blocks do. Find the Absolute block in menu on the left side (Library tab) and place it somewhere around your Generator and Output.



I guess I've missed something?

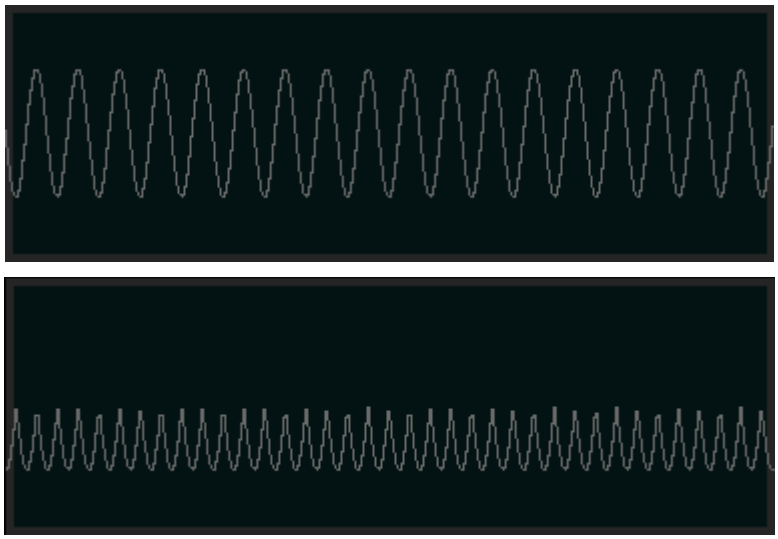


Now it's better. Link your Generator to the Absolute, and Absolute to the Output using Linking tool. Here's how the process works now: Generator generates simple 400Hz sine waveform, and passes it to the Absolute by link. Absolute processor gets the input, calculates it to an absolute value and sends to the Output by its outgoing link. The result is $|sin(t)|$, if the math speaks to you in a better way. Press Return key to hear the output.

Visualizations

No, I didn't forget about the graphic representation of your sounds. Go to the General tab and press Select Visualization button. The dialog will appear, and you can select Oscilloscope, Specter or Vectorscope tool - one at the time - click on the image on the left side, or Cancel on the bottom to exit the dialog. (None) option is to disable the present visualization tool.

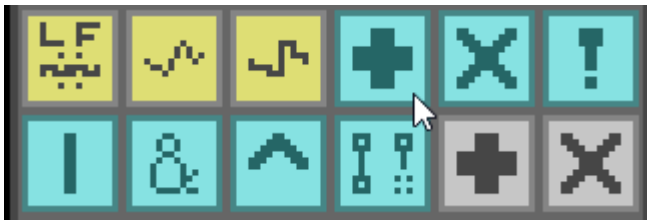
The selected visualization shows up in a right-bottom corner, near the help text (if enabled). Here's the comparison of waveform on Oscilloscope from the first and second algorithms, *sin(t)* on the top and |*sin(t)*| on the bottom.



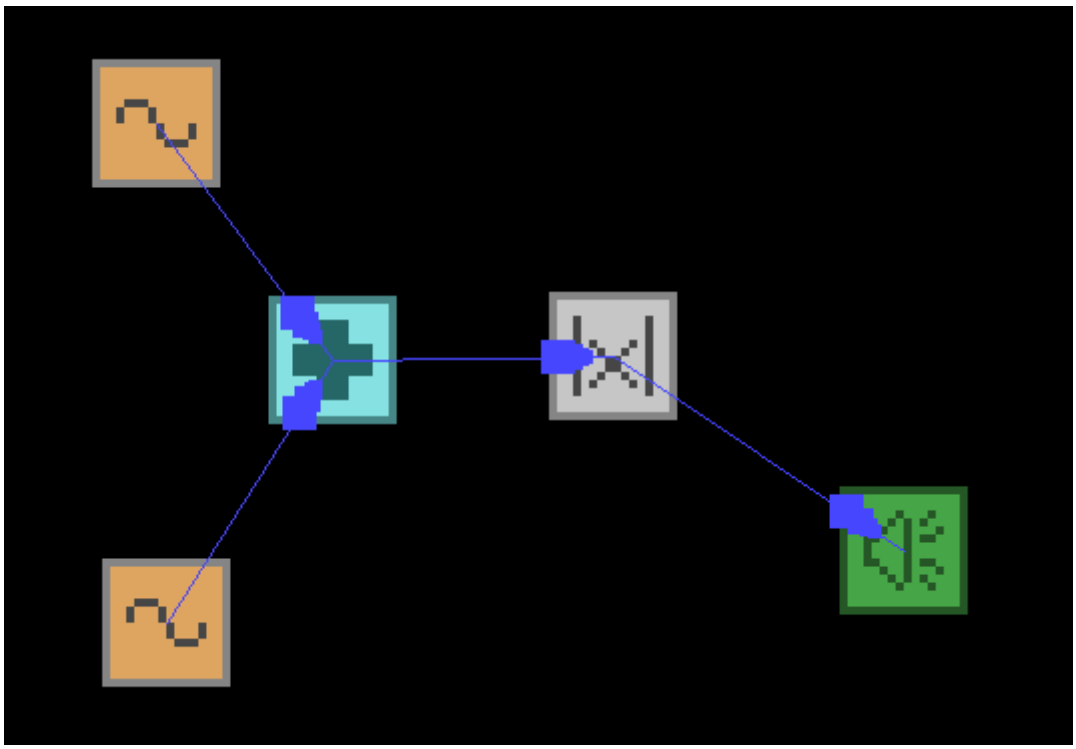
Although the |*sin(t)*| might look more like -|*sin(t)*|, it's just a matter of display, the processing block Absolute calculates the samples in a proper way. If it doesn't bother you, let's go to the next chapter.

4. Sums and Gates

You can place all these block, and set them into chain, calculating |||||*sin(t)*||||| and further... but what about calculating, for example, a sum of two inputs? Of course the Output does it on its own, but what if we want an absolute of the sum, like |*sin(t1)+sin(t2)*|? Go to the Library tab.



Here, marked in light blue (or cyan, I dunno), we have gates that can sum up to N inputs. Yeah, even 1000 or more, if your computer will manage that. The cyan gate with adding symbol is the Adder gate. Click it and put aside of the Absolute from our previous algorithm.



...and you guessed out, we need a separate generator. Of course you can add any generator you like, but I'll stay with sine. Link two Sine Generators to the Adder, then Adder to the Absolute and Absolute to the Output, so it will first sum up the two incoming inputs, then calculate absolute of the processed sum, and play it on your sound system. Yup - nothing special, these generators are just generating constant, annoying bleep, and nothing else. Let's tune them up a little.

Setting parameters

Switch to the Parameter Value Control tab - the third from the left, or the middle one. Got it? Now click on one of the generators.

For the sake of readability, frequency value was shortened to *Freq*. It says 400 [Hz], if you didn't change anything before. Click on the edit field to catch input focus - the field marks green if you have focus on it - and write, for example, 400.5, then press Return or simply click outside of the text field. It should result with two sines interfering with each other. Just as regular two waves with similar frequencies. Now click on the edit text once again and raise the value to 777 (backspace to erase characters). Now this is annoying sound, huh? You can always set any parameter listed in this window to any value.

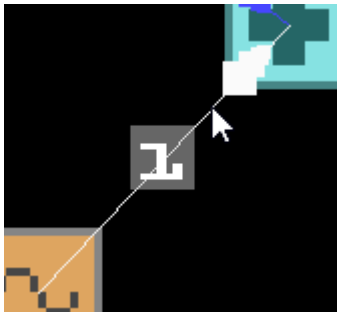
No more hard clipping?

In the example above there is parameter *Vol*, a short for volume. Try to set frequency back to 400.5, then the volume to 0.9. Do you hear how the sound is ocassionally unmuffling? That's because the peak sum of two volumes in the example is *0.5 + 0.9 = 1.4*, which is way above the maximum *1.0* level. Hard clipping is an additional condition that if the volume is more than 1.0, cut it down to the level of 1.0. Here, the audio data is buffered as *short* type (16bit value), and calculations are performed onto *floating-point* values. Because floating data range is a little wider than *[-32768..32767]*, the value is automatically rounded as a modulo of 16 bits (which probably happens when float is converted to int, and from there, the two youngest bytes are selected... *blah, blah*. Please refer to the Wikipedia for more technical informations.)

Hard clipping was ineffective and would result additional impact on performance... but, remember: after all, it's not some techno making program but a *NOISE* software, so more painful sounds are welcome!

Input selection

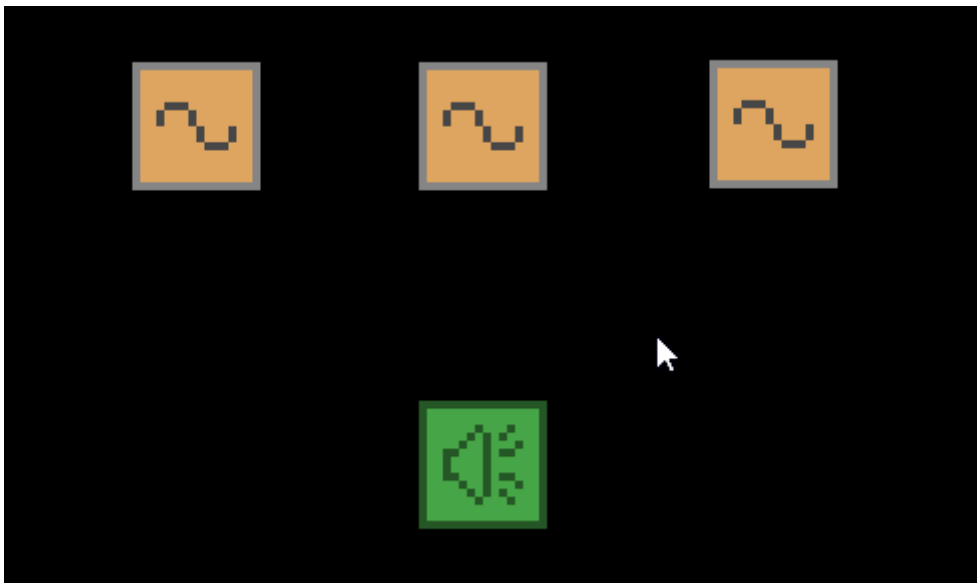
As you probably noticed, arrows become white when you point onto them, and they also display some weird numbers. This number is selected input number, more precisely - its identifier.



It doesn't really matter in such blocks as Adder or Multiplier, which iterates over every input (like for .. each loop if you're a programmer), but it matters for Input Selector block, which outputs the input specified in the parameter.

5. Advanced math

Let's say we need... ehh, go for it. We need a product of two waveforms: the integral of sine at 500Hz, and sum of sine 30Hz with reversed 800Hz. Looks hard? Let's find the solution together. We definitely need three waveforms, so three Sine Generators will fit.

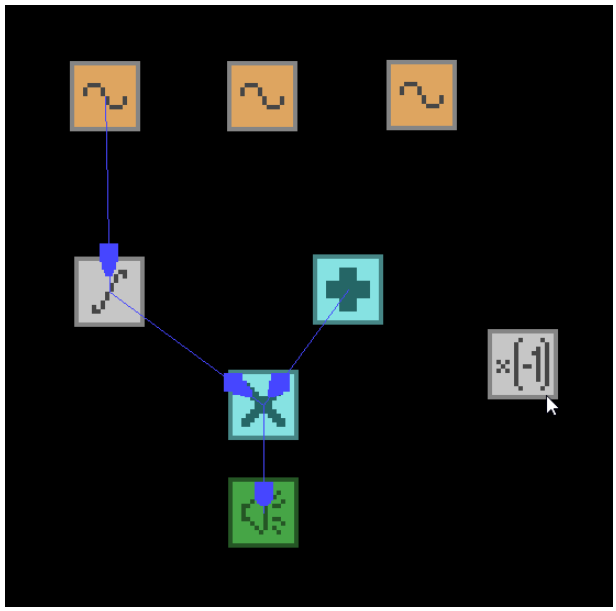


Set first on the left to *Freq=400* and *Vol=0.3*, the middle one to *Freq=30* and *Vol=0.3*, and last one, the far-right to *Freq=800* and *Vol=0.3*. Why the *0.3* volume? See No more hard clipping section if you didn't read it. They all would sum up to *1.5* at the peak, so we lowered it to *0.9*.

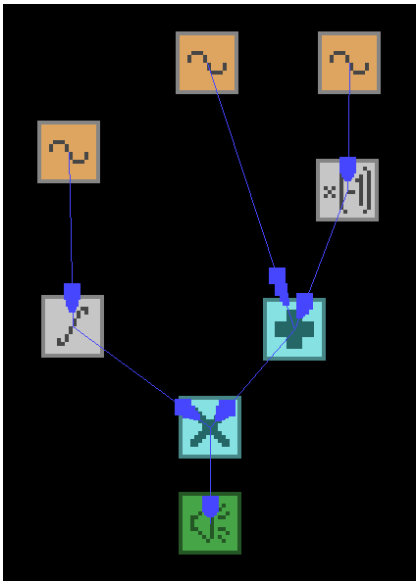
Now we need a product of an integral and sum of two, with one reversed. A product of N inputs is solved by a block called Multiplier (cyan one with X symbol). The Adder we know

already, so we can put Adder and Multiplier on the screen. What's with these integral and reversed things?

An integral is basically calculated "area under the curve", the opposite of the derivative; refer to the Wikipedia for more informations, if you're not into math. It is denoted by ∫ sign. A reverse (reversed polarization) is simply waveform multiplied by *(-1)*. Look for two of these blocks and put them on the screen. Remember that you can delete a misplaced block by Remove node tool (or point over the block and press Del key), or set it's position by clicking and draging the selected block.

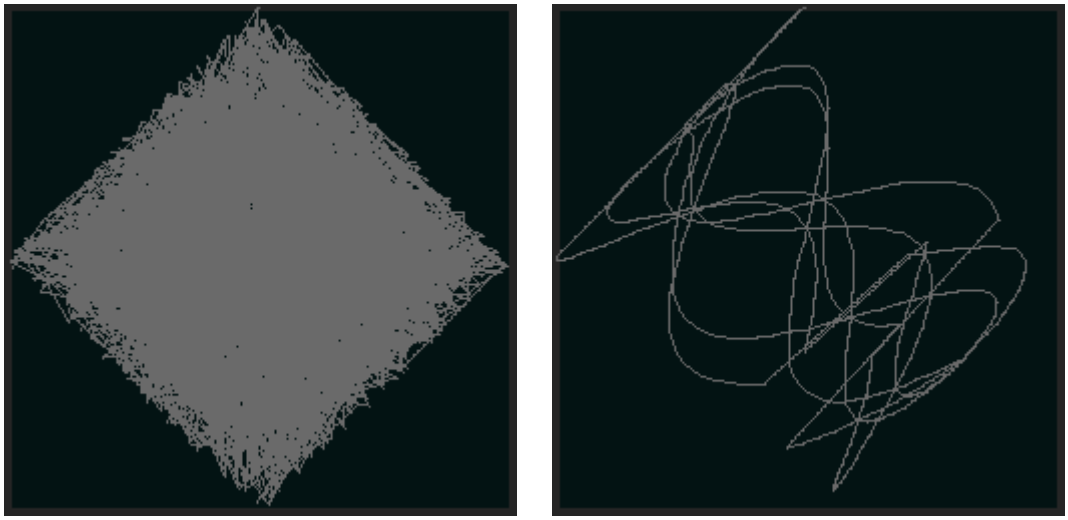


A product of sum and integral can be linked on the beginning. Same with the integral of sine (hey nerd, I know that $\int sin(t)dt$ is pointless, as it leads to $-cos(t)+C$, but that's not the point now) - linking from the top-left generator. Where to put the Reverse Polarization block? As we need a sum of the middle generator and "reversed" right generator, link third generator to reversing block, and this block to the sum, together with the second generator:



Stereo control and Vectorscope

Remember that every sound generated in this program, even it sounds like a mono, is in fact always a stereo - 2 channels. It means you can swap, reverse, join or divide them - try replacing Reverse Polarization block from previous screen to Channel Inverter and compare outputs on vectorscope visualization. Why? The Reverse Polarization block multiplies both channels by *(-1)*, but Channel Inverter does that with one channel only, specified in parameter, and it yields to highest possible difference between channels (aka *destructive interference*).



The vectorscope shows the corelation between left and right channel: just as in FL Studio, the vertical axis *(Y)* shows sum, and horizontal axis *(X)* shows difference* between these channels, so the resulting point is:

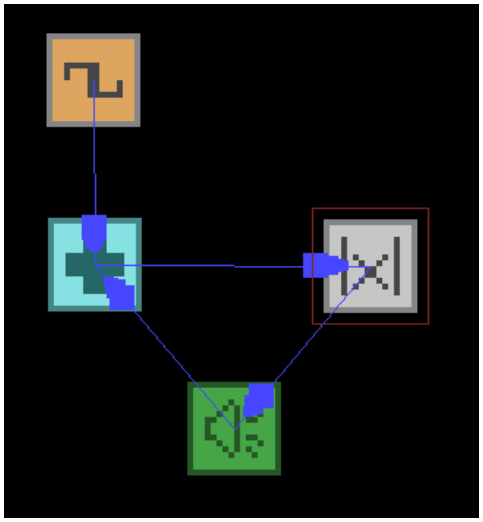
$$P(x,y) = P(L-R, L+R) \qquad (L, R) \in \mathbb{R}$$

This is the output for current point. Draw line to the point of previous sample, and you will start to draw your own vectorscope with that.

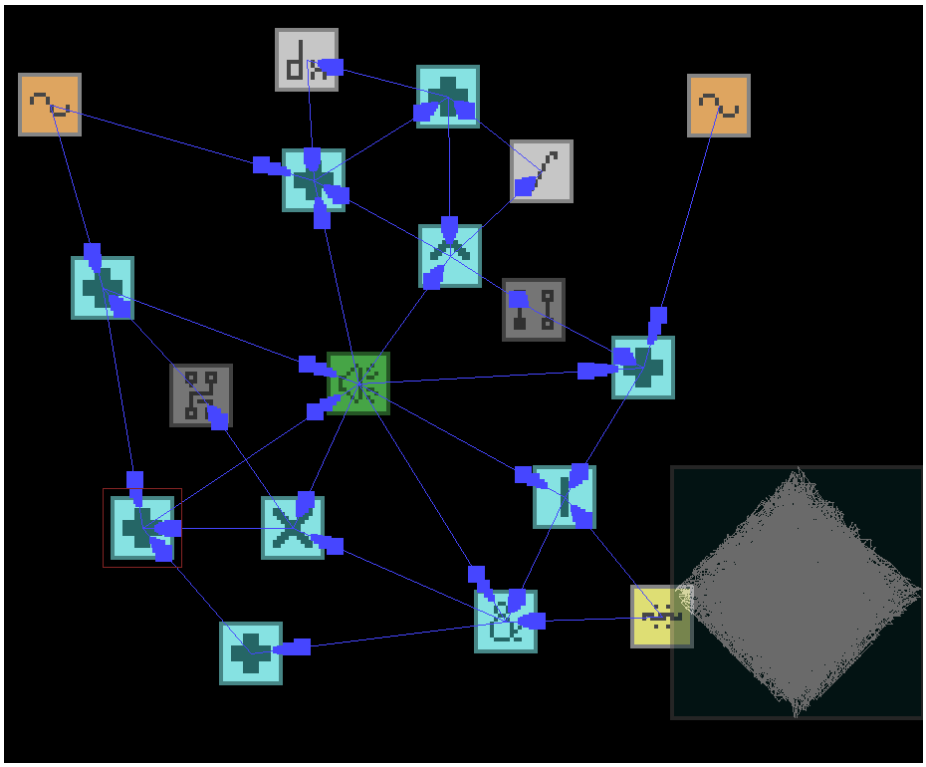
**Note that this is FL Studio-style vectorscope. Graphic representation is usually different, but one stays as a rule, that first line is for sum, and the second for difference between L/R channels.*

6. Feedback calculations

What if... we linked Output as an input for something, and link to the Output again? There is no problem, you are allowed to link any node to any other node if there is no limitation for inputs; let's consider how it works in real life:



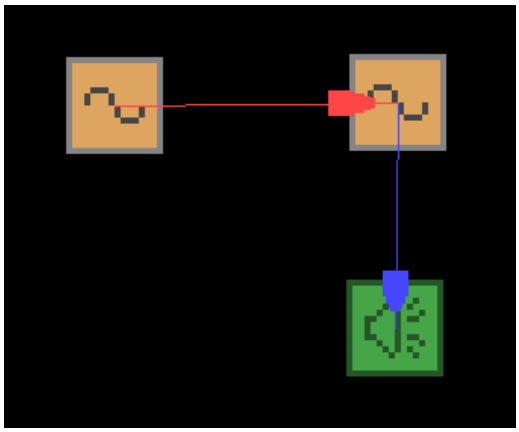
you can wire some cables together to make a circuit, you can join some pipes to keep water flowing over and over - if you are a programmer, you might worry about infinite detection loop now, but algorithm of determining processing chain is reliable, and any structure can be set. Even the one with many smaller circuits inside, where the (mere) human could easily get lost. I'd bet 100$ that there is no such structure where processing couldn't be determined (I don't, because I'm a mere human too, I didn't check *anything* that is possible).



Yeah... I can't really describe what it does. Except that it works. Works - just some data is transferred with a bit of latency, as the output buffer from previous tick is being used. You can even notice the small "8" loop containing Integrator and Derivator, it looks like an infinite one - but the data there goes in such a loop, and walks through each node and link continuously.

7. Output to parameter - automation control

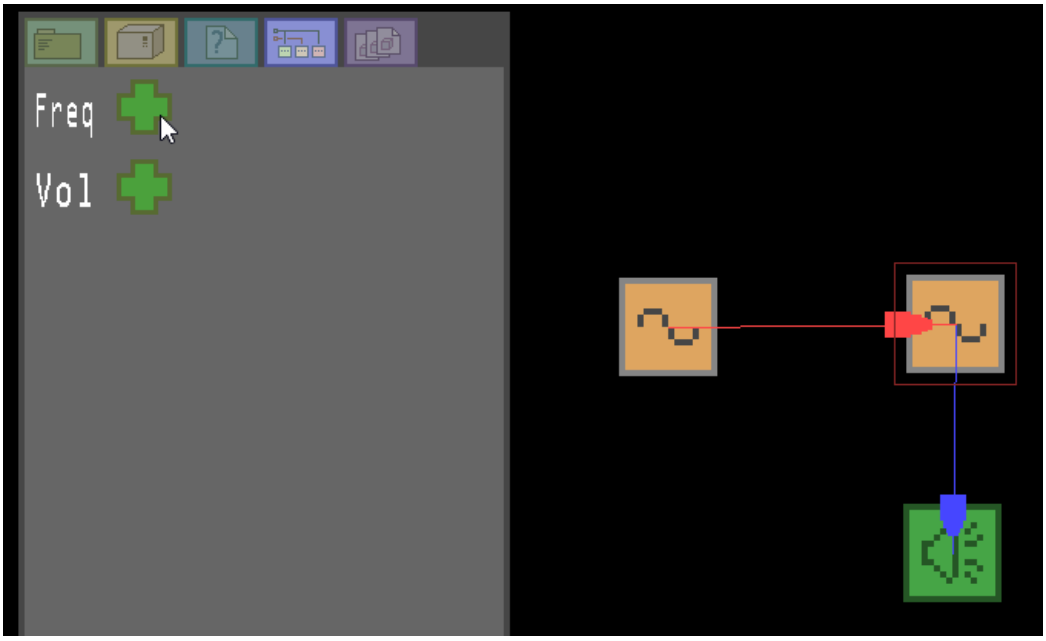
Tired of manual setting the parameter values already? Parameter links, here to help! Any parameter can be linked with anything that outputs data (that is, with everything, because there are no "consumers", only "producers" and processors). If you, for example, need a Generator that does not emit constant sine wave, but the pitch goes up and down, same with volume, you can link another sine generator to *Freq* parameter. Go to the Library, set up two sine generators and link one generator to the Output; select Linking input tool - shortcut P - then create link as a normal link, that is, left click on source (controller) node and hold, point over the destination (controlled) node and release the mouse button.



Now it corresponds the expression of *sin(A*sin(t))*. You might have noticed that there is no input number on the parameter link (red arrow). That's because parameter linking consists of two stages: linking nodes and input selection.

"WTF, it doesn't work!"

Yup. Not yet. But we can get it fixed. Click on the sine generator linked to Output to select it, then switch to the Parameter Input Control tab.

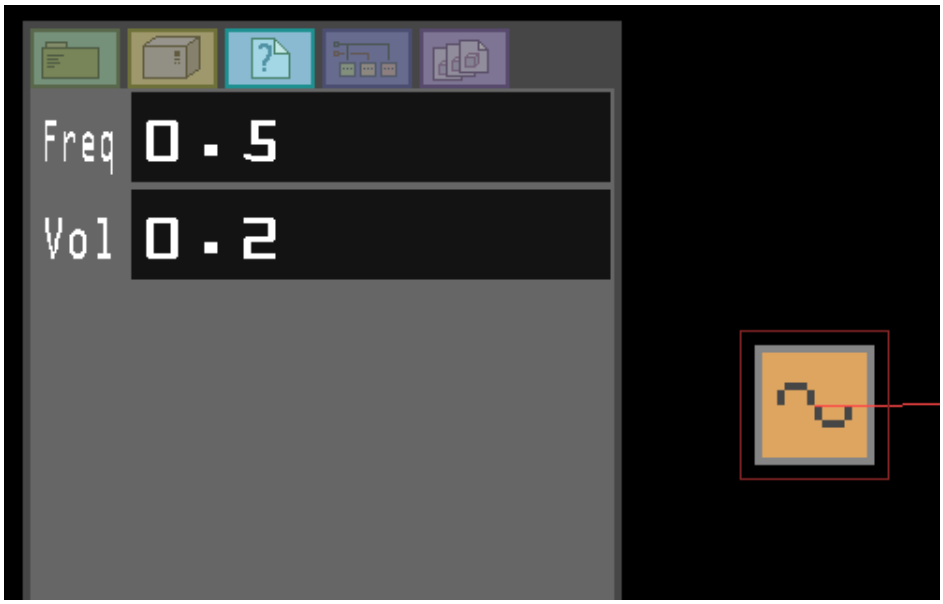


Here, when you click on the green "+" next to *Freq* parameter, the dialog opens:

This is where you can select the other node, which will provide input for this parameter. If you no longer need parameter link, click *(Unlink)* or simply remove link pointing on the red arrow and pressing Del, or using Remove link tool. Be aware that clicking *(Unlink)* won't remove your node from the selection list, it only "dismounts" the other node; link still exists. Let's pick the *Sine Generator [2]* for now; the *[2]* number is just ID, to differentiate the possible source nodes if they have the same name - although I wouldn't reccomend holding 500 possible sources unless necessary.

Okay, the parameter *Freq* is now "tied" with sine generator. Let's switch to Parameter Value Control tab for a while.

As you can see, the edit text value is marked red - that means it's not editable right now, the reason is that the parameter is controlled by an external node. Let's move to that node.



The *Freq* parameter here means that frequency of the resulting sine wave generated by the other node is changing over time, in a period specified by frequency of this generator's sine wave. To make it noticeable, I set it to 0.5 [Hz], which makes an LFO. The *Vol* parameter determines this sine wave amplitude, so it is audible as *how hard the resulting frequency drops*. You can put some Constant Adder or Constant Generator to increase/decrease the LFO middle point. For example, if you sum this *Vol* with constant *0.2*, it means that pitch amplitude and frequency will stay, but the

resulting tone will be higher. And here comes another thing - you can control that by another parameter, and that one by another, and another... imagination is your limitation.

You can set any wave as an input for the parameter - you can even do it in the loop, as the ***Feedback*** chapter shows.

8. Templates import/export

I accidentally (closed) the template, how can I have it back?

Well, the first step is to make sure that you export your work regularly - that's a good thing. This program is not very stable, although it can work for more than 3 hours ahead (checked), there are still some unresolved problems. There is no "Safety parachute" that some programs drop, when something fails. If this program fails, it's always a *hard way* - luckily not so often.

To export your template, go to Import/Export tab. Input file name in the black text field near the *e* button. If you don't specify extension, it becomes a .dat by default.

Click Export button e. The dialog should message you with the result. Be aware that you can't export to the existing file - that was done to avoid overwriting previous template by user accident. To Import your exported template, click i button above. The dialog will pop up:

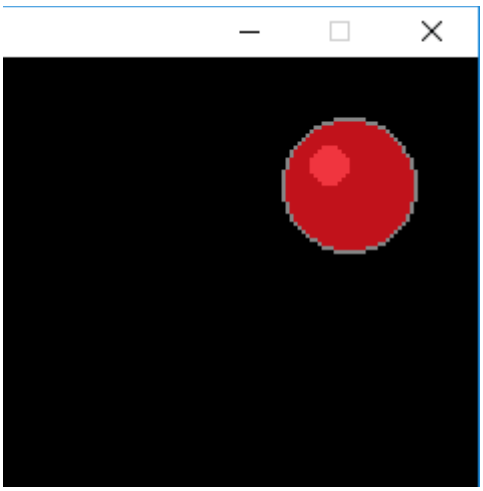
In this dialog, you can select your template to import. Click on it and work.

IMPORTING ANY TEMPLATE WILL OVERWRITE CURRENT TEMPLATE WITHOUT ASKING. Same as if you exit the program - the unsaved data gets lost.

All template files are stored in data\userTemplates folder, non-recursively. No file extension is specified, so you can load any file.

9. Recording your sound

You don't have to use third-party audio recording software. Just switch to General tab and click Record button, the one in the left bottom corner of UI panel, with a yellow dot. Now, as you can see, the bigger red dot popped out in the top-right corner of the window. This means that your audio is now recording.



Do anything during the audio record to complexify your sound. Or don't. But be aware that if you press Pause (or Return key while playing), the Recording tool will not record the break - in opposite to generating quiet waveform, where a flat line will be recorded. All the recording is performed in the real-time, if the sound is playing. Note that the break time might be not so precise, as the generated buffer is written first to file, then played. This might lead to little overhead in the file; but it's really low, as one buffer is approximately 1/43s long.

The resulting filename is *recorderOutput_<<timeInMillis>>.wav*, e.g. *recorderOutput_1573709594.wav*. The file is located in main folder of an .exe.

DON'T FORGET TO PRESS RECORDING BUTTON AGAIN BEFORE QUIT. Otherwise the file will be damaged. Pressing the button stops recording and closes the file in a clear way, then the .wav is ready to be played or put in another software. Good luck!

10. Useful shortcuts

Space (hold) - play sound until pressed
Return - play/pause sound
H - quick help (toggle); informations about items pointed by mouse
Backspace - remove link under the mouse pointer
Delete - remove node (block) under the mouse pointer
L - node-to-node linking
P - node-to-parameter linking
M - move node (click and hold instead of select and click again and hold)
[**1**..**5**] - quick access menu tab (1-General, 2-Library, 3-Parameter Value Control, 4-Parameter Input Control, 5-Import/Export)
Escape (when in dialog) - exit dialog; same as pressing Cancel button
Arrows - move panel (moves display of the whole algorithm already; useful for extended schemas)

11. Additional informations / Bug reporting

In case of any infos required, ideas you came up with or bugs you've discovered, contact me on Facebook or YouTube. If there is any bug, please also provide as many details as possible - remember, it's C++, not Java, it's hard to track an error when someone says "it returned with 0xC0000005", as it can be thrown anywhere, at any time.

As of writing this manual, the release was not stable, and some problems were caused by multithreading. Please be patient, I'm working alone over all these sources and stuff ;)

And, the last but not least - any feedback appreciated! I hope this tool will help you with samples for harsh noise, speedcore and extratone music. Regards!

Diabarha