SOVNDSPEAR

USER MANUAL

Formula

IMPORTANT:

The software, when used in combination with an amplifier, headphones, or speakers, may be able to produce sound levels that could cause permanent hearing loss. DO NOT operate for long periods of time at a high level or at a level that is uncomfortable. If you encounter any hearing loss or ringing in the ears, you should consult an audiologist.

TABLE OF CONTENTS

PLUGIN OVERVIEW

Compatibility

Windows	**11**	**Compatible**
	10	**Compatible**
	7, 8	Untested
	< 7	Not Compatible
macOS (Intel, M1)	**11, 12** **10.11 to 10.14**	**Compatible**
OS X / macOS	10.7 to 10.11	Untested
OS X / macOS	< 10.7	Not Compatible

Functional overview

Within Formula, you can create your own plugins inside your DAW and access hundreds of pre-made plugins from the community.

🎸 **Music composers, sound engineers:**

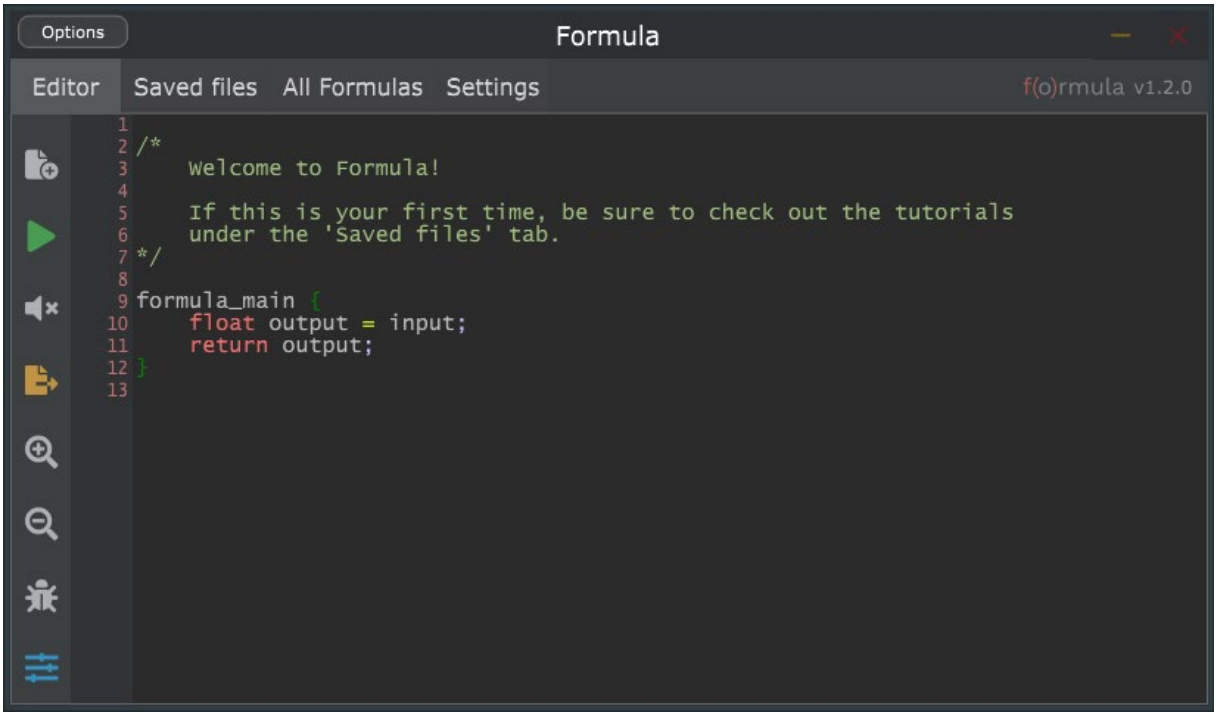
Use hundreds of bundled Formula effects made by developers from around the world.

👨‍💻 **Developers:**

Live code, debug and test your effects right inside your DAW.

INTERFACE

The plugin can either be used as an effect plugin inside your DAW (VST3 or AU), or as a standalone application. In standalone mode, you can load an audio file that will be used as an input to the effects from the Settings tab



Tabs overview

There are 4 different tabs you can access through the top navigation bar:

- **Editor**: Modify the code of the active formula and launch it. Access the knobs and switches to interact with the code.
- **Saved Files**: Access and load the formulas that are saved on your computer. The application is shipped with a few formulas and tutorials under this tab.
- **All Formulas**: Browse and use hundreds of formulas created by the community.
- **Settings**: Application settings.

Editor

You can modify the active Formula in the Editor tab. The typical workflow is to load or modify a formula and press the play button on the sidebar ( ). Then play a sound from your DAW and use the knobs panel ( ) to modify the formula parameters.

The active Formula and the knobs settings will be saved by your DAW within your project.

Code editor

Within the code editor, you can create your own formulas using a simplified version of the C programming language. More information about creating your own formulas is available under the DEVELOPER GUIDE section.

Whenever you want to test your changes, click on the button in the sidebar to launch your formula.

```
21
22 formula_main {
23     float bias = KNOB_1;
24
25     int neg = input < 0;
26     input = fabs(input);
27     float x[4] = {input*input*input, input*input, input, 1};
28
29     for (int i = 0; i < 2; i++) {
30         y[i] = 0;
31         for (int j = 0; j < 3; j++) {
32             if (input > domains[i][j]) {
33                 continue;
34             }
35
36             for (int c = 0; c < 4; c++) {
37                 y[i] += x[c] * splines[i][j][c];
38             }
39             break;
40         }
41     }
42
43     float output = y[0]*0.5 * (1-bias) + y[1]*bias;
44     if (neg) {
45         output = -output;
46     }
```

Sidebar

The sidebar offers several actions related to the editor:

Discard the current formula and create an empty one.

Launch the current formula. You must click this button when you make a change.

Mute the application output.

Save the current formula to your *Saved Files* tab. If it is new, you will have to input a name and a description.

Increase the zoom of the application.

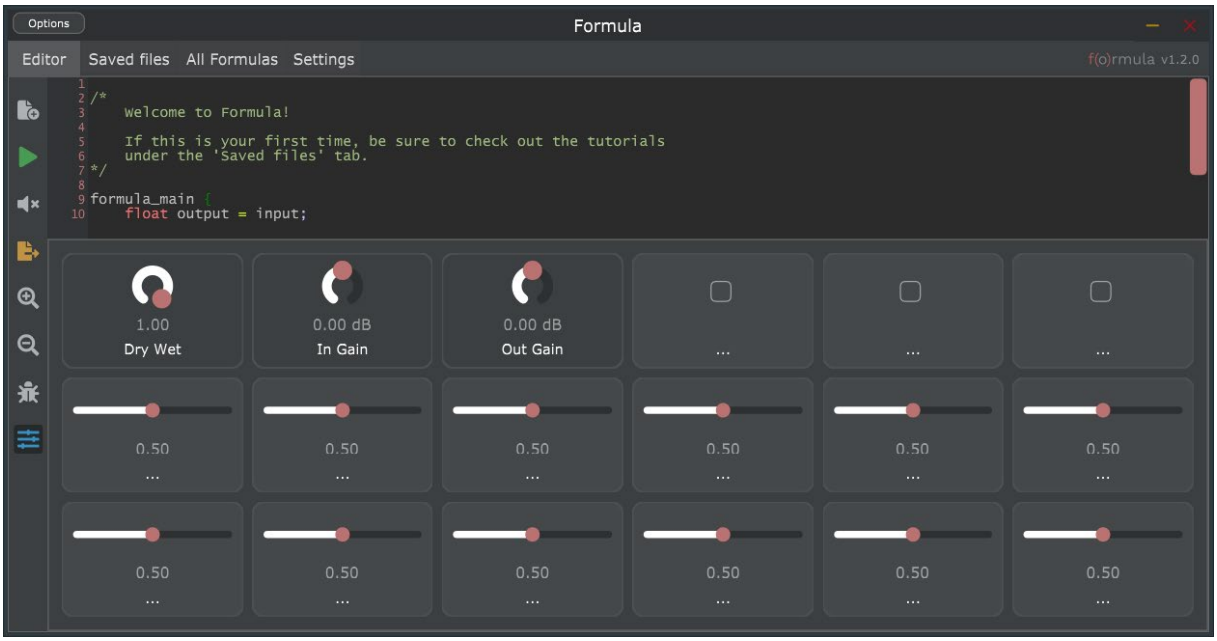
Decrease the zoom of the application.

Show the debugging output. Only useful to developers.

Open the knobs panel. You can tweak the knobs and switches of the current formula.

Knobs panel

As with traditional audio plugins, the formula behaviour can be controlled or automated using knobs and switches. When toggling the knobs panel with the button, you can change the value of those knobs and switches.



If you are using an existing formula, the formula author should have labelled the knobs and switches his formula is using. Typically, the knobs and switches that have the … label are not used within the formula.

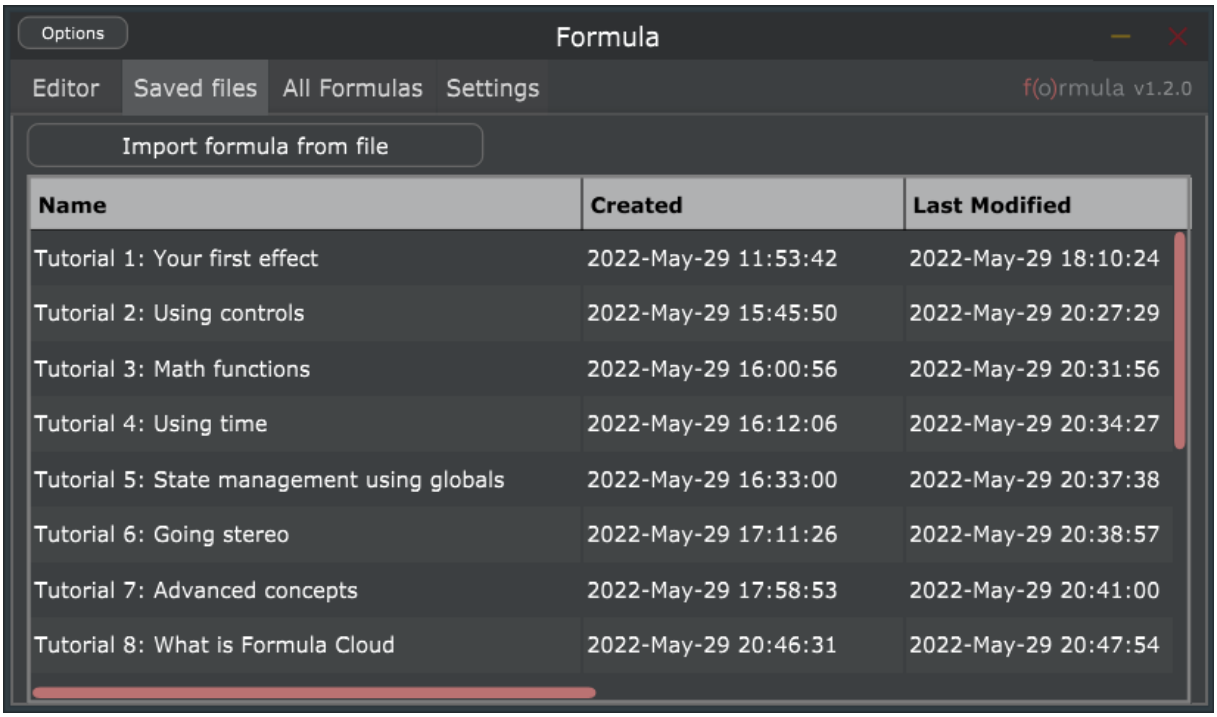
Hence, you know what the use of each knob is.

Inside your DAW, the knobs are available as parameters labelled from *Knob 1* to *Knob 12* with values ranging from *0%* to *100%*, and the switches are labelled from *Switch 1* to *Switch 3* with values being *On* or *Off*.

If you are creating your own formula, the value of every knob is available within the editor under the KNOB_1, KNOB_2, …, KNOB_12 variables and the switches under SWITCH_1, SWITCH_2 and SWITCH_3.

Saved files

Under the Saved files tab, you can find the formulas you saved from the editor using the  button.



Default files

If it is your first time lauching Formula, you will find 7 tutorials in case you wish to learn how to craft your own formulas.

Import and export

Your saved formulas can be exported to standalone files and imported back from another computer:

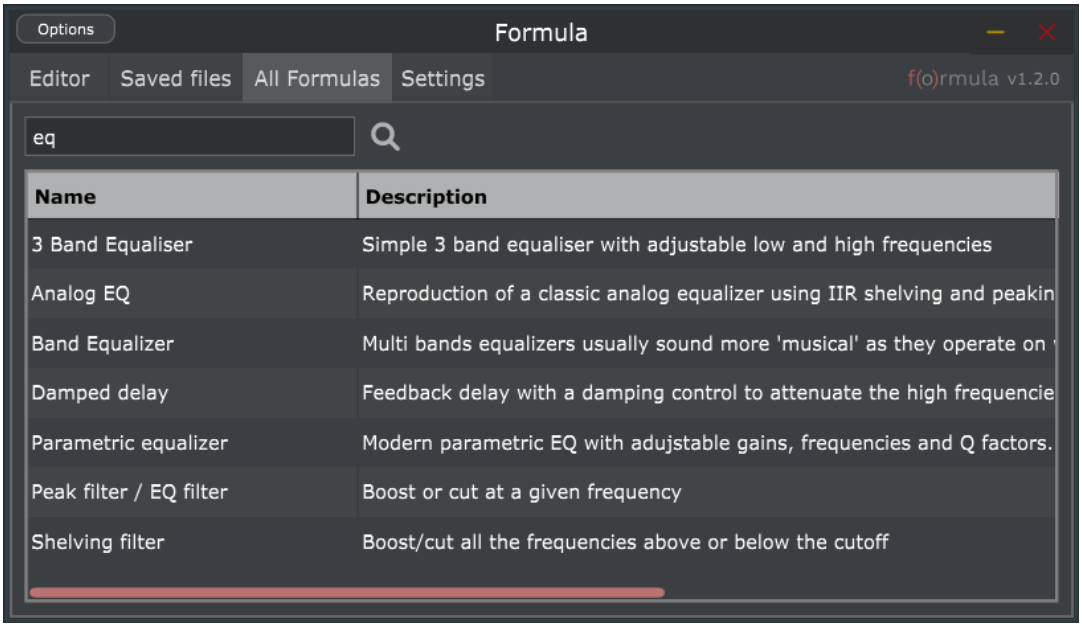
- Load a standalone formula file by clicking on the *Import formula* from file button on top of the tab.
- Click on a formula and choose *Export to file* to save it as a standalone file.
- Click on a formula and choose *Delete* to remove it from your local files.

You can make your formula available to everyone through our open-source repository at https://github.com/soundspear/formula.

All Formulas

In the All Formulas, you can find all the community-created formulas, that were exported and further shared in our open-source [Git repository](#).

- List and search formulas published by other users using the search bar on the top and clicking on the search button 🔍.



- View the details of any published formula by selecting a formula.
- Load a published formula into the editor by clicking the *Load formula* button after selecting a formula.

Settings

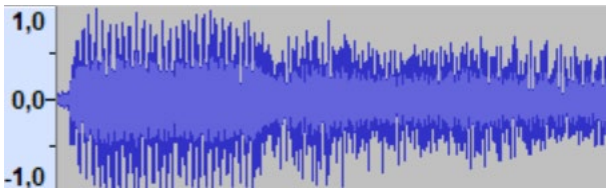
You can change application settings under the Settings tab.

DEVELOPER GUIDE

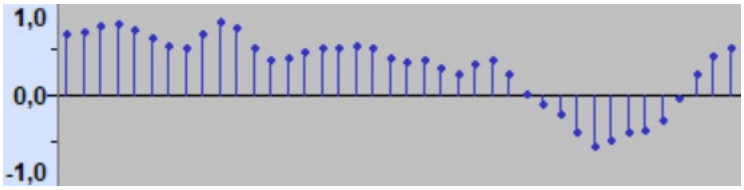
Audio programming 101

In Formula, you will either create or modify an audio signal.

An audio signal looks like this:

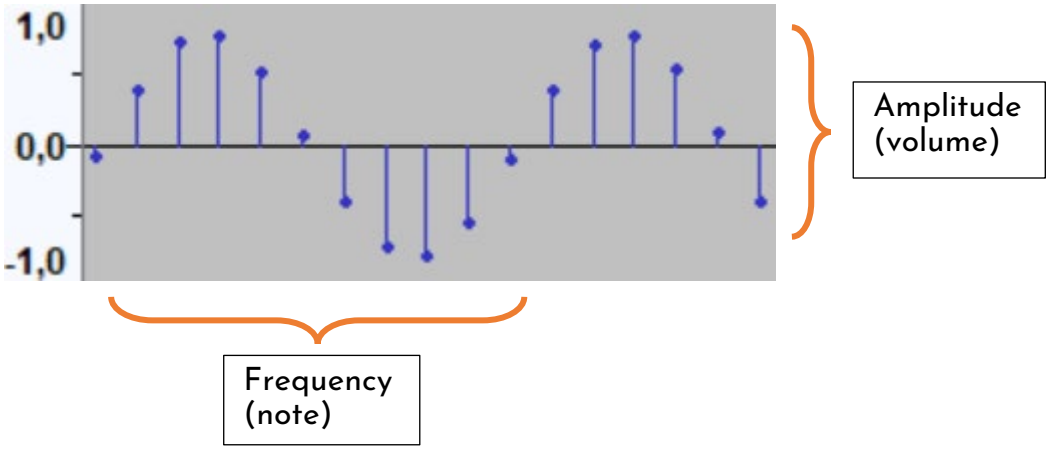


If we zoom a bit, you can see a **collection of points**. Every point has a value **between -1 and 1** and moves ahead in time given a fixed step.



For instance, the first point is 0.68 at time 0. The next point is 0.69 at time 10 milliseconds. The next one is 0.73 at time 20 milliseconds. The time step between points is called the sampling rate.

This audio signal/collection of points is made of several **harmonics** that are added altogether. A single harmonic is a sine signal with a given note and volume:



By adding many harmonics together with different pitches and volumes you will find back the above audio signal.

The goal of an audio processor is to have a function that modifies or generates a single audio point. This function will then be called for every point coming from the audio signal.

Programming in Formula

C language

Formulas are programmed using the C language. Other languages are available for corporate builds with restrictions mentioned below being removed:

- **Entry point**: Unlike in a typical C program, the entry point is not the main function. Instead, you would use an entry-point macro: either `formula_main` or `formula_main_stereo` (*see Formula architecture*).

- **Macros**: Interaction with the DAW (user control, sample rate, …) is made using macros (*see Macros*).

- **Includes and preprocessor**: For safety reasons, we have disabled the C preprocessor and it is not possible to include libraries in a formula. However, both the standard C library (**stdlib.h**) and the standard math library (**math.h**) are included in all formulas.

- **Security**: Along with disabling the preprocessor, several security measures have been enforced: dynamic memory allocation and management, system or exec calls and inline assembly are not allowed.

Formula architecture

As explained in the previous sections, we need to process each point of the audio signal. In formula, this is done in the `formula_main` block which is delimited by two curly brackets. In this block, you can access the audio point you need to modify using the `input` variable. Once you are done with processing your point, you must return it using the `return` instruction:

```
formula_main {

    float output = input;
    return output;

}
```

If you hit the ▶ button after entering this code, you will notice that the original signal is left untouched. That is normal, as we are not modifying the input samples.

Since increasing the amplitude of every point will increase the volume (refer to Audio programming 101), the simplest formula would be to multiply the input points by a fix value to increase the overall volume:

```
formula_main {

    float boostedInput = input * 2;
    return boostedInput;

}
```

Macros

Several values can be retrieved from macros that are specific to Formula:

Macro	**Usage**	**Value range**
`KNOB_1, KNOB_2, … KNOB_12`	Value of a user knob from 	From 0.0 to 1.0
`SWITCH_1, … , SWITCH_3`	Value of a user switch from 	Either 0 or 1
`SAMPLE_RATE`	Sample rate value used by the host DAW (or the operating system in standalone mode)	Positive floating point value
`TIME`	Time in seconds elapsed since the launch of the application	Positive floating point value
`DEBUG(x)`	Print the value of a variable in the debug panel	Any integer or floating value

Example:

```
formula_main {

    float output = input * KNOB_1;
    return output;

}
```

State management

You might need to store values between samples processings. For instance, a filter might need to reuse the previous outputs and inputs.

The way to do so in Formula is through global variables that you define outside of the `formula_main` block.

```
float globalVariable = 0;
formula_main {
    globalVariable++;
    DEBUG(globalVariable);
}
```

By running this code and triggering the debug pane (), you will see that `globalVariable` is increasing over time, as this variable keeps its state between consecutive samples.

Multi-mono and Stereo

By default, Formula runs in multi-mono mode. It means the same code will be executed for every channel (typically stereo left and right) and the global variables will not be shared between those channels.

However, you might need to process the two channels in stereo at the same time. To do so, you have to use the `formula_main_stereo` entry point instead of the `formula_main` entry point. The `input` variable will not be a `float`, but a `Stereo struct` containing two floating points variables: `left` and `right`. You also must return the same struct.

Example:

```
formula_main_stereo {
    float mid = (input.left + input.right) / 2;
    float sides = (input.left - input.right) / 2;
    mid = mid * KNOB_1;
    sides = sides * KNOB_2;

    Stereo output;
    output.left = mid + sides;
    output.right = mid - sides;
    return output;
}
```