FabDist

Group ID: 6 - FabFour

Amico Stefano Antonio, 10937333

Anand Abhijeet, 10859656

Pletti Manfredi, 10493741

Portentoso Alice, 10664207

May 5, 2023

1 Introduction

The Oberheim OB-X was the first of Oberheim's OB-series polyphonic analog subtractive synthesizers. First commercially available in June 1979, the OB-X was introduced to compete with the Sequential Circuits Prophet-5, which had been successfully introduced the year before.
The OB-X was the first Oberheim synthesizer based on a single printed circuit board called a "voice card" (still using mostly discrete components) rather than the earlier SEM (Synthesizer Expander Module) used in Oberheim semi-modular systems, which had required multiple modules to achieve polyphony. The OB-X's memory held 32 user-programmable presets. The synthesizer's built-in Z-80 microprocessor also automated the tuning process. This made the OB-X less laborious to program, more functional for live performance, and more portable than its ancestors.

The OB-Xd is based on the Oberheim OB-X. It attempts to recreate its sound and behaviors, but as the original was very limited in some important ways a number of things were added or altered to the original design. The OB-Xd is written in C++ and utilizes the Juce Framework (v. 7.0.3 suggested to compile).

2 Key components and functionality

Initialization: The ObxdAudioProcessor main class initializes the synthesizer's components, such as the voice manager, LFOs, filters, and oscillators.

User Interface: The ObxdAudioProcessorEditor class handles user interaction with the synthesizer's controls, such as knobs and buttons. The Juce framework processes the user input and updates the synthesizer parameters accordingly.

MIDI Input: The MidiMap class handles MIDI events, such as note-on and note-off messages, and interacts with the voice manager to handle these events.

Voice Management: The ObxdVoice class handles voice allocation and updates for each voice. It processes the output of individual voices, including envelopes, oscillators, filters, and LFO modulation.

DSP Components: The Engine folder contains the code for the synthesizer's DSP components, such as oscillators, filters, and envelopes. These components are utilized by the voices to generate and shape the sound output.

Audio Output: The mixed output of all active voices is sent to the audio output provided by the Juce framework in the processBlock() function of PluginProcessor.cpp.

Patch Management: The Juce framework handles patch saving and loading. Users can save and load patches using the host DAW's preset management features.

3 Innovation - Continuous blendable multimode filter

The continuous blendable multimode filter is a type of audio filter that can smoothly transition between different filter types. In this case, the filter types are High Pass (HP), Notch, Band Pass (BP), and Low Pass (LP) filters. The term "multimode" refers to the ability to switch between these different modes. The filter provides two configurations: 12 dB per octave and 24 dB per octave. In the 12 dB mode, it offers HP-Notch(BP)-HP filters, while in the 24 dB mode, it uses a 4-pole LP filter that can transition to a 1-pole HP filter.

The 12 dB mode uses a second-order filter (two poles), and the 24 dB mode uses a fourth-order filter (four poles). The "dB per octave" refers to the rate at which the signal is attenuated once it goes beyond the cutoff frequency of the filter. In a 12 dB per octave filter, the signal level drops by 12 dB for every octave the frequency moves away from the cutoff frequency.

To understand the math behind these filters, we need to look at the transfer function of each filter type:

High Pass filter: $H_{hp}(s) = \frac{s}{s+\omega_c}$, where s is the complex frequency and w_c is the cutoff frequency

Low Pass filter: $H_{lp}(s) = \frac{\omega_c}{s+\omega_c}$

Band Pass filter: $H_{bp}(s) = \frac{s}{s^2 + \frac{s \cdot \omega_c}{Q} + \omega_c^2}$, where Q is the quality factor of the filter.

Notch filter: $H_{notch}(s) = \frac{s^2 + \omega_c^2}{s^2 + \frac{s \cdot \omega_c}{Q} + \omega_c^2}$

The continuous blendable multimode filter essentially takes a weighted average of these filters, allowing for smooth transitions between them.

The original code is handled in the Filter class (Engine/Filter.h).

We have tested this blendable filter in python; some configuration plots:

TO BE TESTED

4 Conclusion

The OB-Xd synthesizer effectively emulates the sound and features of the classic Oberheim OB-X analog synthesizers while incorporating modern improvements. Its well-structured codebase, built on top of the Juce Plugin Framework, ensures compatibility with various DAWs and platforms. The project's open-source nature allows for further development, feature additions, and customization by the community, but precompiled version are commercially distributed for various platform. The continuous blendable multimode filter has been rewriten in python and tested in depth. It could be for sure inspiration for other projects.

5 Bibliography

https://en.wikipedia.org/wiki/Oberheim-OB-X

https://www.discodsp.com/obxd/

https://oberheim.com/

https://www.vintagesynth.com/oberheim/obx.php