Software Modular Synthesizer

Computer Sound Production Laboratory Assignment Intermediate Report

Gregor Kovač
Faculty of Computer And Information Science, University of Ljubljana
Email: gk1854@student.uni-lj.si

I. Introduction

Synthesizers are electronic musical instruments that generate audio signals [3]. They do this by producing a waveform, like a sine wave, and then shape it using different techniques. The first proper analog synthesizers were created in the 1960s [2], but they were replaced by digital synthesizers in the 1980s. As of then, they have been widely spread in the computer sound production world. Here we will focus on *modular synthesis* - a popular technique in the analog domain, where one takes a collection of different smaller synthesizers called *modules* and connects them together with cables to route the audio signal. This allows for a lot of creativity and freedom in creating new interesting sounds. There exists many different modules ([1] presents some of them) which makes this field of sound production complex. On top of that, building such a system can be rather expensive. The goal of this project is to design a simpler all-in-one software alternative to a modular synthesizer. In this report we will describe the work we have done so far and what we plan to add in the future.

II. Goals

The goal of this project is to build a standalone application that simulates modular synthesis. Some software alternatives like this already exist, like N*ative Instruments Reaktor*[6] and *VCV Rack*[7], but they seem complex and intimidating at first, just like usual modular synthesizers. The purpose of our app will be exploration, creativity and learning, therefore we will strive to develop a beginner-friendly learning curve.

III. Implementation

Here we will provide a list of all features we have implemented so far. Our application is being developed in the programming language *Python* with the help of libraries *PYO* [4], a library for digital sound production, and *PyGame* [5], a library for visualizations. From the initial plan we replaced the library *MatPlotLib* with PyGame, since we found out that working with it is easier, especially when it comes to handling user input and interactions.

Let us continue by describing a list of current features.

1) **Master synth.** The master synth is the core of our application. It is a class that handles user input, graphics and modules. It allows us to add new modules (currently implemented with numbers on the keyboard), remove them, patch them together and control their parameters.
2) **Module base.** We have created a base class that all of our modules are derived from. It contains information about the location and size of the module, its inputs, outputs and controls. It also has a function for drawing and for checking if the user clicked on it. All of these features are expanded with each derived module. The master synth contains a collection of these modules and calls the same functions for all of them.
What actually enables us to create sounds are the so-called pins and potentiometers, which are classes used in all modules.
    a) A *pin* is an input or an output of a module. It can either output a signal or receive an input signal. The input signal can either be manipulated and sent to the output or it can control one of the parameters of a module.
    b) A *potentiometer* allows us to manually control a parameter of a module.
3) **Modules.** Here we will describe the four modules we have implemented so far.
    a) *Master out* is the simplest module, containing a single input. It takes the input signal and sends it to the computer's sound card. This module is the final one in each patch we create and cannot be removed.
    b) *Voltage controlled oscillator (VCO)* is the essential sound generator that outputs a signal. Currently it supports only a sine wave. It has a single control knob that changes the frequency of the oscillator. It has an input for its amplitude and frequency parameters.
    c) *Low frequency oscillator (LFO)* is an oscillator that has a much lower frequency than the VCO. It is good for controlling parameters and adding movement. It currently has no inputs, but it has two potentiometers for controlling the shape and frequency. It has 9 different shapes supported by PYO, like saw, square, triangle and pulse.
    d) *Mixer* has 4 inputs and a single output. Each input has a potentiometer that controls its amplitude. This module allows us to mix together multiple different signals into a single sound.
4) **Creating a sound.** In Figure 1 we can see an example of a simple patch we have created with our application. We have added the modules with number keys and connected them together by clicking on a pin and dragging a "cable" to another pin. We have changed

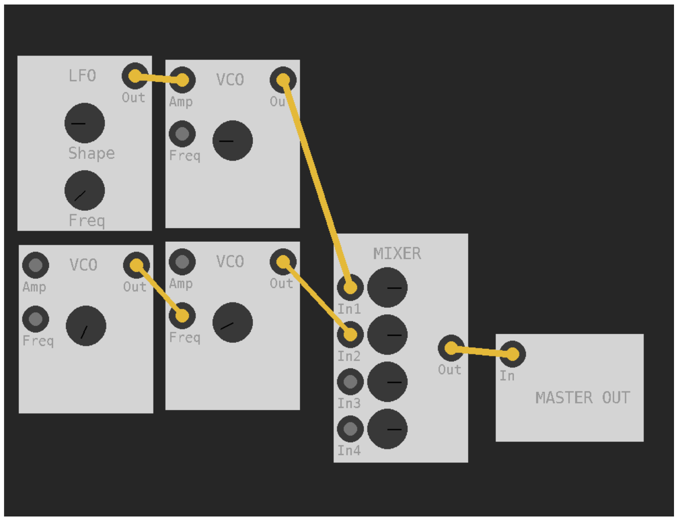

Fig. 1. An example of a sound patch.

some potentiometer values. For the patch itself, we can see that we first have a VCO, with an LFO controlling its amplitude, giving a sort of a ”wavy” sound. We have mixed this sound together with a VCO, for which the frequency is controlled by another VCO which gives a metallic ringing sound. This technique is referred to as FM (frequency modulation) synthesis.

IV. Future work

We have spent a lot of time developing the basis for our application (the master synth and the module base), but after that the development seemed to start moving faster. In the future we plan to implement a few other modules, for example:

1) *Envelope generator (EG)* creates a one-shot signal shaped by attack, decay, sustain and release (ADSR) parameters.
2) *Voltage controlled filter (VCF)* applies standard filters (low-pass, high-pass, band-pass,...) to and input signal.
3) *Sequencer* outputs a rhythmic sequence of frequencies (a melody).
4) *Visualizer* features visualizations of an input signal, like a frequency graph or an oscilloscope.

We will also add more controls and options for our existing modules, like inputs for the LFO and other wavetables for the VCO.

We plan to improve the user experience and graphics by improving controls, adding a dedicated menu for adding modules and improving the look of the modules. We also want to add optional tooltips that guide a new user towards creating their first sound.

Although our application seems to work properly at this point, we still plan to do more thorough testing. Currently our evaluation is purely reliant on our own listening abilities, checking if a sound we create makes sense. A good additional way of testing will be through the visualizer module.

Even though our implementation currently contains only four modules, we have found out that we are able to create a wide variety of different sounds just with these basic assets. We are happy with the progress so far and we are looking forward to seeing how this project will turn out when finished.

References

[1] James, A., *The Secret World Of Modular Synthesizers*, Sound On Sound, 2013. https://www.soundonsound.com/reviews/secret-world-modular-synthesizers
[2] Analog Synthesizer, Wikipedia, 2024. https://en.wikipedia.org/wiki/Analog_synthesizer
[3] Synthesizer, Wikipedia, 2024. https://en.wikipedia.org/wiki/Synthesizer
[4] PYO, Ajax Sound Studio, 2021. http://ajaxsoundstudio.com/software/pyo/
[5] PyGame, 2024. https://www.pygame.org/
[6] Native Instruments, *Reaktor 6*, 2015. https://www.native-instruments.com/en/products/komplete/synths/reaktor-6/
[7] VCV, *Rack 2*. 2021. https://vcvrack.com/