# Assignment #2: Digital Polyphonic Synthesizer

Pedro Esteves

*Faculty of Engineering of the University of Porto*

up201705160@fe.up.pt

## I. Introduction

A Synthesizer is an electronic musical instrument that generates audio signals. There are two main types of synthesizers, analog and digital, and they both can be monophonic or polyphonic, depending on the number of notes they are capable of sounding.

This report describes a Digital Polyphonic Synthesizer, created with *Pure Data*, for the *Sound Synthesis for Digital Media* (SSMD) course (Master in Multimedia, FEUP).

## II. Research

With only two objects, *poly* and *clone*, we can create a digital polyphonic synthesizer with Pure Data. Everything else can be considered as extra features that almost every synthesizer has nowadays.

With this thought in mind, the first step to create this synthesizer was to search for Polyphonic Synthesizers and their features. Most synthesizers have knobs to control the volume and the keys' tempo, a midi keyboard, with a switch to control its current octave, filters, amplitude envelope generator, and sequencers. Other features include the *Voltage-Controlled Oscillators* (VCOs), the *Low-Frequency Oscillators* (LFOs), and voice modes, which were not considered relevant to this assignment due to their complexity.

After finding several polyphonic synthesizers, I chose the *KORG minilogue - Polyphonic Analogue Synthesizer* as the main inspiration for my synthesizer's interface due to its simplicity in colors and positions (Figure 1).

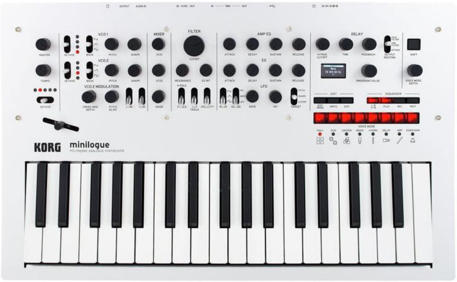

Figure 1: Korg minilogue - Digital Polyphonic Synthesizer

## III. Synthesizer

As seen in the previous section, the synthesizer created was inspired by *Korg minilogue*. The following subsections describe in more detail the features implemented.

### A. On / Off Switch

An analogue synthesizer always has a power button, allowing the user to turn on / off the device. For this synthesizer, I developed something similar that enables / disables Pure Data's DSP (Digital Signal Processing), and changes the button's color from red to green and vice versa, allowing the user to quickly check the state of the synth.

### B. Master Knob

The master's knob, or slider, allows the user to control the output volume of the synthesizer. Its values vary between 0 and 1, and, using amplitude modulation (AM), they are multiplied by the output signal.

### C. Tempo Knob

Tempo allows the user to define the *tempo*, or playing speed, of the sequencer. This feature is used in a lot of synthesizers and is useful to recreate pieces of music like Pink Floyd's *On The Run*, from the first assignment.

### D. *Wave Switch*

Most synthesizers have VCOs and LFOs with a parameter to control their waveforms. This switch has a similar effect as that, but it changes the waveform of the output sound. Each position of the switch has an image of the wave it represents, implemented with the *image* object of the *Moonlib* package. The synthesizer creates polyphony using the *poly* and *clone* objects, cloning the *poly_voice* patch. Inside this patch there's a sub-patch to select the output waveform of the *poly_voice* according to the one selected by the user.

### E. *AMP EG - Amplitude Envelope Generator*

This section represents a new sub-patch that, with the help of AM, controls the sound's envelope. There are 4 sliders, one for each envelope's parameter: attack, decay, sustain, and release.

### F. *Filter*

Subtractive Synthesis allows us to recreate several sound effects. In the SSMD classes we learned the 4 basic types of filters (low-pass, high-pass, band-pass, and notch), but after my research I found that most synthesizers use a filter called *Low-Pass Resonant Filter*. This filter has at least the *cutoff frequency* and the *resonance* (also called Q or Peak), the ones used in this assignment. It was implemented with the use of the object *bob~*.

### G. *Keyboard*

The MIDI keyboard was developed using the *keyboard* object, and it has some limitations. For starters, the keys sometimes won't appear with the red color, meaning that it was pressed, while using the PC's keyboard. If the user keeps pressing a key, the last note is repeated, because *Pure Data* doesn't have good support for the input key down event. The comments above each key, indicating input - keyboard mapping, disappear when changing the keyboard's octave.

Besides the previous limitations, the keyboard is very responsive and produces a good result, by allowing the user to play at most 5 keys at a time (polyphony).

### H. *Octave Switch*

The octave switch is also present in several synthesizers. In this case you can change between 5 octaves, as in the *Korg minilogue*. The middle octave corresponds to the one starting in C3.

### I. *Sequencer*

In the last assignment, I learned how to create sequences of notes and reproduce them. Since most synthesizers contain a sequencer, I decided to create one that allowed me to recreate *On The Run*. To record notes, the user needs to click on one or more channels, and then click on the record button. Then the user can press the play button to listen to the sequence.

## IV. Using the Synthesizer

To prove the potential of the synthesizer, I recorded a video using it, available in this drive.

The synthesizer's interface is simple to use, and I think someone without any experience in this field can quickly learn it by experimenting with it.

To assure that nothing breaks during the experimentations, the user must install, besides Pd, the following libraries: *cyclone*, *else*, *audiolab*, and *moonlib*. I have tried other libraries, such as *Gem* and *list-abs*, to solve some problems regarding the image displaying and the keyboard input, but they have been discarded from the final result.

## V. Conclusions and Future Work

Although the synthesizer isn't perfect and as complex as the one's seen in the industry, with the help of this assignment, I have learned a lot more about digital, and analogue, polyphonic, and monophonic, synthesizers.

In future work, I could improve the keyboard, and create a synthesizer more similar to *Korg minilogue*, by creating VCOs and LFOs. I also think that playing with a real synthesizer can also help improve this work.

## VI. References

1. KORG. (2016). Korg minilogue - Polyphonic Analogue Synthesizer - Owner's Manual. https://cdn.korg.com/us/support/download/files/23be1d77cac2c42e6f6c586cf0f75ffb.pdf