1

Subtractive Synthesis: The Beginning

To simplify the search, I have summarized in Table 1.1, for each of the exercises in this chapter, the hardware and software chosen and the sections where you can find them.

Exercise	Hardware/software	Section
No. 1. 1 VCO, 1 VCA	Behringer Neutron	1.1.1
	Behringer 2600 (ARP2600)	1.1.2
	Max/MSP	1.1.3
	Pure Data	1.1.4
	VCV Rack	1.1.4
No. 2. 1 VCO, 1 VCA, 1 EG	Behringer Neutron	1.2.1
	Behringer 2600 (ARP2600)	1.2.2
	Max/MSP	1.2.3
	Pure Data	1.2.4
	VCV Rack	1.2.5
No. 3. 1 VCO, 1 VCA, 1 EG, 1 VCF	Behringer Neutron	1.3.1
	Max/MSP	1.3.2
	Pure Data	1.3.3
	VCV Rack	1.3.4

Table 1.1. *Summary of the different exercises*

Before starting, it seems appropriate to clarify that the objective of the exercises, in all chapters, is to learn about subtractive sound synthesis in order to understand it

better and master it, but also to allow your ears to discover the sensations and the sounds reproduced by each of the signals that you are going to create. What is the point of doing sound synthesis if not to discover new sounds or understand what it means to reproduce a sound, a timbre, or an already existing tone such as that of a musical instrument or a familiar noise? We need to remember to listen rather than hear, pay attention and discover the richness of the sound universe surrounding us by synthesizing our main constituents.

NOTE.– As the exercises are presented, specific comments will not be repeated to avoid overloading the explanations. In case of doubt or misunderstanding, I advise the reader to go back and read the previous exercises to find the explanatory elements that may be missing.

1.1. Exercise 1 – generate sound with a single oscillator

The basics of sound synthesis are to generate sound from an oscillator. A classic configuration contains an oscillator (VCO) and an amplifier (VCA), which can be preceded by an input controller, often a keyboard.

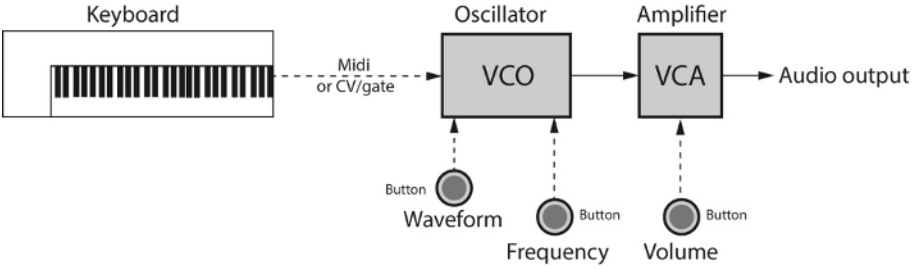

Figure 1.1. *Subtractive sound synthesis with an oscillator. The command signals are represented with dotted lines*

1.1.1. *Behringer Neutron*

The Behringer Neutron semi-modular synthesizer (see Volume 1, section 4.1.3) has two oscillators whose signals can be mixed. For this exercise, we will use Oscillator 1 (Oscillator 2 would have worked the same way, as these two oscillators are identical) and a patch cable, on the matrix, to route it to the VCA. The oscillator has five waveforms, available over three octaves.

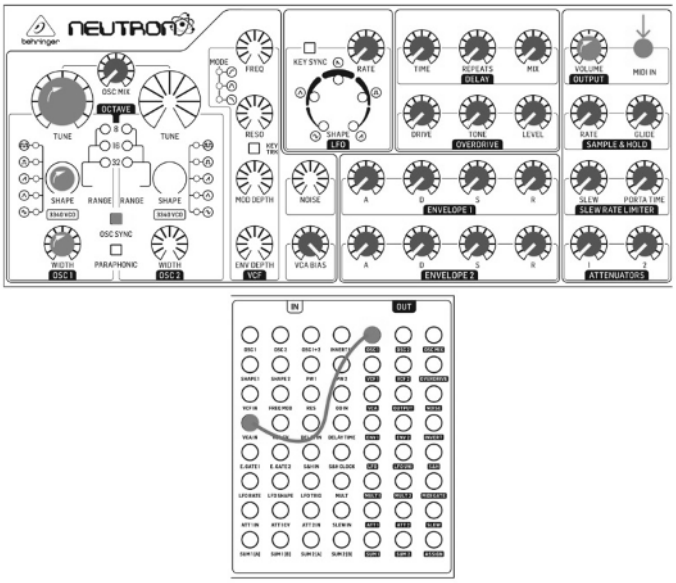

Figure 1.2. *Exercise 1 for the Behringer Neutron. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– connect a MIDI keyboard to the MIDI IN input (on the front panel) and switch it on. Check that the keyboard and the Neutron are placed on the same MIDI channel number (1–16). On the Neutron, the adjustment is made with the DIP switches located on the back;

– connect the audio output (OUTPUT) on the back to an amplification system;

– switch on the Neutron;

– connect the output (OUT) of oscillator 1 (OSC1) with a patch cable to the input (IN) of the VCA (VCA IN) on the input–output matrix to the right; this will route the signal produced by the oscillator directly to the amplifier input;

– the controls of oscillator 1 (OSC1) are all accessible. We find the agreement (TUNE), the waveform (SHAPE: mode-tonal, square-pulse, sawtooth, triangle, sinusoid) and the width (WIDTH), which modifies the width of the pulse or the tonal mode when chosen as the waveform and the octave (RANGE – 8', 16' or 32');

– the oscillator mix control knob (OSC MIX) must be turned all the way to the left;

– the VC BIAS button (influence of the VCA) must be turned all the way to the right to open the VCA to the maximum;

– the VOLUME OUTPUT knob can be adjusted to adjust the overall audio output volume to your liking;

– the GATE knob of the LFO must be turned all the way to the left; otherwise, it modifies the signal of the mode-tonal and impulse waveforms;

– all DELAY, OVERDRIVE, ENVELOPE 1 and 2, SAMPLE & HOLD, SLEW RATE LIMITER and ATTENUATORS knobs should be turned all the way to the left.

NOTE.– Instead of connecting a MIDI keyboard, it is possible to use a virtual MIDI keyboard or the one in some DAWs (digital audio workstations). In this case, the connection is made via the USB-MIDI ports of the microcomputer and the Neutron. For the latter, it is located at the back of the device. Again, the MIDI channel numbers must be identical.

1.1.2. *Behringer 2600 (ARP 2600)*

The Behringer 2600 is one of the ARP 2600 clones (see Volume 1, section 4.1.1) and has similar, if not superior, functionality to today's synthesizers. It is semi-modular but has no separate matrix, and the connection points are distributed on each front panel module.

For this exercise, we will need a patch cable connected to the VCA, the other end of which can be moved to obtain the various signals generated by the oscillator since the equipment does not have a waveform selector.

Oscillator 2 was chosen because it has four waveforms; oscillator 3 would have provided the same results, with these two oscillators being equivalent.

The audio output of the 2600 is stereophonic by default, so you can choose to use both channels or just one. The slide control, PAN, manages the left–right balance.

NOTE.– The names of the commands are not always specified on the serigraphy of the front panel of the ARP 600, Behringer 2600, Korg 2600 or other clones. You can refer to these in the figures for each exercise.

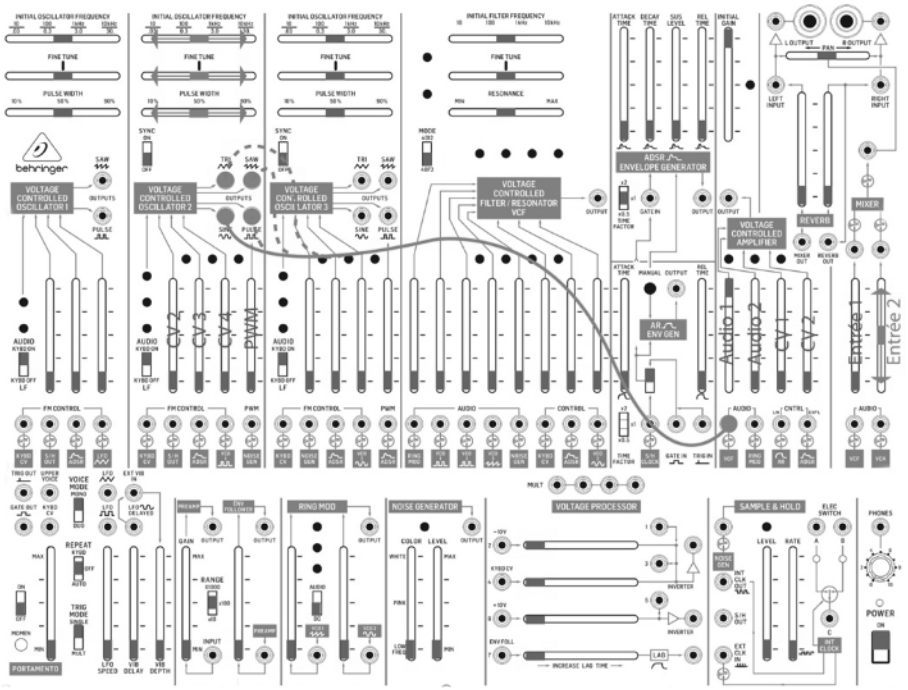

Figure 1.3. *Exercise 1 for Behringer 2600 (ARP 2600). For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– connect a MIDI keyboard to the MIDI IN input on the back and switch it on. Check that the keyboard and the 2600 are set to the same MIDI channel number (1–16). On the Behringer 2600, the adjustment is made using the DIP switches on the back. If you are on an ARP 2600, a KORG 2600, or another clone, use the appropriate keyboard;

– connect the one or two audio outputs (L OUTPUT – R OUTPUT) located at the front to an amplification system;

– switch on the 2600;

– connect, using a patch cable, one of the four outputs, TRI, SAW, SINE, or PULSE to input 1 of the VCA;

– the INITIAL OSCILLATOR FREQUENCY, FINE TUNE and PULSE WIDTH controls are active and modify the audio signal. The level 2 control of the MIXER module input controls the output volume. The PULSE WIDTH control, which defines the width of the waveform square, is active if you have chosen the PULSE output of the oscillator;

– the LFO SPEED, VIB DELAY, VIB DEPTH and REVERB controls must be set to zero (lower position of the slider);

– all oscillator 2 input level controls must be set to zero;

– the AUDIO switch of oscillator 2 must be placed on KEYB ON so that the keyboard is active;

– the SYNC switch of oscillator 2 must be set to OFF;

– the INITIAL GAIN control must be placed at its maximum (uppermost position of the slider) to obtain the optimal gain of the VCA;

– the audio signal level control 1 entering the VCA must be placed at its maximum (uppermost position of the slider) to obtain a maximum incoming level;

– the PAN control can be placed in the center position to obtain an audio signal of equal intensity on the right and left outputs.

1.1.3. *Max/MSP*

The Max/MSP exercises in this book have been tested and designed with version 8.5.4, they may also work on different versions but noticeable differences may occur in some cases[1].

You will find in this exercise some of the main Max/MSP functions.

NOTE.– In all Max/MSP exercises, we will not use Live objects[2] (Ableton) to allow all users to carry out the work.

1 Available at: www.cycling74.com/.

2 Live is a software music sequencer developed by Ableton that has been available since 1999. Since 2009, Ableton and Cycling ’74, the distributor of Max/MSP, have been working together and have developed “Max for Live”, a version of Max, integrated with Live. In 2017, Ableton acquired Cycling ’74.

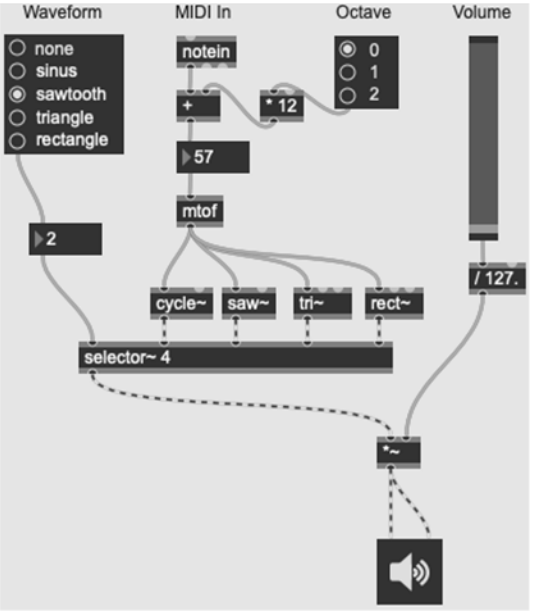

Figure 1.4. *Exercise 1 with Max/MSP*

Instructions and comments are as follows:

– `Waveform`: `radiogroup` object outputting 0, 1, 2, 3, or 4 based on user choice (0 for no signal, 1 for sine signal, 2 for sawtooth signal, 3 for triangle signal, 4 for square wave signal);

– `notein`: MIDI IN acquisition of MIDI notes and information from the keyboard;

– `+`; `* 12`, `Octave`: allows you to increase or decrease the notes by +/– 2 octaves by multiplying the output of the vertical object `radiogroup` by 12 and adding it to the note number[3] received, visualized by the object `number` connected to the output of the object `+`;

– `mtof`: converts the MIDI note number to a frequency in hertz;

– `cycle~`: oscillator that outputs a sine waveform;

– `saw~`: oscillator that outputs a sawtooth waveform;

3 In Appendix 4, you will find the different correspondences of the MIDI note numbers (piano note, frequency, name, etc.).

– `tri~`: an oscillator that outputs a triangular waveform;

– `rect~`: an oscillator that outputs a rectangular waveform;

– `Selector~ 4`: it routes the waveform chosen by the user to the volume control and then the output;

– `*~`: audio operator combining two signals, here a frequency signal and a floating number between 0 and 1 for volume management (`0` by default);

– `Volume`: vertical slider to adjust the volume. By default, it generates values between `0` and 127. Its size is 128, and its minimum output value is 0;

– `/127.`: divides the value of the slider by 127 so that the volume varies between 0 and 1;

– two whole `number` objects and one floating number `flonum` object are present to display the choice of the waveform, the note number and the volume value;

– `Ezdac~` (icon: speaker): it generates an audio signal as output by default on two channels (right and left);

– the texts (waveform, MIDI IN Octave, etc.) are present as comments (`Comment`).

NOTE.– In the event of a malfunction, remember to check the MIDI and audio configuration of Max/MSP in the "Options" menu, "MIDI Setup" sub-menu and "Audio Status". Also, do not forget to click on the "Audio On/Off" icon to the bottom right of the patcher window to activate the sound, and check that the sound is activated on your computer if this is the output you have chosen by default.

1.1.4. *Pure Data*

This exercise covers the main functions in creating a Pure Data patch (see Volume 1, section 4.2.3): object, selector, slider, number, message, etc.

All of these functions will be used throughout the exercises. In Appendix 2, you will find some patches intended to improve or modify the exercises: oscilloscope, keyboard, etc.

NOTE.– You may have seen previously, in section 1.1.3, the term "patcher" for a new project in Max/MSP, whereas here we call it "patch". The terminology is different, but the meaning is the same. It may be an evolution of the language of the developers over time or a problem of rights after the departure of Miller Puckette from the Max/MSP project in 1996 (see Volume 1, sections 4.2.2 and 4.2.3).

The Pure Data exercises in this book have been tested and designed with version 0.53.2; different versions may also be suitable but with some notable differences[4].

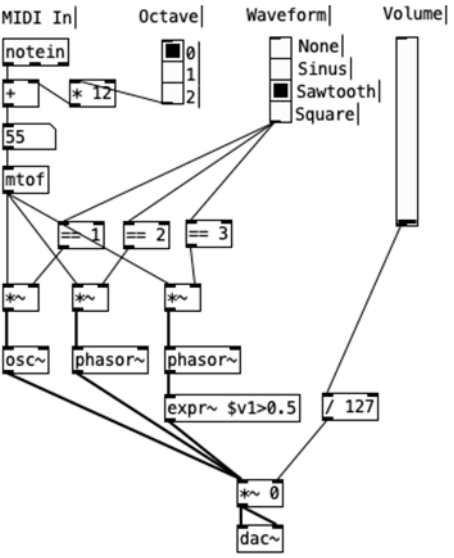

Figure 1.5. *Exercise 1 with Pure Data*

Instructions and comments are as follows:

– `notein`: acquisition of note numbers and MIDI information from the keyboard;

– `+`; `* 12`, `Octave`: it allows you to increase or decrease the notes by +/– 2 octaves by multiplying the output of the vertical object `vradio` by 12 and adding it to the received note number, visualized by the object `number` placed at the output of the object +;

– `mtof`: it converts the MIDI note to a frequency in hertz;

– `*~`: audio operator combining two signals, here the frequency and a binary indicator (0 or 1) validating or not the passage of the signal;

– `osc~`: an oscillator that outputs a sine waveform;

– `phasor~`: an oscillator that outputs a sawtooth waveform;

4 Available at: www.puredata.info/.

– `expr~ $v1>0.5`: a square wave oscillator has a shape that alternates between 0 and 1. Since there is no square wave object in Pure Data, the latter is created by transforming the output of the object `phasor~`. If it is greater than 0.5, the object `expr~` outputs 1; otherwise, it outputs 0. This creates the high (1) and low (0) states of the square wave;

– `*~ 0`: an audio operator combining two signals, here the frequency and an integer for volume management, default 0;

– `dac~`: it generates a default audio output signal on two channels (right and left);

– `== 1`, `== 2`, `== 3`: binary operators testing for equality (0 if not equal, 1 if equal);

– `Waveform`: selector outputting 0, 1, 2 or 3 according to the choice of the user (0 for no signal, 1 for a sinusoidal signal, 2 for a sawtooth signal, 3 for a square signal);

– `Volume`: vertical slider to control the volume. By default, its range of values is between 0 and 127 on a linear scale;

– `/127.`: it divides the slider value by 127 to control the volume between 0 and 1;

– Text (MIDI IN Octave, etc.) is present as comments (`COMMENT`).

NOTE.– In the event of a malfunction, remember to check the MIDI in and audio in configuration in the "Pd" menu, "Preferences" sub-menu, "Audio Settings…", and " MIDI Settings… ". Also, do not forget to check the DSP box or go to the "Media" menu to activate the sound by checking DSP ON, and check that the sound is activated on your computer if this is the output you have chosen by default.

1.1.5. *VCV Rack*

This exercise is designed with the library (fundamental blocks library) provided by default with the VCV Rack. The exercises in this book are for VCV Rack version 2.0, but they may work perfectly on an earlier version, with slight differences.

You will need five modules as follows:

– MIDI-CV: a MIDI input module allowing the connection of peripherals, audio interfaces, keyboard and the selection of the MIDI channel;

– OCT: an octave change module;

– VCO: an oscillator generating four waveforms (sine, triangular, sawtooth and square);

– VCA: a signal amplification module;

– AUDIO: an output module redirecting the signal to the microcomputer or another device such as an audio interface.

Figure 1.6. *Exercise 1 with VCV Rack. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– place the different modules within the rack;

– connect the V/OCT output of the MIDI-CV module to the V/OCT input of the OCT module;

– connect the OUT output of the OCT module to the V/OCT input of the VCO module;

– connect one of the four outputs, SIN, TRI, SAW, or SQR, of the VCO module to the IN input of the VCA module depending on the waveform you want;

– connect the OUT output of the VCA module to the L/MON input of the AUDIO module. The L/MON input directs the signal to both channels (right and left);

– do not forget to adjust the level of the VCA module (bar graph);

– set the output level, LEVEL of the AUDIO module;

– you can act on the FREQ (frequency) controls if you have chosen SIN (sinusoid), TRI (triangle) or SAW (sawtooth) on the VCO module;

– the PULSE WIDTH control of the VCO module is only active for the SQR output (square).

NOTE.– In the event of a malfunction, remember to check the MIDI configuration in the MIDI-CV module and the audio output configuration in the AUDIO module. Also, check that the sound is activated on your computer if this is your chosen output.

1.2. Exercise 2 – associate an envelope

Exercise 1 provides continuous sounds without an envelope (see Volume 1, section 3.3). This new exercise includes an envelope generator (EG), which will be applied to the VCA, as shown in Figure 1.7.

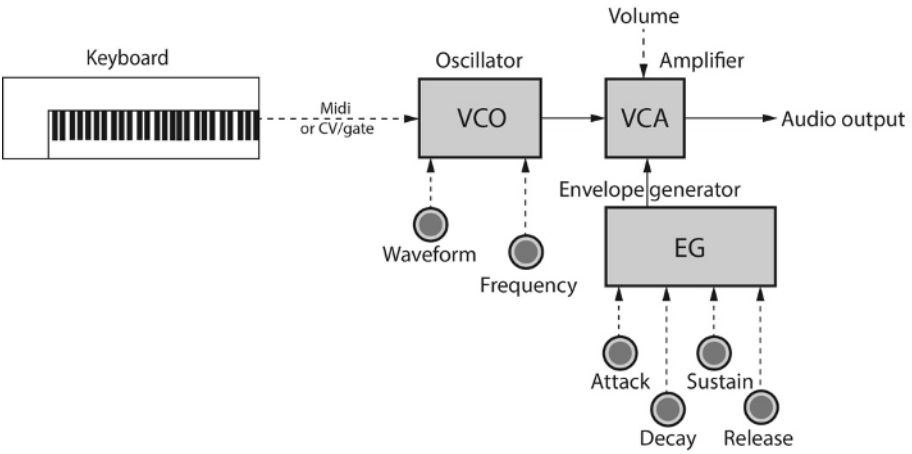

Figure 1.7. *Subtractive sound synthesis with an oscillator, an envelope generator and an amplifier (VCO – EG – VCA). The command signals are shown with dotted lines*

You will find the classic audio signal shaping parameters: attack, decay, sustain, and release (ADSR). The input controller remains a keyboard.

1.2.1. *Behringer Neutron*

We keep the patch cable between OSC1 OUT and VCA IN for this second exercise on the Neutron matrix. A pre-routing of the EG on the VCA already exists by default.

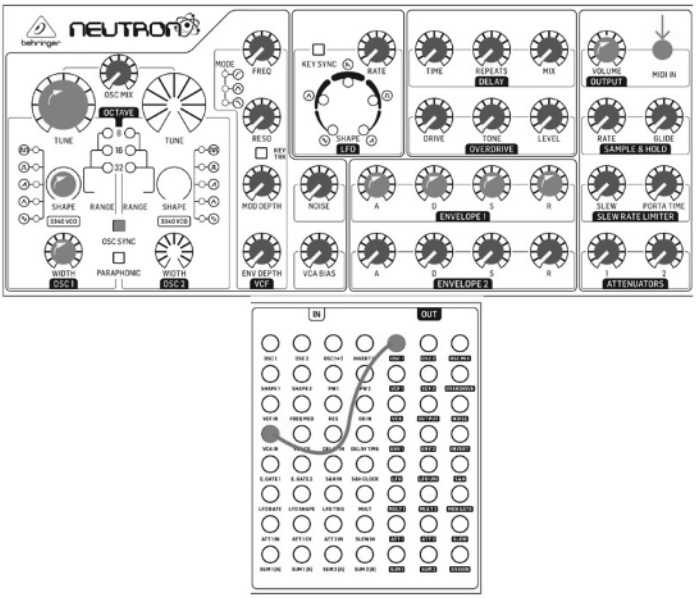

Figure 1.8. *Exercise 2 for the Behringer Neutron. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– keep the configuration in exercise 1;

– as before, the controls for oscillator 1 (OSC1) are all accessible, as well as the general volume knob;

– the VC BIAS knob (VCA influence) must be turned all the way to the left to close the VCA to the maximum; otherwise, the signal coming from the oscillator will be diverted and the EG will have no effect;

– the VOLUME OUTPUT knob can be adjusted to adjust the overall audio output volume to your convenience;

– the A, D, S, R buttons allow you to adjust the envelope parameters. Note that it is envelope 1 that acts on the VCA (see Volume 1, section 4.1.3, Figure 4.24).

1.2.2. *Behringer 2600 (ARP 2600)*

To set up the EG acting on the VCA, we will modify the configuration adopted in exercise 1.

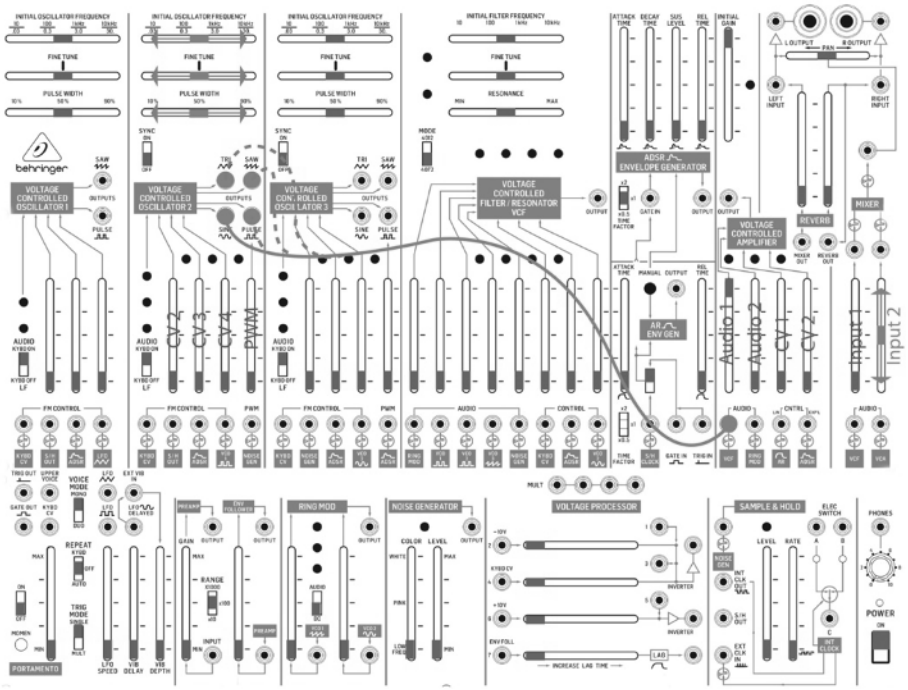

Figure 1.9. *Exercise 2 for Behringer 2600 (ARP 2600). For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– keep the wiring from the configuration in exercise 1;

– set the INITIAL GAIN control to the down position to completely reduce the overall gain of the VCA;

– the audio signal input control CV2 must be placed at its maximum (highest position of the slider);

– the ATTACK TIME, DECAY TIME, SUS LEVEL, REAL TIME (ADSR) controls of the EG can be manipulated to define the envelope curve of the input signal;

– the audio input 1 control of the MIXER module controls the output volume.

1.2.3. *Max/MSP*

Exercise 2 will enrich the patcher already created in exercise 1 with new functionalities.

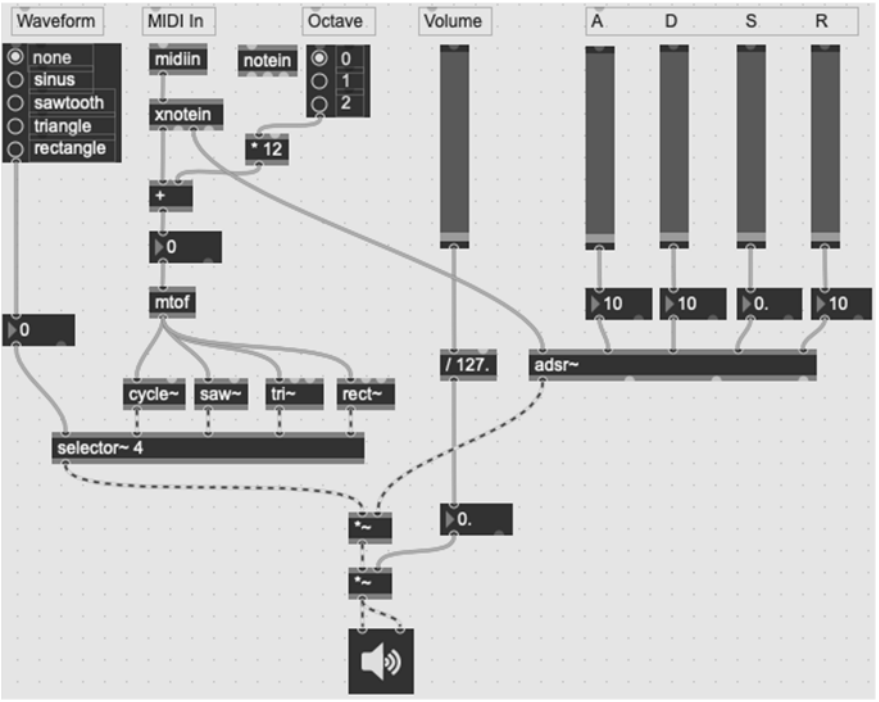

Figure 1.10. *Exercise 2 with Max/MSP*

Instructions and comments are as follows:

– `midiin`: replace the object `notein` to capture the notes coming from the keyboard;

– `xnotein`: connected to the `midiin` object. The `xnotein` provides an additional logical indicator (0 or 1) with respect to `notein`. It detects if a key has been pressed on the keyboard. This indicator will be connected to the input of `ADSR~` to signal to it the state of the keyboard keys, pressed or released (MIDI note-on, note-off);

– `A`: the `slider` that controls the attack (A). Its size is 10,000, and its output minimum (min) value is 10. It represents a time between 10 ms and 10 s;

– D: the `slider` that controls the decay (D). Its size is 1,000, and its minimum output value (min) is 10. It represents a time between 10 ms and 1 s;

– S: the `slider` that controls the sustain (S). Its size is 1, and its minimum output decimal value (min) is 0. It represents a percentage between 0 and 10%;

– R: the `slider` that controls the release (R). Its size is 10,000, and its minimum output value (min) is 10. It represents a time between 10 ms and 10 s;

– add three whole `numbers` connected to `slider` A, D and R to display the values according to the position of the slider pointer;

– add a `flonum` (floating number) connected to the output of slider S to display the values according to the position of the slider pointer;

– add an `ADSR~` object to recall commands A, D, S, R and generate the envelope;

– `*~`: audio operator combining (multiplying) two signals, here a frequency signal and the envelope to be applied.

1.2.4. *Pure Data*

Creating an EG in Pure Data has certain subtleties; this language does not have an ADSR object directly integrated into its library, unlike Max/MSP.

The same goes for detecting the note-off/note-on signal, which is not embedded in an object.

Instructions and comments are as follows:

– A: `slider` that controls the attack (A). Its linear range is between 10 and 10,000. It represents a time in milliseconds;

– D: `slider` that controls the decay time (D). Its linear range is between 10 and 1,000. It represents a time in milliseconds;

– S: `slider` that controls the sustain level (S). Its linear range is between 0 and 1. It represents percentages (0–100%);

– R: `slider` that controls the release time (R). Its linear range is between 10 and 10,000. It represents a time in milliseconds;

– the four `number` objects present behind the `slider` objects display the values of each of them;

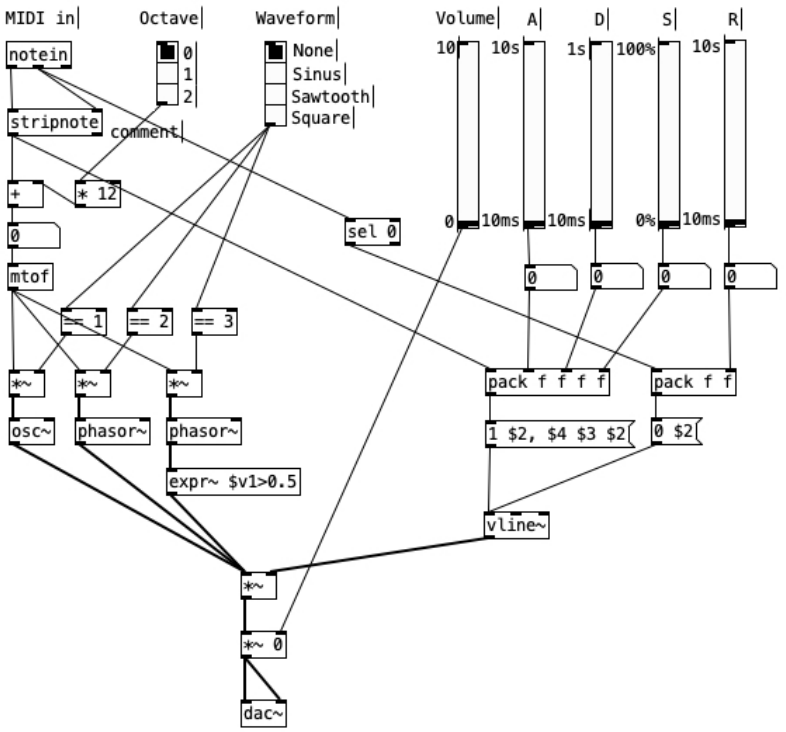

Figure 1.11. *Exercise 2 with Pure Data*

– `Stripnote`: detects the presence of a pressed key (note-on) to trigger the `pack` object, which will define the attack, decay and sustain (A, D and S);

– `Sel`: makes it possible to test if its input, the velocity, is equal to the constant 0. If so, the left output is true (1); otherwise, the right output gives the velocity. When velocity disappears, release can take place;

– `Pack` (on the left): concatenates all the entries, that is, they are placed one after the other. The input data are the note number, attack, decay and sustain (A, D and S). As the output, `pack` will provide the value of 1, that is, an upward ramp, necessary for the attack, followed by the three arguments;

– `Pack` (right): if its right input is true (1), that is, if the key has been released (zero velocity), then the output is equal to 0 followed by the release (R). The value 0 indicates a downward ramp, which is required for a release;

– behind the two `pack` objects place a `message` object, which retrieves the sequence of values to be transmitted to the `vline~` object;

– `vline~`: this object retrieves the data on its input to transform it into a variable linear ramp whose unit is milliseconds;

– `*~`: combines the audio signal and its envelope.

1.2.5. *VCV Rack*

Adding an EG controlling the VCA to the previous exercise is very simple. With VCV Rack, just install an ADSR module.

Figure 1.12. *Exercise 2 with VCV Rack. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– place an ADSR module between the VCO and the VCA;

– connect the GATE output of the MIDI-CV module to the GATE input of the ADSR module;

– connect the ENV output of the ADSR module to the control input of the VCA;

– by acting on the four commands of the ADSR module, you can vary the contours of the output signal.

1.3. Exercise 3 – add a filter

To continue, we will enrich our work with a filter, which will process the signals from the VCO. We will choose a low-pass filter with cutoff frequency and resonance management, as shown in Figure 1.13.

Other filters are possible (high-pass, band-pass, with different types and more complex settings). They are often present on synthesizers, with each manufacturer trying to offer more sound processing possibilities.

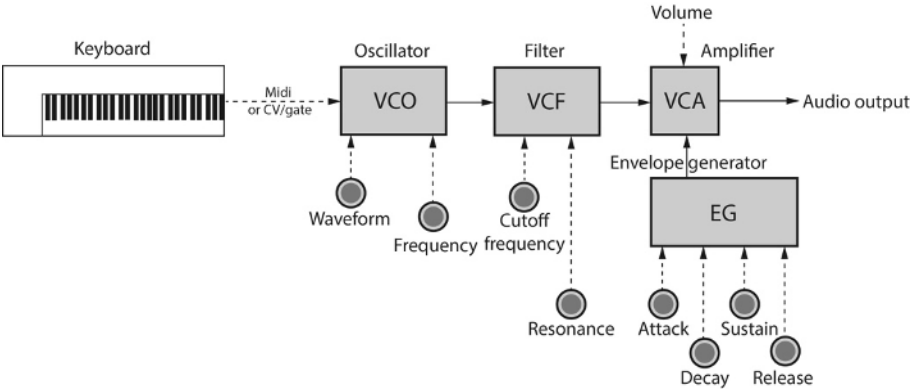

Figure 1.13. *Subtractive sound synthesis with an oscillator, filter, envelope generator and amplifier (VCO – VCF – EG – VCA). The command signals are shown with dotted lines*

1.3.1. *Behringer Neutron*

Compared to the previous exercise, placing one or more patch cables on the Neutron matrix is unnecessary. By default, a pre-routing exists between oscillator 1 (OSC 1) and the filter (VCF).

Instructions and comments are as follows:

– remove the patch cable from exercise 2;

– turn the LEVEL knob of the OVERDRIVE entirely to the right to authorize the output of the audio signal to the VCA;

– the three buttons of the VCF allow you to adjust the cutoff frequency of the filter (FREQ), its resonance (RESO) and the modulation depth (MOD DEPTH);

– the type of filter can also be changed by pressing the MODE button (from top to bottom: high-pass filter, band-pass filter, low-pass filter);

– the setting of envelope 1 (ADSR) relating to the VCA always remains active, like the settings of oscillator 1 and the control of the general volume;

– do not forget to place the VCA BIAS button at its minimum to preserve the action of the EG 1.

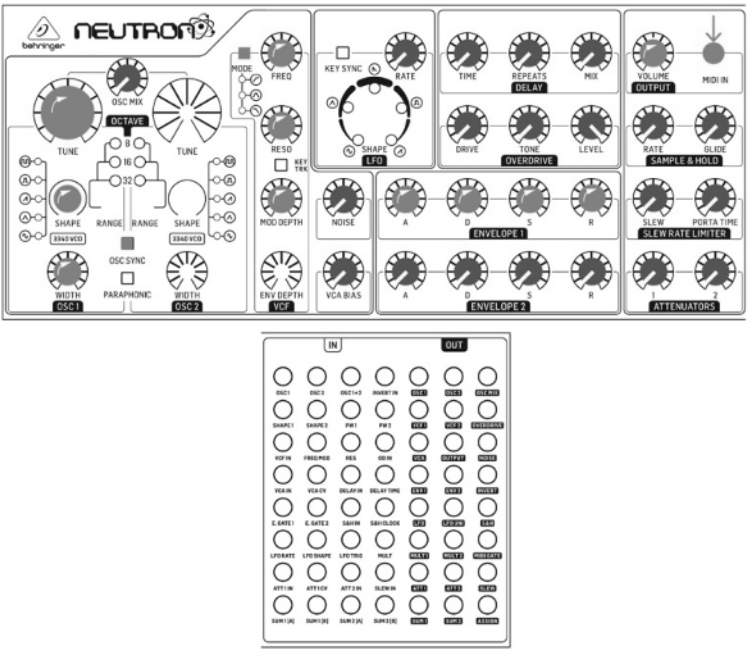

Figure 1.14. *Exercise 3 for Behringer Neutron. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

1.3.2. *Max/MSP*

For this exercise, we are going to add a low-pass resonant filter, whose cutoff frequency and resonance we can adjust by integrating an object into our patcher `lores~`.

Instructions and comments are as follows:

– repeat the patcher from exercise 2;

– `Cutoff frequency`: `slider` controlling the cutoff frequency of the low-pass filter. It has a size of 50 and a minimum value of 0;

– `Resonance`: `slider` controlling the resonance of the low-pass filter. It has a size of 1 and a minimum decimal value of 0;

– add a whole `number` to display the value of the `slider Cutoff frequency` object;

– add a floating number object `flonum` to display the value of the `slider Resonance` object;

– `lowres~`: resonant low-pass filter, the right input picks up the frequency signal, the center input picks up the cutoff frequency and the left input picks up the resonance.

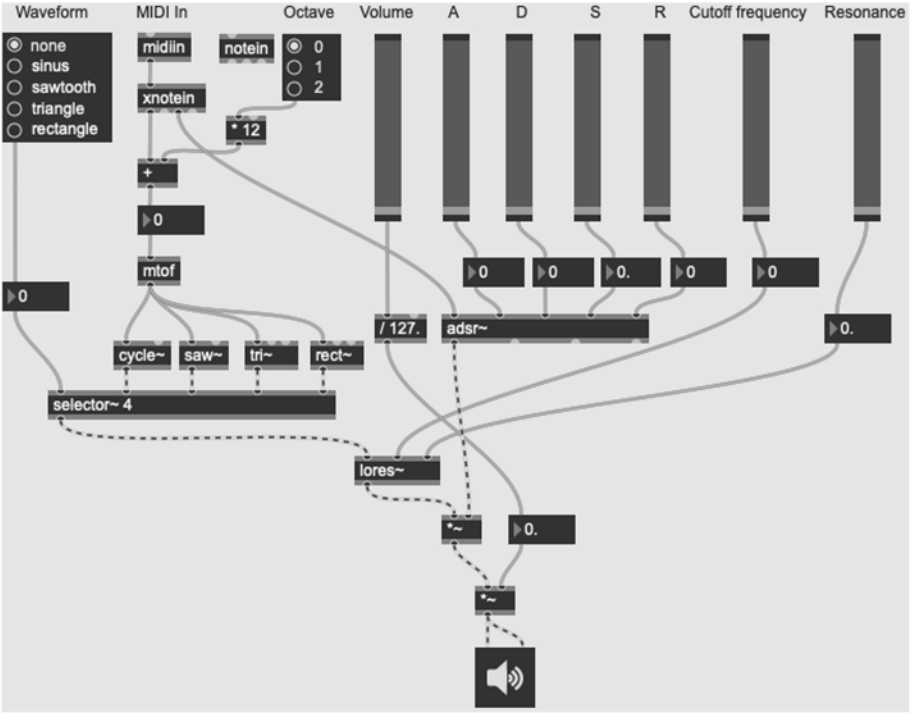

Figure 1.15. *Exercise 3 with Max/MSP*

1.3.3. *Pure Data*

There are several objects that can act as a filter. In this exercise, we will use the `bob~` filter (this name pays homage to Robert Moog, one of the inventors of the first analog synthesizer).

Its characteristics make it a filter well suited to subtractive synthesis, it is a Runge-Kutta type[5] 4 (RK4) filter and it perfectly simulates the resonant filter used on Moog synthesizers.

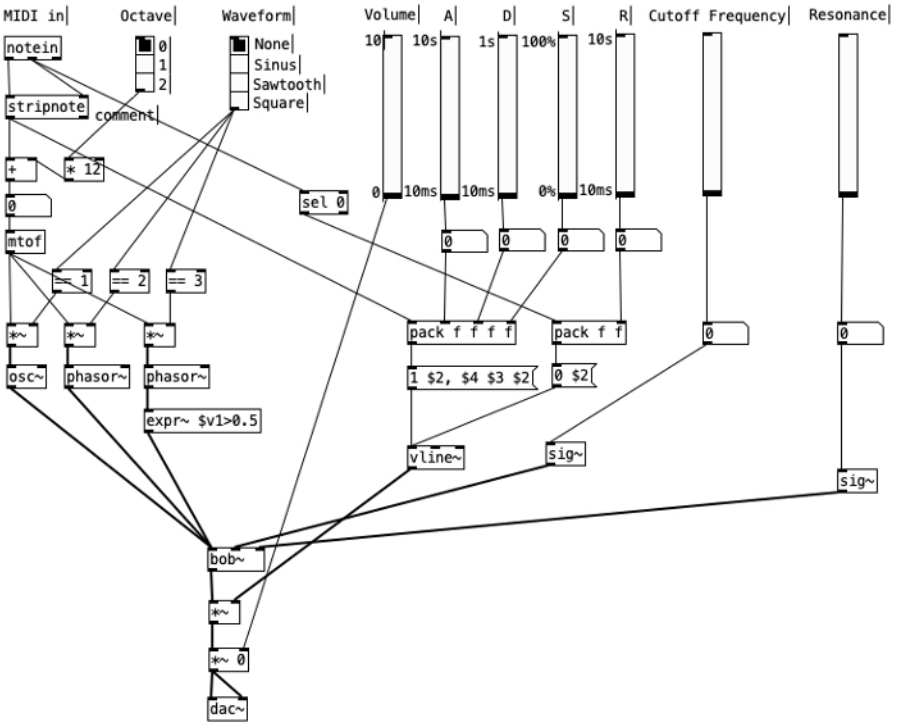

Figure 1.16. *Exercise 3 with Pure Data*

Instructions and comments are as follows:

– resume the setup of the previous exercise;

– `Cutoff frequency`: `slider` that sets the filter cutoff frequency. Its logarithmic range is between 0 and 10,000;

– `Resonance`: `slider` that controls the filter cutoff frequency. Its linear range is between 0 and 6;

5 Numerical approximation method developed in 1901 by Carl Runge and Martin Wilhelm Kutta, widely used, among other things, in filter design calculations.

– `sig~`: an object that converts a number into an audio signal;

– `bob~`: filter whose three inputs, from left to right, are the frequency signal, the filter cutoff frequency and the resonance. Beyond 4, the filter goes into self-oscillation;

– both `number` objects display the values of each of the `Cutoff frequency slider` and `Resonance slider` objects.

1.3.4. *VCV Rack*

Let us continue to evolve our virtual modular synthesizer by adding a VCF module.

Figure 1.17. *Exercise 3 with VCV Rack. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– repeat exercise 2;

– remove the patch cable between the VCO and the VCA;

– add a VCF module;

– connect one of the VCO outputs (SIN, TRI, SAW or SQR) to the input (IN) of the VCF;

– connect the HPF (high-pass frequency) output of the VCF to the IN input of the VCA. You can also test the LPF (low-pass frequency) output to listen to the influence of each filter on the sound reproduction.

1.4. A little further with Max/MSP, Pure Data and VCV Rack

Below are additional elements for creating envelopes and filters with virtual synthesis software, Max/MSP and Pure Data.

A section will also be devoted to the VCV Rack to introduce us to some modules that differ from those provided by default (fundamental blocks library).

We carefully study these different patches, which we can integrate into previous exercises.

1.4.1. *ADSR with Max/MSP*

Here is another simpler way (no `midiin` or `xnotein` object) to generate an ADSR envelope using the velocity of a `notein` object to check the note-off/note-on triggering of the keyboard.

By default, the chosen oscillator produces a rectangular signal, but it is easy to adapt this example.

Instructions and comments are as follows:

– object `/127.`: it divides the velocity, which varies between 0 and 127, to generate, at the output, a decimal value between 0 and 1. When it is not 0, a key on the keyboard is detected as pressed, and the release is not considered;

– the `flonum` object placed behind the object `/127.` displays the decimal value of the division (between 0 and 1);

– the four `dial` objects are rotary cursors that vary each of the parameters A, D, S and R. Their sizes are 2001, 1001, 1 (decimal) and 2001. Their minimum values are all equal to 0;

– to accurately control the parameter values, three `number` objects for A, D, R and one `flonum` object for S connect to the `dials` located above;

– the object `rect~` already encountered in the exercises generates a rectangular signal.

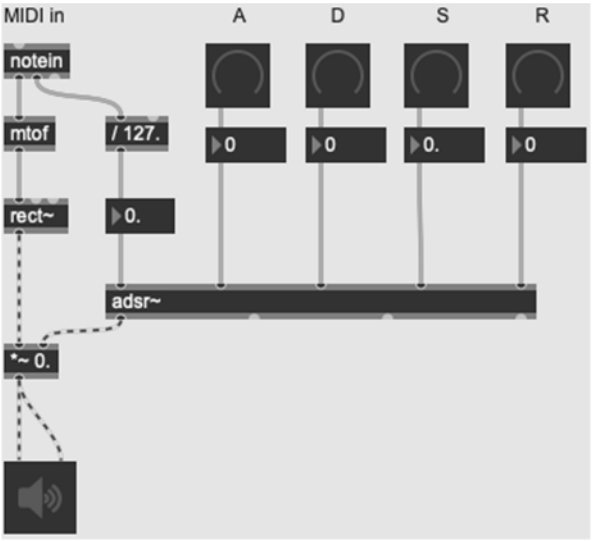

Figure 1.18. *Easy ADSR management*

1.4.2. *Filter with Max/MSP*

For this example, we use an `svf~` object, a multi-state filter that produces four outputs: low-pass, high-pass, band-pass or notch.

Instructions and comments are as follows:

– the object `/127.`: it divides the velocity, which varies between 0 and 127, to generate, at the output, a decimal value between 0 and 1. When it is not equal to 0, a key on the keyboard is detected as pressed, and the release is not considered;

– the `flonum` object, which is placed behind the object `/127`, displays the decimal value of the division (between 0 and 1);

– the two `dial` objects are rotary sliders that vary the filter cutoff frequency and resonance. Their sizes are, respectively, 20,000 and 1 (decimal). Their minimum values (min) are equal to 0;

– to manage parameter values more precisely, two `number` objects for the cutoff frequency and the resonance connect to the dials located above;

– the object `svf~` is the filter; it has four outputs corresponding to each filter type. From left to right, the order is low-pass, high-pass, band-pass and rejector;

– an `umenu` object generates the drop-down menu. It has five items: `None`, `Lowpass`, `Highpass`, `Bandpass` and `Notch`;

– the object `selector~ 4` allows you to choose one of the four filter outputs;

– the object `rect~` already encountered in the exercises generates a rectangular signal.

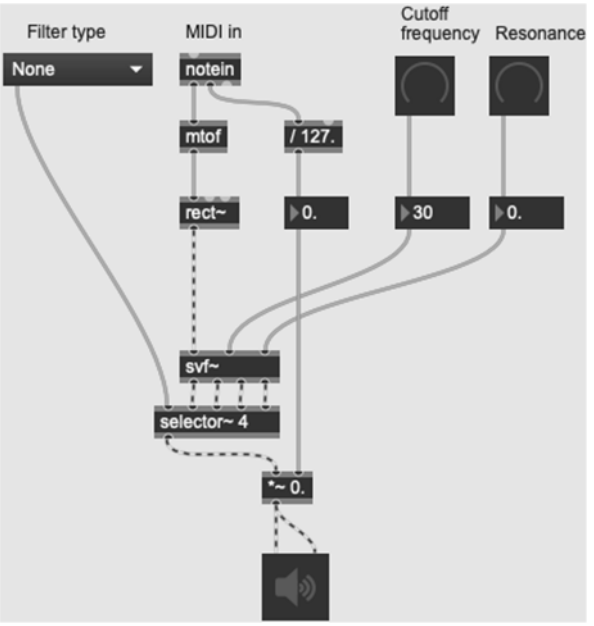

Figure 1.19. *Using a multistate filter*

1.4.3. *Configurable filter with Max/MSP*

This example uses a powerful feature of the Max/MSP, the `filtergraph~` object, which provides a graphical interface allowing you to fully configure a `biquad~` or `cascade~` filter.

Here, we will use the `biquad~` (biquadratic) object, a linear recursive second-order filter with two poles and two zeros.

A `spectroscope~` object displays the spectrogram of the signal without filtering. This object can also display a sonogram depending on its configuration.

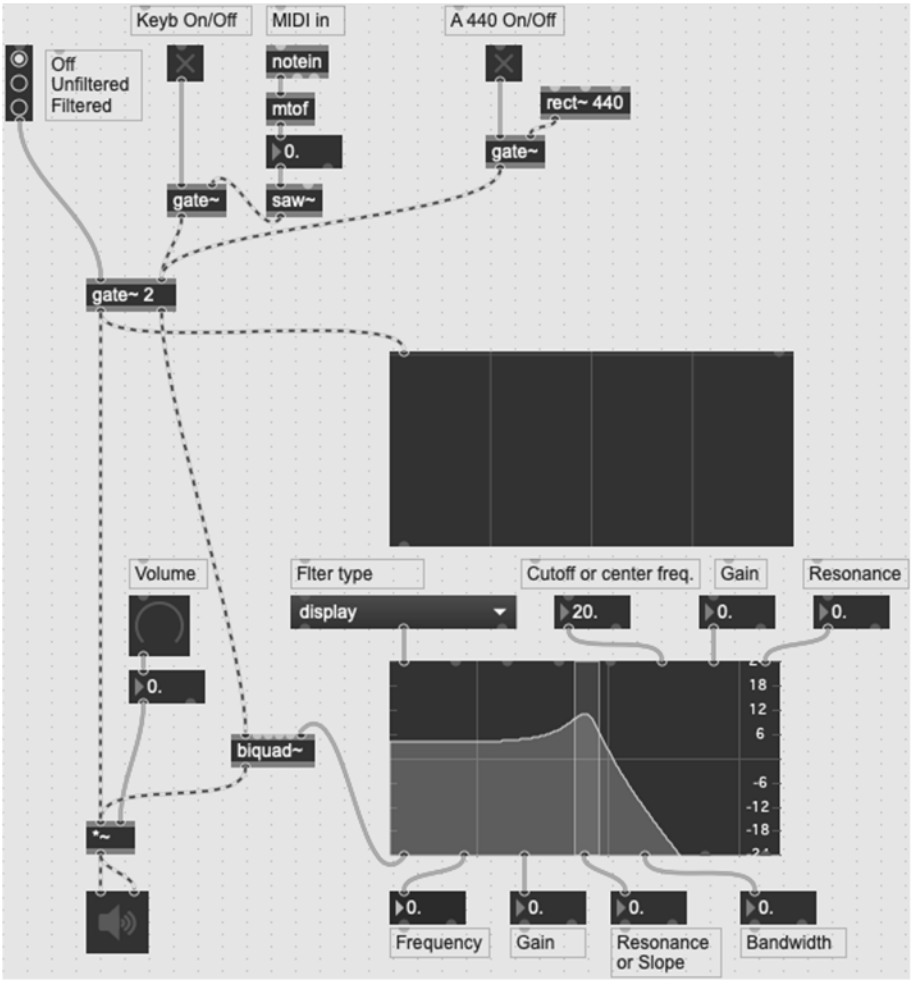

Figure 1.20. *Use of a configurable filter*

Instructions and comments are as follows:

– the `rect~` object generates a square wave signal;

– `Keyb On/Off` and `A440 On/Off` represent two `toggle` objects, that is, on/off switches (0 or 1), which each control a `gate~` object;

– the `gate~` object directs its right input to its output if its left input receives information 1;

– the `toggle Keyb On/Off` object allows the use of a MIDI keyboard;

– the `toggle A440 On/Off` object triggers the sending of an A 440 (A440);

– the `radiogroup` object allows you to choose signal blocking (`Off`), unfiltering (`Unfiltered`) of signals, or the use of a filter (`Filtered`);

– the `gate~ 2` object is a multi-position switch that routes the content of its left input according to its right input. If it is equal to 0, no signal passes; if it is equal to 1, the signal is directed to its left output in the direction of the `biquad~` object; and if it is equal to 2, the signal is directed to its right output in the direction of the `spectroscope~` object;

– the associated `dial Volume` and `flonum` objects control the sound intensity;

– the `dial Volume` object has a size of 1 (decimal) and a minimum value of 0;

– the `spectroscope~` object displays the spectrogram of the signal when it is unfiltered (`Unfiltered`);

– the `filtergraph~` object manages all the filter parameters, the type (low-pass, high-pass, band-pass, etc.), the cutoff frequency, the gain and the resonance. These four parameters can be modified via the `umenu` and `flonum` objects placed above the display window or directly by clicking with the mouse on the different parts of the graph;

– below the graph, the changes made are displayed in four `flonum` objects. The left output of the `filtergraph~` object configures the `biquad~` filter object.

1.4.4. *Custom waveforms with Pure Data*

As you may have noticed in the previous exercises, Pure Data can generate classic, sinusoidal waveforms (`osc~`), sawtooth (`phasor~`) or square (see section 1.1.4).

It is possible to define a custom waveform by acting on the filtering of the audio signal, as subtractive synthesis skillfully does; however, there is another way, using waveshaping, as shown in Figure 1.21.

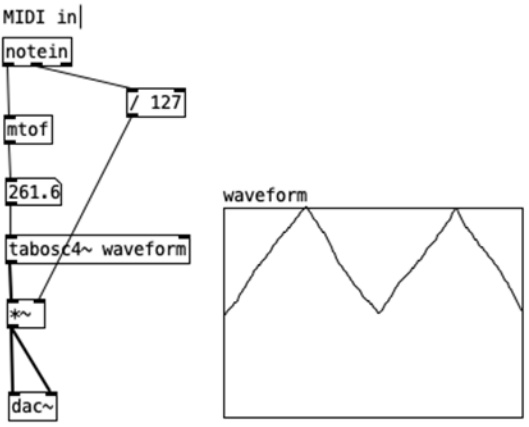

Figure 1.21. *Custom waveform*

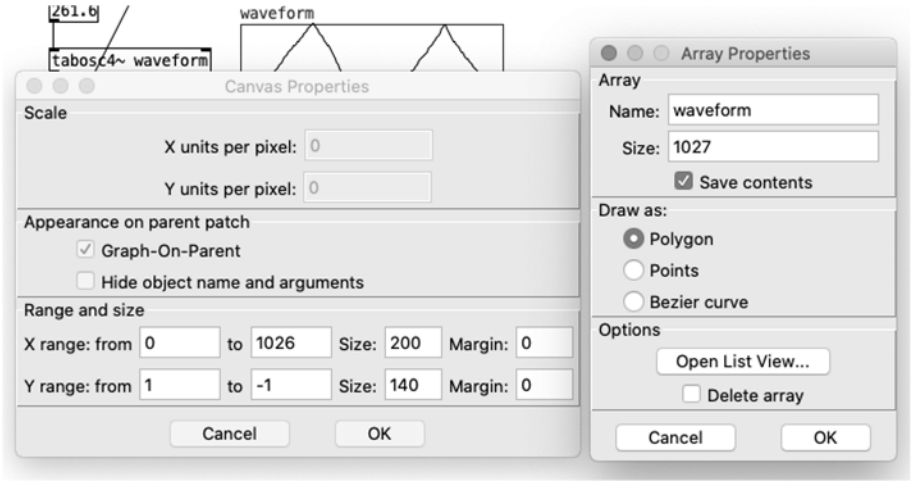

Figure 1.22. *Waveform array parameters. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

Instructions and comments are as follows:

– the `notein`, `mtof`, `number`, `/127`, `*~` and `dac~` objects have already been encountered in previous exercises. We will not go back over their use;

– `tabosc4~` applies to the `waveform` table. It is a wavetable oscillator operating on a 4-point polynomial interpolation. The number of values (points of the

plot) in the table must equal a power of 2 to which 3 points are added. For example, for 29, we will have 512 + 3 = 515 values and for 2^{10}, 1,024 + 3 = 1,027 values;

– `waveform` is a table whose settings are shown in Figure 1.22. For Array Properties, the name is waveform, the size is 1,027. For Canvas Properties, the range of X varies between 0 to 1,026, or 1,027 values, and the extent of Y varies from 1 to –1;

– with your mouse, you can draw the curve of your choice in the frame defined by the `waveform` object.

1.4.5. *ADSR with Pure Data*

In exercise 2 (see section 1.2.4), we set up an ADSR (attack, decay, sustain, release) EG. In this example, we will use another solution based on delays to shape the audio signal, while keeping the same parameters.

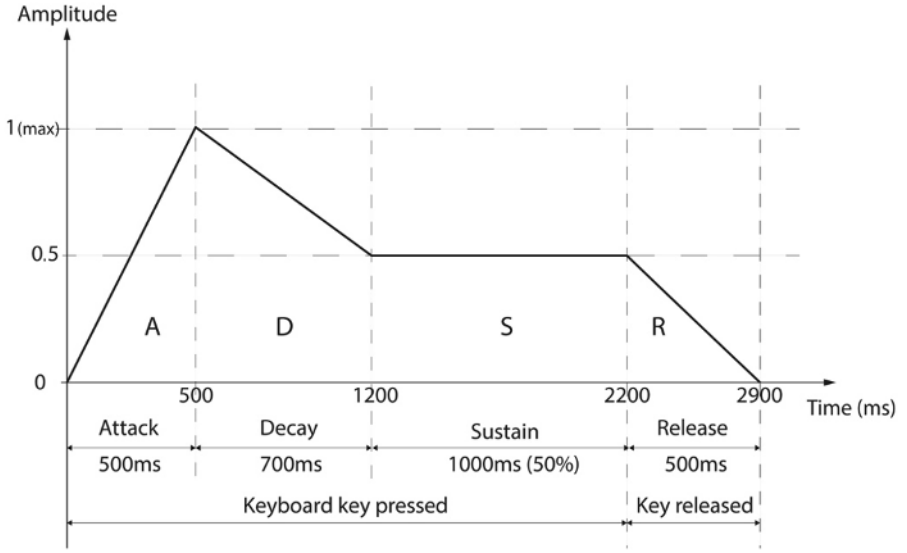

Figure 1.23. *ADSR curve*

The principle is built around the succession of different time ranges linked by `delay` objects (abbreviated `del`).

Take the example of Figure 1.23; the selected ADSR curve respects the following parameters:

– time: A = 500 ms, D = 700 ms, S = 1,000 ms and R = 500 ms;

– intensities: A varies from 0% to 100% (1), D varies from 100% to 50% (0.5), S remains at 50% (0.5), and R varies from 50% to 0% (0).

In our Pure Data patch, we will implement different delays and intensities.

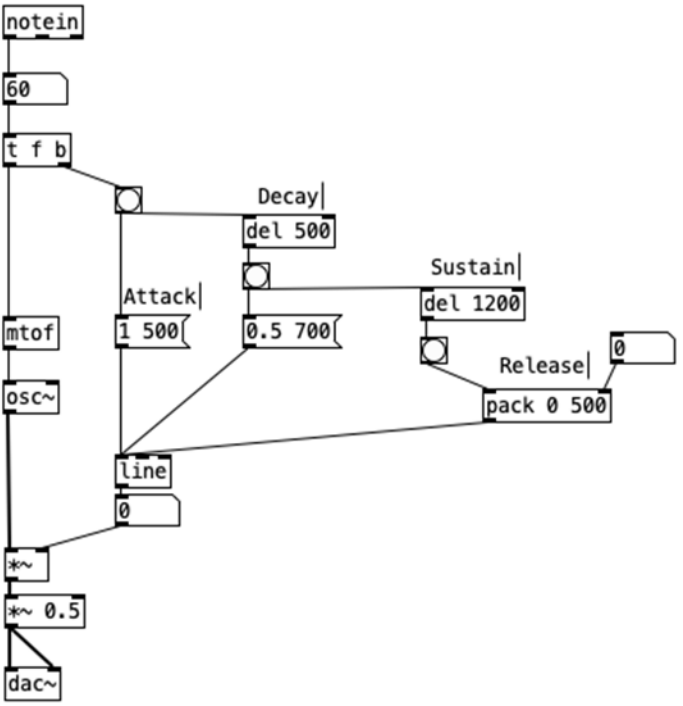

Figure 1.24. *ADSR and delays*

Instructions and comments are as follows:

– the objects `notein`, `number`, `mtof`, `osc~`, `*~`, `*~ 0.5` and `dac~` have already been seen in the exercises;

– the three `bang` objects are there to indicate the sequence of the different stages that the ADSR curve follows;

– the `message` object (`Attack`) transmits an intensity ramp equal to 1 (100%) and of 500 ms duration;

– the `del` object (`Decay`) sets a wait of 500 ms corresponding to the duration of the attack;

– the `message` object (`Decay`) transmits an intensity ramp of 0.5 (50%) and a duration of 700 ms;

– the `del` object (`Sustain`) defines a wait of 1200 ms, which corresponds to the sum of the duration of the attack (A: 500 ms) and the decay (D: 700 ms);

– the `pack` (`Release`) object defines a release (R) to be transmitted with a duration of 500 ms;

– the `number` object connected to the right `pack` input allows the user to vary the duration of the release (R);

– the `line` object generates the different ramps, which are multiplied by the audio signal;

– the `number` object placed at the output of the `line` object shows the numerical sequence of the different temporal steps (ramps) of the ADSR curve.

1.4.6. *Signals and filters with Pure Data*

We have seen that filters are vital parts of subtractive sound synthesis; all synthesizers have at least one, with advanced settings.

The following example will allow the user to discover the main types of filters present within Pure Data: `vcf~`, `lop~`, `hip~`. and `bp~`.

To feed these filters, the user will have two types of waveforms at their disposal: triangular or symmetrical sawtooth.

Instructions and comments are as follows:

– only the Pure Data objects necessary to understand this example will be presented;

– for the oscillator generating triangular signals, we use four objects: `phasor~` (generation of the sawtooth signal), `clip~ 0 0.5` (signal truncation), `*~ 2` (signal amplification by 2) and `-~` (signal subtraction);

– Figure 1.26 shows the progression of the waveform through these four objects;

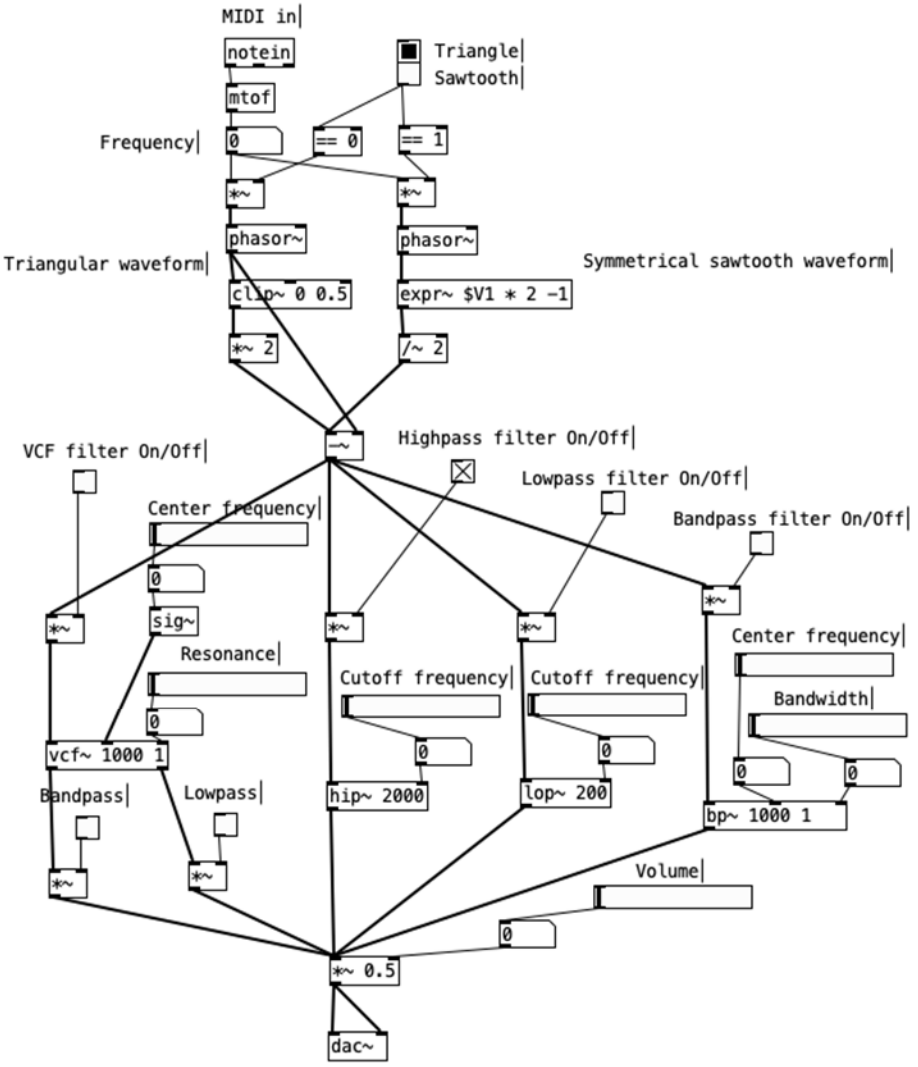

Figure 1.25. *Signals and filters*

– for the oscillator generating sawtooth signals, we use three objects: `phasor~` (generation of the sawtooth signal), `expr~ $V1 * 2 - 1` (transformation of a unipolar signal into a bipolar signal) and `/~ 2` (signal attenuation by 2);

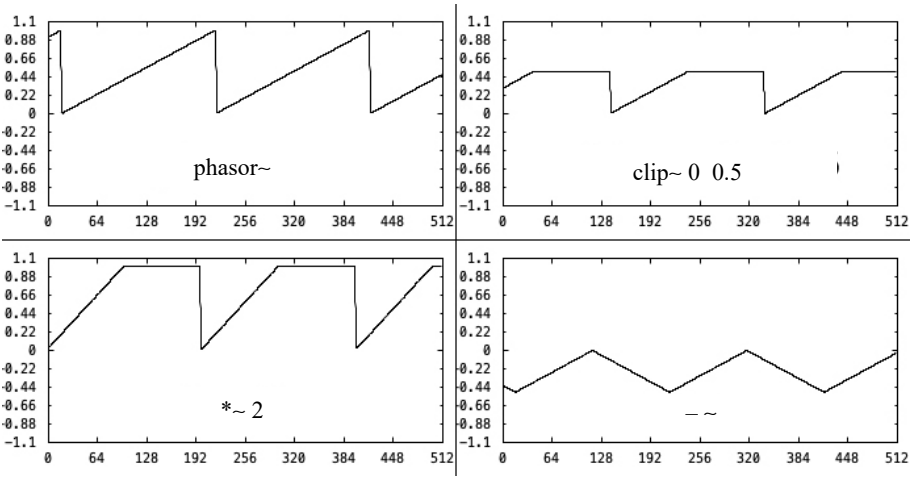

Figure 1.26. *Successive waveforms to obtain a triangular signal*

– Figure 1.27 shows the successive forms of the waveform transformation;

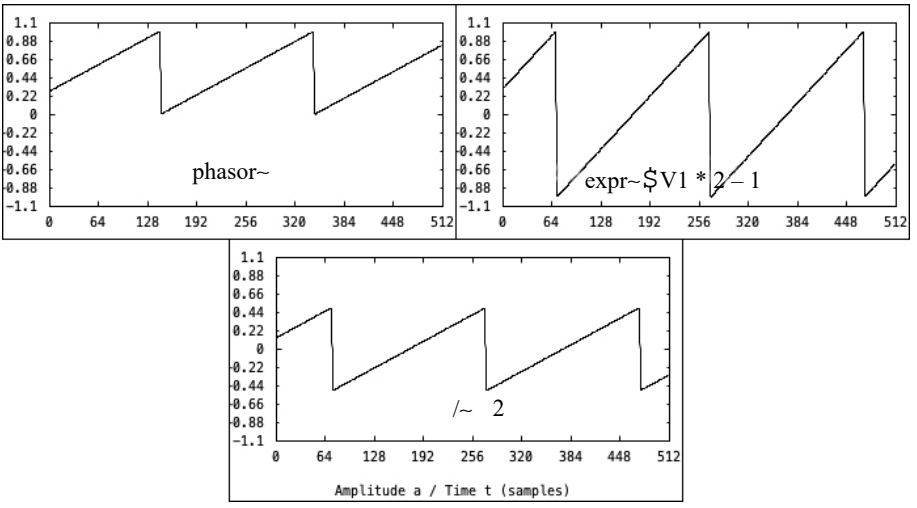

Figure 1.27. *Successive waveforms for the sawtooth signal*

– the four `slider` `Cutoff frequency` and `Center frequency` objects have a logarithmic range of values between 10 and 10,000;

– the `Resonance slider` object has a range of linear values between 0 and 20;

– the `Bandwidth slider` object has a range of linear values between 0 and 10;

– the `volume slider` object has a range of linear values between 0 and 1;

– the four objects `toggle`, `VCF Filter On/Off`, `Highpass filter On/Off`, `Lowpass filter On/Off` and `Bandpass filter On/Off` allow you to select the filter you want to activate. They are cumulative;

– the two `toggle` objects, `Bandpass` and `Lowpass`, allow you to choose the type of filter to use for the `VCF~` object;

– under each `slider` object, a `number` object indicates the chosen value;

– by default, each filter has pre-existing parameters, for example, a cutoff frequency of 1,000 Hz and a resonance equal to 1 for the `VCF~` object;

– do not hesitate to try all the possible settings; you will be able to better understand the use and the modifications made to the sound reproduction by each of the filters on waveforms with different harmonics, odd for the triangular signal and even/odd for the sawtooth signal;

– for the curves corresponding to each signal, see Appendix 2, section A2.1, to integrate a control oscilloscope patch.

1.4.7. *Additional modules with the VCV Rack*

The VCV Rack has a vast library of modules, many of which are virtual copies of actual off-the-shelf hardware modules.

Many of these software modules are free and often have much more advanced functions than those provided in the fundamental blocks library.

In Figure 1.28, you can see a configuration showing an alternative to exercise 3 using the VCO module “Macro Oscillator” from Audible Instruments (free) and the VCF module “Stabile” from Vult (free).

Instructions and comments are as follows:

– wire the two modules;

– try the different settings on each of the modules;

– for the Macro Oscillator, edit the different waveforms by turning the EDIT button;

– for Stabile, you can use each of the outputs (LP: low-pass, BP: band-pass, HP: high-pass and SEM: semblance).

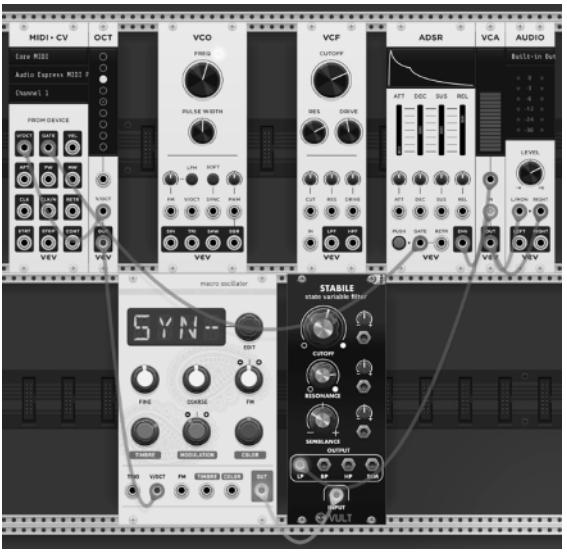

Figure 1.28. *Macro Oscillator and Stabile in a VCV Rack. For a color version of this figure, see www.iste.co.uk/reveillac/synthesizers2.zip*

1.5. Final remarks

In this chapter, we have seen the beginnings of subtractive synthesis on different pieces of hardware and software, but we are still far from having synthesizer models worthy of being called such.

The various exercises presented provide solutions but they are not the only ones, particularly with regard to the VCV Rack, Max/MSP and Pure Data, which are more open than the Behringer Neutron or the 2600.

It is interesting to find other configurations that can provide answers to the three exercises proposed. Consult the reference section at the end of the book to deepen your knowledge. Do not hesitate to ask questions on the many specialized forums in the Internet links section.