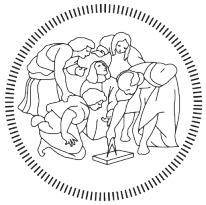
LBNL-O

CMLS Homework N.1 - Assignment 4: Octaver

Group: **Last but not Least**

Professor: **Fabio Antonacci**
Assistant: **Marco Olivieri**

A.Y. 2021/2022



POLITECNICO
MILANO 1863

Contents

1 Introduction

An octaver is a type of special effects unit which mixes the input signal with a synthesised signal whose musical tone is one or more octaves lower or higher than the original. The aim of this assignment is to create an octaver effect and a GUI interface to control it using SuperCollider.

2 Specifications Summary

The implemented special effects octaver unit has the following specifications:

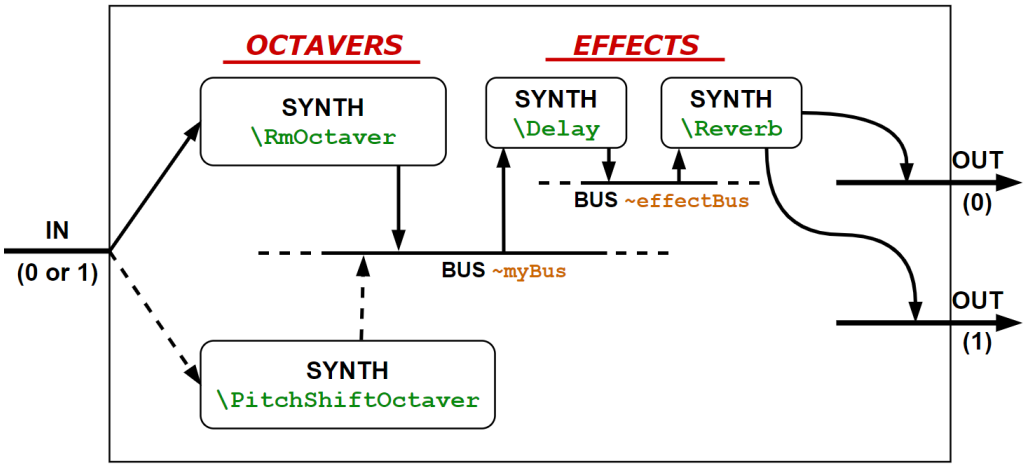


Figure 1: Scheme of the implementation

- **Input**: chosen by the user between the first or second channel of the input bus
- **Octavers**: input signal is processed by one of two octavers, chosen by the user
  - **RM Octaver**: works on single pitch sounds
  - **Polyphonic Octaver**: works on every type of sound
- **Octaves**: both octavers process a total of four added octaves (1 and 2 octaves below, 1 and 2 octaves above the input signal)
- **Effects**: the user can choose to apply one or two effects on top of the octaver-processed signal (**Delay** and **Reverb**)
- **Output**: final processed signal is outputted on both channels of the output bus (stereo)
- **GUI**: a simple Graphical User Interface that takes inspiration from a polyphonic octaver, the Electro Harmonix Pog2 guitar pedal.

Two different types of octavers are used to create a better personalization experience on the user side and to show different methods of implementation.

3 The Octavers

The basic function of an octaver is to take a sound and translate its pitch into one or more octaves lower or higher than the original input signal. Depending on how the octaver is implemented, this process can be constructed in many ways and can lead to different results.

In order to show an example of how the output sound changes as a consequence of the octaver's implementation, the user can choose between two different types of octaver: an **RM Octaver** (which only works with single pitch sounds) and a **Polyphonic Octaver**.

3.1 RM (Ring Modulation) Octaver

The fist type of octaver, used for single pitch sounds, is implemented as a synthesizer which uses SuperCollider's `Pitch` class to find the signal's pitch in order to create and process the added octaves.

Pitch Class

`Pitch` is a pitch follower. It returns two values: a `freq` which is the pitch estimate of the signal and `hasFreq`, which tells whether a pitch was found.

Algorithm

The algorithm is divided into five main steps:

1. Input signal is read from the input bus
2. Pitch frequency of the signal is found using the `Pitch` class
3. Four `SinOsc.ar` signals are created, each having frequency:

$$f_{-2} = \frac{3}{4} \cdot f \qquad f_{-1} = \frac{1}{2} \cdot f \qquad f_{+1} = 2 \cdot f \qquad f_{+2} = 4 \cdot f$$

   where f is the frequency of the previously found pitch.
4. Each new sinusoidal signal is multiplied by the input signal [1]
5. Generated signals along with the input sound are merged together and written on a supplementary bus ready to be further processed.

Each octave has its own amplitude (passed as argument to the synth), so that the user can decide the output volume of all five components (octaves + input signal) of the output sound.

[1] Signal having frequency f_{-1} is summed directly to the input signal in order to keep every spectral component of the input sound intact.

3.2 Polyphonic (PitchShift) Octaver

The second, more versatile type of octaver, which can be used for virually any sound, is implemented with SuperCollider's `PitchShift` class.

PitchShift Class

`PitchShift` is a time domain granular pitch shifter. The method used in the implementation, `PitchShift.ar`, has the following input arguments:

- `in`: input signal
- `windowSize`: size of the grain window
- `pitchRatio`: ratio of the pitch shift
- `pitchDispersion`: deviation of the pitch from `pitchRatio` (*set to zero as the pitch needs to be precise*)
- `timeDispersion`: delay of each grain
- `mul`: gain of the generated signal
- `add`: value to be added to the output

Algorithm

The polyphonic octaver algorithm is divided into three main steps:

1. Input audio signal is read from the input bus
2. Four new signals are generated, one for each octave
    - Apart from `timeDispersion` and `mul` (which can be modified by the user), all parameters are set to obtain an optimal sound
3. Generated signals along with the input sound are merged together and written on a supplementary bus ready to be further processed.

Through the GUI the user can change the `mul` parameter to set the desired gain for each octave, and the `timeDispersion` parameter to personalize the generated sound (changing this parameter can alleviate a hard comb filter effect due to uniform grain placement).

4 Audio effects

To extend the capabilities of the previously constructed octaver, other effects are added to the final mix so that the user can have a more round, complete experience when personalizing the type of sound he wants to be outputted.

4.1 Effect chain

In order to implement the two effects, it was necessary to define an effect chain. The signal is taken from the octaver on the bus `myBus`, than the delay effect is applied and successively, the new signal is sent to the bus `effectBus`. From this bus the delayed signal is taken and modified by the reverb effect. The output signal is then sent to the output bus in order to be played.

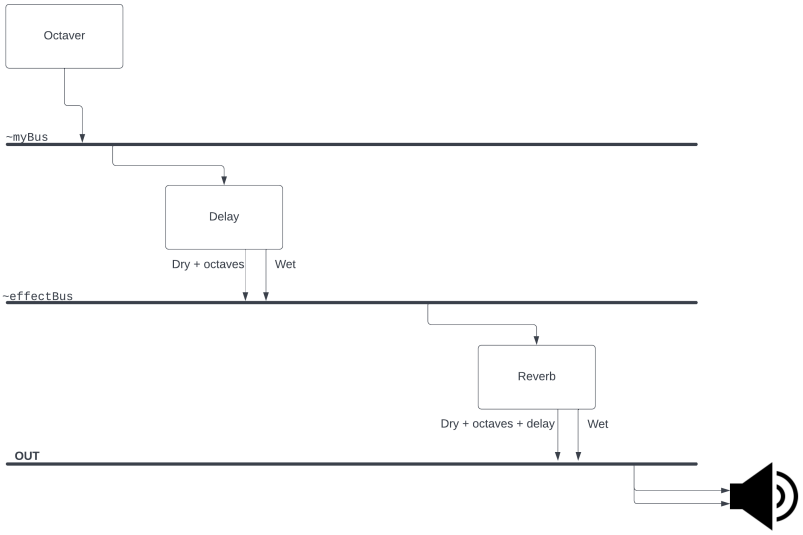


Figure 2: Effect chain

4.2 Delay

The delay effect is implemented through the `PingPong` UGen. This UGen requires a buffer that must have the same size of the input. In this case `[input, input]` is used as input because the input signal is mono. The arguments of the `SynthDef` are:

- `inBus`: number of the bus where the signal is taken from, in this case `myBus`.
- `outBus`: number of the bus where the signal is written, in this case `effectBus`.
- `delayTime`: time in seconds between one repetition and the following one.

- `Feedback`: used to modify the duration of the repetitions of the signal. When it is set to zero the delay effect is turned off, while if it is set to 1 the repetitions go on forever.
- `Wet`: volume of the delay signal, it goes from 0 (no effect is heard) to 1.

Both the input signal (with added octaves) and the purely-delayed signal are written on the supplementary bus `effectBus` which is the input bus of the reverb effect.

4.3 Reverb

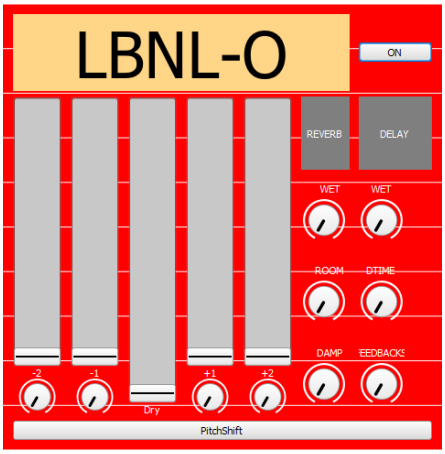
This effect applies to the signal a `FreeVerb` UGen taking as input the delayed signal, generating a reverberated sound which is outputted on the output bus, which is the hardware bus of the server. It uses the following parameters:

- `wetR`: volume of the reverb effect, it goes from 0 (no effect is heard) to 1.
- `room`: room size, goes from 0 to 1.
- `damp`: controls how long the higher frequencies last, goes from 0 to 1.

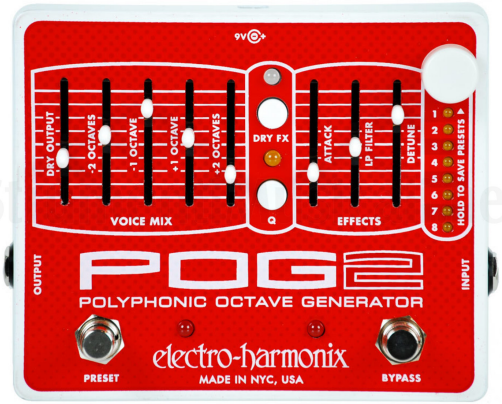
5 Graphical User Interface

Having an intuitive, easy to understand Graphical User Interface (GUI) is crucial for the application to express its full potential.

5.1 Looks



(a) GUI



(b) Electro Harmonix POG-2

The look of the Graphical User Interface is in part inspired by the Electro Harmonix POG-2 octaver guitar pedal. The design is simple yet easy to understand and is composed of four main parts:

- **Title**: written in a big rectangle, an on-off button can be seen next to it
- **Octaves sliders**: each octave has its own volume slider and knob to control time dispersion
- **Effects knobs**: three knobs for each effect control the effects' parameters
- **Octaver switch**: a button is used to switch between the two different octavers

5.2 In Detail

More specifically, the GUI is composed of many different parts:

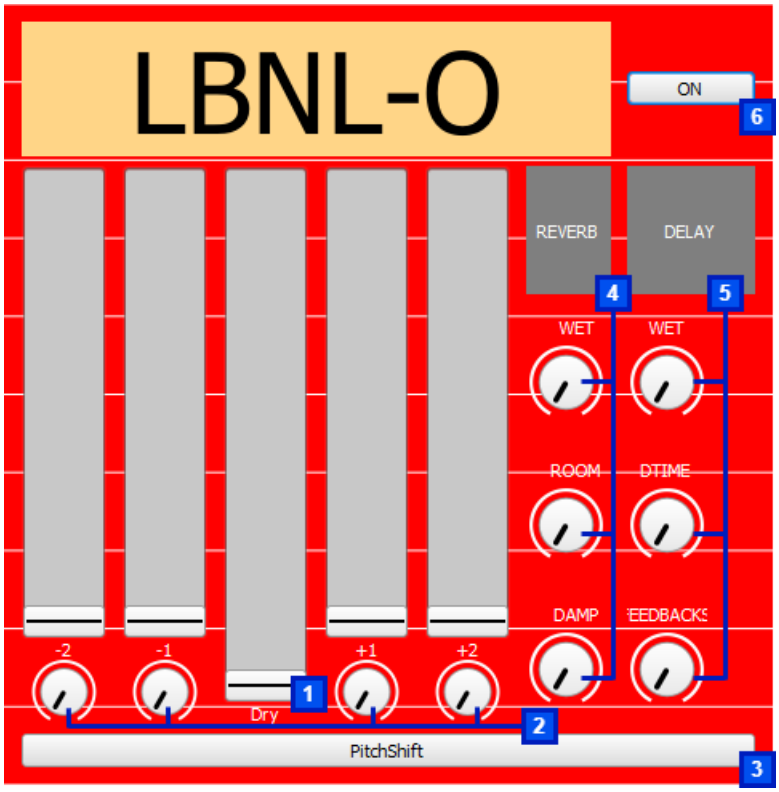


Figure 3: GUI details

1 volume slider of the dry output (which is the unaffected input signal)

2 volume sliders of the octaves with a knob which controls the `timeDispersion` parameter of the `PitchShift` octaver

3 button used to switch between the RM octaver and the PitchShift octaver

4 three volume knobs to control reverb parameters. Respectively: gain, room and dampening

5 three volume knobs to control delay parameters. Respectively: gain, delay time and number of feedbacks

6 button used to turn on and off the overall octaver effects

6 Some considerations

While the polyphonic octaver works well for all four added octaves, the RM octaver only works as intended for the two lower octaves. The RM octaver is based on **Ring Modulation**, which cannot be used to generate a faithful recreation of the input sound's spectrum at higher octaves, as it keeps all the integer multiples of the original signal. As such, the only octave which has the desired octaver effect is the first octave below the original signal (-1), while **the others are to be intended as effects** that modify the sound's colour and characteristics.