L'Ottavatore

CAPR group:

Claudio, Eutizi (`ID: 10812073`)
Andrés, Bertazzi (`ID: 10488849`)
Pierluigi, Tartabini(`ID: 10845797`)
Riccardo, Martinelli (`ID: 10456202`)

SuperCollider Homework assignment #4
Computer Music: languages & Systems
Politecnico di Milano

April 4, 2022

Contents

1 Introduction **3**

2 GUI **3**

2.1 OTTAVATORE 4

2.2 PANNATORE 5

2.3 RESET 5

3 Synth structure **6**

3.1 Source group and pitch shift group 6

3.1.1 PitchShift & \hiOctaver and \lowOctaver 6

3.2 Panning group, Il Pannatore 7

3.3 Mix and master groups 7

1 Introduction

The **Ottavatore** is a guitar-pedal-designed octaver application, implemented using **SuperCollider 3.12.2**. Its main functionalities are addition and mixing of an analog input audio signal from an audio card (e.g microphone or guitar) with synthesised signals that reproduce upper or/and lower octaves.

Furthermore, a panning section, the so-called **Pannatore**, is introduced on the right part of the pedal in order to modify the position of each voice in the stereo field statically and dynamically.

2 GUI

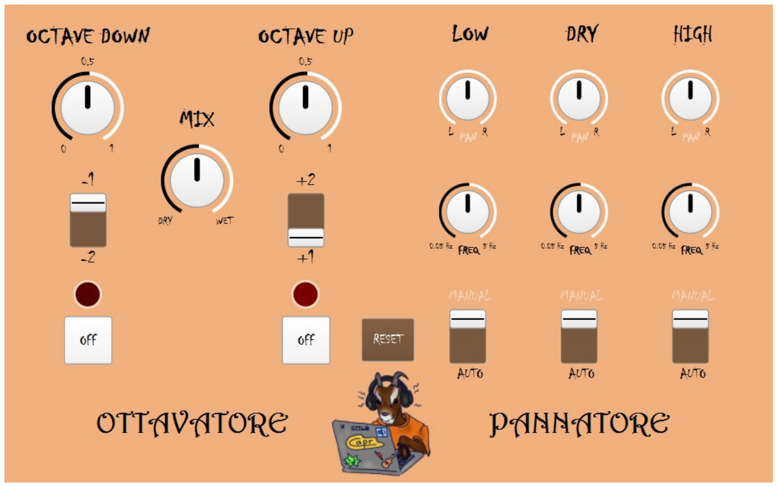

Figure 1: Ottavatore GUI.

The GUI of the pedal has been made with a rectangular window (`Rect`) and has been divided into two sections:

1. The "*Ottavatore*", which manages the presence of the octaves inside the output signal.
2. The "*Pannatore*", which manages the panning left/right of the output signal.

The two sections are highlighted in the code by two titles ("OTTAVATORE", "PANNATORE") using the class `StaticText`.
They are shown on the bottom of the window at the sides of the logo.
This one has been created with an object of the class `Image` which reads the file `"LOGO Capr.png"` from a specific path. Then the image is printed on the screen by using the method `.drawAtPoint_()`.

2.1 OTTAVATORE

- **Button & Leds**
  The buttons on the bottom of the window are used to turn on/off the higher and lower octaves of the input signal. It has been used the class `Button` to create the rectangle and the attribute `.states` in order to create the strings "ON" and "OFF" respectively.
  Since it is inspired from a guitar pedal, the switches of the buttons allow to color the leds above from a darker red (OFF) to a brighter one (ON).
  In order to do this, it has been used the `.drawFunc_()` method of the `Window` class.
  So as to manage correctly the switches of the element's colors, some `if-else` conditions has been implemented on the values of the specific states.

- **Switches**
  The switches has been implemented through the class `EZSlider` and positioned above the related buttons and leds. Both of them allow to control the octave's number through which the input signal is modified.
  The left switch allows to decide the lower octaves (*-1, -2*) and the right one the higher octaves (*+1, +2*). In particular, if the state of the button is "ON", the chosen octave's number is highlighted and alternates with the respective position of the switch.
  This is made with the `.action_()` method.

- **Octave Down & Octave Up Buttons**
  The knobs the volumes of each chosen octave of the output signal. They are made with the `Knob` class and have a range of values from 0 to 1 (the initial value is 0.5).
  When the state of the buttons is "ON", the related octave's title is highlighted.

- **Knob MIX**
  The central knob controls the presence of both octaves in the output signal with DRY & WET parameters.

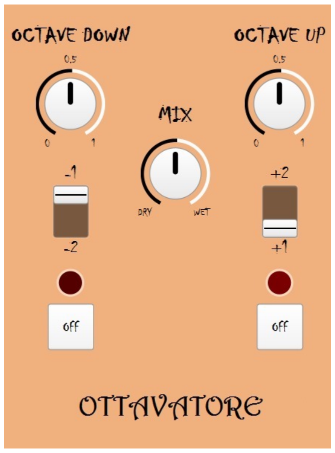

Figure 2: The "Ottavatore" section.

2.2 PANNATORE

The "Pannatore" section is divided into three columns: Low, Dry, High.
The first refers to the lower octaves, the second to the dry signal, and the last one to the higher octaves.
Each column has two parameters:

- **Pan Knob**
  This knob modifies the panning of the specific octave to left (L) or right (R) and is controlled by the switch's value "MANUAL".

- **Freq Knob**
  This knob modifies automatically the panning of the specific octave to left (L) or right (R) with a frequency whose range starts from 0.05 Hz to 5 Hz.
  It is controlled by the switch's value "AUTO".

  The switch highlights the specific knob's title when it is activated.
  It has been used the same classes (`Knob`, `EzSlider`) described in the *subsection 2.1*

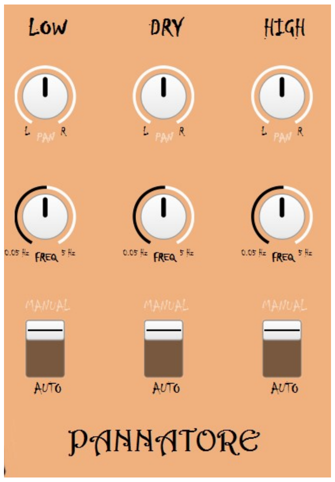

Figure 3: The "Pannatore" section.

2.3 RESET

The **Reset** button is in the middle of the window (as it is shown in *Figure 1*).
It allows to reset the default values of each element.
In particular, all the knobs of the application resets themselves at the value defined in the `.valueAction_()` function.
The Ottavatore's switches instead get back to +1/-1 octave values and the Pannatore's ones to "MANUAL" mode.
It is important to say that the reset button does not change the ON/OFF states of the buttons.

3 Synth structure

The main aim of the developed application is to provide an interface and a signal path management that makes the user experience as complete and user-friendly as possible.

For this purpose, the signal path from the audio input to the master output has been managed creating different **audio and control busses** and **groups**. These last respectively transport signals from one synth instance to the next one, regulate the order of execution of the synths and group these synths together.

In particular, a high-level flow chart of the application's implementation can be seen in *Figure 4* and its details will be described in the next sections.

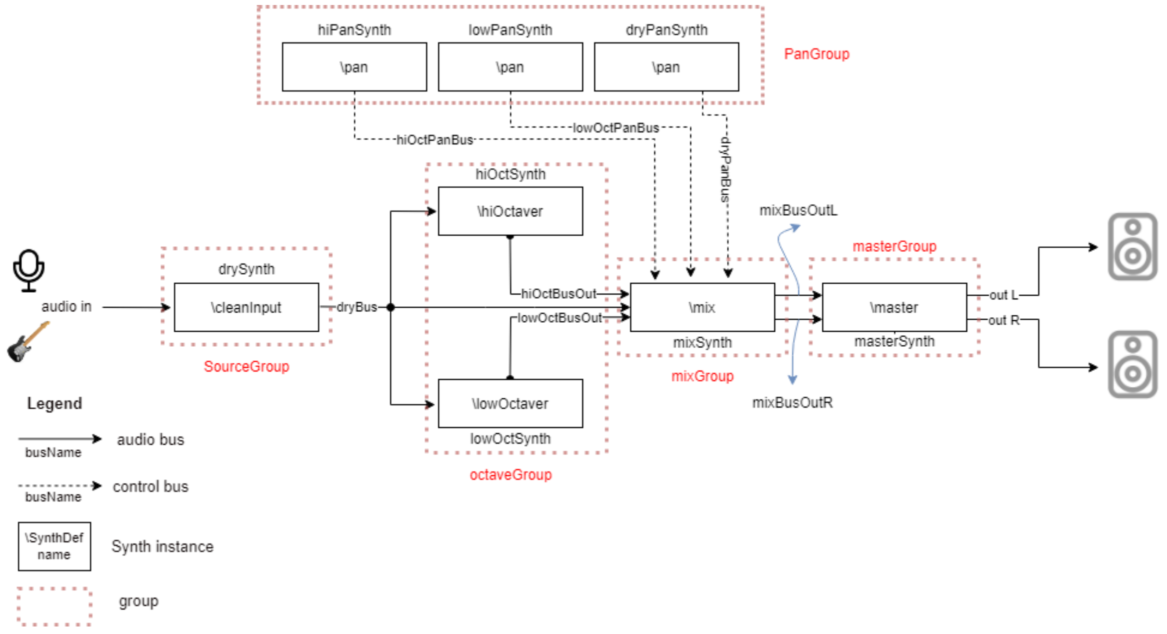

Figure 4: the Ottavatore high-level flow chart.

3.1 Source group and pitch shift group

The audio coming from the audio card is fed into a very first `SynthDef` instance of name `\cleanInput` that forwards the signal to an audio bus, `dryBus`, which will send it to the pitch-shifting group. This first Synth belongs to the `sourceGroup`, the first of the groups execution queue.

The subsequent group, `octaveGroup`, contains the instances of `\hiOctaver` and `\lowOctaver` `SynthDefs`: the key players of this section. Both of these have been implemented using the `PitchShift` class.

3.1.1 PitchShift & \hiOctaver and \lowOctaver

The class `PitchShift` is *"A time domain granular pitch shifter. Grains have a triangular amplitude envelope and an overlap of 4:1, and use linear interpolation of the buffer"* (1). In this particular implementation, the best audio quality of the pitched signals, whether in the higher or in the lower octaves, has been reached using a windowSize parameter of 0.08. The pitchRatio parameter, responsible for the amount of pitch variation, can be dynamically set via the GUI switches, choosing between the first and the second lower/higher octave of the input sound.

However, during the tests, has been found that choosing the second octaves ruins the output signal, introducing an unwanted delay and sort of a distortion. This phenomenon will be studied in detail in the next updates of the application.

Other valuable parameters of these two SynthDefs control the volume of each shifted octave signal and their on/off states. Each of these can easily be handled using the GUI knobs and controllers described above.

The synthesized output audio signals downstream of the just described synths are sent to two different mono audio busses, hiOctBusOut and lowOctBusOut, and they will be two audio inputs of the mixing stage.

3.2 Panning group, Il Pannatore

In order to provide the user with the possibility to move independently the three voices (dry signal, higher and lower octave) in the stereo field, **three independent synths** have been allocated in the panGroup, one for each voice. These last are instances of the SynthDef \pan.

With the GUI switch set to *MANUAL*, the stereo position of each sound is a fixed parameter panPos $\in [-1(L), 1(R)]$, chosen by the *PAN* GUI knob.

Setting the switch to *AUTO*, an **auto pan** effect is activated and the pan value will depend on a FSinOsc.kr(panFreq) definition. The chosen signal will travel from L to R with a certain panFreq frequency that the user can manually tune on the GUI knobs. The three voices can be manually or dynamically panned independently one from each other.

The output of the three synths is sent to three different control busses: hiOctPanBus, lowOctPanBus and dryPanBus.

3.3 Mix and master groups

The mixGroup contains the \mix synth that takes as audio input the signal carried by dryBus, hiOctBusOut and lowOctBusOut. In this SynthDef, the wet GUI-controlled parameter sets the amount of dry and wet signal on the synth output. Subsequently, a Pan2 class is used to pan the signals using the information carried by the control busses as pos argument. These busses come from the panGroup. The panned L/R components of the signals are then respectively summed and sent through the audio mixBusOutL and mixBusOutR mono busses.

These busses are connected as input to the last master synth, that belongs to the masterGroup and sends the signals it receives to the main output busses.

References

[1] SuperCollider, “Pitchshift class documentation.”

Many thanks to ***Marco Di Mambro*** *for the CAPR logo design and drawing.*