CMLS Homework 3 - Voice Harmonizer

Group 11: E. Castelli, E. Intagliata, A. Rizzitiello, G. Zanocco

May 29, 2021

1 GitHub Repository

https://github.com/ElisaCastelli/CMLS-HW3-Group11

2 Introduction

The aim of the third homework assigned to our group is to develop a vocal harmonizer. This application, whose features reconcile different disciplinary areas (such as object-oriented programming, sound and music), converts the voice into a musically richer version of itself. In fact, the input audio captured by the microphone is returned in polyphonic form with respect to some reference chords, resulting in multiple overlapping versions of the same voice, harmonically coherent with each other. We also decided to implement two sound effects which can be applied to the output voice (delay and reverb).

As specified in the guidelines, this tool was developed using *SuperCollider* [1], a free and open source software for sound synthesis and real-time control, along with its communication protocol *OSC*. Concerning the implementation of the graphical user interface, we decided to use *Processing 3* [2] for managing the parameters of the harmonization and the sound effects.

3 Vocal Harmonizer

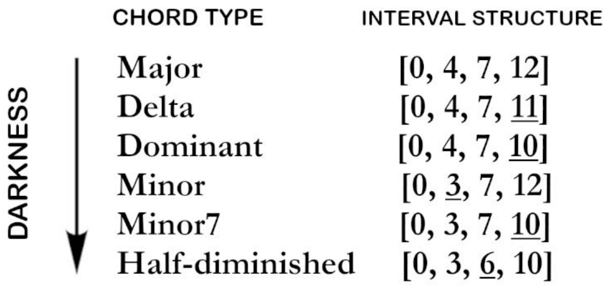
Music, often defined as organized sound [3], is characterized by multiple sound events aimed at stimulating our perception. This process is two-dimensional. In fact, it develops both horizontally (sequentially) and vertically (synchronously). Harmony is the relationship that is established in parallel between several sounds. First of all, to automatically harmonize the voice, it was necessary to choose certain chords consistent with a reference harmonic syntax (in our case the tonal harmony). Specifically, we have chosen to use the six types of quadriads obtained from the harmonization of the major scale which are the following:

- Major

- Minor
- Major7 (delta)
- Minor7
- Dominant
- Half-diminished

In this way, the sound we hear isn't just more pleasant (because it is more familiar to our ears), but also more diversified than what it would be using only triads (three notes) or bichords (two notes). For the first two types of chords (Major and Minor), we chose to repeat the root in the upper octave in order to obtain a complete four-note chord.

In addition, to accommodate those who do not have specific musical knowledge, we also decided to arrange the chords in a scale according to their degree of "darkness/brightness". For example, a minor chord can be considered "darker" than a major chord, so it will occupy an higher position in this scale. This order is based on a musical principle, concerning the inherent interval structure of any group of notes. By lowering one or more notes of a chord by a semitone, we can obtain new chords with different sounds, generally gloomier. The following figure explains in detail this musical principle and shows how it was applied as metric for ordering the chords:



CHORD TYPE	INTERVAL STRUCTURE
Major	[0, 4, 7, 12]
Delta	[0, 4, 7, $\underline{11}$]
Dominant	[0, 4, 7, $\underline{10}$]
Minor	[0, $\underline{3}$, 7, 12]
Minor7	[0, 3, 7, $\underline{10}$]
Half-diminished	[0, 3, $\underline{6}$, 10]

Figure 1. Darkness diagram

The number underlined represents the note which has been lowered with respect to the previous chord. In this way, the user also has the possibility to harmonize their voice by easily passing from one type of chord to the next by means of this musical principle.

4 Implementation

As anticipated in the Introduction, we used SuperCollider to implement both the functions described in the previous chapter and the communication with the user interface (using the *OSC* protocol).

The three main sections of the code are given by the three *SynthDef* which represent the core of the program. The first is called **\vocarmonizer** and is responsible for generating the harmonized voices with respect to the selected chord. It is characterized by five arguments, specifically the *bus* from which the input signal is read, the chord chosen for vocal harmonization (by means of the relative degree of *"darkness"*) and *gain* of the voices that are generated respectively. After defining the interval structure of each chord, the *pitch shift* of the voice is implemented through the UGen *PitchShift*, computing the relative *pitchRatio*. Once we have generated the harmonized voices, we proceed to mix them together using the UGen *Mix* and the signal is returned to the initial bus thanks to the UGen *Out*.

As for the sound effects, they are defined by the *SynthDef* **\delay** and **\reverb**. The first takes *bus* and *delaytime* as input arguments. The *delay* effect is applied by the UGen *DelayN* input signal. Instead the second has as input parameters *bus*, *roomSize* and *roomWetDry*. The *reverb* effect is applied by the UGen *FreeVerb* input signal. In both cases, the modified signal is returned on the same *bus* they receive as input.

The diagram below shows the internal architecture at a high level of abstraction.

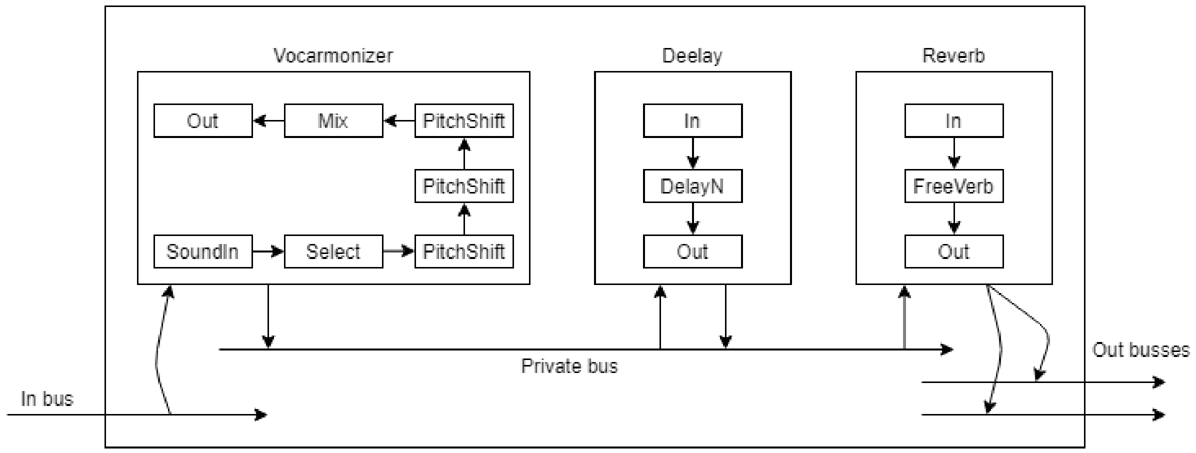


Figure 2. Internal architecture

In order to connect *Processing* with *SuperCollider* and control the parameters of the harmonization or of the effects, we use the *OSC* communication protocol. The *GUI* is articulated in three different sections. Starting from the left side, the user is free to select a specific chord for the voice harmonization and to change the gain of a specific generated voice. On the right side there are the sliders which control the effects: *delay time*, *reverb room size* and *reverb wet/dry*. On the bottom you can select the chord with the related *degree of darkness*. The appearance of the graphical user interface is shown in the following figure.

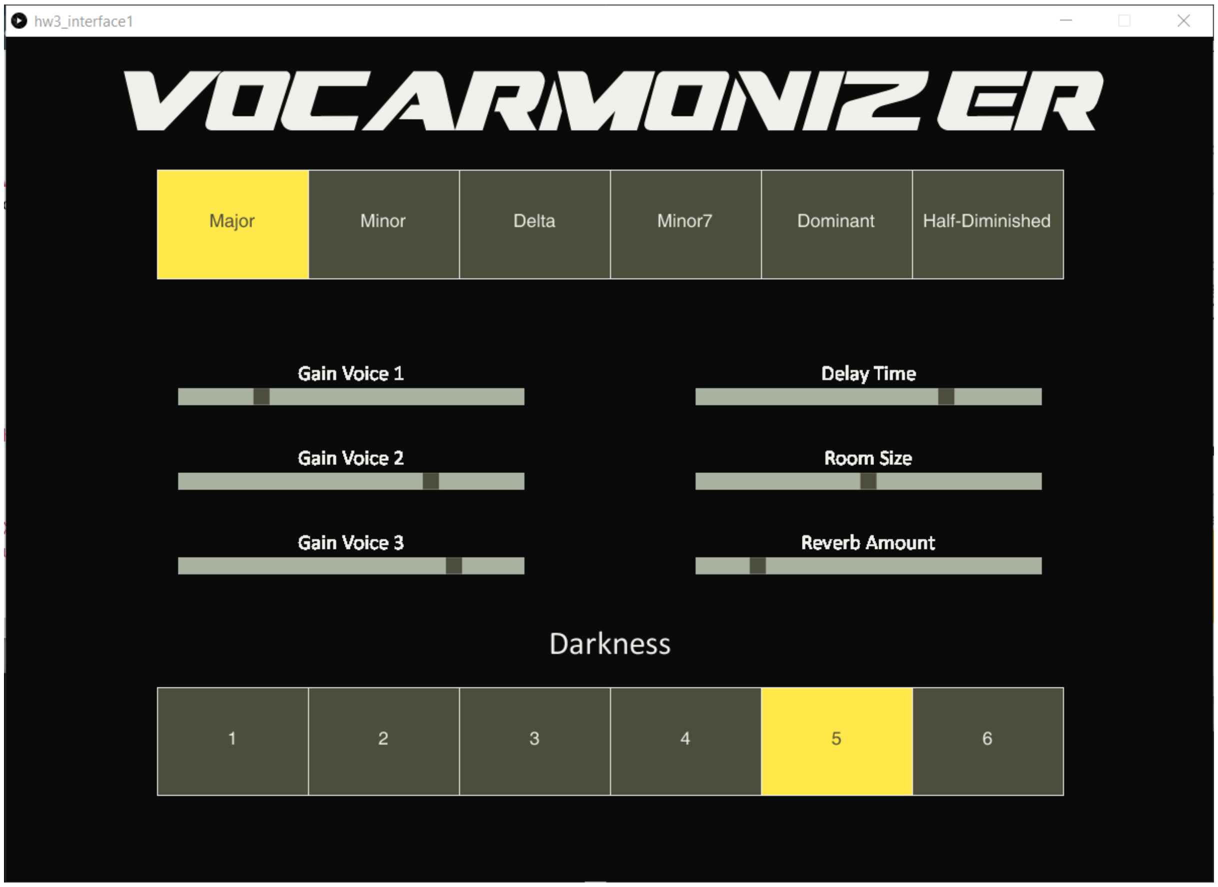


Figure 3. Graphical user interface

5 Conclusions and future work

In this work we have developed a tool capable of harmonizing the voice with respect to a set of reference chords (specifically, the quadriads obtained from the harmonization of the major scale). We used the *SuperCollider* software both from the operational point of view (to capture the sound, shift the voice, create different notes, and then add them back to the original version), and to communicate with the user interface through the *OSC* communication protocol. In fact, for controlling the parameters of the harmonization and the effects we have developed an intuitive user interface through the *Processing 3* software. Moreover, we tried to arrange some graphic objects basing on musical principles so as to make the tool interesting and fun even for those who do not possess particular skills in the field of music. We have also implemented the ability to apply two effects to the harmonized voice, which are *deelay* and *reverb*.

The main future developments can be included in two categories. The first concerns all the improvements from an artistic point of view, such as:

- implementation of new types of more "exotic" chords, using more dissonant tensions

- implementation of new effects such as spectral effects (EQ, panning etc.) or modulation effects (chorus, tremolo, flanger etc.).

Instead, from a technical point of view:

- ability to record, store and play your own performance
- improvements from the point of view of networking, to guarantee multiple users to interact simultaneously.

References

[1] SuperCollider is available at the folliwing link: `https://supercollider.github.io`. Cited at page 1.

[2] Processing 3 official web page link: `https://processing.org`. Cited at page 1.

[3] Edgard Varèse, Concepts of Organized Sound. Jan. 1982. Cited at page 1.