**Nonlinear Quasistatic Quintet**

Prompt 3: Harmonizer

Graphical user interface, application

Description automatically generated

Group members: Marco Cristofalo, Marco Bernasconi, Francesco Barraco, Bram Steenbergen, Geoffrey Horn

04/04/2022

Department of Music and Acoustic Engineering

Politecnico di Milano

The group known by the moniker of Nonlinear Quasistatic Quintet (NQQ) has been tasked with the creation of a vocal harmonizer within the Supercollider system. Given the freedom to model a harmonizer within a loose set of constraints, NQQ set to work creating the general logic that the Supercollider code was to follow.

From previous experience using harmonizers, primarily of the guitar variety, it is known that the basics of a harmonizer involve reading an input signal, identifying the frequency being inputted, shifting the note given amount within a specified key, mixing all voices selected by the user, then outputting the input signal along with the shifted harmonized signal to create a harmonized output. The user is then given the option to apply after-effects, such as delay and reverb, to tune the sound to their specification. Therefore, our programming logic follows the logic presented in Figure 1.

Diagram, schematic

Description automatically generated

**Figure 1: UGen Flow Diagram**

Many common harmonizers follow the logic presented above. However, the information found in Figure 1 is an over-simplification of the background programming and music theory required to create an effective and efficient harmonizer.

This harmonizer works with modal scales, meaning that it will dynamically change the pitch shift ratio of every voice in order to let them remain diatonic. Four pitch shifters are running in parallel, each of them harmonizing a specific interval: 3rd, 5th, 6th and 7th, which are the most common voices used for harmonizing. This means that every pitch shifter must cover two possible intervals: major and minor or, in the case of the 5th, perfect and diminished.

Instead of tracking the selected scale in semitones and then make decision based upon the entire scale, a chordal approach utilized. For each voice, a flag array of 12 elements has been created. Within each array, elements are marked with 0’s and 1’s in which a 0 represents a major (or perfect) interval and a 1 represents a minor (or diminished) interval. The element’s position in the array represents the note’s position in the scale, which has been extended with non-diatonic notes in order to have a complete mapping of the possible cases. These arrays are first created with respect to the C Ionian scale and then modified according to the key and mode Graphical User Interface (GUI) controls.

Take for example the chord harmonization of the C Ionian scale:

C – Dm – Em – F – G7 – Am – Bm(b5)

The progression is then extended to include non-diatonic root notes (C# , D#, F#, G#, A#) in such a way that their harmonization is still in key (except for the root).

C – C#m(b5)maj7 – Dm – D#(b5) - Em – F – F#m(b5)maj7 – G7 – G#m(b5)maj7 - Am – A# – Bm(b5)

From this, only the 3rd interval progression is considered with respect to the root of each chord: (3: major b3: minor)

3 – b3 – b3 – 3 – b3 – 3 – b3 – 3 – b3 – b3 - 3 – b3

The progression is then transcribed in the following flag array: [0,1,1,0,1,0,1,0,1,1,0,1].

Repeating this process for every voice, four arrays are created:

~scaleThird = [0,1,1,0,1,0,1,0,1,1,0,1];

~scaleFifth = [0,1,0,1,0,0,1,0,1,0,0,1];

~scaleSixth = [0,1,0,1,0,0,1,0,0,1,0,1];

~scaleSeventh = [0,0,1,0,1,0,0,1,0,1,0,1];

It is now sufficient to translate the detected pitch from frequency to a number from 0 to 12, where 0 = C and 12 = B. From then on, that value is used to access the array to obtain the correct diatonic harmonization for each voice.

It is not necessary to repeat the whole procedure whenever there is a key or mode change. Since this harmonizer works only with major scale harmony, it is sufficient to shift the array. A change in key is equivalent to a right shift while a change in mode to a left shift. Consider as an example the third voice array:

C Ionian = [0,1,1,0,1,0,1,0,1,1,0,1]

D Ionian = [0,1,0,1,1,0,1,0,1,0,1,1] (right shift x2)

C Dorian = [1,0,1,0,1,0,1,1,0,1,0,1] (left shift x2)

The pitch is always decoded with reference to C, so the value for accessing the array will not change when the mode or the key is changed. Notice, however, that Phrygian and Lydian modes are separated by a single semitone instead of two. Therefore, this must be compensated for by applying an additional right shift.

As an additional feature, it is possible to use the harmonizer as a fixed pitch shifter. This feature is useful when harmonizing with 5th only and constantly switching key while playing. With this option is selected, the arrays are simply switched to all 0s (major/perfect) or all 1s (minor/diminished), in order to keep using the same algorithm.

While all the above music theory is critical, unlike a guitar which is played within a certain key, the human voice is much more volatile. Even if the user attempts to sing a B, per se, most singers are not skilled enough to hit that specific note perfectly. Fortunately, the Pitch function in Supercollider presents a simple solution to the issue of identifying pitch. The Pitch function, which is an algorithm pre-programmed into supercollider, takes the input from the sound car/microphone, and returns a ‘freq’ variable. This frequency variable can also be translated into a MIDI note.

Setting the frequency to a MIDI note allows the program to easily shift the pitch as required. Since MIDI notes are given as a number from 0 – 128, each of which has a specified frequency, the frequency is multiplied by the specified pitch ratio for each key for simple computation.

Supercollider has a built-in PitchShift feature, which utilizes the pitch ratios as specified in the previous section. After reading about the feature, it seemed to be the perfect solution. The PitchShift feature is a time domain granular pitch shifter in which an input signal is from the sound card is received, then allows the programmer to specify the window size, pitch shift ratio, pitch dispersion, time dispersion, and an input multiplier. This feature lets the program shift the pitch as specified per each necessary ratio according to the type of harmony that is called.

While the PitchShift feature was effective in accomplishing its task, there was difficulty with specifying the proper window size for a vocal input, as the PitchShift function is better suited for a guitar input. After some online research, others had similar problems, and one developer created the PitchShiftPA pseudo-UGen. Unlike PitchShift, it allows the user to shift the pitch while maintaining the formant structure of the sound. With a vocal input, this allows the program more consistent pitch shifting. However, one of the most important aspects of the PitchShiftPA implementation is having good quality tracking.

To keep track of the simultaneous inputs, the implementation of buffers is paramount in ensuring that our harmonic information was kept independent of each other, and the output sound is clear. In implementing the buffer system, it is critical to understand the usage of a buffer, and how it works within SuperCollider. A buffer is used to hold sampled audio, such as the input signal from the microphone, before being used as the input for the SynthDefs. These buffers are then called depending on which key is selected by the user.

Additionally, busses are critical to keep output signals organized based on the user’s selections for the harmonization settings. Therefore, each harmonizer setting requires its own bus allocation. A series of busses are utilized, each acting as a gate to ensure all data flows properly along each step from input to output. Due to the large amount of data being processed and the various channels that could be activated, 28 busses were utilized in the creation of the harmonizer. Control bus are allocated for the 5 mixers and the pitch data, whereas audio busses are for all other signals, a total of 6 control busses and 22 audio busses.

All input data is processed according to the user’s specification by the “mixer”, which can be seen in Figure 1. This mixing strip is responsible for specifying the information to be directed from the harmonizing code to the output channels. These specifications are dictated by the user through the utilization of the GUI.

Paramount to the creation of a successful GUI is ease of use. While simplicity is the main goal with the final implementation, although the GUI certainly requires a personal touch. Hence, upon final implementation of the program, the user interface is simultaneously simple to interact with and visually stunning.

First, the user interacts with the drop-down menus for Key and Harmonic Mode to select how each setting is to function. Once selected, the drop-down menus display text to ensure the user knows which setting is selected. The user next encounters a series of knobs and sliders, organized into five columns which control the effects of the various harmonic voices to be combined with the original vocal track. For each column, the top knob controls ‘Panning’, the second controls ‘Reverb amount’, and the third controls ‘Delay amount’. Additionally, two two-dimensional arrays change the ‘Echo’ and ‘Room size’ of the harmonies. A large start/stop button indicates the current state of the program; however, all settings can be modified in real time as the program continues to run. The sliders operate in tangent with level indicator bars, which visualize the input signal for the user, going from green to yellow to orange to red as levels increase.

Many existing vocal and guitar harmonizers make use of a series of knobs, presumably because it is a simple way to ensure that the correct setting is selected while performing a concert. Many of these knobs are discrete, such as selecting which key the user will be playing in. However, Supercollider only has the option for continuous knobs. The first iteration was to make it so that each key had its own button, and only one button could be selected at a time to prevent overlapping functions to be running simultaneously. Upon the discovery of the drop-down menu, it seemed a more practical option. As with a discrete knob, it contains multiple options to choose from as the input, but only allows one to be selected.

Of course, a team with the name of Nonlinear Quasistatic Quintet cannot make anything simple, and the backsplash of the GUI reflects this mentality. The background is made of a series of five concentric rainbow pinwheels, each of which has a behavior that is linked to the position of one of the sliders. When the slider moves, it may adjust the transparency or rotational position of the corresponding pinwheel. In the case of this GUI, the flamboyant visuals enhance the user experience by making it simultaneously amusing and satisfying to adjust the various settings. It provides a unique way to visualize the levels versus a traditional harmonizer while not being intrusive to the function of the system.

Throughout the process of creating the harmonizer, hurdles were certainly encountered, but were always conquered. Upon reflection of the process, it was certainly a challenge to combine the group knowledge of music theory within Supercollider. While Supercollider certainly possesses useful built-in tools for certain applications, it also works within certain confines that at times caused delays in the programming process.

Although the program works as intended there are still useful features that may be implemented in a possible future version. Adding more exotic modes can surely help to broaden the artistic scope of the project. A mode which constantly harmonizes the input tuned to a selected pedal note could also be used for different artistic endeavors like adding a pedal bass note to add tension in a composition. Finally, an intelligent pattern recognition algorithm could be implemented to better choose how to harmonize an out of key note. The system currently implemented wants to create a chord with the least amount of out of key notes possible, but a more musically informed algorithm might make musical choices based on what has already been played using techniques like chromatic passages or voice leading.

Ultimately, including the transfixing GUI, the harmonizer works exactly as intended. The GUI functions in a simplistic manner that is easy for the user. All vocal settings function exactly as they were programmed. Just with any harmonization program, some background knowledge is required to operate the program with finesse. Therefore, the Nonlinear Quasistatic Quintet Harmonizer is the perfect harmonizing tool for any musician on a quest for the most effective, efficient, and eloquent vocal and guitar harmonizer on the internet.