Computer Music - Languages and Systems
Supercollider - Homework #1

Group 11 - BeetleJUCE:
Rebecca Superbo, Alessandro Gorni, Adriano Farina, Giuseppe Risitano, Enrico Regondi

April 2022

1 Goal

The idea of this project is that of creating an instrument based on Additive Synthesis through Supercollider which can be easily controlled using an intuitive interface.
Through the control part the user will be able to change the synthesis parameters, e.g. harmonics selected, their amplitude, detuning factor, envelope shape, and also use external MIDI devices.

2 Overview

Sounds that we usually hear everyday are not characterized by a single frequency. Instead, they consist of a sum of pure sinusoids at different frequencies, each one of them being also at different amplitudes. This combination of specific features constitutes the characteristic timbre of what we actually hear and it can be modeled as an additive synthesis. Musical instruments also rely on this powerful model, as well as software synthesizers.
For the purpose of this project, we decided to focus on all the major features of additive synthesis, while exploring the engaging field of Auditory illusion.

2.1 Shepard-Risset Glissando - a brief explanation

Everything comes to an end.
Everything, except the Shepard Tone, which may be the one thing that truly doesn't stop.

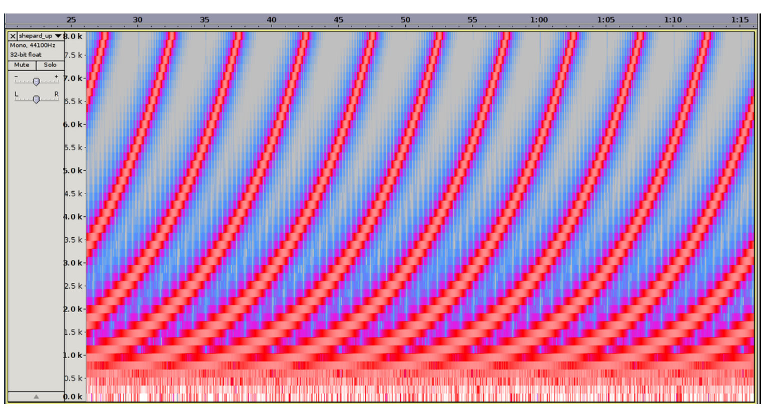

Figure 1: Visualization of the Shepard Tone

This audio illusion is not the newest among discoveries, however filmmakers like Christopher Nolan and musicians like the Pink Floyd and the Franz Ferdinand have embraced it in their work. From sound effects to the musical score, nothing pulls up tension like the Shepard Tone.

Interestingly enough, the Shepard Tone (or Shepard Scale) is named after the scientist Roger Shepard and its major traits can be summarized in three points:

1. Creates the illusion of continuous rising or falling notes or frequencies;
2. Achieved by simultaneously playing scales separated by one octave;
3. Scales are played at different volumes

Taking the sum of these points, it creates the auditory illusion of a tone that seems to continually ascend or descend in pitch, yet which ultimately gets no higher or lower.
Risset subsequently created a version of the scale where the tones slide continuously, and it is appropriately called the continuous Risset scale or Shepard–Risset glissando, the one implemented in this project. When done correctly, the tone appears to rise or fall continuously in pitch, instead it returns to its starting note.

3 Implementation

The general structure of the project can be easily divided in four main categories:

- Variables definition: in this first part, global variables have been declared. The most relevant is indeed the declaration of the SynthDef, the root of the whole project.
- MIDI setup: perhaps the simplest part, yet an useful one, the MIDI setup enables the user to connect his own external devices and explore all the features with tools even closer to the music field, recalling the central role played by the additive synthesis. When a noteOn message is received, the note number is converted to a frequency, which becomes the new base frequency. All the harmonics are then recomputed. The difficulty was that the MIDI setup happens in its own thread, so changes to the rest of the program has to happen through AppClock.
- Main window: here the GUI is declared, composed by the Main window along with a Stethoscope and a Frequency Analyzer leading to the concept of visualization of all the implemented sounds.
- GUI elements: through this part the user can directly interact with several buttons, each of them bounded to a specific feature that we decided to carry out. The simpler ones are indeed represented by the STOP and RESET buttons and we will not discuss them in much detail because their role is already crystal clear. However, the PLAY button has been implemented by taking into account all the possible values of the main parameters in which a sound can be described (detune, envelope and amplitude). In particular, we wanted to normalize the energy of a sound (i.e. its perceived loudness, aka volume), distributing it on each n harmonic in order to have 1 as a final result. So, it strictly depends on the number of activated sliders: each one of them equally contributes to the sum. In other words, each active slider is a weight for the total sum in order to get 1 as a final result of summing every single contribution of each harmonic of the signal. Eventually, we decided to implement a continuous detune effect. In spite of what the word seems to represent, detune actually "sounds good" because it more closely approximates what happens in "real world" instruments we're so used to hearing, e.g. the traditional string section, or pair of orchestral instruments, or multiple voices, double and triple-tracked guitars and so on. Concerning the implementation we opted for representing this feature through a series of knobs that with their movement express the continuity previously described. Also, the range in which each knob works is different for each harmonic, recognising that higher frequencies are perceived as to be more solid.

In the next section, we will describe in detail the most relevant components belonging to the previous bullet list.

3.1 SynthDef

A SynthDef represents an object in the server, i.e. the synthesizer, and can be seen as the engine of the whole implementation. However, we decided to keep it minimal with a fixed audio-rate sine wave at 440 Hz, corresponding to the A note, and a *mul* factor working on the amplitude of the signal. Only continuos sounds use SynthDef as their source, modifying both the frequency and the amplitude in specific ways.

3.2 GUI

Here we will provide some images of the GUI following different moments of the user experience.
Once the server has been booted and it is ready, we can compile the whole project and three windows will show up.
This first image pictures only the Main window at its default state. On the left side we have a list of buttons, each of them labelled after their specific feature, that we will discuss in a minute. The predominant part in the middle is represented by a series of sliders and knobs, each of them associated to specific harmonics.

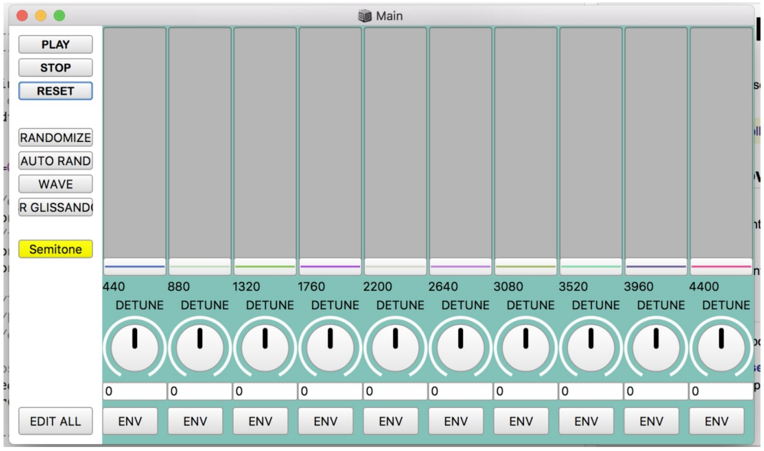

Figure 2: Default GUI

Selecting one of the available features, the slider will move up to specific positions in respect of each harmonics amplitude value. In the following image we can see that the Shepard-Risset Glissando has been selected and the sliders arranged themselves accordingly to the calculated values.

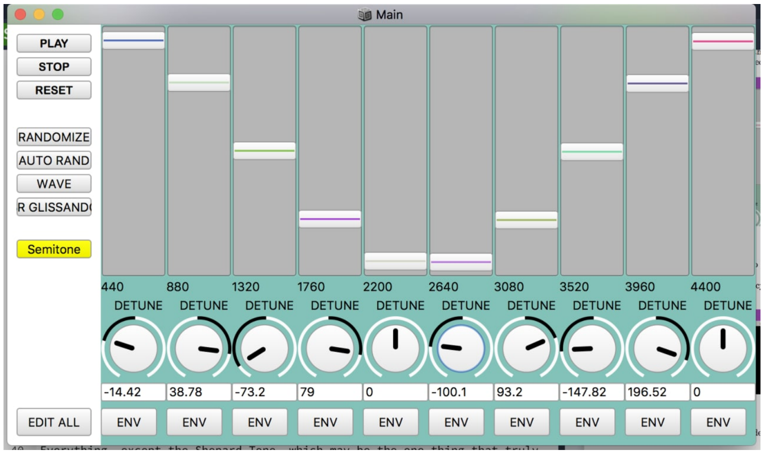

Figure 3: In action GUI

As said before, we decided to provide also some kind of visualization in order to appreciate the working application in real time.
Here we have first a Frequency Analyzer showing the frequency spectrum of the harmonics in the specified audio bus.

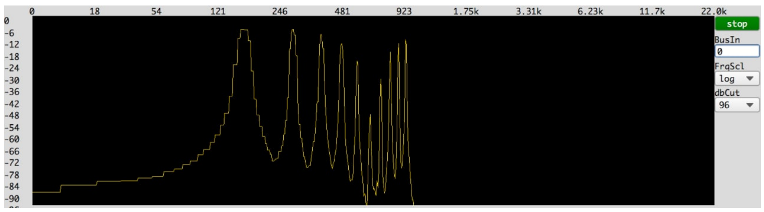

Figure 4: Frequency Analyzer

Also an oscilloscope is provided displaying a window containing a bus-plotting ScopeView and an interface to configure the plotting and choose among the buses.

Figure 5: Stethoscope

3.3 Implemented features

3.3.1 Randomize and Auto Randomize

Through the RANDOMIZE button, a randomic combination of harmonics in a single envelope is created, which could be either result in a pleasing or unpleasing hearing experience.
To make it sound more interesting, we took a further step toward randomic sounds and added the AUTO RANDOMIZE button which creates a randomic series of harmonics combinations, possibly leading to interesting combinations and outcomes.

3.3.2 Edit all envelopes

In general, we want total control over how parameters of a sound vary over time. This is can be achieved by using envelopes. Through EnvelopeView we managed to create a view in which one can graphically display nodes at x/y coordinates, connection lines, cross-connections, node markers, and labels. This view is typically used to make editable envelopes interfaces, which is our goal for this feature. The final result can be appreciated by means of the ”EDIT ALL ENVELOPES” button.

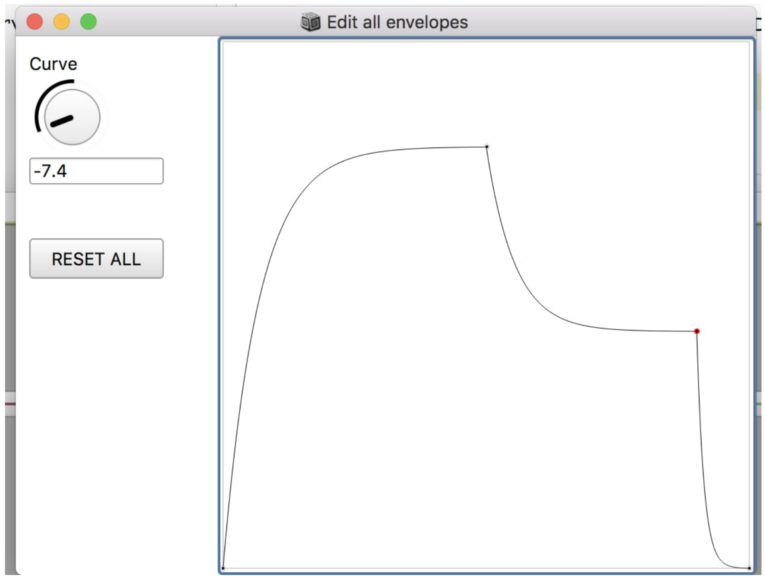

Figure 6: Edit All Envelopes interface

3.3.3 Wave

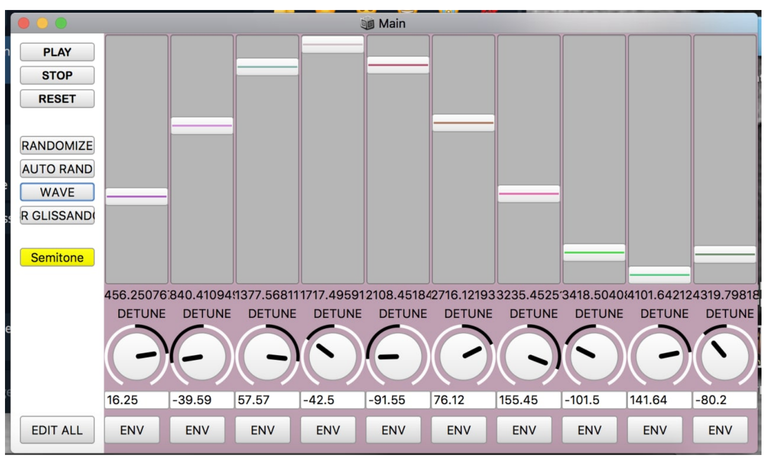

Figure 7: Wave

The wave is another automated example. Pressing the correspondent button, this feature changes the amplitude of each harmonic in time through a sinusoidal function. The final result consists in a periodical timbre modulation and the graphical effect obtained is a wave, thus the name. Regarding the implementation, the infinite cycle is managed by AppClock.

3.3.4 Shepard-Risset Glissando

In order to implement the Shepard-Rissett Glissando, we have used the known equation to get semitone-spaced frequencies scale (described below in the "Semitone" section) from a starting reference frequency. In more detail: each tone consists of two sine waves with frequencies separated by octaves whose intensity is a raised cosine function of its separation in semitones from a peak frequency.

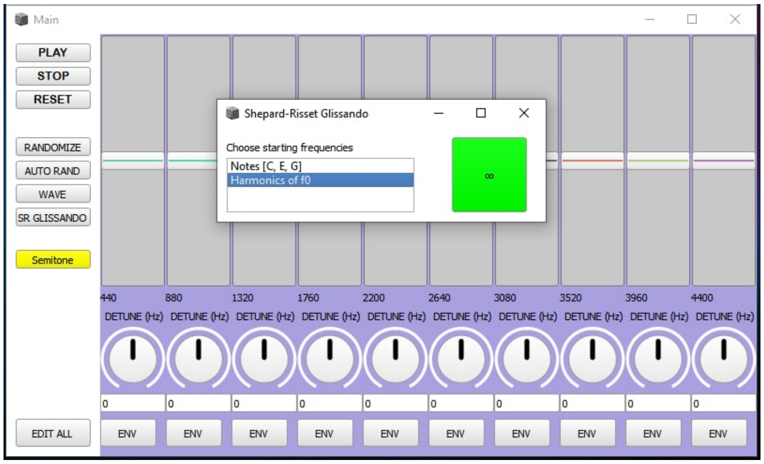

Figure 8: Shepard-Risset Glissando

3.3.5 Semitone

Let's call f_0 the reference frequency from which we aim at calculating the corresponding frequencies for each desired interval of n semitones. In order to get these values by selecting the "Semitone" button, we use the following equation:
$f_n = 2^{(n/12)} * f_0$
From an implementation point of view, we've used a for cycle with stepsize f_n.

4 Conclusions and future work

Eventually, this project proposes some of the fundamentals of Additive synthesis, together with an example of those features at work, i.e. the Wave, and an appealing one in the field of the auditory illusions. Keeping the GUI rather simple and intuitive, we believe that the final users could be encouraged in exploring all the provided features and be able to explore all the possible combinations that additive synthesis has to offer, leading to unexpected yet interesting results. Concerning future work, the GUI can be modified in order to be more user-friendly. Also more feature regarding additive synthesis can be added, as well as other kind of auditory illusions investigating through the field of sound design.