GRANULATOR 5000 – COMPUTER MUSIC LANGUAGES AND SYSTEMS- HOMERWORK 1

Juan Camilo Albarracín, Michele Murciano, Emanuele Pavese,
Camilo Trujillo, Francesco Zese

1. INTRODUCTION

Among the different ways to synthesize sound there is the *"Granular Synthesis"*; this process consists in the segmentation of an input sample into very small pieces of audio that are denominated "grains" (often between 1 and 100 milliseconds long). Said method differs from other standard synthesis approaches (additive/subtractive, FM, sampling, etc.) mainly because of the fact it generally does not take single sinusoids or noises as input, instead, it uses samples and processes them in a particular way to create new sounds.

Taking this into account, this document presents an implementation of Supercollider to develop a tool to create Foley sound effects for retro video games based on granular synthesis. The first section is devoted to describing the problem when there is no "direct sound" available and the approach followed by the group to create a useful tool in this context. Later, the implementation in Supercollider is described as well as how the different UGens interact in the programming environment for structuring the features that aid the synthesis process. The aspects regarding the creation of a graphical interface that allows users to manipulate the most important parameters and the different kinds of inputs are also discussed in the subsequent sections.

In the last part, the output of this project is presented, it consists of a granular synthesizer along with a collection of seven presets that correspond to different Foley sounds that can be loaded in the program by default. Conclusions and further developments are proposed by the end of this report.

2. PROBLEM DESCRIPTION AND OBJECTIVES

Unlike the sound production for movies, making sound for video games is a task that is usually performed exclusively in a recording studio in a post production stage, since there is no shooting that takes place, it is not possible to have recorded "direct sound". During the post production, the sound designer can create sound effects from scratch or emphasize the characters actions by doing Foley. For these last purposes, there are several tools and techniques available, such as the previously mentioned technique, granular synthesis. With that, the objectives of the project are:

- Developing a virtual instrument based on granular synthesizing that allows the user to employ diverse kinds of input signals.
- Implementing a user interface that allows mixing the different signals and to control the synth's parameters, such as density and length of grains, envelope and pitch to obtain usable custom Foley sounds for "retro" video games.
- Create a collection of Foley "retro" sounds that can be loaded by the user as presets in the program itself.

3. APPROACH

In order to acquire the perspective of the final user, documentation and references were used as means of gaining a more general comprehension of the Foley recording process as well as the granular synthesis usage. Through this exercise it was possible to identify the most promising properties of each one of these methods, as well as the common ground shared by them and on which a solution as the one suggested could find its optimal place for action. Just by having a quick glance at any Foley reference, it is possible to determine the real complexity behind the process of achieving a proper specific sound from scratch. Evidently, in order to best serve the emulation of a Foley process, the result of this project would require to incorporate features that broaden the spectrum of usable base samples (from pre-recorded files to signals generated directly in SuperCollider), just as mixing capabilities that allow the final user to synthesize both simple sound effects and complex compositions like soundscapes. The decision was made to work towards the inclusion of these features into the final product.

Brainstorming sessions were carried out in order to further grasp what kind of features a user might want to find during utilization of this tool. As a result, a first draft of the user interface was obtained.

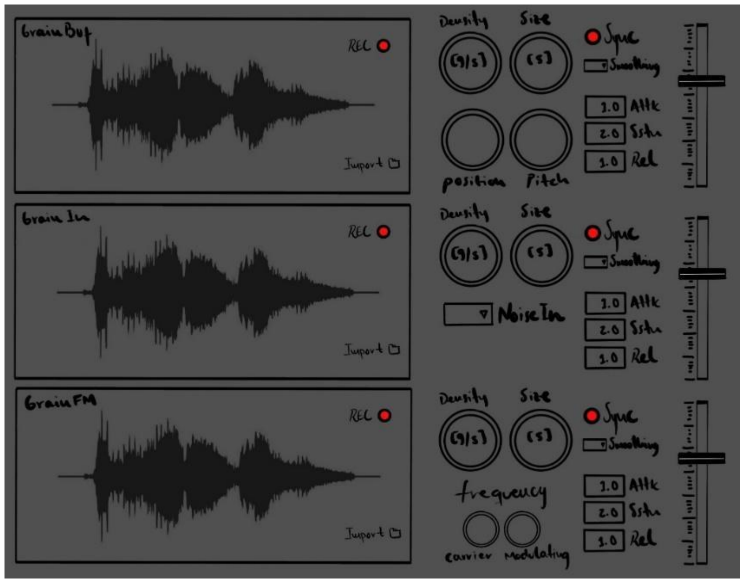


Figure 1. Granulator 5000 - First GUI draft

From the first GUI draft obtained, and aiming for the delivery of a tool featuring a simple interaction with the user but that would allow a comprehensive manipulation of the main granular synthesis parameters, it was concluded that the final interface should contain the following sections and basic controls:

- Import functionality for using pre-recorded samples
- Scoping section for placing the graphs of the manipulated signals
- Granular Synthesis basic controls such as grain density, size and position
- Editable envelope parameters for both grain smoothing and general output
- At least three different channels with individual volume controls for mixing purposes

These were selected as the base features from which the successive project outlining and implementation was generated.

4. IMPLEMENTATION

For each of the features defined at the conceptual stage, an oriented research was conducted for determining what would be the best way to structure each particular idea in the context of the code to be built and drawing upon the existent elements included within the SuperCollider libraries.

Considering that granulation is the core concept that enables an optimal synthesizer performance, special care was to be put during the selection of the modules for signal generation and processing inside the SynthDefs. Whereas allowing flexibility to the user in terms of sound composition and input type was a focal point during the implementation, three of the pre-defined UGens from the granular class library of SuperCollider were selected. The idea behind this was to remove as many constraints as possible during utilization of the synth while providing a wide range of possibilities for achieving complex sounds. Naturally, for accomplishing this purpose, it was necessary to create three different SynthDefs from which the user would have the opportunity to select and load onto each one of the synthesizer's constitutive channels.

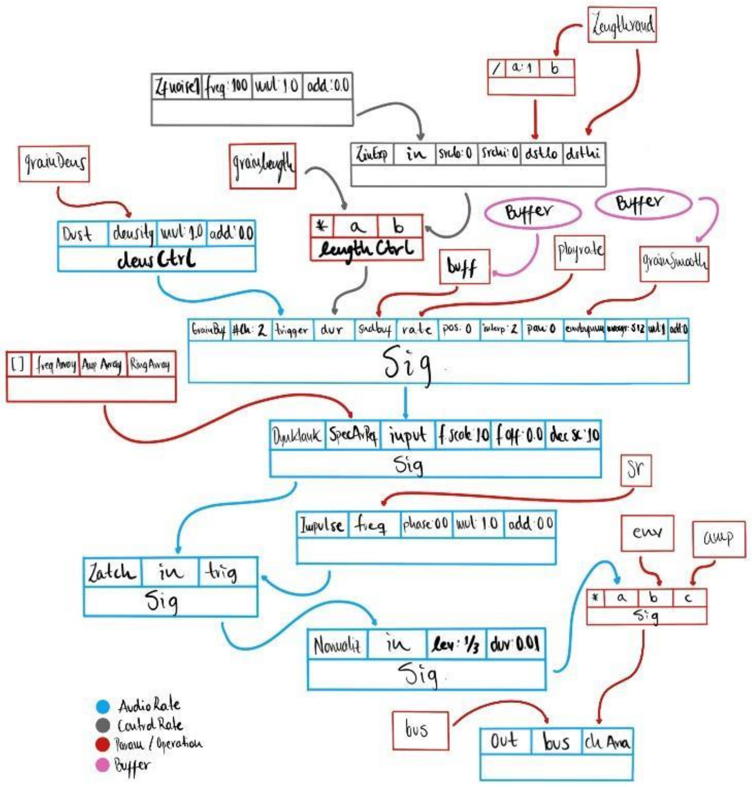
Explanation of particularities about UGens and interaction:

- GrainBufSample: As indicated by its name, this SynthDef is modeled grounding on the GrainBuf UGen included in the granular class library of SuperCollider. Selecting this synth, will allow the user to employ an external mono or stereo signal as a base sample for synthesis. Such signal is to be generated from an audio file that needs to be stored in a buffer. The following are the arguments and methods of the referenced class:

  ```
  GrainBuf.ar(numChannels: 1, trigger: 0, dur: 1, sndbuf, rate: 1,
  pos: 0, interp: 2, pan: 0, envbufnum: -1, maxGrains: 512, mul: 1,
  add: 0)
  ```

  For the specific case of the GrainBufSample SynthDef, the arguments *trigger* and *dur* (responsible for setting grain density and size) have a particular structure. As input for *trigger* GrainBufSample uses *densCtrl*, which in its turn is generated by a Dust UGen that is fed directly by the user through the parameter *grainDens*. The outlined flow introduces a randomized factor for the generation of new grains after the start of the synth which serves the purpose of enhancing the complexity of the audio output. Similarly, the variable *lengthCtrl* introduces an automated variation of the grain duration - *dur* - through the usage of a noise generator. Some other worth highlighting features of GrainBufSample are related with pitch modification capabilities (directly controlled by the user through variable *playRate*) and the possibility of choosing a custom grain smoothing envelope from a list of predefined arrays drawing upon the *grainSmooth* variable control features.

  Downstream of GrainBuf, this synth deploys a series of other UGens such as DynKlank, which allows the stimulation of particular resonant modes of the treated sample and Latch which brings the possibility of obtaining the 'Lo-fi' effect typical of the retro video games applying a process that can be understood as a downsampling of the signal: The function holds and send to the output the value of the current sample of the input signal until it receives a new trigger, to this end, an impulse signal was used as a trigger. Finally, Normalizer serves the purpose of consolidating the output audio features without distortion previous to the appearance of general amplitude and envelope customizing capabilities that are included before obtaining the definitive output. The interdependence of these modules and their parameters are shown by the UGen graph below.



LFNoise1 freq: 100 mul: 1.0 add: 0.0
/ a: 1 b
lengthround
LinExp in srclo: 0 srchi: 0 dstlo dsthi
grainDens
grainLength
Dust density mul: 1.0 add: 0.0
densCtrl
* a b
lengthCtrl
Buffer
Buffer
buff
playrate
grainSmooth
GrainBuf #Ch: 2 trigger dur sndbuf rate pos: 0 interp: 2 pan: 0 envbufnum maxGr: 512 mul: 1 add: 0
Sig
[] freqArray AmpArray RingArray
DynKlank SpecArray input fscale 1.0 foff: 0.0 decsc: 1.0
Sig
SR
Impulse freq phase 0.0 mul 1.0 add 0.0
Latch in trig
Sig
env
amp
* a b c
Sig
Normalize in lev: 1/3 dur 0.01
Sig
bus
Out bus chArray
Audio Rate
Control Rate
Param / Operation
Buffer

Figure 2. GrainBufSample UGen Graph

- GrainFMSample: Based on the GrainFM UGen from the granular class library of SuperCollider. This generator permits the application of the Granular Synthesis method over frequency modulated sine tones directly generated and processed by this module. The following are the arguments and methods of the given class:

  ```
  GrainFM.ar(numChannels: 1, trigger: 0, dur: 1, carfreq: 440, modfreq: 200, index: 1, pan: 0, envbufnum: -1, maxGrains: 512, mul: 1, add: 0)
  ```

  Analogously to the GrainBufSample, GrainFMSample emulates the proposed structure for the *trigger* and *dur* parameters within the GrainFM UGen, as well as the grain smoothing features previously explained. In addition, this synth allows the user to directly manipulate the carrier and modulating frequency of the grain generator internal oscillator through the variable *freqCtrl* and *modfreq* respectively. Similar signal treatment is carried out downstream of GrainFM.

The interdependence of the mentioned UGens is given by the graph below.

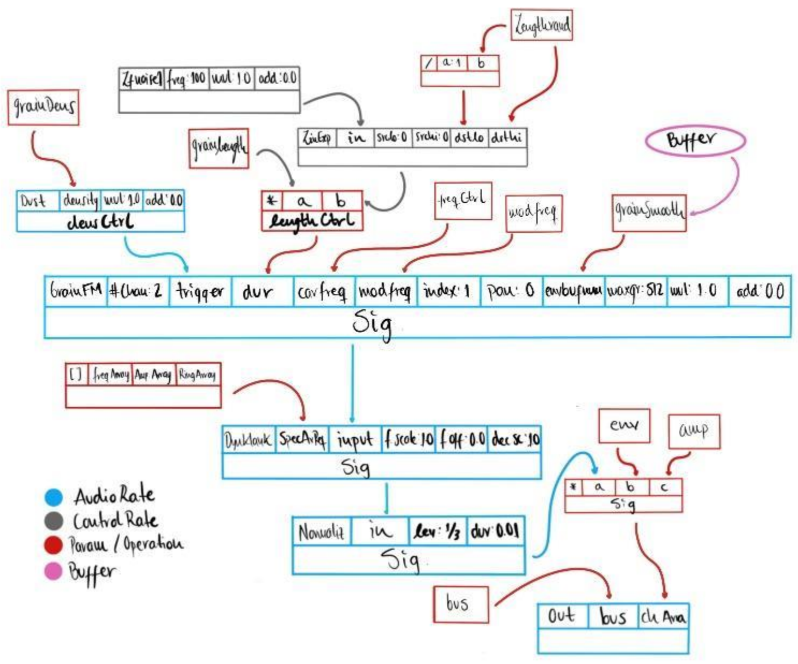

LengthRand
/ a: 1 b
LFNoise1 freq: 100 mul: 1.0 add: 0.0
grainDens
LinExp in srclo: 0 srchi: 0 dstlo dsthi
Buffer
grainLength
Dust density mul: 1.0 add: 0.0
dens Ctrl
* a b
length Ctrl
freq Ctrl
mod freq
grainSmooth
GrainFM #Chan: 2 trigger dur car freq mod freq index: 1 pan: 0 envbufnum maxgr: 512 mul: 1.0 add: 0.0
Sig
[ ] freq Array Amp Array Ring Array
DynKlank SpecArray input f scale 1.0 f off: 0.0 dec sc: 1.0
Sig
env
amp
* a b c
Sig
Audio Rate
Control Rate
Param / Operation
Buffer
Normaliz in lev: 1/3 dur: 0.01
Sig
bus
Out bus ch Arra

Figure 3. GrainFMSample UGen Graph

- GrainInSample: GrainIn is a pre-defined UGen that is also contained in the granular class library of SuperCollider. GrainIn allows the user to granulate any input signal that has been previously stored in a buffer. In this case, the SynthDef structured around this generator will allow the final user to manipulate noise signals for one-channel granulation or with the purpose of adding textures to a composition. The following are the arguments and methods of the given class:

  ```
  GrainIn.ar(numChannels: 1, trigger: 0, dur: 1, in, pan: 0,
  envbufnum: -1, maxGrains: 512, mul: 1, add: 0)
  ```

  GrainInSample features the same operation principles that GrainBufSample and GrainFMSample regarding grain density and size, as well as grain smoothing pre-defined envelopes. The main particularity encountered for the input arguments of GrainIn, is related to *in*. For this case, a Select UGen was implemented for allowing the user to choose the type of input noise to granulate from a pre-defined array using the *noiseType* variable index. Regarding the treatment of the signal downstream of GrainIn, a BLowPass UGen was implemented for attenuating the output frequencies. GrainInSample also features DynKlank, Latch and Normalizer for final output treatment, as well as custom amplitude and envelope controls, as shown in the following UGen Graph.

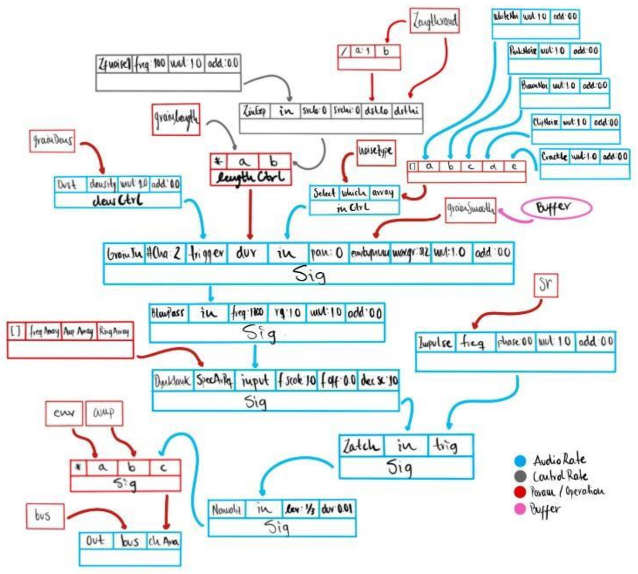
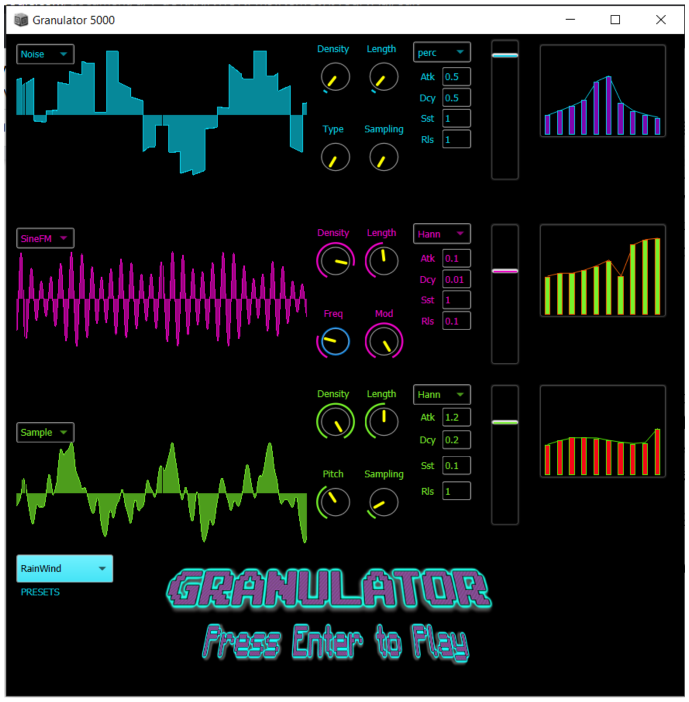

LengthRand
/ a:1 b
LFNoise1 freq:100 mul:1.0 add:0.0
LinExp in srclo:0 srchi:0 dstlo dsthi
grainLength
grainDens
* a b
length Ctrl
Dust density mul:1.0 add:0.0
dens Ctrl
noisetype
Select which array
in Ctrl
WhiteNoise mul:1.0 add:0.0
PinkNoise mul:1.0 add:0.0
BrownNoise mul:1.0 add:0.0
ClipNoise mul:1.0 add:0.0
Crackle mul:1.0 add:0.0
[] a b c d e
grainSmooth
Buffer
GrainIn #Cha 2 trigger dur in pan: 0 envbufnum maxgr: 512 mul:1.0 add: 0.0
Sig
BLowPass in freq:1200 rq:1.0 mul:1.0 add:0.0
Sig
[] freq Array Amp Array Ring Array
DynKlank SpecArray input f scale 1.0 f off: 0.0 dec sc: 1.0
Sig
Sr
Impulse freq phase:0.0 mul:1.0 add:0.0
Latch in trig
Sig
env
amp
* a b c
Sig
bus
Out bus ch Array
Normalizer in lev: 1/3 dur 0.01
Sig
● Audio Rate
● Control Rate
● Param / Operation
● Buffer

Figure 4. GrainINSample UGen Graph

5. RESULTS

General description of GUI elements:

The designed user interface responds to the objective of creating a mixing environment while giving the feel of a *Channel Strip* in which it is possible to monitor the behavior of each channel's output signal while having the possibility of making adjustments to its parameters in order to achieve the desired composition. With this in mind, the final graphic interface looks like this:



Noise
perc
Atk 0.5
Dcy 0.5
Sst 1
Rls 1

SineFM
Hann
Atk 0.1
Dcy 0.01
Sst 1
Rls 0.1

Sample
Hann
Atk 1.2
Dcy 0.2
Sst 0.1
Rls 1

RainWind

Figure 6. Final graphic interface

Density and length of the grains knobs are common for all the SynthDef . When a Sample is loaded, the parameters assigned to knobs are: Play Rate (pitch) and Sampling. For the Noise, instead of Pitch a knob for selecting the desired Type of Noise is implemented. Finally for the SineFM, the knobs control the frequency of the sinusoid and the frequency of the modulator signal. For each channel, a Multisliderview was implemented in order to control the amplitudes of the resonances set in the DynKlank function (the resonances are set to be octave bands).

The sound input visualization is done via the ScopeView which is an element that allows the integration of the oscillogram of a signal into a window. The latter is achieved by the constant exchange of data between ScopeView and ScopeOut2. While the synth is running, the ScopeOut2 Ugen is writing information into a buffer from which ScopeView reads for updating the displayed graph. In order to allow the user to monitor the behavior of each separate signal, the designed GUI includes three different ScopeView elements (one for each channel). For achieving this, a SynthDef *Scoping* had to be defined as shown in the following figure.

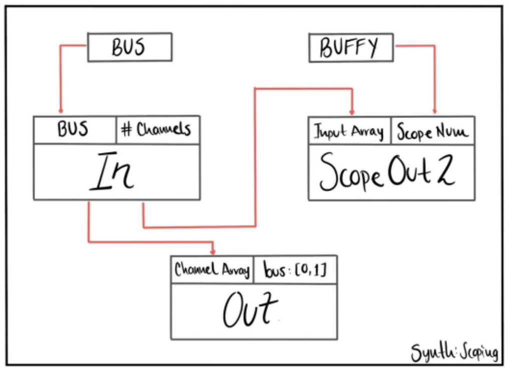


Figure 5. Scoping SynthDef flow graph

Once added to the server, it was possible to instantiate a *Scoping* Synth for each of the ScopeView elements to be included in the interface. At this stage, it's worth mentioning that it was necessary to invoke the *.defer* method for delaying the evaluation of the code included in the function, intending to make time for the instantiation of the SynthDef on the server. The featured solution becomes useful in applications with more than one ScopeView object that are intended to display signals coming from different Synths, while the output of each of them needs to be sent to the main channels [0,1].

Foley sounds

As mentioned in previous sections, one of the set project's targets was creating a collection of sounds that gives the feeling of "retro" using the different sound sources and possibilities inside the developed instrument, while the instrument also allows the user to freely play with the synthesizer to create its own sounds. The collection can be found in a series of presets that can be called into the program. The list of preset and a brief description is the following

- FootSteps: Consist in a short sound characterized by a low-mid frequency content generate using a noise type crackle in the input. The length of the grains is no does not exceed the 0.06 seconds and the density is set to 88.25 grains per second
- Cricket: It is also a short sound, but in this case is characterized by the high frequency content and higher density of grains (325 grains per second) and again uses a crackle noise as input. The signal is downsampled to around 4300.

- Spooky: This soundscape consists in a sine of 356 Hz modulated in frequency by another sinusoid of 13 Hz. In this case the decay (6s), sustain (5s) and release (4s) times are longer than the attack (0.5s).
- CreepGlass: Foley characterized by a short attack time (0.01s) and the percussive envelope of the grains. This effect uses a voice sample as input, the play rate (pitch) is set to 1.25 while the sampling is set to 7655.
- Explosion: Again based in a sineFM and a voice sample, where the sampling is set to 4664. The 80 Hz of the sine is modulated by 20 Hz. For both signals the density and length are set to the maximums.
- RainWind: This soundscape uses the 3 channels. The white noise has the sampling at its maximum and the density and length to the minimum, while the play rate of the sample (the song) is set to 0.63. The percussive envelope to the grains in the noise channel helps to achieve the sound of the water tears and the hanning to the other two benefit smoother sounds to the wind and background.
- LaserGun: This foley is achieved by emphasizing the high frequencies using the multislider in the sample. Another main characteristic is the modulated sinusoid that gives the feel of a "moving away" sound. Overall, the attack and decay times are way shorter than the sustain and release ones.

6. CONCLUSIONS

- SuperCollider, as a programming environment, offers a considerable amount of resources that enable the outlining of a wide range of software solutions in audio applications with relative ease. For the purpose of obtaining the best possible result from the development process, without drifting apart from the optimal progress track, it is convenient to kick off the implementation phase of the project solely once the conceptual stage has been successfully completed and a clear idea has been attained as a general goal for the team.
- It was achieved to create a virtual instrument that has an interface that allows users to load, mix and change the properties of different signals using granular synthesis and its different parameters, such as grain and length density.
- Through the use of the virtual instrument, it was possible to create a collection of Foley sounds useful in the audiovisual context, specially for "retro" video games sound design.
- The incorporation of the control of the sampling for the sample and noise inputs success to bring the instrument the 'Lo-Fi feel' characteristic sound of the 'retro' video games.

7. FURTHER DEVELOPMENTS

- Regarding the implemented features, it would be interesting to expand the input capabilities of the program by including a recording function to allow the user the granulation treatment of a sample directly captured by the software.
- Another future implementation could be the inclusion of an interface that permits to generate synchronized events in time code (Hours:Minutes:Seconds:Frames) since this is a tool meant to be used in an audiovisual context.
- In case of further developing the presented program, lowering the processing cost is suggested. Achieving a more efficient use of memory and processing power could result in the possibility of introducing new features to the program that would ultimately enrich the synthesis process.