Politecnico di Milano (2023), 1–7

HOMEWORK #1 REPORT

Granny's Kitchen

Enrico Dalla Mora, Rocco Scarano, Nicolò Chillè, and Federico Caroppo

Group Name: ***La Lobby*** **- Group ID: 3**

Abstract

The purpose of this document is to describe the process of creating *foley sounds* by means of granular synthesis in the `SuperCollider` environment and implementing an integrated graphical user interface in order allow, to some extent, the customization of the sound parameters by the user. Given that foley sounds are usually employed in audiovisual content as a mean to enrich the soundscape of a scene, the sounds analyzed herein were developed to mimic the sounds that are usually perceived in a specific environment.

Keywords: granular, synthesis, foley

The synthesis process started with the selection of the environment to which the synthesized sounds would belong. The choice fell on an ordinary life soundscape: the one of a ***kitchen***. Through granular synthesis, the sounds associated to the following objects were recreated:

- a **gas stove** being turned on - ①
- a **knife** cutting something on a cutting board - ②
- two **glasses** clashing - ③
- **boiling water** - ④

Even though granular synthesis is, to an extent, affected by the micro-sampled track, the choice of said track was solely based on the audio context and fell on "*Le Festin*" by Camille and Michael Giacchino, a song from the soundtrack of the hugely popular animated movie "*Ratatouille*" (2007), which is set in a kitchen for the most part.

The first section of this report will focus on the approach adopted to achieve the synthesis of the aforementioned sounds in the SuperCollider coding environment.

1. Granular Synthesis and `SuperCollider`

The sound synthesis method known as **granular synthesis** is based on the production of a high density of small acoustic events (the so-called 'grains') that are typically under 100 *ms* in duration, obtained via the *microsampling* of a given audio signal. Adding a large number of grains together and organizing them will result in the synthesis of a texture, therefore generating a sound from a macroscopic standpoint.

SuperCollider (which will be referred to, from now on, as SC) provides a UGen, called `GrainBuf`, that implements granular synthesis starting from the audio signal stored in a buffer (∼`b1`). The class is structured as follows:

Code 1. The `GrainBuf` 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);
```

The most significant parameters, i.e. the ones that were involved in the development of the sounds, are:

- `trigger`: the **trigger** to start a new grain (either at audio rate or at control rate). Two kinds of trigger were used, namely:
  - the ***impulse oscillator*** `Impulse`, which is more suited for impulsive audio events such as sounds ② and ③;
  - the ***random impulses generator*** `Dust`, more fitting for rough, noisy sounds such as ① and ④.
- `dur`: indicates the temporal **size** of the grain (in *seconds*);
- `rate`: the **playback rate** of the sampled sound, affecting pitch and timbre;
- `pos`: the **position** for the grain to start with, relative to the input sound (range [0, 1] - where 0 represents the beginning of the sample and 1 corresponds with its end).

The sound synthesized by each `GrainBuf` object class was further shaped by an **envelope** and, possibly, by effects (e.g. *reverb* or *resonance*). In particular, the purpose of the reverb is to enhance the spatial details linked to the perception of the sounds in a real room.

2. Implementation

In the present section, each one of the foley sounds will be discussed individually and in detail. Ath the end of each dissertation, the corresponding UGen graph will be reported.

2.1 Gas Stove ①

The objective was to reproduce the complex sound of a gas stove being turned on, from the gas flow rustling to the flame rumble, including the regular pulsing of the spark module. Thus, the synthesized signal is structured in four superimposed components, namely the gas flow, the ticking of the sparks, the burst of the lighting flame and the flames rumble. The transition between said components is managed via a number of envelopes.

2.1.1 Signal 1 - Gas flow

The first component of the final signal corresponds to the *rustling of the gas flow*. Being this a **noise-like sound**, the trigger signal was generated using the `Dust` UGen. The average number of impulses (density) given as input is very high, and thus the duration of the grains must be low to avoid problems linked to overlapping. As said before, the `rate` parameter affects the perceived pitch. *No envelope* was used to further shape the sound.

Code 2. The gas flow signal.

```
sig1 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(1469.89), dur: 0.04912,
     sndbuf: ~b1, rate: 1.89, pos: 0.00000, interp: 2, pan: 0, envbufnum: -1,
     maxGrains: 512);
```

2.1.2 Signal 2 - Spark module

The base signal is still noise-like, but it is characterized by a much higher density of grains in order to achieve a *richer high-frequency spectrum*. The pulsing effect is achieved via an **impulse-shaped periodic envelope** (Figure 1) applied to the signal amplitude.

Code 3. The ticking sparks signal and envelope.

```
sig2 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(4000), dur: 0.00000, sndbuf:
     ~b1, rate: 1, pos: 0.28855, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
env2 =  EnvGen.ar(Env([0, 1, 0], [0.005, 0.015], curve: 0), Impulse.kr(2));
```

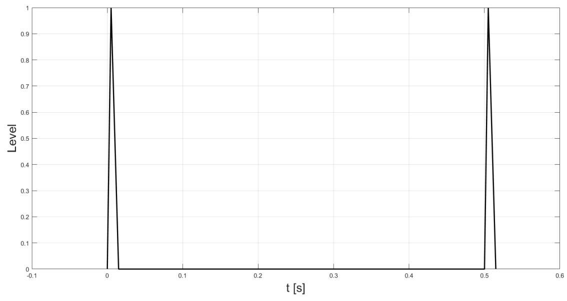

Figure 1. Spark ticking amplitude envelope.

2.1.3 Signal 3 - Flame rumble

The third signal represents the flame rumble that follows the flame lighting burst (i.e. signal 4). The peculiarity of this sound is the way the transition between the spark ticking and the ramble is handled: the envelope named `env3` (Figure 2-a) is a `LFPulse` characterized by a very low frequency and a long duty cycle, and it manages the ignition of the flame in correspondence with the last spark, while `env31` is an inverted square envelope (refer to Code 4) with respect to `env3` and handles the deactivation of the spark ticking after the igniting one. The duty cycle and the initial phase are slightly different from the `env3` ones in order to allow the last spark (the igniting one) to be heard.

Code 4. The flame rumble signal and envelopes.

```
sig3 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(388), dur: 0.00391, sndbuf:
     ~b1, rate: 0.29, pos: 0.16565, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
env3 = LFPulse.ar(0.08, 0.8, 0.8);
env31= 1 - LFPulse.ar(0.08, 0.79, 0.79);
```

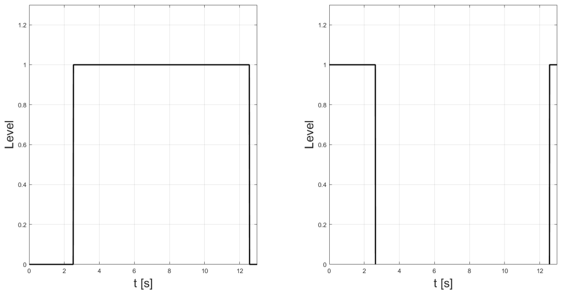

Figure 2. `env3` and `env31` amplitude envelopes.

2.1.4 *Signal 4 - Ignition burst*

Code 5. The flame burst signal and envelope.

```
sig4=GrainBuf.ar(numChannels: 2, trigger: Dust.ar(388), dur: 0.01291, sndbuf: ~b1
     , rate: 0.30, pos: 0.16565, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
env4=EnvGen.ar(Env([1, 0], [0.7], curve: 0), env3);
```

The last signal is also a rough sound, and thus employs `Dust` as a trigger signal generator, but its input density is much lower to the flame rumble one. This is due to the necessity of a lower pitch to mimic the fire blaze. The audio event must be impulsive, as the overall signal transitions into the lighted stove rumble: this effect is achieved by means of a single decaying envelope (for code reference, see Code 5 and Figure 3).

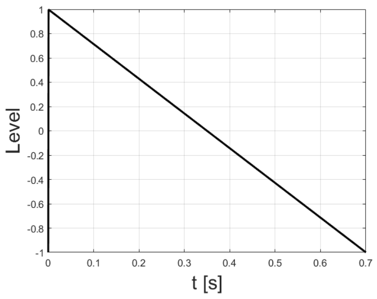

Figure 3. Blaze amplitude envelope.

2.1.5 *Final sound - Igniting stove*

To generate the complete sound, the four signals are amplified (via four amplitude factors), modulated by means of the dedicated envelopes and added together to be run through a simple reverb and sent out as a stereo output.

Code 6. The flame rumble signal and envelopes.

```
sig = (sig1*amp1) + (sig2*amp2*env2*env31) + (sig3*amp3*env3) + (sig4*amp4*env4*
     env3);
Out.ar(0, FreeVerb.ar(sig, 0.2, 0.3, 1)*amp);
```

2.2 *Cutting Knife* ②

The sound in question is comprised of *two components* (signals): the knife cutting through an hypothetical vegetable - `sig1` - and the dull thud of the knife hitting the cutting board - `sig2`. Each of this sound components if of **periodic and impulsive character**, thus the chosen trigger signal generator is `Impulse`. The trigger of the board thud is, obviously, slightly delayed with respect to the one relative to the first signal. In addition, the duration of `sig2` is really small in order to avoid overlapping.

The overall signal is then amplified and shaped through a square-shaped envelope, before being sent in output through a reverb.

Code 7. The cutting knife signals and envelope.

```
sig1 = GrainBuf.ar(numChannels: 2, trigger: Impulse.ar(1), dur: 0.4, sndbuf: ~b1,
      rate: 0.5, pos: 0.27, interp: 2, pan: 0, envbufnum: -1, maxGrains: 512);
sig2 = GrainBuf.ar(numChannels: 2, trigger: Impulse.ar(1, 0.70), dur: 0.02524,
     sndbuf: ~b1, rate: 0.23, pos: 0.95064, interp: 2, pan: 0, envbufnum: -1,
     maxGrains: 512);
sig = (sig1*amp1) + (sig2*amp2);
env = EnvGen.ar(Env([1, 0], [4], curve: [500]), 1, 1, 0, 1, 2);
```

2.3 *Glasses clashing* ③

The glass clinking effect was achieved by ***feeding a single, very short grain to a resonator*** UGen: `DynKlank`. The purpose of said UGen is to simulate the resonant modes of an object: each mode is given a ring time (corresponding to the time necessary for the mode to decay by 60 dB). The signal's amplitude is shaped through two envelopes, one applied before sending it to the resonator (a square-shaped one) and one after (decaying hyperbolic) - see Figure 4 for reference - and is then fed to a reverb.

Code 8. The clashing glasses signal and envelopes.

```
sig = GrainBuf.ar(numChannels: 2, trigger: Impulse.ar(319.09), dur: 0.09134,
     sndbuf: ~b1, rate: 1.61, pos: 0.17028, interp: 2, pan: 0, envbufnum: -1,
     maxGrains: 512);
env1 =  EnvGen.ar(Env([1, 0], [0.03], curve: [500]), 1, 1, 0, 1, 0);
sig = DynKlank.ar([[800*1.2, 2684*1.2, 4612*1.2, 6892*1.2], [1, 0.5, 0.23,
     0.125], [5, 4, 3, 2]], sig*env1*0.1);
env =  EnvGen.ar(Env([1, 0], [6], curve: [-5]), 1, 1, 0, 1, 2);
```

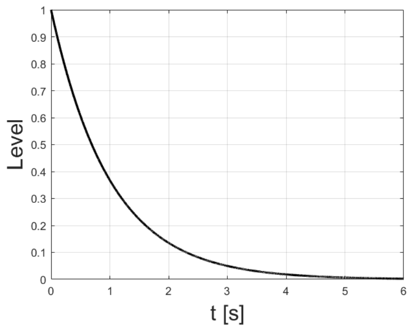

Figure 4. Glass amplitude envelope.

2.4 Boiling water (4)

The idea behind the boiling water sound is to smooth a superimposition of four different rough, noisy signals. For each one of them, a `Dust` trigger signal generator was employed (using various input densities) and a **very small duration** of the grains was set.

Code 9. The clashing glasses signal and envelopes.

```
sig1 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(20), dur: 0.01751, sndbuf:
     ~b1, rate: 1, pos: 0.06094, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
sig2 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(10), dur: 0.01751, sndbuf:
     ~b3, rate: 2, pos: 0.16094, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
sig3 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(30), dur: 0.01751, sndbuf:
     ~b3, rate: 3, pos: 0.26094, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);
sig4 = GrainBuf.ar(numChannels: 2, trigger: Dust.ar(855), dur: 0.01234, sndbuf:
     ~b3, rate: 9.61, pos: 0.31664, interp: 2, pan: 0, envbufnum: -1, maxGrains:
     512);

sig = (sig1*amp1) + (sig2*amp2) + (sig3*amp3) + (sig4*amp4);
env =  EnvGen.ar(Env([1, 0], [5], curve: [500]), 1, 1, 0, 1, 2);
```

3. Graphical User Interface (GUI)

The GUI that has been designed for the assignment is shown in Figure (...). In the top-left corner of the window was placed a button (1) whose purpose is to reproduce the source track ("Le Festin", by Camille and Michael Giacchino). The rest of GUI is divided in four vertical sections, one for each of the synthesized sounds, and designed to resemble a kitchen top - an oven, specifically. The "Play" starts the selected sound playback, while the sliders allow the user to control the volume and the panning of the played sound. The square corresponding to the oven door is the plot of the stereo sound (left and right channel) either in the time domain or in the frequency domain. Button (7) is deputed to switch the domain of representation of the sound. The two rows of knobs control the sound parameters customizable by the user: the first row controls the granular synthesis parameters, while the second contains the reverb controls.

4. Conclusions

What became clear during the process is that only by means of a deeper understanding of granular synthesis, achievable only through experience and time, the sounds generated with this technique could have been more accurate. Understanding how to modify the granulator parameters, during the first phases of the work, in order to cause certain effects on the sound features, was not at all easy. It was therefore difficult to choose how many and which sounds it was possible to reproduce, and which parameters could be customized by the user without dramatically changing the character of the sound.

Nevertheless, the authors feel confident about the results, both in terms of the quality of the sounds and the both for the usability of the application.