Politecnico di Milano

Computer Music: Languages and Systems

A.Y. 2022/2023

HW1 – Assignment #4

GRANULAR SYNTHESIS FOR FOLEY SOUNDS

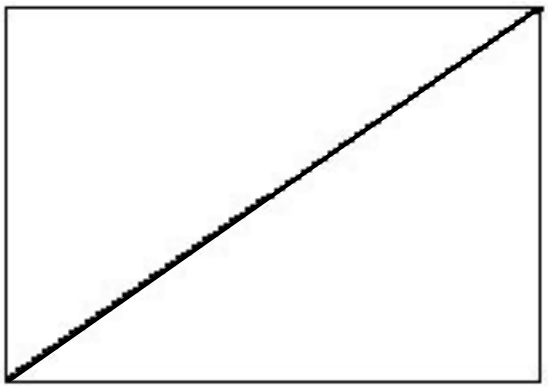
Emma Coletta 214391

Federico Ferreri 221557

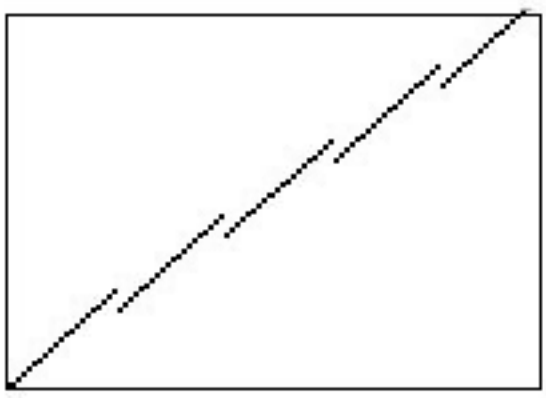
Lorenzo Previati 222058

- ## Granular Synthesis

Granular synthesis is a sound manipulation technique that uses audio fragments to generate new sounds and pattern. These fragments are called *grains* and can last from 1 up to 100 ms. The rapid succession of the grains rebuilds the original sample, but it's also possible to control some parameters, without changing the length of the output signal.



Original Sample



Sample after granular synthesis

Multiple grains can be layered on top of each other and may play at different speeds, phases, volume and other parameters, resulting in the generation of a completely new sound.

Each grain is governed by an envelope that allows to control the temporal overlap of grains, with the aim of reconstructing the original sample in the most adherent way.
In general, longer grains tend to maintain the timbral identity of the original sound and shorter grains tend to be impulsive.

- ## Supercollider implementation

Steps:

1. Assigning each sample to a buffer;
2. Synth definition with parameters enriched by components of randomness;
3. Writing a GUI able to modify, through graphical interface, the synth parameters and to activate the playback of each buffer.

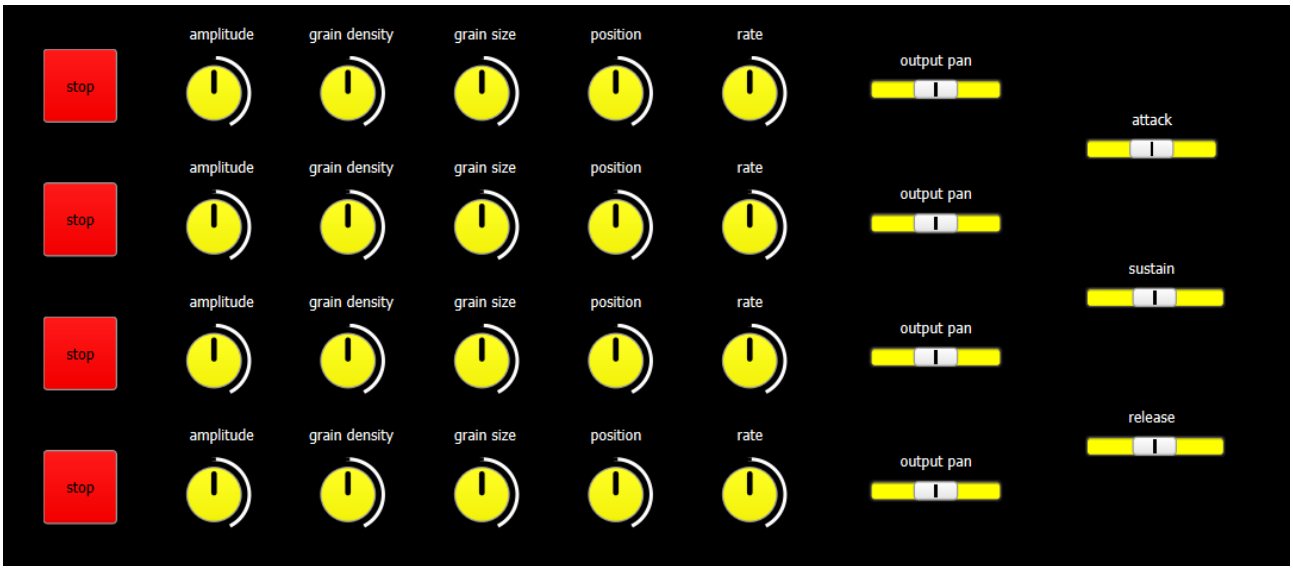
The implementation of the code for the granular synthesis passes through the use of the GrainBuf function, which requires the allocation of samples in buffers, associating to it a useful number for the passage of the buffer in the Synth.

Allocated the buffers, we have therefore defined a Synth through the SynthDef function that contains the granular synthesis function. The latter operates according to the definition allof the following parameters:

- **numChannels**: number of channels to output → numChannels=2 (audio pan is possible)
- **trigger**: trigger to start a new grain. Allows to control grain density
- **dur**: size of the grain [s]
- **sndbuf**: number of buffer holding the audio signal
- **rate**: playback rate of the sample sound
- **pos**: playback position for the grain to start
- **interp**: 2 (linear interpolation)
- **pan**: where to pan the output
- **envbufnum**: -1 (Hann window for grains envelope)
- **maxGrains**: maximum number of overlapping grains

An envelope is also defined for the output signal and attack, sustain and release values are user editable.

A GUI is created for this purpose, showing the following user interface:



Editable parameters:

- Amplitude: controls sound intensity;
- Grain density
- Grain size: decides how much each grain lasts;
- Playback position: controls the playback starting point of the audio file;
- Rate: playback rate of the samples sound;
- Pan: where we're panning the playing output;
- Output envelope time values (attack, sustain and release).

The code is implemented so that the user can modify the granular synthesis parameters for each buffer in the SynthDef. It's also possible to activate and deactivate the playback of each Synth.

All Synths are initialized at the same parameters values.

▪ Conclusions

We used granular synthesis to create a modular ambient sound, in which each buffer is independently defined through the parameters described above.

Real-time modulation of the parameters allowed us to create a complex ambient sound.

The GrainBuf function was our choice because, compared to the other functions suitable for granular analysis, it allowed us to modify many more parameters. We have also introduced components that can create unpredictability in the reproduction of grains making the output sound more like a real human perception of sound.

We also added a phasor that can run through the grain buffer.

▪ Reference functions

Buffer allocation

Buffer.readChannel(server, path, startFrame: 0, numFrames: -1, channels, action, bufnum)

Granular Synthesis

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)

Envelope generator

EnvGen.ar(envelope, gate: 1.0, levelScale: 1.0, levelBias: 0.0, timeScale: 1.0, doneAction: 0)

Phasor

Phasor.ar(trig: 0.0, rate: 1.0, start: 0.0, end: 1.0, resetPos: 0.0)