Computer Music – Languages and Systems

HW1

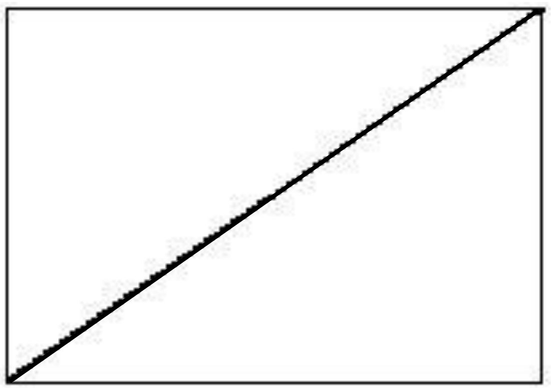
Assignment #4

GRANULAR SYNTHESIS FOR FOLEY SOUNDS

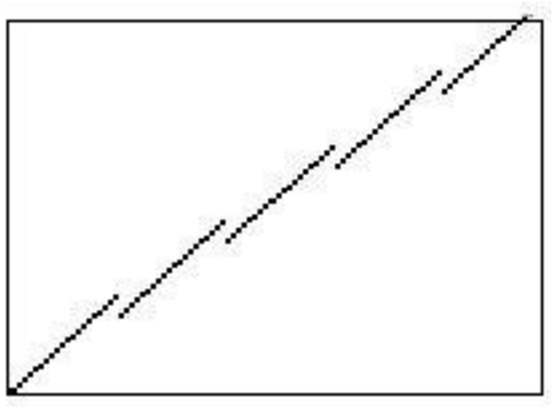
Emma Coletta	214391
Federico Ferreri	221557
Lorenzo Previati	222058

• Granular Synthesis

Granular synthesis uses samples fragments to generate sounds. These fragments are called grains and can least from 1 to 100 ms. The rapid succession of the grains rebuilds the original sample, but it's possible to control some parameters without changing the length of the output signal.



Original Sample



Sample after granular synthesis

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

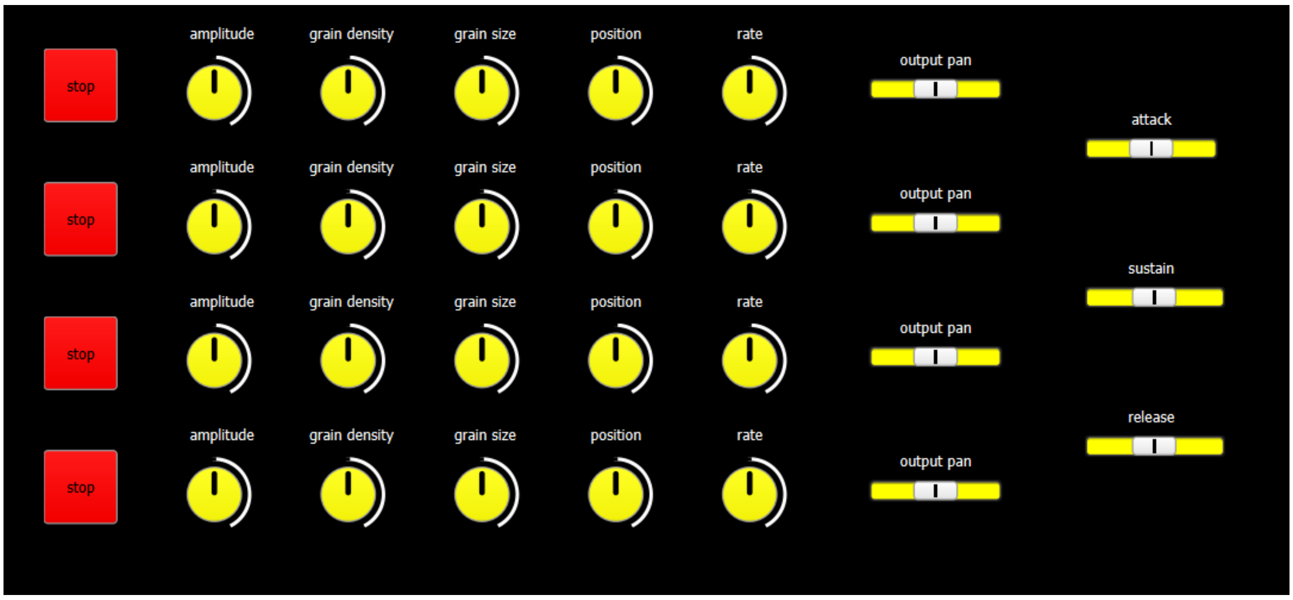
The implementation of the code for the granular synthesis passes through the use of the GrainBuf function, which requires the allocation of samples in buffers, associating to it a useful number for the passage of the buffer in the Synth.

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 of the following parameters:

- numChannels: number of channels to output. → numchannels=2 (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
- 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 times are user editable.

A GUI is created for this purpose:



Editable parameters:

- Amplitude
- Grain density
- Grain size
- Playback position
- Rate
- Pan
- Output envelope times

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

- ## 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 whole more like a real human perception of sound.

A phasor has been introduced 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)
```