Spatial Granular Synthesis With Ambitools and Antescollider

Pierre Lecomte, José Miguel Fernandez

SPATIAL GRANULAR SYNTHESIS WITH AMBITOOLS AND ANTESCOLLIDER

Pierre Lecomte [*]

Ecole Centrale de Lyon, CNRS, Universite Claude Bernard Lyon 1, INSA Lyon, LMFA, UMR5509, 69130, Ecully, France
Lyon, France
pierre.lecomte@ec-lyon.fr

José Miguel Fernandez [†]

IRCAM, STMS, UMR9912, 75004, Paris, France
jose.miguel.fernandez@ircam.fr

ABSTRACT

This paper presents a spatial granular synthesis tool developed in FAUST and part of the Ambitools v1.3 library. This tool generates a swarm of spatialized sound grains in a spherical shell sector using the Higher Order Ambisonic format. The grains can be played from pre-recorded sound files or from a circular buffer of the input signal for live use. This tool is then integrated into the AntesCollider library to offer fine control over the evolution of the grain swarm as well as its visualization.

1. INTRODUCTION

Granular synthesis, originally based on the pioneering work of Gabor [1] and Xenakis [2], is a sound synthesis technique that breaks down an audio signal into small segments called "grains". Building on the general concepts developed by Truax [3] and Roads [4], granular synthesis has been extensively used by composers and musicians to create new sounds from pre-recorded audio (whether concrete or synthesized). This technique allows for the creation of complex sounds, rich textures, and evolving sounds by manipulating various parameters such as grain duration, overlap time, envelope type, and playback position within the audio file. The use of granular synthesis in spatial audio, or granular spatialization, is a natural evolution of this concept. Beyond its original capabilities, it enables the creation of diverse spatial sound morphologies. For instance, one can easily generate sound masses that move not only in timbre but also in space, such as a sound point exploding into the surrounding space, multiple sounds transforming and converging towards a specific spatial point, or using random positions to generate an immersive spatial sound. Although spatial granulation is not a new concept [5, 6], the implementation we propose, based on the Ambitools library [7], allows for the generation of many grains in a Higher Order Ambisonics (HOA) format of any order. This approach enables the dynamic creation of various granular synthesis modules in real-time from an audio file or live input. The spatial granular synthesizer being written in FAUST, many plug-ins formats are available which can be integrated in multiple environments. We present here an integration of this tool in AntesCollider [8], which provides an interface for precise control over the evolution of a swarm of grains within the 3D spatial environment. AntesCollider is a library dedicated to electronic music composition and temporal sequencing, enabling, among other things, synchronization with a musician through score follower or gesture follower. It also allows for 3D visualizations through the integration of the OpenFrameworks library. The paper is organized as follows: Technical details on the FAUST implementation of the spatial granular synthesis engine is given in Sec. 2. Then, the integration to AntesCollider is presented in Sec. 3. Conclusions and future works are given in Sec. 4.

[*] https://sekisushai.net/ambitools

[†] https://josemiguel-fernandez.com/

2. THE GRANULATOR

The spatial granulation tool is written in the FAUST language [9]. It is part of the ambitools plug-in suite [7] v1.3 as the "Granulator" (`granulator.dsp`) [1]. The tool generates N^2 parallel signal streams of sound "grains", spatialize each grain individually in a swarm whose geometry define a spherical shell sector. An example of Graphical User Inteface (GUI) for the Granulator is shown in Fig. 1.

This section detail the construction of a stream of grains, present extra parameters for each grain and their spatialization. A discussion of the FAUST compilation time is carried out.

2.1. Grain Stream Generation

Each of the N streams of grains is a signal made of concatenated grains: once a grain is played, a new one is generated and so on. The grains are read from a buffer which can be either a sound file (by using the `soundfile` primitive), or a circular buffer fed with an input signal (by using the `rwtable` primitive). The user can choose which sound source to use at runtime. For both cases, a "read index" signal, which gives the samples index over time, is needed and constructed according the following grain parameters:

- duration,
- reading speed,
- starting index in the buffer,
- reading direction: forward or reverse.

An example of read index signal as well as the duration, starting sample and reading speed parameters for a grain stream is shown in Fig. 2

For each grain of each stream, the parameters are set randomly using decorrelated random noise signals using `no.rnoises` primitive [3]. The random signals are scaled and shifted according to the parameters ranges which can are tuned using sliders in the User Interface (see Fig. 1). They are fed into a sample and hold function. The latter is denoted `trig` and a block diagram is shown in Fig. 3. The `trigger` signal in Fig. 3 is an impulse which is

[1] https://sekisushai.net/ambitools/docs/granulator.html

[2] $N \in \mathbb{N}$ is set at compilation time.

[3] Note that we don't use of `no.noises` primitive here as it would produce the same grains sequences at each plugin initialization because the noises generator seeds are the same.

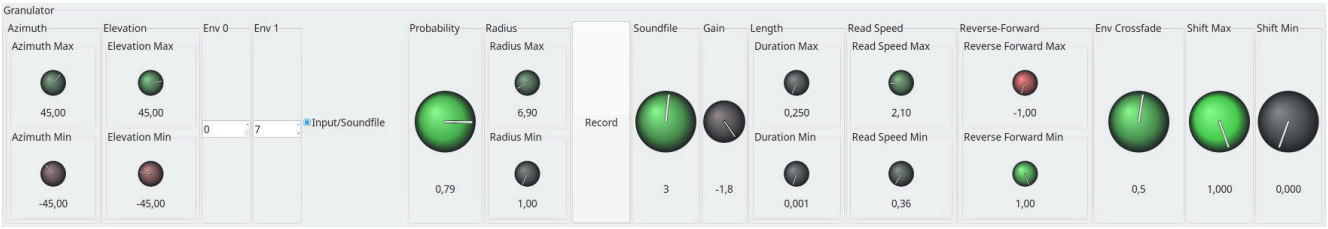

Figure 1: *GUI of the Granulator compiled with* `faust2jack` *script.*

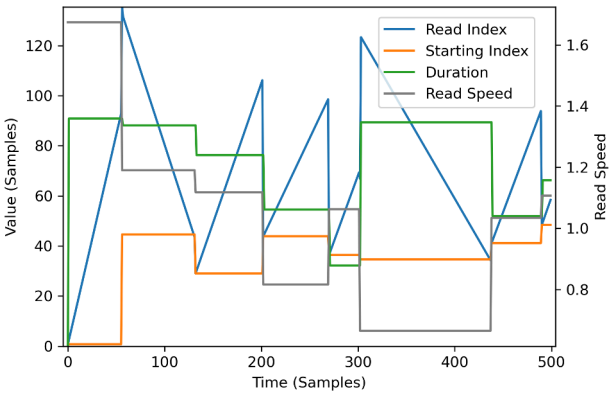

Figure 2: *The read index signal for a stream of grains (in blue): the slope is proportional to the read speed. Its sign gives the read direction. Once the read index minus the starting sample equals the duration, a new set of parameters is randomly chosen within the parameter ranges.*

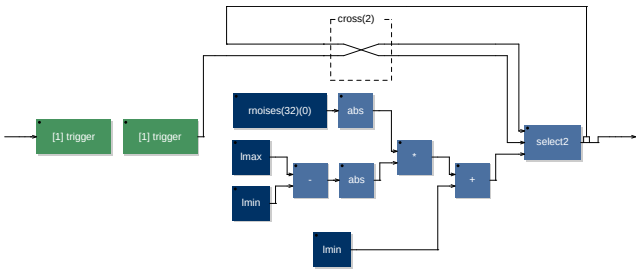

Figure 3: *Block diagram of the* `trig` *function, i.e., a sample and hold function triggered by a impulse signal. Here the 0-th noise among 32 noises signals (*`rnoises(32)(0)`*) is scaled and shifted to give a random signal between* `lmin` *and* `lmax`*. A value of this signal is hold until an impulse is received (*`trigger`*).*

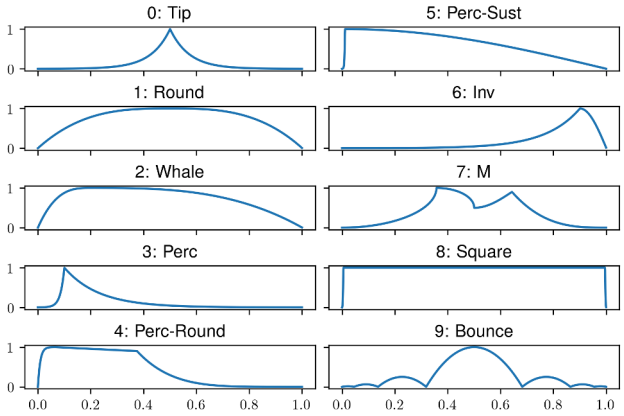

Figure 4: *The various envelopes used on a grain stream. A cross-fade allows interpolation between two envelopes among this bank at runtime.*

non-zero only when the read index value minus the starting index equals the grain duration. At this instant the `trig` function of Fig. 3 samples the current value of the random signal and hold it until the next impulse. Thus, a new set of grain parameters is randomly set at the end of each grain.

2.2. Extra Parameters

Grain Envelopes Since the reading index signal exhibits discontinuities (see Fig. 3), audible clicks may occur at each grain change. To prevent this phenomenon, but also to provide control over grain dynamics, an envelope starting and ending at zero is applied to each grain. The same envelope is used in the N streams. This envelope is constructed at runtime with a cross-fade between two envelopes taken from a bank. The envelopes bank can be seen in Fig. 4.

Grain Probability To control the density of grains played simultaneously in the N streams, a `Probability` slider (see Fig. 1) is used (between 0 and 100%). The `trig` function of Fig. 3 is used with `lmin = 0` and `lmax = 1` and this value is compared with the `Probability` slider value. If the value is higher, the gain grain is set to 0 and the grain is not played.

Markers The starting index in Fig. 3 is by default chosen randomly in the buffer. However, in the case of sound file as the sound source, it is possible to use "markers" given as a list of samples index (using the `waveform` primitive). In this case, the starting

index of each grain is chosen randomly among these markers. This feature helps the composer to select time instants in the soundfile where the sound grains are of interest. An example of such markers in a sound sample are shown in Fig. 5:

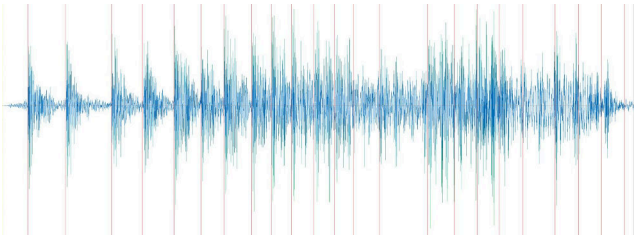

Figure 5: *Markers in red are used to supervise the starting index choice for each grain in the sound sample.*

2.3. Grain Spatialization

For each grain, the spatialization stage is performed in HOA format as a source point with the ambitools encoder (`encoder.dsp`)[4]. The HOA order $L \in \mathbb{N}$ is set at compilation time. The spherical coordinates are randomly picked within intervals set in the user interface at runtime (see Fig. 1). To do so, the `trig` function of Fig. 3 is used. In this way, the radius, azimuth and elevation ranges define a spherical shell sector in which the grain swarm evolves. Optionnally, the grain coordinates and amplitudes signals are forwarded into `bargraphs` to transmit these values through Open Sound Control (OSC) or as output signals in SuperCollider, using the `faust2supercollider` script, unlocking the swarm visualization in AntesCollider (see Sec. 3.2).

2.4. FAUST Compilation

The Granulator in its current version uses 8 decorrelated random noises signals[5] for each of the N stream. In addition, each of the N monophonic stream is encoded into $(L+1)^2$ HOA signals. Moreover, to switch between recorded buffer or sound file at run time, both signal are computed at runtime, as well as 20 signals per stream for the grains envelopes. Finally, there are $28N + (L+1)^2$ audio rate signals to compute at runtime. As L and N increase the FAUST compiler takes a rapidly increasing time to evaluate and progagate the code and produce the binary output, if at all. This can be seen in Fig. 6 for increasing N and L. Therefore, we suggest compiling the Granulator keeping the value of N low and launching several instances of the plug-in in the host software. Note that use of the `no.rnoises` primitive in the code is therefore essential to use different seeds for the noise generators of the various plug-in instances.

[4] `https://sekisushai.net/ambitools/docs/encoder.html`

[5] One random signal for each of the following parameters: duration, reading speed, starting index, reading direction, probability, radius, azimuth and elevation.

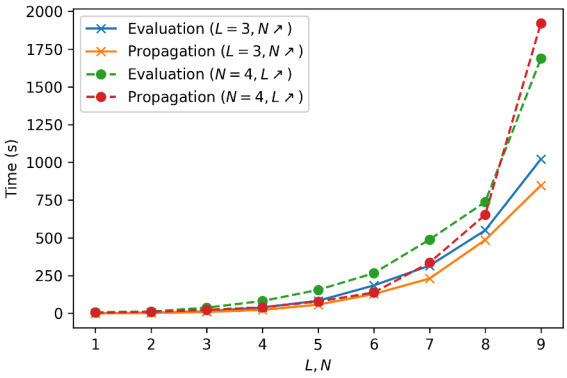

Figure 6: *The time spent by the* FAUST *on the evaluation and propagation steps for: A constant HOA order $L = 3$ and increasing number of grain streams N; A constant grain streams number $N = 4$ and increasing HOA order. The values are obtained using `faust -time -t 0 granulator.dsp` on a conventional laptop.*

3. ANTESCOLLIDER INTEGRATION

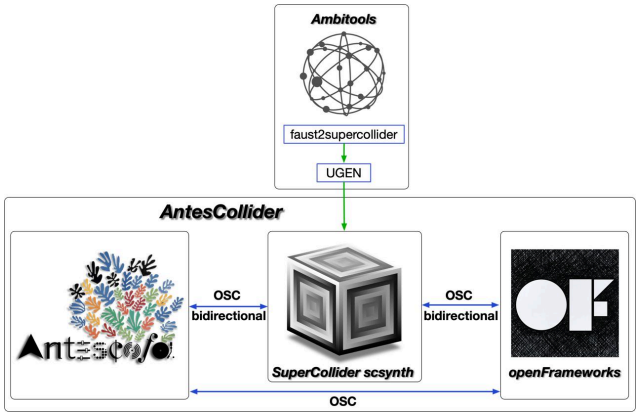

Figure 7: *Interaction between Ambitools and Antescollider.*

AntesCollider [8] is a library for composing and writing electronic music, created using the Antescofo [10] programming language. It consists of two elements: an audio rendering engine, the SuperCollider [11] servers (scsynth), and a synchronous programming language to control them, Antescofo. The goal of this integration is to dynamically create real-time audio processing chains with fine control over parameters over time. The expressiveness of the Antescofo language and its temporal control allow for efficient and concise creation and on-the-fly restructuring of audio processing, simplifying and adding flexibility to synthesis control. The main motivations behind the creation of this library are both musical and compositional. They aim towards the conception of an environment that can extend the composer's palette and imagination in order to create music with a strong component of interactivity, thanks to score following and data processing derived from performance, such as audio signal analysis and gesture tracking of performers. This interactivity can also be easily extended to other

media, such as video, through the integration of communication protocols like OSC. AntesCollider integrates advanced functions:

- dynamic audio chain creation,
- preset saving and loading,
- graphical monitoring,
- spatial composition thanks to the integration of the Ambitools [12, 7] library,
- algorithmic control (with examples of physical models (boids, mass-spring), KNN in parameter spaces, etc.).

3.1. Granulator Unit Generator

The Granulator tool of Sec. 2 is integrated within Antescollider as a Unit Generator (UGEN) in SuperCollider (see Fig. 7). To do so, the `granulator.dsp` code is compiled using the `faust2supercollider` script. An example of use within AntesCollider is shown in the code of Fig. 9.

3.2. 3D Visualization

To visualize the grain swarm from the Granulator, we use the OpenFrameworks [6] C++ library for real-time image and video synthesis. To receive data from the Granulator UGEN, an OSC connection is used with the SuperCollider scsynth (see Fig. 7). The UGEN sends the spherical coordinates as well as the amplitude in dB of each grain in real-time at the audio rate. It is then converted into OSC messages via the `SendReply.ar` command in a SynthDef in SuperCollider. This implementation allows for real-time visualization of the 3D position of each grain in space, as well as their amplitude: the grain dynamically changes size based on the amplitude[7]. In parallel with this implementation, a 3D Ambisonic energy visualizer is used to display energy on a spherical surface. The UGEN for this energy visualizer uses a sampling decoder from the `sampling_decoder.dsp`[8] tool of the Ambitools library on a 974-node Lededev grid [13]. This 3D Visualization tool is shown in Fig. 8:

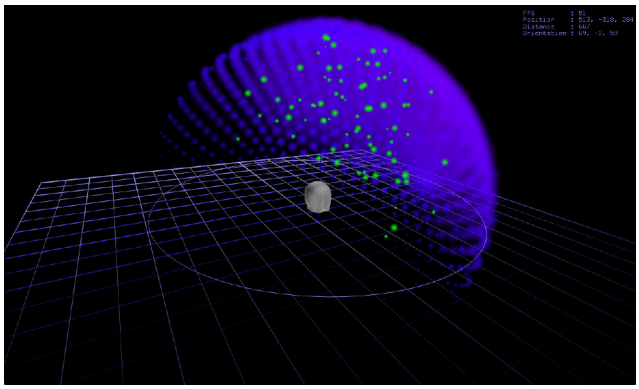

Figure 8: *The 3D visualization of the grain swarm. Here $N = 30$ grain streams are used. The grains are represented in green, their size is proportional to they amplitude. The acoustic energy of the resulting HOA scene is shown in purple. A dummy head facing the front direction is placed at origin and represented in grey.*

4. CONCLUSIONS

We have developed a new spatial granular synthesis tool, the "Granulator", integrated into the Ambitools v1.3 library and implemented in the AntesCollider library. The "Granulator" is the result of research and creation at GRAME[9]. The development and addition of various parameters were carried out by considering elements of spatial and musical perception, with the aim of using it in real-time, both with audio files and live input. Resource optimization through programming in FAUST and its deployment in SuperCollider (using the `faust2supercollider` script) allows for a versatile, dynamic, HOA, multi-grain granulator where all parameters can be modified in real-time, providing great flexibility and richness both in timbre and spatial impession. Its implementation in the AntesCollider library, thanks to the Antescofo synchronous programming language, enables the creation of intuitive spatial morphologies in direct relation to instrumental performance and/or with fine control of all parameters in a multimodal way. Future developments include the possibility of creating spatial zones based on timbre, through the use of audio descriptors. This involves the idea of creating and recreating spatial soundscapes based on characteristics such as pitch, spectrum, dynamics, roughness, and more. The Granulator will be extensively used for the creation of a new piece for trumpet and live electronics, "Gnomon", commissioned by GRAME and to be premiered in June 2025 in Lyon, France.

5. ACKNOWLEDGMENTS

The authors would like to thank GRAME for hosting them during artistic residencies in 2024. Most of this work was carried out at that time. In particular we would like to thank Stéphane Letz for his technical support on FAUST and `faust2supercollider` script. Part of this work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme. ERC REACH: Raising Co-Creativity in Cyber-Human Musicianship, Grant agreement #883313

[6] `https://openframeworks.cc/`

[7] Note that if the grain amplitude is 0 (as when its probability is 0), it disappears in the visualization.

[8] `https://sekisushai.net/ambitools/docs/sampling_decoder.html`

[9] `https://www.grame.fr/`

```
// Example of creating a track in AntesCollider and initialize a Granulator:

// First of all, we instantiate a 'mix_group' (a group of tracks) in ambisonic,
// in this case, in 4th order on the scsynth server 'server1'
obj::mix_group_HOA(''group_hoa1'', ''server1'', ''HOA_GRAME_ADTN3D'', 4)

// Then, instantiate the track 'granulator_track' in the group 'group_hoa1'.
// All tracks instantiated in this group will inherit the order of the group (in
    this case, 4th order).
obj::crea_track_HOA(''granulator_track'', ''group_hoa1'', fade_in = 1, amp = -6,
    encoder = false, doppler = 0)

// Retrieve the audio bus and assign it to the Antescofo variable '$hoa_bus'.
$hoa_bus := $tracks(''granulator_track'').$hoa_inter_bus

// Add the module (SuperCollider SynthDef) 'Granulator8_4' to the track '
    granulator_track', a UGEN with 8 grain streams in
// 4th order and set the initialization parameters for the granular synthesis module
     'Granulator8_4'
$tracks(''granulator_track'').mod_add([''Granulator8_4'', ''globTBus'', $hoa_bus,
    radius_max, 1, radius_min, 0.1, azimuth_max, 180, azimuth_min, -180,
    elevation_max, 45.0, elevation_min, -45.0, read_speed_max, 1.5,read_speed_min,
    0.5, duration_max, 0.51, duration_min, 0.001, shift_min, 0.0, shift_max, 0.5,
    sound, 4, env_0, 8, env_1, 0, env_crossfade, 0])

// Set the initialization parameters for the granular synthesis module '
    Granulator8_4'.
$tracks(''granulator_track'').set(''Granulator8_4'', [radius_max, 1, radius_min,
    0.1, azimuth_max, 180, azimuth_min, -180,
elevation_max, 45.0, elevation_min, -45.0, read_speed_max, 1.5, read_speed_min, 0.5,
     duration_max, 0.51, lduration_min, 0.001, shift_min, 0.0, shift_max, 0.5, sound
    , 4, env_0, 8, env_1, 0, env_crossfade, 0])

// Change parameters in real-time (live coding)
$tracks(''granulator_track'').set(''Granulator8_4'', [record_input, 0])
$tracks(''granulator_track'').set(''Granulator8_4'', [grains_probability, 0.05])
$tracks(''granulator_track'').set(''Granulator8_4'', [reverse_forward_max, 1])
$tracks(''granulator_track'').set(''Granulator8_4'', [reverse_forward_min, -1])
$tracks(''granulator_track'').set(''Granulator8_4'', [record, 0])
$tracks(''granulator_track'').set(''Granulator8_4'', [azimuth_min, -30])
$tracks(''granulator_track'').set(''Granulator8_4'', [azimuth_max, 30])
$tracks(''granulator_track'').set(''Granulator8_4'', [elevation_min, -20])
$tracks(''granulator_track'').set(''Granulator8_4'', [elevation_max, 40])
$tracks(''granulator_track'').set(''Granulator8_4'', [radius_max, 10])
$tracks(''granulator_track'').set(''Granulator8_4'', [radius_min, 1])

//Move parameters with continuous controls, in this case at different speeds using
    random LFOs.
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', azimuth_min, 0, 360, 0, ''
    linear'', 120)
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', azimuth_max, 0, 360, 0, ''
    linear'', 90)
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', elevation_min, -30, 0, 0,
    ''linear'', 30)
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', elevation_max, 0, 90, 0, '
    'linear'', 55)
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', radius_min, 0.1, 2, 0.1, '
    'linear'', 160)
$tracks(''granulator_track'').rand_lfo(''Granulator8_4'', radius_max, 1, 10, 1, ''
    linear'', 40)
```

Figure 9: *Example of the Granulator usage in AntesCollider.*

6. REFERENCES

[1] Dennis Gabor, "Acoustical quanta and the theory of hearing," *Nature*, vol. 159, pp. 591–594, 1947.

[2] Iannis Xenakis, "Formalized music. bloomington, indi-ana," 1971.

[3] Barry Truax, "Real-time granular synthesis with a digital signal processor," *Computer Music Journal*, vol. 12, no. 2, pp. 14–26, 1988.

[4] Curtis Roads, *The computer music tutorial*, MIT press, 1996.

[5] Scott Wilson, "Spatial swarm granulation," in *ICMC*, 2008.

[6] Nicholas Mariette, "Ambigrainer-a higher order ambisonic granulator in pd," in *Ambisonics symposium*, 2009.

[7] Pierre Lecomte, "Ambitools: Tools for Sound Field Synthesis with Higher Order Ambisonics - V1.0," in *International Faust Conference*, Mainz, 2018, pp. 1–9.

[8] José Miguel Fernandez, Jean-Louis Giavitto, and Pierre Donat-Bouillud, "Antescollider: Control and signal processing in the same score," in *ICMC 2019-International Computer Music Conference*, 2019.

[9] Yann Orlarey, Dominique Fober, and Stéphane Letz, "FAUST: An efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, vol. 290, 2009.

[10] Arshia Cont, "Antescofo: Anticipatory synchronization and control of interactive parameters in computer music.," in *International Computer Music Conference (ICMC)*, 2008, pp. 33–40.

[11] James McCartney, "Rethinking the computer music language: Super collider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[12] Florian Grond and Pierre Lecomte, "Higher order ambisonics for supercollider," in *Linux audio conference*, 2017.

[13] Vyacheslav Ivanovich Lebedev and AL Skorokhodov, "Quadrature formulas of orders 41, 47, and 53 for the sphere," in *Russian Acad. Sci. Dokl. Math*, 1992, vol. 45, pp. 587–592.