Particle synthesis – a unified model for granular synthesis

Øyvind BRANDTSEGG
Music technology
NTNU - Department of Music
NO-7491 Trondheim, Norway
oyvind.brandtsegg@ntnu.no

Sigurd SAUE
Music technology
NTNU – Department of Music
NO-7491 Trondheim, Norway
sigurd.saue@ntnu.no

Thom JOHANSEN
Q2S Centre of Excellence
N-7491 Trondheim
Norway
thomj@alumni.ntnu.no

Abstract

The article describes an implementation of a synthesis module capable of performing all known types of time based granular synthesis. The term particle synthesis is used to cover granular synthesis and all its variations. An important motivation for this all-inclusive implementation is to facilitate interpolation between the known varieties of particle synthesis. The requirements, design and implementation of the synthesis generator is presented and discussed. Examples of individual varieties are implemented along with a longer interpolated sequence morphing between them. Finally an application, the Hadron Particle Synthesizer, is briefly presented.

Keywords

Granular synthesis, Particle synthesis, CSound

1 Introduction

Granular synthesis is a well established technique for synthesizing sounds based on the additive combination of thousands of very short sonic grains into larger acoustics events [1]. Its potential for musical and sonic expression is abundantly rich through fine-grained (!) control of properties in both the time- and frequency-domain.

The foundation for granular synthesis was laid by the British physicist Dennis Gabor in his studies of acoustical quanta as a means of representation in the theory of hearing [2]. The idea of using grains of sound in music was later expanded into a compositional theory by Iannis Xenakis in his book *Formalized Music* [3].

In its basic form granular synthesis offers low-level control of single grains through parameters such as waveform, frequency, duration and envelope shape, and it typically provides global organization of grains through another set of parameters such as density, frequency band and grain cloud envelope.

There are several variations of the basic scheme. A comprehensive survey of different granular techniques can be found in Curtis Roads' excellent book "Microsound" [4]. We will present a brief summary in the next section. The book suggests the term *particle synthesis* as a general term covering granular synthesis and all its variations. Although not a formal definition, we will adopt that usage in this paper. Hence our all-in-one implementation of all these techniques is aptly named *partikkel*, the Norwegian word for 'particle'.

Due to its popularity numerous implementations of granular synthesis have been made available through the years, starting with the pioneering works of Roads (see [4]) and later Truax [5]. Today we find real-time granular synthesis modules included in commercial software synthesis packages such as Absynth and Reaktor from Native Instruments [6] or Max/MSP from Cycling '74 [7]. Granular synthesis is also a household component of open-source, platform-independent audio programming languages such as CSound [8], PureData [9] and SuperCollider [10].

Common to most of these implementations is that they focus on a particular variety of granular synthesis, for instance sound file granulation or asynchronous granular synthesis. The opcode[1] *partikkel* [11] that we have implemented in the audio processing language CSound, is an attempt to support all known types of time based granular synthesis. To our knowledge it is the only open-source, platform-independent all-in-one solution for granular synthesis.

This paper will motivate the design of our particle generator by extracting requirements from known particle synthesis varieties. After some additional considerations we present the *partikkel* implementation. Finally we briefly introduce the Hadron Particle Synthesizer that provides a powerful and compact user interface to the particle generator.

[1] An opcode is a basic CSound module that either generates or modifies signals.

2 Particle synthesis

The term particle synthesis covers all the varieties of granular synthesis as described by Roads [4]. In this section we will take a closer look at each variety, starting with basic granular synthesis. We will focus on specific design requirements posed by the variety as input to an all-including implementation.

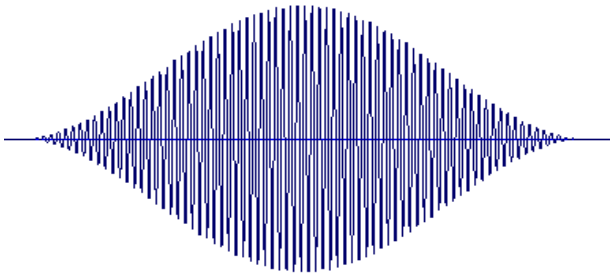

Figure 1: A sinusoidal grain with Gaussian envelope

2.1 Basic granular synthesis

The building block of granular synthesis is the grain, a brief microacoustic event with duration near the threshold of human hearing, typically in the range 1 to 100 milliseconds [4]. Figure 1 shows a typical grain: a sinusoidal waveform shaped by a Gaussian envelope. The parameters necessary to control the grain are:

- Source audio: arbitrary waveform (sampled or periodic)
- Grain shape: envelope function for each grain
- Grain duration
- Grain pitch: playback rate of source audio inside the grain
- Phase (or time pointer): start position for reading the waveform inside each grain

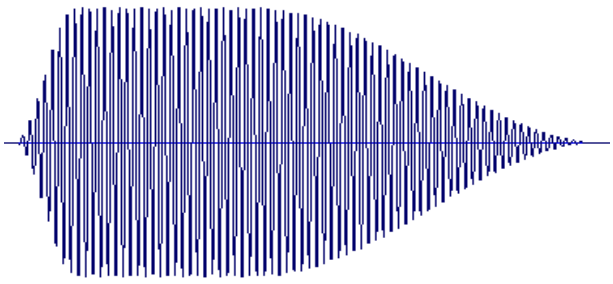

Figure 2: Sinusoidal grain with irregular envelope and sustain

The grain shape does not have to be symmetric. Figure 2 shows a grain envelope with independently specified attack and decay, and a sustain portion in the middle. A flexible implementation should permit updates of both grain envelope and waveform during playback.

The global organization of grains introduces one more parameter:

- Grain rate: the number of grains per second

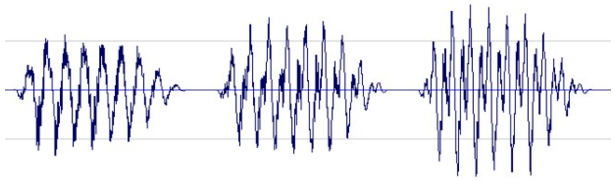

Figure 3: Synchronous granular synthesis of audio sample, the time pointer into the source waveform is updated on each new grain

In synchronous granular synthesis, the grains are distributed at regular intervals as shown in Figure 3. For asynchronous granular synthesis the grain intervals are irregularly distributed, and in this case it might be more correct to use the term *grain density* than *grain rate.* An all-including implementation should permit various degrees of soft or hard synchronization.

The concept of a granular *cloud* is typically associated with asynchronous grain generation within specified frequency limits. The latter can easily be controlled from outside the grain generator by providing a randomly varied, band-limited grain pitch variable. Similarly the amplitude envelope of a cloud of grains may be implemented as external global control of the individual grain amplitudes.

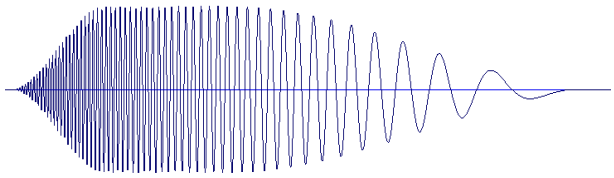

Figure 4: A typical grain in Glisson synthesis

2.2 Glisson synthesis

Glisson synthesis is a straightforward extension of basic granular synthesis in which the grain has an independent frequency trajectory [4]. The grain or glisson creates a short glissando (see Figure 4 above). In order to meet this requirement the granular generator must allow specification of both start and end frequency for each individual particle and also allow control over the pitch sweep curve

(the rate of progression from starting pitch to ending pitch).

2.3 Grainlet synthesis

Grainlet synthesis is inspired by ideas from wavelet synthesis. We understand a wavelet to be a short segment of a signal, always encapsulating a constant number of cycles. Hence the duration of a wavelet is always inversely proportional to the frequency of the waveform inside it. Duration and frequency are linked (through an inverse relationship). Grainlet synthesis is based on a generalization of the linkage between different synthesis parameters.

Obviously, the greater the number of parameters available for continuous control, the greater the number of possible combinations for parameter linkage. The most common linkage of grainlets is the frequency/duration linkage found in wavelets. More exotic combinations mentioned by Roads [4] are duration/space, frequency/space and amplitude/space. The space parameter refers to the placement of a grain in the stereo field or the spatial position in a 3D multichannel setup.

Grainlet synthesis does not impose additional requirements on the design of the granular generator itself, but suggests the possibility of linking parameters, which can conveniently be accomplished in a control structure external to the actual granular audio generator unit.

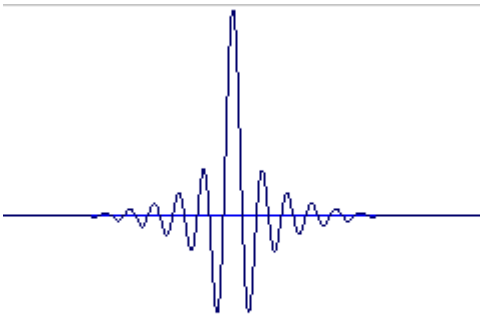

Figure 5:Band-limited trainlet pulse

2.4 Trainlet synthesis

The specific property that characterizes a trainlet (and also gives rise to its name) is the audio waveform inside each grain. The waveform consists of a band-limited impulse train as shown in Figure 5. The trainlet is specified by:

- Pulse period (or its counterpart, the base frequency)
- Number of harmonics
- Harmonic balance (chroma): The energy distribution between high and low frequency harmonics

In terms of designing a general purpose granular synthesis generator, as we set out to do in this paper, it should be noted that the trainlet waveform has to be synthesized in real time to allow for parametric control over the impulse train. This dictates that the trainlet must be considered a special case when compared to single cycle or sampled waveforms used in the other varieties of particle synthesis.

2.5 Pulsar synthesis

Pulsar synthesis introduces two new concepts to our universal particle synthesis engine: duty cycle and masking. Here the term pulsar is used to describe a sound particle consisting of an arbitrary waveform (the *pulsaret*) followed by a silent interval. The total duration of the pulsar is labeled the pulsar period, while the duration of the pulsaret is labeled the duty cycle. The pulsaret itself can be seen as a special kind of grainlet, where pitch and duration is linked. A pulsaret can be contained by an arbitrary envelope, and the envelope shape obviously affects the spectrum of the pulsaret due to the amplitude modulation effects inherent in applying the envelope to the signal. Repetitions of the pulsar signal form a pulsar train.

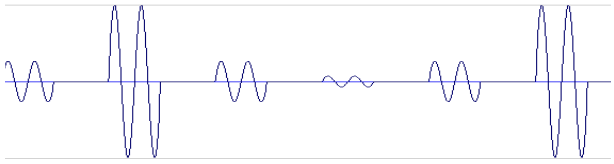

Figure 6: Amplitude masked pulsar train

A feature associated with pulsar synthesis is the phenomenon of *masking*. This refers to the separate processing of individual pulsars, most commonly by applying different amplitude gains to each pulsaret (see Figure 6 for an example). Masking may be done on a periodic or stochastic basis. If the masking pattern is periodic, subharmonics of the pulsar frequency will be generated. To be able to synthesize pulsars in a flexible manner, we should enable grain masking in our general granular synthesizer.

2.6 Formant synthesis

Granular techniques are commonly used to create a spectrum with controllable formants, for example to simulate vocals or speech. Several variants of particle-based formant synthesis (FOF, Vosim, Window Function Synthesis) have been

proposed [4]. As a gross simplification of these techniques one could state that the base pitch is constituted by the grain rate (which is normally periodic), the formant position is determined by the pitch of the source audio inside each grain (commonly a sine wave), and the grain envelope has a significant effect on the formant's spectral shape. Formant wave-function (FOF) synthesis requires separate control of grain attack and decay durations, and commonly uses an exponential decay shape (see Figure 7). These requirements must be met by the design of our all-including granular generator.

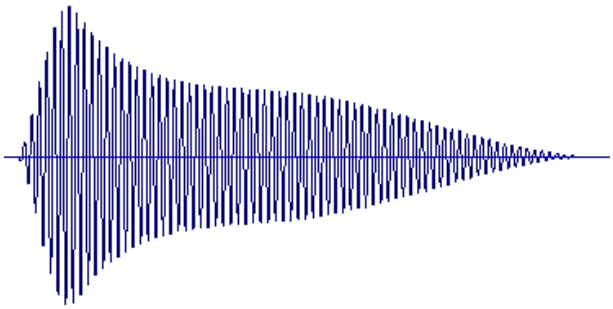

Figure 7: Grain shape with complex envelope. The envelope is made up of an overall exponential decay combined with sinusoidal attack and decay segments.

3 Design considerations

3.1 Grain clock

Different varieties of particle synthesis use different methods for organizing the distribution of grains over time, from periodic grain dispersion to asynchronous scattering of grains. A general purpose granular generator must be able to dynamically change the rate and the periodicity of the internal clock used for grain generation. Generation of truly asynchronous grain clouds may require that an external clock source is allowed to trigger grain generation (possibly by disabling the internal clock). In any case, enabling an optional external clock source to control grain dispersion ensures maximum flexibility of grain scheduling. In order to support exotic and yet unknown synchronous granular synthesis varieties it would be useful to add the possibility to gradually synchronize internal and external clocks.

When deliberating the question of the most flexible clock source for our granular generator, we should also consider making the clock adjustable at audio rate[2], so as to enable frequency modulation effects on the clock rate. Obviously, the effect of continuously modulating a clock rate is only manifested at the actual tick output of the clock. Hence the clock rate could be considered as some kind of "clock modulation sampling rate". Frequency modulation of the grain rate will be the source of further investigation in later research projects.

3.2 Grain masking

The masking concept introduced in relation to pulsar synthesis could be extended to other parameters than amplitude. We could for instance dynamically distribute individual grains to different locations in space. Thus our particle generator could provide a channel mask option and thereby allow individual grains to be routed to specific audio outputs. This feature would also enable the possibility to apply different signal processing effects (for instance different filtering) on individual grains by post-processing the output channels of the generator.

3.3 Waveform

One important reason for designing an all-including particle generator is to enable dynamic interpolation between the different varieties. As we have already pointed out, the generator should support arbitrary waveforms within the grains. As a matter of fact the grain waveform is a distinguishing characteristic of several varieties. In order to morph between them the particle generator must support gradual transitions from one waveform to another.

The most obvious approach to waveform transitions is crossfading. Crossfading between two different waveforms would be sufficient, but it might be interesting to investigate the effects of simultaneously crossmixing even more waveforms into each grain. We also need a crossfading option for trainlet sources, since trainlet synthesis must be treated as a special case. The masking technique discussed in the previous section can easily be extended to include source waveform mixing: a wave-mix mask for truly exotic pulsars.

Providing several simultaneous source waveforms for each grain would naturally also require independent transposition and phase (time pointer) control for each source wave to enable flexible mixing and matching of source waves.

As a simple extension to the already flexible playback and mixing of source audio material within each grain, the generator could add support for frequency modulation of the source waveforms.

[2]Audio rate corresponds to the sample rate (as opposed to control rate which normally is orders of magnitude slower)

It is computationally cheap, but its effects in granular synthesis have been sparsely explored. Frequency modulation of source waveform playback pitch could be implemented as phase modulation, using an external audio input to modulate the reading position of the source audio waveform(s).

4 The *partikkel* CSound opcode

A generalized implementation enabling all known varieties of particle synthesis in one single generator will facilitate new forms of the synthesis technique. To enable the use of such a generalized granular generator in a broader context, it seems apt to implement it in an already existent audio processing language. To broaden the context as much as possible it would be preferable to use an open source language with a large library of signal processing routines already in place. To comply with these requirements, the authors chose to implement the new generator as an opcode for the audio programming language CSound. The opcode was given the name *partikkel*.

We will now try to sum up the features of *partikkel*. Where appropriate, we will refer to the specific type of particle synthesis that each feature originates from.

The basic parameters of granular synthesis are grain rate, grain pitch and grain shape/duration, as well as the audio waveform inside each grain. We decided to enable grain rate modifications at audio rate since this might open up new possibilities for frequency modulating the grain rate. The internal grain clock may also be disabled completely for truly asynchronous grain clouds, or it may be run as an internal clock with soft synchronization to an external clock source. For simpler grain displacements (semi-synchronous), a separate grain distribution parameter has been implemented, moving single grains within a time slot of 1/grainrate seconds.

Grain pitch should be relatively straightforward, defined as the playback speed of the audio waveform inside each grain. However, since we use four separate source waveforms[3] we need four separate pitch controls, in addition to one master pitch control. Grain pitch can also be modified at audio rate via a separate frequency modulation audio input parameter to *partikkel*. Trainlets (or pulse trains) can be used as a fifth source, and we actually need a separate pitch control for them too.

As glisson synthesis requires pitch glissandi within each grain, an additional layer of pitch control with start and end pitch for each grain has been added. This type of control over individual grains is implemented in a general manner via a grain masking method. We will return to that topic later.

Different varieties of particle synthesis require different source audio waveforms, and to enable the interpolation between different synthesis types *partikkel* has the ability to crossfade between different waveforms. Separate phase control over the four source waveforms completes this design requirement. Trainlet synthesis requires a special source waveform of band limited pulse trains. This waveform is synthesized in real time to allow for parametric control over harmonics and chroma.

Both pulsars and formant synthesis require flexible grain envelopes with separate control over shape and time for both the attack and decay portion of the envelope. As a further adjustment to the envelope shape, a sustain time parameter has been added, where the grain amplitude is at maximum for the duration of the sustain segment. To enable even more flexibility, a second enveloping window (full grain length) might be used on top of the primary attack, sustain and decay shape.

Pulsar synthesis introduces a grain masking feature. Normally, this masking would be confined to amplitude and output channel modifications. In *partikkel*, the masking methods have also been extended to include source waveform mix, pitch glissandi (with separate start and end pitch values), and frequency modulation index masks. The masking feature is implemented by using tables of successive values, *partikkel* reading one value for each grain before progressing onto the next table index. Start and end/loop indices are also part of this data set, so the mask length and content can be continuously modified while the generator is running. For simplified random particle bursts, a separate parameter (random mask) can be used to randomly mute separate grains.

Grainlet synthesis has not been explicitly accounted for so far. This is because we chose to design the core granular generator to be as generic as possible, and as part of that design decision we determined that any parameter linkage should be left to external implementation. Still, the parameter set and the supported rate of change for each parameter have been designed with parameter linkage in mind.

[3]The choice of four source waveforms is a more or less arbitrary trade-off between complexity and expressivity.

4.1 Implementation notes

The processing in the *partikkel* opcode consists of two primary phases: grain scheduling and grain rendering. The grain scheduler will place grains according to the time parameters, with each grain being given attributes according to the parameters describing pitch, amplitude, etc.

Grain rendering consists of synthesizing the actual grain waveforms. Despite the large number of parameters utilized by partikkel, the core grain rendering itself is quite simple, and consists of the following stages:

1. interpolated sample reading or DSF[4] synthesis for trainlets
2. frequency modulation
3. frequency sweep (glisson) curve synthesis
4. applying envelope
5. mixing to output buffer(s)

Most of the internal parameters these stages depend upon are calculated on creation of the grain and stored away in a linked list containing one entry per grain, and will not be modified until the end of the grain's life cycle. This is a tradeoff, meaning that partikkel cannot (with the exception of waveform FM) alter any properties influencing a grain during its lifetime, but also means that all the most demanding calculations are performed one time per grain, leaving most processing power to render as many grains as possible at the same time. This might at first seem a limitation, but it can be argued that granular synthesis is at its most promising exactly when grains are allowed to be different, and evolve in succession rather than simultaneously.

5 Examples

A number of implementation examples [12] accompany this paper. The examples are intended to show how different varieties of particle synthesis can be implemented using the generalized technique as described in the paper. First we present a number of individual examples, followed by a long morphing sound, gluing together all the individual examples into a long, continuous transformation.

[4] Discrete Summation Formulae (DSF) (see for instance [17])

5.1 Example 1: Basic granular synthesis, sample player with time stretch

In this example, a sound file is used as the source waveform for grains and we use a flexible time pointer (moving phase value) to set the starting point for waveform reading within each grain. This technique is commonly used for time stretching and other time manipulations.

5.2 Example 2: Single cycle source waveform

A sine wave is used as source waveform for each grain. In itself this is a trivial example, but is included to show the transition (in example 10) from reading sampled waveforms to single cycle waveforms. The transition can be considered nontrivial for most oscillators. Not only must the oscillator waveform change on the fly, but the pitch ratio for sampled sounds and single cycle waveforms are usually very different.

5.3 Example 3: Glissons

Glissons in this example have a converging pitch sweep profile. Each single glisson may start on a pitch above or below, gliding quickly and stabilizing on a central pitch.

5.4 Example 4: Trainlets

Trainlets with 20 partials and chroma varying from 1 to 1.5.

5.5 Example 5: Simple pulsars/grainlets

This example shows simple pulsar synthesis. A pulsaret is generated at periodic intervals, followed by a silent interval. The duration of the pulsaret is inversely proportional to the pulsaret pitch, and the pitch gradually changes over the duration of the example. The waveform of the pulsaret is a trainlet.

5.6 Example 6: Pulsar masking

The example starts with a pulsar train similar to the one in example 5. By grain masking we gradually reduce the amplitude of every second grain, then gradually creating a stereo pattern of grains (left-center-right-center). Towards the end of the example, stochastic masking is added.

5.7 Example 7: Formant synthesis

Using granular techniques similar to the classic FOF (Fonction d'onde formantique) generators, where the grain rate constitutes the perceived pitch. The grain pitch (transposition of the waveform inside each grain) controls the placement of a formant region, and the grain shape controls the spectral contour of the formant region. Since our

Figure 8: Graphical user interface for the Hadron Particle Synthesizer

granular generator allows 4 separate source waveforms with independent pitch, we can create 4 formant regions with one granular generator. The example generates formants as found in the vowel "a" in a male basso voice.

5.8 Example 8: Asynchronous GS

A gradual transformation from synchronous to asynchronous granular synthesis. An asynchronous clock pulse is generated by using a probability function, this clock pulse is used to trigger individual grains.

5.9 Example 9: Waveform mixing

Crossfading between 4 sampled waveforms. From a vocal sample to distorted vibraphone, to cello and finally to a synth pad.

5.10 Example 10: Morphing between all previous examples

One continuous long transformation moving through the granular techniques explored in each previous example.

6 Performing with the partikkel opcode

The all-including implementation of particle synthesis in a single generator encourages further experimentation with granular synthesis techniques. Interpolation between the different granular varieties may reveal new and interesting sonic textures, as would experimentation with some of the more exotic design considerations suggested in section 3.

The flexibility of the *partikkel* opcode comes with a price. The parameter set is large and unwieldy, particularly in a live performance setting. There seems to be an unavoidable trade-off between high-dimensional control and playability. We have therefore investigated various strategies for meta-parametric control to reduce parameter dimensionality and with that performance complexity, greatly inspired by research on mapping in digital musical instruments [13-15].

The *partikkel* opcode takes on the role as the fundamental building block in a particle-based digital instrument where the mapping between performance parameters and opcode variables plays a significant part. Not only to increase playability, but also to provide particle synthesis features external to the core generator, such as the parameter linkage of grainlet synthesis.

The most recent "front-end" for *partikkel* is the Hadron Particle Synthesizer. It adds several new features including a number of modulators such as low-frequency oscillators, envelopes and random generators, all interconnected by a dynamic modulation matrix [16]. A simplified control structure was developed to allow real-time performance with precise control over the large parameter set using just a few user interface controls (see Figure 8).

Hadron is freely available, open-source software [12] and will be publicly released in 2011. The screen shot in Figure 8 is taken from the Max For Live version. Other plug-in formats such as VST, RTAS and AU will be supported. Hadron can also be run as a standalone instrument under CSound. Expansion packs with additional parameter states will be made available for purchase at www.partikkelaudio.com.

Acknowledgements

The *partikkel* opcode was designed by first author Øyvind Brandtsegg. It was implemented as a diploma project by students Thom Johansen and Torgeir Strand Henriksen under the supervision of Sigurd Saue.

References

[1] Curtis Roads. 1988. Introduction to Granular Synthesis. In *Computer Music Journal* 12(2): 11-13

[2] Dennis Gabor. 1947. Acoustical quanta and the theory of hearing. In *Nature* 159(4044): 591-594

[3] Iannis Xenakis. 1971. *Formalized Music*. Indiana University Press, Bloomington, Indiana.

[4] Curtis Roads. 2001. *Microsound*. MIT Press, Cambridge, Massachusetts.

[5] Barry Truax. 1986. Real-time granular synthesis with the DMX-1000. In P. Berg (ed.) *Proceedings of the International Computer Music Conference*, The Hague, Computer Music Association

[6] Native Instruments. Homepage at: http://www.native-instruments.com

[7] Cycling '74. Homepage at: http://cycling74.com/

[8] CSound. Information at: http://csounds.com/

[9] PureData. Information at: http://puredata.info/

[10] SuperCollider. Information at: http://supercollider.sourceforge.net/

[11] CSound opcode *partikkel*. Documentation at: http://www.csounds.com/manual/html/partikkel.html

[12] Hadron source code and audio examples available at: http://folk.ntnu.no/oyvinbra/LAC2011partikkel/index.htm

[13] D. Arfib, J. Couturier, L. Kessous and V. Verfaille. 2002. Strategies of mapping between model parameters using perceptual spaces. *Organised Sound* 7 (2): pages 127-144

[14] A. Hunt, M. Wanderley and M. Paradis. 2003. The importance of parameter mapping in electronic instrument design. *Journal of New Music Research* 32 (4): pages 429-440

[15] A. Momeni and D. Wessel. 2003. Characterizing and controlling musical material intuitively with geometric models. In *Proceedings of the New Interfaces for Musical Expression Conference* (NIME-03) (Montreal, Canada, May 22-24, 2003)

[16] Ø. Brandtsegg, S. Saue and T. Johansen. 2011. A modulation matrix for complex parameter sets. In *Proceedings of the New Interfaces for Musical Expression Conference* (NIME-11) (Oslo, Norway, 2011)

[17] T. Stilson and J. O. Smith. 1996. Alias-free synthesis of classic analog waveforms. In *Proceedings of the 1996 International Computer Music Conference*, Hong Kong, Computer Music Association: pages 332-335. Available at: https://ccrma.stanford.edu/~stilti/papers/blit.pdf