Parameter Field Spatialization: The development of a technique and software library for immersive spatial audio

by Miguel Cerdeira Marreiros Negrão

Lic., M.Mus.

February 2016

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

in the School of Creative Arts

Faculty of Arts, Humanities and Social Sciences

Queen's University Belfast

Abstract

This thesis describes *parameter field spatialization*, a novel technique for creating and controlling spatial sound patterns formed along a surface, when working with sound synthesis and signal processing in a computer-music environment. Its main purpose is the creation of spatially-dynamic immersive sources in electroacoustic composition. This technique generates multiple decorrelated signals from a given sound process definition, which when spatialized at different locations can create a single enveloping auditory event with large width and heigth. By modulating parameters of the sound process differently for each signal it is possible to create a s*patial surface pattern*. The modulation signals are generated based on a mathematical model which assumes a surface encompassing all the loudspeakers and describes an abstract pattern in this surface through a mathematical function of time and surface coordinates, called *parameter field*. This research investigates whether parameter field spatialization can successfully create and precisely control spatial surface patterns, and how these patterns can be made into a compositional parameter in computer music.

The technique was implemented in *ImmLib*, a software library for the *SuperCollider* audio-synthesis environment. Several specific examples of the combination of parameter fields with sound processes were investigated from a perceptual point of view, in listening sessions using two loudspeaker systems, one spherical and the other a vertical rectangular grid. A group of composers was invited to use the software for their own work, producing three pieces presented in public, which were analysed regarding the use of the technique. From this practical work findings relating to the most effective strategies regarding the use of parameter fields were outlined.

Acknowledgements

This research has been carried out at the Sonic Arts Research Centre, Queen's University Belfast between January 2012 and September 2015. It was made possible by generous funding from *Fundação para a Ciência e Tecnologia*, belonging to the Portuguese Ministry of Education and Science, which supported my work during these 4 years.

I would like to thank my current primary supervisor Michael Alcorn for his insight and for helping me become more familiar with the task of technical writing. I am thankful to my second supervisor Stéphanie Bertet for supervising me through these four years, giving me motivation, questioning my ideas and making sure I was on the right track. I would also like to thank my previous supervisors Gary Kendall and Eric Lyon, who having left SARC could not be present throughout the entirety of the Ph.D., but which guided me through the initial steps of the study.

I am thankful to Michael Dzjaparidze for accepting to collaborate on this project, having the patience to use an experimental tool for his compositional work and enduring many days of bug tracking. His feedback and insight into the practical and aesthetic issues arising from using my tool proved to be very useful.

Many thanks go to Scott Wilson, Tim Blechman, Scott Carver, Julian Rohrhuber and the SuperCollider developer community for putting aside their own time to help with fixing complex issues that were uncovered in *SuperCollider* through the use of *ImmLib*. I would like to thank Rohan Drape for helping me with his package rd_dot for ugen graph visualization. I am thankful to the *Game of Life Foundation* for having funded the initial development of *UnitLib* on which *ImmLib* is based, as well as to Wouter Snoei, for being a mentor and providing me with experience in large scale *SuperCollider* projects and for his continuous support in maintaining *UnitLib*.

I would like to thank Luísa Cerdeira, Tomás Patrocínio, José Barata Moura, Trevor

Agus and Wouter Snoei who were kind enough to proof read parts of this text, providing me with very useful feedback.

Special thanks to my dear friend Pablo Sanz for putting up with my complaining through the ups and downs of the Ph.D. experience, and for never ending encouragement. His enthusiasm and willingness to use *ImmLib* for his work led to important contributions for this study. Many thanks to Matilde Meireles for motivating me to keep making music while in Belfast. Special thanks also to all the friends and family who have enriched my life during these four years.

Much gratitude to my parents, who supported me in many ways through these 25 years of studying, for believing in me even when perhaps they didn't fully understand my choices.

QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL PORTUGAL 2007.2013

PROGRAMA OPERACIONAL POTENCIAL HUMANO

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

Published work

Some ideas and arguments present in this thesis have already appeared in the author's following publication:

M. Cerdeira Negrão, "ImmLib - A new library for immersive spatial composition," Int. Computer Music Conf. Proc., vol. 2014, Athens, 2014.

Contents

List of Figures

Acronyms

ADSR	Attack Decay Sustain Release
BRIC	Binaural Room Impulse Response
DAW	Digital Audio Workstation
ESE	Ensemble Source Envelopment
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FM	Frequency Modulation
FRP	Functional Reactive Programming
GUI	Graphical User Interface
HOA	Higher-Order Ambisonics
HRTF	Head Related Transfer Function
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
ILD	Interaural Level Difference
ISE	Individual Source Envelopment
ISW	Individual Source Width
ITD	Interaural Time Difference
LEV	Listener Envelopment

LFO	Low-Frequency Oscillator
NFC-HOA	Near Field Compensated Higher-Order Ambisonics
OOP	Object-Oriented Programming
OSC	Open Sound Control
SARC	Sonic Arts Research Center
VBAP	Vector Base Amplitude Panning
WFS	Wave Field Synthesis

Chapter 1.

Introduction

There is a rich history in electroacoustic music of using spatialization technologies as tools serving a compositional idea [1]. The defining characteristic of electroacoustic music is the generation of sound by transduction of electrical signals into acoustic waves usually using loudspeakers. Tape music, an important subgenre of electroacoustic music, separates music creation from music diffusion, which naturally invites investigation into ways of diffusing sound using multiple loudspeakers, with the spatial sound diffusion becoming another compositional concern. Common spatial compositional strategies include using trajectories, serializing the location parameter, live diffusion, simulation of acoustics, resonating spaces or using sounds that are evocative of specific physical locations [2]. When presenting an electroacoustic sound work, there are varied ways to acoustically radiate audio signals onto a physical space, creating a spatial scene to be perceived by a listener. When using multiple loudspeakers, an audio signal can be sent directly to a single individual loudspeaker, it can be sent to multiple loudspeakers simultaneously (an approach common in sound diffusion), or it can, at least in theory, be placed anywhere between loudspeakers using a multichannel panning technique, such as Ambisonics [3], vector base amplitude panning (VBAP) [4] or wave field synthesis (WFS) [5]. Multichannel panning techniques generally conceive sources as points, with sound emanating from a well-defined narrow region of space. On the other hand, real-world sound sources are perceived as having spatial attributes, which tend to be subtle, such as envelopment, apparent source width (ASW), extent (width and height) and locatedness [6, p. 241][7]. Real-world sound sources display significant diversity in spatiality, from very localizable sources perceived as points, to non-localizable sources, to sources which seem to have extent or even surround the listener.

Creating a point source through electroacoustic means can be straightforward: a single audio signal played through a single loudspeaker is perceived in most cases essentially as a point source. It is not as straightforward to create an extended source, with width and height, which is significantly larger than a point source. Psychoacoustic research has shown that multiple decorrelated sources with similar spectral characteristics when presented simultaneously can create the perception of a source with extent [8, p. 241]. For instance, Plenge [9] demonstrates that two decorrelated broadband-noise signals, played anechoically from two loudspeakers at 60° from each other, create a single auditory event with width sensibly identical to the distance between loudspeakers. Inversely, it has also been shown that the signals from two microphones capturing sound in the diffuse field will be decorrelated [10], and that when listening to a sound source in a reverberant enclosed space, away from the source in the diffuse field, the auditory event created is enveloping and has low locatedness. It appears therefore that the perception of enveloping and wide sources is related with the presence of decorrelated signals. It is also not unreasonable to suppose that in the real world auditory events with significant extent, such as those created by sea waves, are created through merging of decorrelated signals from many small sounding bodies, in the case of waves, out of many individual drops of water colliding.

Recently, spatial audio and sound reproduction techniques have been developed to create more enveloping sound sources or to control source extent, usually through the generation of multiple decorrelated copies of an original signal [11, 12]. In computer music research specific creative techniques have also been developed, such as spatial swarm granulation [13] and spectral spatialization [14], which seem to be able to create sources with extent or spatial textures, exercising some level of control over the properties of the sound at different points along the source.

When multiple audio signals are played each from a different loudspeaker, either a single or multiple auditory events are manifested (see figure 1.1). There are different causes, to be reviewed later, for auditory event segregation which relate not just with spatial properties. Playing two completely unrelated signals on two loudspeakers creates two distinct auditory events, located at each of the loudspeakers. When identical audio signals are played on two loudspeakers, under certain conditions, summing localization

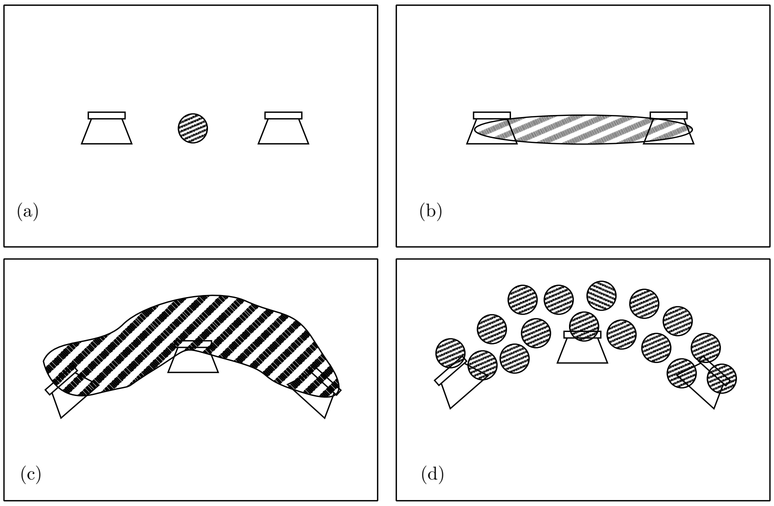


Figure 1.1.: Geometric shapes representing the size of auditory events created from different sets of signals. a) two correlated signals b) two decorrelated broadband-noise signals c) three decorrelated broadband-noise signals d) multiple virtual sources with decorrelated signals with auditory event segregation.

will occur and only one sound source with narrow extent will be perceived, somewhere between the two loudspeakers. On the other hand, under certain conditions, when two decorrelated but spectrally similar signals are sent to two loudspeakers, if the angle between loudspeakers is small enough, a single auditory event is perceived with large width. This phenomenon can also happen with more than two loudspeakers or with virtual sources (using a panning technique) with the source extent becoming wider than with only two loudspeakers. Under certain circumstances, sound signals can be decorrelated and spectrally similar but segregation still occurs. Although multiple auditory events are created, it is possible that a texture is formed which is still perceived as a whole, where individual auditory events rise above and fall back into the texture, depending on the level of attention. This texture can also have extent, therefore in both cases the sound is perceived as occupying a region of space significantly larger than a point, possibly even surrounding the listener from all directions. When the loudspeakers are not coplanar but instead form a surface in three-dimensional space, as is the case with a dome, and in the conditions just mentioned, the auditory event will have both width and height. It will occupy an area of a surface defined by the loudspeakers.

When a sound source has extent and covers a large area, it can sound identical everywhere along its extent or it can sound different in different zones. Perhaps it is brighter in one zone than in another, or louder, or sparser. If it sounds different in different zones, those differences form a pattern, which I will call *spatial surface pattern*. This pattern is a spatial attribute of the auditory event and it can be static or dynamic, evolving in time. A well-defined sharp spatial surface pattern occupying a large area is not so common in everyday environmental sound, the reader will therefore not necessarily be aware of the perceptual phenomenon that I am describing. One notable example which might be useful to mention in order to grasp what spatial surface patterns are, is the sound of sea waves crashing on a sandy beach, something which almost everyone has experienced. When the tip of the wave collapses and hits the water below, there is a change in spectral content, with the high frequencies increasing in energy. This increase seems to propagate from the region of first contact progressively to the left and right, this can be considered a simple one-dimensional spatial pattern. Unfortunately, it is not straightforward to reproduce spatial surface patterns through recorded media and headphones, the binaural stereo sound examples included in the annexed media are not entirely representative of the real experience, and are included only as a reference[1]. It might therefore be the case that the reader will go through this document without being able to entirely perceive the main perceptual property that the thesis deals with, in which case I will have to appeal to the reader's imagination.

To the best of my knowledge, there are at the moment no software tools available to deal with spatial surface patterns in a systematic and general way, the ones that exist are usually tied to a single synthesis/processing technique[2]. The work presented here therefore aims at developing and generalizing the techniques and tools available in computer music for the creation and precise control of spatial surface patterns.

1.1. Parameter Field Spatialization

A computer music environment connected to a multichannel sound system is capable of generating multiple parallel audio signals spatialized at different locations in space. When certain processes, such as stochastic processes, for instance broadband-noise generators, are used in the signal generation, the signals can be decorrelated. Let us

[1] These examples were generated using direct convolution and IRCAM's *listen* head-related transfer function (HRTF) database [15]. Binaural preview through direct convolution is not currently part of the released version of *ImmLib*.

[2] For instance Wilson's spatial swarm granulation [13] and Kim Boyle's spectral spatialization [14].

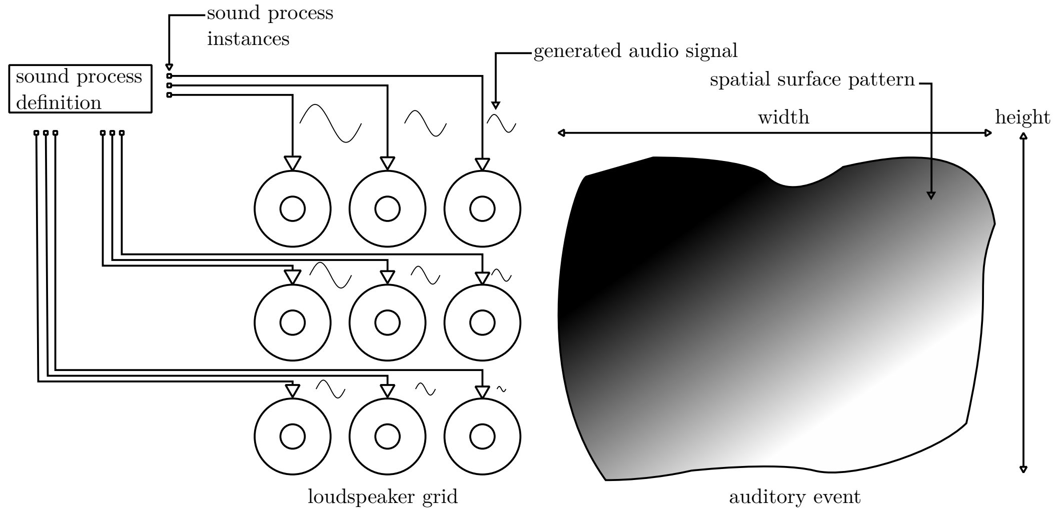


Figure 1.2.: On the left, creation of a decorrelated bundle through multiple instantiations of the same sound process definition, generating multiple decorrelated signals which are sent to different loudspeakers. Corresponding auditory event on the right.

use the term *decorrelated bundle* to refer to a group of spectromorphologically-related[3] decorrelated audio signals spatialized simultaneously at different locations in space. A spatial sound scene consisting of perceptually dissimilar signals in space, that is, multiple clearly distinct auditory events, does not interest us here. We are concerned with signals which are strongly related, possibly but not necessarily becoming a single auditory event, while still decorrelated. If the locations chosen for spatialization lie on a surface in three-dimensional space, such as on a sphere, then the decorrelated bundle can be perceived as occupying a non-trivial region of the surface. If each of the audio signals is made to have slightly different temporal-spectral characteristics it might be the case that it is possible to control the spatial surface pattern that arises. It would be interesting for spatialization in computer music to be able to create sound sources featuring these spatial surface patterns. If the patterns are controllable, then they can become a compositional parameter. Such a technique should have generality, that is, it should be applicable to a wide range of sound synthesis and signal processing techniques, although not necessarily to all. Spatial surface patterns should be controllable to a degree similar to how features of timbre can be controlled in sound synthesis.

This thesis is concerned with a method for generating spatial surface patterns using decorrelated bundles in a computer music system, such that the method is capable of precise control of such patterns. In the method presented here decorrelated bundles

[3]Meaning that the evolution of the spectral qualities is similar.

are automatically generated from a single sound process[4] definition[5] (see figure 1.2). In order to create a spatial surface pattern one parameter of the sound process is selected and multiple auxiliary control signals are generated and applied to this parameter, in the instances of the sound process which correspond to different spatial locations. Parameters of a sound process can be properties such as “amplitude”, “frequency”, or “grain duration”.

In the method presented in this study the control signals are generated from a mathematical model of spatial surface patterns. This model conceptualizes the spatial surface pattern as a continuous pattern or “image” defined on a surface. As any model, it will not always entirely match the underlying phenomena, but in general there should be some correspondence between the abstract mathematical model and the experienced auditory event. From a creative point of view, even when the model fails it can still be useful, as it can nevertheless create an interesting spatial perception according to a certain compositional intention. The mathematical model presented here is based on the differential geometry of surfaces. Intuitively a pattern is a bidimensional image which rests on a curved surface, formally it is a real-valued function on a differentiable surface. A function mapping each point on some space to a real number is sometimes called a *scalar field,* hence we will call the abstract mathematical patterns *parameter fields* and the technique will be named *parameter field spatialization.* This study will present an algorithm for generating the multiple control signals from the abstract mathematical pattern with the aim of generating and controlling spatial surface patterns. This algorithm is generic and can be implemented in any capable sound-synthesis programming language. It will also present a concrete implementation of the algorithm as a software library (*ImmLib*) for the audio-synthesis environment *SuperCollider.* This library presents a complete digital audio workstation (DAW) environment[6], with timelines, user-definable sound processes, extensive scripting capabilities, graphical user interfaces (GUIs) for most functionality. An event is spatialized by selecting a surface, which must be geometrically compatible with the loudspeaker setup, and as-

[4]By sound process it is meant any digital audio algorithm that can be implemented in a typical computer music environment, encompassing synthesis using waveform generators, stochastic (noise) generators, linear and non-linear input-output systems, file playback, soundcard input processing and other algorithms.

[5]The sound process definition must meet certain criteria, presented later, in order for a decorrelated bundle to be created.

[6]Most of the DAW functionality comes from *UnitLib* on which *ImmLib* is based.

signing a parameter field to a parameter of a sound process. The use of parameter field spatialization in electroacoustic music was explored through a set of perceptual tests and examples, and through a collaboration with composers where publicly presented sound works were created using *ImmLib*.

1.2. Research Questions

The research questions which motivated this study can now be presented :

- Can parameter field spatialization successfully create and precisely control spatial surface patterns?
- How can spatial surface patterns be made into a compositional parameter in computer music?
- Does the aforementioned technique represent a novel approach to spatialization in electroacoustic music? In what way does it differentiate itself from other approaches?
- Does the use of parameter fields bring advantages over direct use of decorrelated bundles[7]?
- Is it possible to find strategies that are particularly effective[8] in terms of combining specific types of sound materials with specific uses of the technique ?

This thesis will address these questions in the following chapters.

1.3. Document structure

The thesis is organized as follows:

Chapter 1 is the current chapter which introduces the study.

Chapter 2 presents an overview of the relevant fields of research for this work. It provides a review of spatial audio techniques and presents different approaches

[7] Direct in the sense that no algorithm is used to change the audio signal taking into account the spatialization position. For instance randomizing a parameter of the sound process across all audio signals would constitute direct use of decorrelated bundles.

[8] One would consider more effective those combinations that create patterns with well-defined shapes and movements or more vague but nevertheless enveloping and dynamic spatial sensations, and less effective those that cannot be distinguished from direct use of decorrelated bundles.

for spatialization in the context of electroacoustic and computer music. It presents a review of the literature on the perceptual analysis of spatial attributes in electroacoustic music, and it reviews the relevant psychoacoustic research on spatial hearing and perceptual grouping. It analyses previous approaches related to decorrelated bundles, in computer-music software and artistic practice.

Chapter 3 describes a mathematical model for spatial surface patterns and introduces an algorithm for spatialization in computer music based on this model. It introduces the mathematical formulation of parameter fields, which is based on the differentiable geometry of surfaces. It details the algorithm for the generation, from a given parameter field, of multiple concurrent control signals controlling a parameter of a sound process. Alternative control-signal generation methods, such as discrete sequences are also examined. Finally, specific parameter fields featuring different shapes and behaviours are introduced.

Chapter 4 introduces *ImmLib*, a library for the *SuperCollider* audio-synthesis environment, which implements the algorithm described in Chapter 3. It presents an overview of the affordances of the library and reviews relevant implementation details. It describes the parameter field evaluation strategy which is based on on the use of functional reactive programming, introducing this programming paradigm. Various workflows for creating computer-music works using the tool are presented, and some of the identified technical issues are reviewed.

Chapter 5 evaluates the practical work realized as part of the research. It examines several specific examples of the combination of parameter fields with sound processes from a perceptual point of view. It presents an overview of the collaborations that took place with sound artists, examining in detail the sound works created. Findings relating to the most effective strategies for parameter field use are outlined.

Chapter 6 summarizes the key ideas of this thesis, addresses the research questions and presents suggestions for further study.

1.4. Contributions

The main contributions of this research can be outlined as:

- A model of spatial surface patterns constructed through the use of mathematical functions on differentiable surfaces together with a computer algorithm for spatialization based on this model.
- A set of specific mathematically-defined abstract spatial patterns, namely parameter fields, showcasing different spatial and temporal behaviours which can be applied to parameters of a sound process.
- The *ImmLib* spatialization software library which implements the aforementioned algorithm.
- Perceptual descriptions of a range of different outputs generated by the technique under different configurations.
- An exploration of possible uses of the tool for electroacoustic music and sound art through collaborations with artists.
- A development of strategies for using parameter field spatialization for artistic purposes in the context of computer music.

Chapter 2.

Background

This chapter introduces different fields of research relevant to this research. I will start by giving an overview of the most common approaches to spatialization in electroacoustic music. This is followed by a perceptual analysis of spatial attributes in acousmatic music which will be important for establishing a vocabulary for spatial sound. A brief overview of the psychology of spatial hearing is given so that the limitations and particularities of the human auditory system are taken into account when discussing the design, implementation and assessment of the spatialization software. The most important techniques in spatial audio and multichannel audio rendering are reviewed, contextualizing the spatial audio technologies used by *ImmLib*. Finally, I will discuss compositional strategies and software implementations similar to the approach proposed in this work, situating it in the field of research.

2.1. Spatialization and electroacoustic music

This thesis is concerned with spatialization strategies in computer music. The simplest definition of computer music is that music which is made with computers [16]. Initially it was a development of electronic music, and given the memory limitations of early computers it was almost exclusively preoccupied with sound synthesis [17, p.187], nowadays the use of computers is so widespread that there is hardly any recorded music which does not use computers to some extent, even if just as a recording device. Although historically spatialization was first realized with analogue means, the use of digital and later computer systems proved to be a more practical approach, and nowadays virtually all spatialization tools are computer-based. I situate the approach presented in this work in computer music to stress that spatialization parameters are to be controllable using computer algorithms created by the composer, going beyond predetermined, simplified and rigid interfaces. At the same time, use of the tool presented here by those not proficient in computer programming should not be excluded, as a lot can be accomplished using comprehensive graphical user interfaces and compositional clarity is far more important than technical prowess.

Computer music is part of the broader category of electroacoustic music where spatialization has played an important role. All music is spatial, and temporal, as sound cannot propagate without space and the propagation necessarily takes time. Since currently it is not yet possible to experience music through direct neuronal excitation, all music is heard or felt after sound has travelled through a medium which occupies some space. Composers and sound artists have become aware of the ways in which sound resonates structures while it travels and activates the acoustics of a place. The contemporary understanding of sound relates it with an expanding vibration which permeates a space occupied by sources and listeners, sound is not teleported instantly from loudspeaker to ear. This is known from acoustics, which studies the propagation of sound waves, yet it is in stark contradiction to the majority of everyday experiences where sound seems to be over there, in the source or spread somewhat around the space, but usually neither in the air all around us nor inside our ears. Sound indeed surrounds us, bathing us as waves propagate in all directions, but we do not have sense organs to measure the vibrations in the space far away from us, the sensory receptors we use to make sense of these vibrations are in the perimeter of our body (the ears and the touch receptors). This is the apparent paradox of spatial sound, to be spatial is must be out there, in a multiplicity of places, yet we can only capture it here, in our body. The auditory system takes the direct stimulus from the sensory receptors and through a complex analysis creates an interpretation of the sound that touches us. Through this analysis sound sources are distinguished or considered part of the same object, and perceived to be at specific locations or not localizable at all. It is worth stressing at this point that the spatial sensations of sound are vast, extending much beyond what our vocabulary can describe. Detecting the position of sources is an important evolutionary capability for obvious reasons: to survive and to make sense of the world it is essential to be able to locate entities which might harm us or with whom we want to interact.

It is therefore understandable that our auditory system is tuned to detect the direction of small sources and that a lot of attention in relevant fields is given to such sources. As important as highly localizable point sources are, we should not forget that there are other types of sources which are as pervasive.

All music is spatial, as there is no other way to transmit it to an audience than through space, but spatialization presupposes an attitude towards the use of space in music, specifically making space a parameter of music composition or performance. Such concerns predate electroacoustic music and spatial audio techniques, the typical example being the spatial antiphony of Adrian Willaert's compositions created for the Basilica di San Marco in Venice, a composer who was part of the 16th century Venetian School [18]. They nevertheless gained new relevance with the appearance of electronic and electroacoustic music, music reproduced through loudspeakers. When talking about spatialization in the context of electroacoustic music, on one hand we have sound reproduction and spatial audio technologies, and on the other hand we have spatialization strategies which represent artistic, compositional decisions. Spatial audio technologies are technical means of sounding electric signals, while spatialization strategies are approaches to placing sound in space in order to achieve a musical goal. Specific spatial audio technologies tend to be associated with a specific understanding of space in music, from a compositional point of view. A particular position on space in music tends to motivate concrete technical decisions, such as the use of mono synthetic sources versus stereo recordings, non-homogeneous loudspeaker orchestras, or homogeneous symmetrical loudspeaker arrays. This motivates the following review of strategies for spatialization in electroacoustic music.

2.1.1. Sound diffusion

The practice of sound diffusion, "the realtime (usually manual) control of the relative levels and spatial deployment during *performance*" [19, p.117], is connected to musique concrète. Musique concrète, largely created by Pierre Schaeffer, introduced the use of sound as the primary compositional resource, and focused on the use of concrete sounds obtained through recording. A possible deeper understanding of the concreteness of musique concrète is that "the composer is working directly with sound" [19, p.117] manipulating a direct representation of sound instead of manipulating the abstract symbolic representations of traditional music notation, which appeal to memory and imagination. In sound diffusion the same signal[1], usually stereo, is sent to multiple pairs of loudspeakers, not necessarily matched, which present the sound from different directions but also with different acoustic and perceptual properties, close-far, narrow-wide, reverberant-dry. Diffusion is done live, taking into account the acoustics of the space during the performance, which can be altered by the audience in the room, and requires skill obtained through extensive practice. Harrison stresses that the use of stereophonic recordings is paramount to creating a sense of "space" and according to him, pieces created out of panned mono signals do not create this sense as they have "little or no phase information on the tape" [19, p.126][2]. Truax argues that computer-based multichannel systems can be used in accordance with the traditional sound diffusion principles, where multiple decorrelated stereo pairs are sent to a network of loudspeakers, arguing the need for computer as opposed to manual control of levels when using multichannel inputs only on grounds of impracticality and as an extension of the stereophonic diffusion practice [21]. More recently, diffusion systems such as BEAST have incorporated hybrid approaches, using computer systems, multichannel instead of stereo recordings, more complex routings and spatial grouping algorithms, and the use of panning techniques such as VBAP [22]. Wilson highlights the use of multichannel stems, submixes which can be diffused separately during performance, these being particularly interesting for this research as they allow the diffusion of multiple decorrelated signals as a cohesive group.

2.1.2. Trajectories

Another approach is the choreography of multiple trajectories of highly localizable point sources through space. The use of trajectories for composition entails the creation of an imaginary space where sound sources move, which is distinct from the listening space. An illusion is created as unseen objects appear to move about. Sometimes virtual acoustics are used to simulate a room or space, inside of which the sound sources move. Varèse's *Poème Electronique,* created for the Phillips pavilion at the 1958 Brussel's

[1] A signal in the field of signal processing is a function which carries information about a property or behaviour of a particular phenomenon [20]. Sound is a mechanical pressure wave propagating through a medium such as air. An audio signal is a representation of sound and it is typically assumed that it contains frequencies in the 20Hz to 20000Hz range, the limits of human hearing.

[2] As this work shows, it is possible nevertheless to create phase information artificially, in order to enhance and manipulate spatiality.

World Fair, is an early example. It was recorded on 3-channel tape and projected by automated electromechanical means using 325 loudspeakers and additional subwoofers [23, p.159-162]. Later, in 1968, John Chowning developed sound trajectories using computer music systems, creating a panning algorithm which simulated the Doppler effect and the reverberant field [24, 1]. This system was used to compose *Turenas* (1972), a piece employing FM synthesis and trajectories in a virtual sound stage.

Trajectory-based composition approaches tend to be used with multichannel systems with symmetricalally-placed matched loudspeakers, as the techniques employed to generate the trajectories often work best in these conditions. Usually such compositions are performed from fixed media with 4, 8 or 16 channels or real-time rendered from object-based spatialization software [25], and the trajectories are not influenced during the performance. The key reason for employing trajectories is perhaps to bring a space imagined by the composer into the concert room. Chowning mentions that "The dream was to be able to compose sound in space that was free of physical constraints and realities, yet would evoke images that were believable" [1, p.10].

2.1.3. Spatial allusion

The discussion above regarding sound diffusion and trajectories is framed in terms of placing sound objects in space, irrespectively of their spectral content and meaning for the listener. Barrett [26] defines this aspect of spatiality as *spatial illusion* and characterizes it as the perception of an apparently real space through the use of phantom images or sound field synthesis, as well the mimicking of the expected behaviour of sound in real spaces, such as the amplitude decay and high-frequency absorption due to varying distance, recreation of reflections and a reverberant field, differences in dry to wet ratios as the sound approaches a listener, or Doppler shifts. To this aspect she contrasts *spatial allusion*: "On hearing a sound that we recognise, or a sound to which we think we have found a clear source-bond, we place it within a space that is appropriate, whether or not that space is actually 'sounding'" [26, p.315]. This relates to what Kendall [7] refers to as the *conceptual event*, the event as the listener conceives it in terms of "actions, objects and agents" and the likely physical context where they take place. This can be quite effective even when due to lack of proper perceptual spatial cues the suggested space or spatial behaviour is in contradiction to what is directly perceived. The use of spatial allusion can on the other hand re-enforce an already effective spatial illusion. Source-bonding [27] and spatial allusion are particularly strong with recorded material which maintains enough integrity through the electroacoustic processing chain to still be recognisable. Stereo or multi-microphone recordings can also enhance the spatial allusion as they preserve some of the cues used by the auditory system for spatial decoding, cues which can suggest spaces quite different from the listening space, including a space that stretches beyond the physical dimension of the listening space [27]. Such a sensation of space can also be created more artificially using a single mono recording and time-shifting to create multiple decorrelated signals [28], or using multiple simultaneous monaural recordings from different positions, perspectives and possibly with quite different types of microphones, where the spatial fingerprint present in the recordings together with the decorrelated signals interact to create a powerful spatial allusion.

2.1.4. Contrasted static positions

Multiple static positions can be contrasted without creating the perception of movement, if the signals being presented are spectromorphologically quite distinct. In this case each sound event happens at a static position and is perceived in relation to other distinct sound events, happening simultaneously or not, at other positions. Technically it can be done either through direct assignment to individual loudspeakers or using multi-channel panning. A similar technical procedure giving rise to quite a different compositional and perceptual result, is the placement at static positions of signals which are sufficiently related spectromorphologically to create a texture or an extended source[3]. In this case the spectromorphological evolution of each signal can cause the perception of movement. This is in fact the basis for the approach described in this work.

[3] We will define an extended source as a source whose extent is non-negligible and therefore cannot be considered a point source.

2.2. Perceptual analysis of spatial attributes in electroacoustic music

The previous discussion of different spatial compositional approaches focused on the conceptual systems and technical means used by composers to structure the spatial element of their work. A different angle is to analyse the spatial qualities of electroacoustic music from a listener perspective without assuming knowledge of the means and ideas employed to produce a work. Having an understanding of different spatial attributes of sound, and associated vocabulary, is necessary for the discussion later in this thesis, of the perceptual results from the use of parameter field spatialization.

Smalley [27] provides a thorough perceptual analysis of the different types of spatial experience that can be found in electroacoustic works (see figure 2.1). He distinguishes between *composed space*, the space intended by the composer and encoded on recorded media, and the *listening space*, the physical space in which the composed space is heard. He divides the listening space between *personal space*, the home listening situation, and *diffused space*, the type of physical space and seating positions usually found in public presentations, where off-centre listening is common. He further subdivides composed space between *internal space*, which is present when a spectromorphology suggests resonant properties of solid material, and *external space* which appears when the existence of an environmental space is made known due to reflections and reverberation. He goes on to specify many more spatial attributes, I mention only the ones more relevant to this work. In terms of *image definition*, a sound image can be *spatially blurred* if the occupancy of space is not well defined, and *spatially clear* if it is, and it can be *concentrated* if activity is mostly present in a small area of space or *diffuse* if it is spread. *Spatial fill* regards the density or sparseness of events in space. *Spatial texture* is divided between *contiguous* and *non-contiguous space*, the former being related with trajectories and spread textures without gaps, the latter with the successive presentation of discrete locations without a sense of motion joining them. He mentions [27, p.124] how contiguity and non-contiguity can depend on the attention of the listener and the time-frame of analysis, something which will be touched upon in chapter 5:

> "The distinction between contiguity and non-contiguity is not necessarily clear-cut. Imagine a very active, ongoing, rough texture, presented across

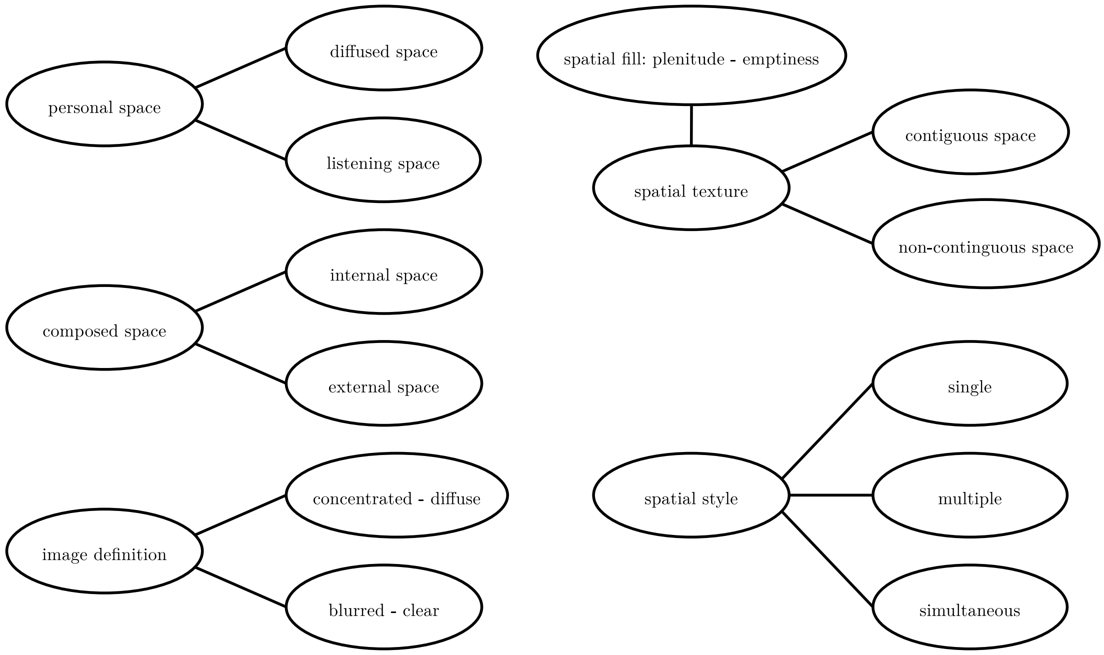


Figure 2.1.: Smalley's grammar of localisation.

> the breadth of stereo space in many scratchy point-sources of varying intensity. These point-sources are presented non-contiguously so that attention may be drawn to any part of the horizontal space where a particular scratch stands out in relief. Thus the attentive ear (eye) darts among non-contiguous positions. Yet, taken over a sufficiently long duration, the individual non-contiguous points of this texture, perceived as a whole, cover a contiguous space. Thus this space is non-contiguously erratic at a low level but contiguous at a higher level, a characteristic spatial behaviour for active textures."

Finally he distinguishes between different possibilities for the global *spatial style* in a work, such as a *single* spatial setting, *multiple* successive spatial settings or *simultaneous* spatial settings.

In another approach, Rumsey [29] lists perceptual attributes of spatial imagery produced with loudspeakers following "a 'scene-based' approach to spatial quality evaluation" [29, p.655]. Although mostly geared towards sound reproduction, it is still quite relevant to electroacoustic music. Rumsey divides spatial attributes between dimensional attributes, *width*, *depth and distance* (to which Kendall [7] adds *height*) and immersive attributes. Some of the immersive attributes, such as *presence* and *environmental envelopment* are related to early reflections and late diffuse reverberant energy characteristic of natural enclosed spaces. These attributes are not very relevant to this work as it is unlikely that the technique presented here will be used to create realistic reproductions of original sound environments as it is more geared towards the creation of artificial spaces. The remaining immersive attributes are more interesting for us: Rumsey distinguishes presence and environmental envelopment from *individual source envelopment* (ISE), the "sense of being enveloped by a single sound source" [29, p.663], and *ensemble source envelopment* (ESE), the "sense of being enveloped by a group of sound sources" [29, p.663], and he specifically mentions that individual source envelopment can be created artificially from dry sources by different spreading techniques. Rumsey also reasons that the type of envelopment reported by listeners in surround sound reproduction from a group of dry sources, or a spreaded single source, is very unlikely to be the same property, related with late reflected sound, of *listener envelopment* (LEV) which appears in traditional literature on reproduction of recorded sound [29, p.662].

Amongst the dimensional attributes, Rumsey also distinguishes between *individual source width* (ISW) and *ensemble width*, where the former relates to the width of what the listener considers to be an individual sound source and the latter to the perceived width of a group of sound sources, while environmental width is related with the sense of how spacious is the enclosing architecture where the sources are placed. Blauert [8, p. 3] defines locatedness as the extent to which one can say that an auditory event is at a particular location. Rumsey [29] makes the observation that sources with small ISW are likely to also have high locatedness, and sources with large ISW are likely to have poor locatedness although he adds the qualification that it is in theory possible that a source has large ISW and has, at the same time, a well defined boundary and is easy to locate. It will be shown later that many sound scenes created with *ImmLib* have exactly this property.

Immersion literally means the action of going under water, metaphorically it evokes the qualities of being completely surrounded by and inside of an environment or experience. In virtual reality and auditory display research it refers to becoming disconnected from the real physical environment and gaining the perception of actually being in a virtual, surrogate world [30]. In that context immersive audio facilitates the creation of the illusion of being in different space. Rumsey includes ISE and ESE amongst "immersion attributes", but Smalley develops the concept further. He notes that in electroacoustic music it relates with sound events which seem to envelope the listener from all directions:

> "One particular type of circumspace positively invites variable orientation. This is immersive space, where the spectral and perspectival space is amply filled, surrounding egocentric space, where the pull of any one direction does not dominate too much, and where the listener gains from adopting, and is encouraged to adopt, different vantage points" [31, p.52].

Reviewing some of the literature relating to spatial attributes, the concept of spatial surface pattern, introduced in chapter 1, does not seem to be a commonly referenced spatial attribute: source extent, width, height, depth and boundary are referenced, but the actual internal pattern of the source does not seem to be widely acknowledged.

2.3. Psychoacoustics and spatial hearing

Spatial composition strategies rely on technical means, spatial audio technologies, to create a desired compositional result. Spatial audio technologies are themselves informed by knowledge about spatial hearing, that is, by how humans localize sounds from sources in the real world. Most of these technologies have limitations, related to properties of the source signal, listener position and other factors, it is therefore important to have a grasp on the main characteristics of human spatial hearing in order to understand how to use a certain spatial technology, and what to expect when using a certain type of sound material.

2.3.1. Localization of a single sound source

Sound emitting objects in the real world generate a sound wave which reaches both ears, activating the auditory system, and producing an auditory event which is perceived by a listener. Although in most cases a single sound event gives rise to a single auditory event, one sound event can produce multiple auditory events, and multiple sound events can cause a single auditory event. *Localization* refers to judgements of the distance and direction of an auditory event. Since good localization performance has been reported in acousmatic conditions, the information used by the auditory system for determining source position must be present in the signals at the eardrums. The main cues carried in these two signals detected so far are the interaural time difference (ITD), the interaural level difference (ILD) and spectral cues caused by head, pinna and torso [8]. The auditory system has a limited spatial resolution, at its highest of around 0.5°[4], significantly lower than the visual system. The following discussion assumes a single sound source[5]. Localization on the horizontal plane[6] presents the least localization blur[7] at $\theta = 0°$[8], in front of the subject. As the sound source is moved to the left or right, localization blur increases until a maximum is reached at $\theta = \pm 90°$, it then decreases again until reaching $\theta = 180°$, where it is higher than at the frontal position. Localization blur in the median plane[9] is lowest directly in front of the listener and increases with an increase of the elevation, reaching a maximum at $\phi = 90°$. Also, it is higher at $\phi = -30°$, at the back, than at $\phi = 30°$, at the front. Localization blur for white noise at $\phi = 0$ is only 4° while for speech it can be as high as 17°. Still in the median plane, for signals with narrow bandwidth, such as sinusoidal tones, the auditory event direction seems to not be related at all with the sound source location, depending only on the frequency of the source signal, these sound events are therefore not localizable in the median plane [8].

In anechoic distance estimation, with training there seems to be good accord between auditory event distance and sound event distance, but as the distance increases the sound event distance is reliably under-estimated with the auditory event distance asymptotically approaching a maximum value. It seems in anechoic conditions a sound is never perceived to be further away than a fixed distance, the auditory event horizon. Attributes of the sound signals at the eardrums are affected by the changes in distance between source and subject in different ways depending on the distance: at distances between 3m and 15m the sound pressure level of the signals decreases following a $\frac{1}{r}$ law, or 6dB for every doubling of distance; at distances longer than 15m the air starts to act as frequency-dependent filter, with high frequencies being more attenuated than

[4]The localization blur depends on the type of signal being used.

[5]It also assumes free-field conditions, as non free-field conditions are equivalent to multiple sound sources, due to the mirror image principle for sound reflections in enclosed spaces.

[6]Sound source at a fixed distance from the subject, free-field conditions.

[7]Localization blur is related with how much displacement of the source can happen before a subject notices that it has moved.

[8]θ is azimuth and ϕ elevation, $(0, 0)$ is front, $(90, 0)$ is right and $(0, 90)$ is up.

[9]The median plane is the plane perpendicular to the axis connecting the two ears and passing through the centre of the head of the subject.

low frequencies; at distances closer than 3 meters the interaction between head and torso with the incoming sound waves causes distance-dependent filtering.

ITD is the most important attribute of the signals at the ears for lateralization[10]. A signal presented with same delay and level at both ears is perceived in the median plane, as the delay is increased in one side, lateral displacement of the auditory event towards the opposite side is linear until 630µs, achieving maximum lateralization with 1ms with the auditory event localized at the entrance of one of the ear canals. Sinusoidal tones above 1.6kHz do not lateralize with changes in phase delay unless they have an envelope. Identical signals presented at the ear canals with a difference in level are more or less linearly lateralized with increasing level differences towards the side with the highest level, until a maximum lateralization is reached at approximately 15 dB–20 dB. Broadly speaking, it is accepted that for broadband sounds, low-frequency components are localized using the interaural time differences, while high-frequency components are localized using interaural level differences.

2.3.2. Decorrelation

It is important for this study to clarify the concept of decorrelation. Correlation measures how "similar" two signals are. Given two signals $x(t)$ and $y(t)$ their *cross-correlation function* is

$$\varphi_{xy}(\tau) = lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} x(t)y(t+\tau)dt.$$

When $y(t) = x(t)$ the expression becomes $\varphi_{xx}(\tau)$, the *auto-correlation function.* This function has the property that $\varphi_{xx}(\tau) \leq \varphi_{xx}(0)$, so auto-correlation is never higher than with zero delay, and it is equal for other values of τ when $x(t)$ is periodic and τ corresponds to the period [32]. The *degree of coherence* k, or *cross-correlation coefficient*, is the maximum absolute value of the normalized cross-correlation function,

[10] Most psychoacoustics experiments relating to spatial hearing are done with headphones in order to precisely control the stimulus signals, in such conditions auditory events are usually localized inside the head, positioned along an axis connecting the two ears. *Lateralization* is the perceived position along this axis. In this situation the possible positions vary between the middle of the head and the entrance of the ear canal. In natural listening conditions, sources are localized outside the head, and head movements improve localization, disambiguating between front and back, or positions in the cone of confusion.

or

$$k = max_{\tau} |\Phi_{xy}(\tau)|,$$

where

$$\Phi_{xy}(\tau) = lim_{T \to \infty} \frac{\int_{-T}^{T} x(t) y(t+\tau) dt}{\sqrt{\int_{-T}^{T} x^2(t) dt \int_{-T}^{T} y^2(t) dt}}.$$

Signals where $k = 1$ or -1 (opposite phase) are fully coherent, signals where $k = 0$ are incoherent, with other values of k they are partially coherent. The cross-correlation is normalized in order to avoid being influenced by the level of the signals [8]. Within this work, incoherent and uncorrelated signals will be considered synonyms. The process of *decorrelation* consists of creating a new signal from an original signal such that the signals are uncorrelated, yet sound the same. The signals are then said to be decorrelated [11, p. 71].

The cross-correlation function applied to the signals at the two ears is called the *interaural cross-correlation function*, while its maximum absolute value is termed the *interaural cross-correlation coefficient* (IACC). For this measurement the absolute value of τ should be smaller than the maximum interaural time difference caused by the physical separation of the human ears, around $\pm$1ms [33]. It is known that this measurement correlates with apparent source width and envelopment [34, 35, 9]. Research has also been conducted in the field of concert hall acoustics with the objective of finding measurements based on IACC which relate to apparent source width [36, 33]. The *inter-channel cross-correlation coefficient* (ICCC) of two signals is in turn related to IACC. For instance, in a stereo configuration when coherent white noise is played in both loudspeakers the ICCC is 1.0 and IACC 0.9, while when each loudspeaker plays incoherent white noise the ICCC is close to 0.0 and the IACC close to 0.2 [12, p. 121]. Therefore playback of decorrelated signals over loudspeakers will generate low values of IACC, which in turn will influence the perception of the source width. For this reason the (inter-channel) degree of coherence of two signals can be taken as crude indicator for the perception of source width.

The degree of coherence compares the entirety of the two signals, when these signals are of long duration the measurement will present a gross indication of correlation. A more detailed picture of correlation can be obtained by using a "running-average" measurement where the cross-correlation is calculated using a time window $[t_1, t_2]$:

$$k' = max_\tau \left| \frac{\int_{t_1}^{t_2} x(t)y(t+\tau)dt}{\sqrt{\int_{t_1}^{t_2} x^2(t)dt \int_{t_1}^{t_2} y^2(t)dt}} \right|.$$

Studies have determined that a time window that matches the resolution of the binaural system should be between 35ms to 243ms with the optimal value estimated between 50ms and 70ms [37]. Henceforth we will consider decorrelated signals those with a "running-average" degree of coherence (considered as a function of time) which remains close to zero through the duration of the signals when measured using a time window smaller than 250ms and $|\tau| \leq 250$ms. This definition is used in order to take into account the case of a non-periodic original signal, for instance an environmental sound recording, and a delayed copy of the same signal. The degree of coherence considering arbitrary values of τ for these two signals will be 1 since reversing the original delay will line up the waveforms. On the other hand considering the definition above which has $|\tau|\leq 250ms$, with a large enough delay it is possible, depending on the content of the original signal, that the two signals are decorrelated since they can be dissimilar when considering distantly separated values in time.

2.3.3. Localization of multiple sound sources

The localization of two sound sources with coherent signals can be divided in several cases:

1. There is only one auditory event,
   a) which depends on the signals of both sources (*phantom image*, *summing localization*).
   b) which depends on the signal of only one of the sources (*precedence effect*).
2. There are two auditory events (echo).

If the time and level differences between the signals at the two sources is small a phantom image appears, this psychoacoustic phenomenon called summing localization is the basis for stereophonic sound reproduction. If the signals are delayed by more than 1ms, only one auditory event is reported, coinciding with position of the closest source, this is the *law of first wavefront, precedence effect* or *Haas effect* [38]. Haas showed that the first reflection can be louder than the primary sound by as much 10dB without being perceivable. Once the delay between signals reaching the ear canals is above a particular threshold, two auditory events are heard, the primary sound and an echo. The echo threshold varies for different types of signals and levels, and is between 2ms and 20ms.

The localization of two incoherent sound sources is quite different from coherent ones. Two partially coherent signals at the ear canals produce a single auditory event with larger extent than with two fully coherent signals. If the degree of coherence between the two signals is decreased, when a certain threshold is crossed two auditory events are produced located at each ear. A study by Damaske [39] mentioned by Blauert [8, p.245] is quite relevant for this work: incoherent signals were played at two loudspeakers, if the loudspeakers were close enough a single enlarged auditory event was produced, if the loudspeakers were farther apart two smaller auditory events were produced. Four equally spaced loudspeakers with incoherent signals produced a single diffuse source spreading in all directions. It is therefore firmly established in the literature that two sources with incoherent signals can produce a single auditory event with large extent, even filling the entire auditory space. This effect is one of the perceptual mechanisms behind the technique described in this work and implemented in *ImmLib*.

In an enclosed space the first wave to reach a listener is the direct sound, this is followed by a group of early reflections which have encountered the walls of the room a small number of times, followed by a dense group of reflections which decay exponentially and are better described statistically, the reverberation of the room. Distance hearing in enclosed spaces, where reflections of the bounding structures (walls, ceiling and floor) are summed to the direct sound, is somewhat more complex. As the direct sound reaches the listener, the auditory system inhibits the formation of further auditory events, according to the law of first wavefront, after the echo threshold has elapsed a strong echo is perceived or if the reverberation field is already strong enough, it is perceived as a diffuse auditory event filling the entire auditory space. The ratio of the level of direct sound relative to diffuse field is an important cue for determining distance of the sound source. Blauert [8, p.278], referring to an experiment by Danilenko [10], states that the coherence between two signals from two microphones placed in proximity of each other but far from the sound source in an enclosed space, is maximum at the arrival of the direct sound and decreases approaching zero as the reverberant field decays. As a listener moves further away from a source in an enclosed space the ratio of the diffuse field to direct sound increases, the signals at the ear canals become more incoherent and the auditory event becomes harder to localize and increases in extent, becoming more diffuse. It has also been shown that the lateral reflections typical of "shoe-box" concert halls cause spatial broadening of the auditory events. This can account for the fact that multiple sound sources with incoherent signals can create a vague perception of a real space with architectural properties, even if the signals were created artificially. This effect can be exploited within *ImmLib* when using a spherical sound reproduction system.

2.3.4. Perceptual grouping

When sound from a single or multiple sound sources reach the ear, the peripheral auditory system is activated, generating neural activity from which the brain creates an auditory percept. In the auditory percept it is often possible to distinguish different discrete sound sources or auditory objects, each with its own location, loudness, pitch and timbre. The brain uses different cues and applies rules such as similarity, good continuation or common fate to analyse a complex mixture of sounds into discrete sources. Bregman [40] distinguishes between *source*, the physical entity which gives rise to the acoustic pressure waves, and *stream,* the psychological organization representing a sequence of acoustic events as being one "source" with internal consistency and continuity which functions as a "whole". The auditory system performs an analysis of the incoming stimulus, separating it in different frequency components, assigning these components to one or different streams. This process is called "auditory scene analysis" or "perceptual grouping". When elements from two different sources are separated into different streams we say "segregation" happened. The grouping of different frequency components emanating from the same source is referred to as "simultaneous grouping" while the grouping in time of events from the same source is called "sequential grouping". When listening to rapid sequences of sounds, if the sounds are grouped into a single stream this is called fusion, if the acoustic stimulus is divided into different streams, fission or stream segregation happens.

Different physical cues are used by the auditory system to separate the auditory stimulus into different streams. Steady complex tones tend to be segregated if the fundamental frequency is different or if one has a spectral regular pattern onto which the other does not fit. Sounds with the same temporal envelope are more likely to be grouped together. Correlated changes in amplitude or frequency, such as when amplitude or frequency modulating two complex tones, can in certain situations cause fusion. Regarding spatial attributes, it is known that if signal and masker have phase or level differences at the two ears it is easier to detect the signal. Some experiments [41, 42] suggest that transitions or static differences in ITD can cause individual sinusoids that are part of a complex tone to segregate, while others [43] showed that pitch shifting a single component of a complex tone would cause the whole tone to be perceived as pitch shifted when the pitch-shifting component was played in the same or different ear from the rest of the tone, suggesting that in this case segregation does not happen. It seems then, that differences in ILD and ITD can in some situation cause segregation while in others seem to have little influence in simultaneous grouping.

2.4. Spatial audio techniques

In order to use spatial attributes as compositional parameters effective spatialization tools and techniques are needed. *ImmLib* is a high-level spatial composition tool, as such it is reliant on lower level spatial audio techniques for effective spatialization of point sources. This section analyses the most relevant spatial audio techniques in the context of this work, and as such does not attempt to be exhaustive.

Spatial audio techniques attempt to faithfully reproduce spatial attributes of sound such as direction, distance, extent or room envelopment using loudspeakers, and often rely on mathematical, physical and psychoacoustic knowledge. The most widely used technique is still stereophony, although the use of multi-loudspeaker setups has become more common, with the use of eight loudspeakers being an informal standard in electroacoustic concerts. The basic goal of spatial audio techniques is to place the centre of a sound source at a particular location in a virtual auditory space (panning), although controlling the source extent, envelopment, directivity and wavefront shape, or doing room simulation is also desirable. Panning methods can be divided between *sound field synthesis*, which attempt to recreate an accurate physical sound field which in turn will create the correct perceptual cues (WFS, Ambisonics), and perception-based methods where an original sound field is not reconstructed, instead equivalent perceptual cues are created (stereo panning, VBAP).

2.4.1. Multichannel panning techniques

Vector Base Amplitude Panning

Vector Base Amplitude Panning (VBAP) is an extension of stereophony to multiple-loudspeaker setups [4]. Like stereophony it is based on summing localization, the psychoacoustic mechanism described in the previous section. VBAP is an amplitude panning law: given a number of loudspeakers it creates a virtual sound source or phantom image by sending the original signal with different amplitudes to a maximum of three loudspeakers in 3D systems, and two loudspeakers in 2D horizontal systems. In stereophonic panning, given two loudspeakers placed no further apart than 60°, the relation between the angle of the phantom image, the angle between the loudspeakers and g_1 and g_2, the gains of the signals, is

$$\frac{\tan\theta}{\tan\theta_0} = \frac{g_1 - g_2}{g_1 + g_2}.$$

Panning more than two loudspeakers in the horizontal plane can be done by considering every pair of loudspeakers as a stereo pair[11]. For 3D setups VBAP extends pair-wise panning to triangles of loudspeakers, generalizing the tangent law using linear algebra and vector bases. In this case only three loudspeakers are active at any given time and as the source moves different loudspeaker triplets are activated and their gains calculated.

The localization of a phantom source agrees quite well with the predicted angle in the median plane, but less so to the sides with areas around the axis connecting the two ears where no phantom source can be placed [44]. In most cases the phantom image is located inside the triangle defined by the loudspeaker triplet [45]. In off-centre listening this means that localization reversal cannot happen, that is, a phantom source cannot be perceived in the opposite hemisphere to the predicted location due to the listener being very close to a single loudspeaker.

[11] Since perceptual tests indicate the angle between loudspeakers should not be higher than 60° a minimum of 6 loudspeakers is needed for horizontal setups.

VBAP panners are available for *Max/MSP* [46], *PD* [46], *Csound* [47] and *SuperCollider* [48].

Ambisonics

Ambisonics is a sound field reproduction technique based on the spherical harmonic decomposition of the sound field [49]. It is capable of spatial sound encoding for reproduction over pantaphonic (2D) and periphonic (3D) multi-loudspeaker systems.

When encoding a single mono sound signal, an n-channel encoded signal is produced, where n depends on the spherical-harmonic order used. First order corresponds to the *B-format* encoding, which uses 4 channels of audio for a periphonic system. The encoded signal is independent of the target reproduction system, and can be decoded to loudspeaker setups with different loudspeaker numbers and geometries, using a set of decoder coefficients. A more faithful reconstruction of the sound field requires higher spherical harmonic orders (*Higher Order Ambisonics* or HOA [3]), which in turn require higher channel counts for the encoded signal and a higher number of loudspeakers. In early formulations only the encoding of plane waves was considered, more recently spherical wave encoding has also been added to the theory (*Near-field Compensated HOA* or NFC-HOA [50]), taking into account the finite distance from the source to the listener. *Focused sources* [51], virtual sources located inside the loudspeaker array and sources with arbitrary directivity [52] have also been described.

Sound fields encoded using the Ambisonics format can be obtained not only by synthetic means, but also using multi-capsule Ambisonics microphones [53]. Once the sound field is encoded different holistic transformations can be applied, such as rotation, zoom or stretch.

The ambisonics technique is quite practical for storage of spatially encoded signals since, for instance, second order ambisonics only requires 9 channels of audio, which can be decoded to a higher number of loudspeakers. The "sweet area" of ambisonics increases with the order used for encoding and decoding, that is, for off-centre listening positions the localization error is lower with higher orders [54].

Ambisonics encoders and decoders are available as plugins to be used in DAWs *[55]* and as ugens for *Max/MSP [56]*, *Csound [57]* or *SuperCollider [58]*. NFC-HOA as been implemented in the *SoundScape Renderer [59]* and Ircam's *Spat* [60].

Wave Field Synthesis

Wave Field Synthesis (WFS) is also a sound field reconstruction technique, proposed by Berkhout [61] and based on the Huygen's principle, which states the equivalence between an original wave and another generated from a distribution of secondary sources located on one wavefront of the original one. This technique employs a grid with a high number of loudspeakers densely packed around a listening area. Although the theory allows for periphonic (3D) reproduction, this would require a very high number of loudspeakers, therefore efforts have focused on reproduction on a plane at the listener's ear level [5]. For plane reproduction usually a rectangle is used, although other shapes are admitted by the theory. Inside this rectangle all listening positions after a certain minimum distance from the loudspeakers are nearly equally performant, as such there is no sweet spot. Apart from known artefacts, an entire region of the intended sound field is correctly reconstructed in the listening area. Because of this, a listener moving inside the loudspeaker array will perceive near-field sources to be completely static even for large changes in position. WFS can place virtual sources in the auditory space with great accuracy, in listening tests Verheijen found that the localization blur for real and virtual point sources behind a loudspeaker array was indistinguishable [62, p.143]. WFS also allows positioning virtual sound sources inside the listening area, *focused sources*, using a time-reversal technique, although this only reconstructs the intended sound field in a subregion of the whole listening area. WFS can generate both plane waves and spherical waves from near-field sources, as well as sources with arbitrary directivity [63]. Trajectories produced in WFS systems can cross seamlessly from outside to inside the loudspeaker array, creating the feeling of complete freedom of movement in an horizontal plane and rendering the loudspeakers almost undetectable[12]. Static focused sources are nevertheless not as convincing as moving sources, specially if the signal is not broadband.

Due to creating physically correct sound fields, sound sources have an overall delay which is proportional to the distance to the listener causing moving virtual sources to pitch-shift due to the Doppler effect, and sources at different distances to be out

[12] After having worked extensively over a period of 5 years with WFS systems, when I transitioned back to using more conventional systems, with much lower number of loudspeakers, I had a very strong impression that the virtual sources were stuck to the surface defined by the loudspeakers, neither receding back nor approaching the centre of the listening room.

of sync. Depending on the compositional goals this might not be desired[13] and it is possible to cancel somewhat these effects[14].

WFS suffers from spatial distortions of the reconstructed wave field due to aliasing, which is caused by using a discrete instead of continuous set of secondary sources, and the truncation effect, caused by the use of finite instead of infinite arrays. Due to aliasing, accurate sound field reconstruction is limited to sources with frequencies below 1000Hz with typical loudspeaker spacing distances. WFS systems currently have a high cost, requiring powerful computers and large numbers of audio channels and corresponding loudspeakers.

Given their size and complexity, these systems are usually owned by institutions or universities, and often use custom software. As an example, the WFS system of TU Berlin university features a rectangular array with 832 audio channels [66] and uses the *sWONDER* software [67], which spatializes audio-signals generated elsewhere in real-time using OSC network messages. The WFS system of the Game Of Life foundation is mobile and mostly used for electroacoustic music concerts, it features an array of 192 loudspeakers and uses the *SuperCollider*-based software *WFSCollider [68, 65]* for real-time spatialization and computer synthesis. As will be mentioned in chapter 4, *ImmLib* makes use of some components of *WFSCollider*[15].

Binaural synthesis

Binaural synthesis generates virtual sound sources using headphones by presenting at the eardrums the same signals which would be created by an equivalent real sound source in free-field [71, 72]. The most common method is to measure the head-related transfer functions (HRTFs) or binaural room impulse responses (BRICs) and convolve the source signal with a pair of HRTFs or BRICs corresponding to the desired virtual

[13]Marimoto explicitly states that his motivation for employing a custom spatialization technique, to be described later, which he termed *NoWFS* was that the Doppler effect “could easily become disturbing and lead to a situation of being surrounded by a large number of accidentally sweeping sounds” [64, p.42].

[14]In *WFSCollider*, a *SuperCollider* based WFS rendering software, there is a panning parameter “Latency compensation / Doppler reduction” which can be continuously changed from no compensation to full compensation. The parameter reduces these effects by removing the overall delay relative to the centre of the room, leaving only the inter-speaker delay differences [65].

[15]I have been developing tools for composition with WFS systems since 2008 [69, 70] and from 2010 to 2012 I worked as a software developer for the Game of Life Foundation, contributing to version 2.0 of *WFSCollider*. This new version extended the software from simple file playback to the full gamut of possibilities allowed by *SuperCollider* synthesis definitions. Version 2 of *WFSCollider* was rewritten almost from scratch in the form of the *UnitLib* library together with a set of WFS panners as synthesis definitions (Udefs).

source position. It is known that head tracking can improve sound source localization in binaural synthesis [73]. The most direct way to implement both head-tracking and moving virtual sources is to perform time-varying interpolation between different HRTFs, although this is technically complex [74]. Binaural synthesis using virtual ambisonics is a simpler alternative requiring only static HRTF convolution [75] where a sound source is first encoded into a set of Ambisonic signals which are then decoded for a set of virtual loudspeakers positioned such that HRTFs are available for those locations. Both head-rotation and source movement can be easily implemented solely on the Ambisonics signal chain. An extension of this method directly associates HRTFs with the plane wave expansion of the encoded sound field [76].

2.4.2. Decorrelation

It was referenced in section §2.3 that multiple sources with decorrelated signals can create an auditory event with enlarged extent and that several such sources together can create a diffuse auditory event enveloping and even surrounding the listener. Decorrelation is therefore an important technique for creating diffuse sources or for controlling source extent[16] in electroacoustic music. It can also be used to create a sense of spaciousness connected with the perception of being inside an enclosed space, which is characteristic of the diffuse reverberation field. In particular, being able to create multiple decorrelated signals is essential to the working of *ImmLib*.

Decorrelation in sound reproduction consists of taking an original recording or live signal and creating an output signal which is decorrelated from the original but sounds essentially the same.

Kendall [11] provides an algorithm for generating multiple output signals with varying degrees of correlation relative to an original signal. The algorithm produces the decorrelated signals by using randomized phase values on the spectral domain. With this technique it is possible to produce an unlimited amount of signals with a correlation measure in a continuous range from $+1.0$ (complete correlation and in-phase) to -1.0 (complete correlation with 180° phase), with the value of 0 (no correlation) being specially important. To obtain the decorrelated signal the original signal is sent through a finite-impulse-response filter (FIR). The coefficients of this filter are obtained via

[16]By source extent is meant the extent of the auditory event created by a physical source or by a panning technique.

inverse fast Fourier transform (IFFT) from a specification spectrum with unity magnitude for all frequencies, and randomized phase. Given a pair of output signals it is possible to manipulate the coefficients in order to have any degree of correlation from 1.0 to -1.0 by interpolating between two different randomized phase specifications. To obtain n decorrelated signals one needs n randomized phase specifications.

Given that the magnitude of the filter frequency response is unitary it is an all-pass filter, and should not alter significantly the timbre of the original source. In practice there is some change in timbre due to this process, with a higher number of points for the filter producing greater timbral neutrality. Nonetheless, a greater number of points also causes the temporal duration of the filter's impulse response to increase, therefore the number of coefficients has to be chosen with care. This technique is used in *ImmLib* for decorrelating recorded signals and live input[17].

An alternative to FIR filters is to use IIR all-pass filters by placing the poles and zeros at random distances from the unit circle in the z-plane. This approach has the advantage that the filter coefficients can be continuously randomized in real-time which causes a "spatial effect akin to the sound of an environment with moving reflecting surfaces or moving sound sources, such as the movement of leaves and branches in a forest or the movement of a crowd within an auditorium" [11, p.77]. There are other, possibly more timbraly-neutral techniques available [12], but currently they do not operate in real-time, and as such are not relevant for this work.

In electroacoustic music and computer synthesis, where there is not necessarily an original signal which must preserved, more aggressive or creative techniques can be used, since in this case all that is required is that the multiple output signals have strong spectral similarities. In computer music this can be achieved by obtaining n decorrelated signals from the same synthesis definition. In Music-N systems, a synthesis definition or instrument, can be instantiated with different values for its parameters[18], creating different signals. Also, typically if the synthesis definition uses stochastic unit generators (ugens) based on random number generators, these are automatically given different random seeds[19]. The use of different random seeds for each instance of the

[17]This technique was implemented by Wilson [22] as the `PV_Decorrelate` plugin in *BEASTMulch* [77] which is used by *ImmLib*.

[18]A synthesis definition is created by interconnecting different sound modules, called *unit* generators. The Music-N architecture, including the concept of unit generator, will be introduced in more detail later.

[19]A computer algorithm cannot generate truly-random numbers, since it is deterministic. Random

definition ensures the signals from each instance are decorrelated, although the amount of decorrelation achieved will depend on how much the stochastic ugens influence the final audio output. A definition consisting of only one white-noise ugen, with different seeds for each instance will create completely decorrelated signals. On the other hand, the signals generated from a definition consisting of one sine oscillator of fixed frequency which is amplitude modulated by a white noise generator whose output is scaled between a and 1.0 will have a decreasing decorrelation measure as a approaches 1.0. It is also possible to achieve decorrelation with definitions containing only deterministic ugens, such as oscillators and other signal generators, by setting a parameter of the definition related to those ugens to different values in each instance. For instance FM synthesis with different values for modulator frequency or amplitude.

When dealing with recordings, besides using Kendall's decorrelation technique one can time-shift $n-1$ times the same recording by $n-1$ different amounts which will create n decorrelated signals [28]. Given the definition of decorrelation being used in this work, which is considers a small time window, it is highly likely that a sound recording and a delayed copy of the same recording will be decorrelated, as long as the delay time is large enough, for instance more than 10s, since in this case the content being analysed in the time window will never overlap. Off course, if the content is periodic it is still possible that the signal and its delayed copy will always match even with small values of τ, in that case this approach will not work. If the recorded material is being processed using other ugens further down the digital signal processing (DSP) graph, we can apply the techniques mentioned in the previous paragraph to those ugens and still achieve decorrelation. Some techniques such as granular synthesis [78, 79], concatenative synthesis [80] or buffer scratching are specially good at achieving multiple decorrelated signals from a single mono recording.

numbers are generated in computer programming using *pseudo-random* generators, which create distributions statistically similar to truly random distributions. These algorithms depend on an initial value, the random seed, and given the same initial value produce the same sequence of values. On some Music-N systems the random seed must always be passed explicitly.

2.5. Working with decorrelated bundles and similar techniques

In this section I will situate *ImmLib* amongst related approaches. There are many examples of sound works making use of what I called decorrelated bundles (see section §1.1) and similar approaches. Different artists and composers use such methods with different goals in mind, although all seem to be interested in a higher level of envelopment or working with spatial attributes other than position. I will start by looking at acoustical or electromechanical works, followed by computer-controlled electroacoustic works and related software.

Iannis Xenakis in his orchestral work, *Terretektorh*, premiered in 1966, places 88 musicians scattered among the audience inside a circle with a radius of about 40 meters with *Nomos Gamma* (1967-1968) having a similar arrangement with 98 musicians [81]. At certain moments four percussion instruments assigned to each player create a stochastically distributed cloud occupying the whole performance space which is “stationary but also permanently changing” [82, p.3]. Maracas and sirens also distributed amongst the players create a texture occupying the listening space but featuring coherent internal movements. This also happens with sustained chords which are transferred from player to player.

Untitled Sound Objects [83], a series of kinetic sound sculptures by Swiss duo Pe Lang and Zimoun, is another entirely acoustic example. The sculptures feature a large number of small DC motors (sometimes as many as 400) driving a sound emitting mechanism placed equally distanced on different surfaces, from a single wall to an entire room including floor and ceiling:

> “Despite the uniformity of presentation and setup, the motors quickly fall out of sync with each other - the chains rattle on at different rates of speed, creating a waterfall-like white noise, and the nubs hit different spots of each cubicle, emitting more of a machine-shop hustle & bustle hum. Likewise, the motions of the chains, which should be nearly identical, are quite varied, each doing their own kind of dance at their own pace.” [84]

Coincidence Engine One: Universal People’s Republic Time (2008) by “The User” collective (Emmanuel Madan / Thomas McIntosh) is a sound installation with 1200 small alarm clocks placed in a structure with the shape of a small amphitheatre that surrounds a single listener. When in the centre of the structure the click sound due to the movement of the "second" hand of each clock creates rhythmic textural waves, since the clocks are not synchronized. In a similar work, *Coincidence Engine Two: Approximate Demarcator of Constellations in other Cosmos* (2008), 96 clocks are organized in a rectangular grid placed on a wall. The clocks are electronically modified, with the "second" hand movement computer-controlled. The choreographed movement of the "second" hand across the multiple clocks, and the click sound thus generated, creates both rhythmic and spatial patterns.

In another electromechanical example, *Pneumatic sound field* (2006) by Dutch artist Edwin van der Heide, a 7×6 $(= 42)$ horizontal over-head grid of independently controllable discrete valves spanning 10 meters by 20 meters creates a spatial choreography of sharp percussive sounds. Compressed air (instead of loudspeakers) is used to produce quick pressure changes, and therefore sound. The valves are activated with different synchronicities in order to create changing spatial rhythms and spectral patterns: "By using different speeds, delays and repetitions a continuum is being created between the spatial rhythmical patterns, spatial localization of sound, movement of sound and the perception of tones and pitches" [85].

We now focus on electroacoustic examples starting with the sound installation *S0UND B1TS* (2002) by Robin Minard, specifically the iteration presented at the Stadtbad Oderberger Straße in Berlin, a public bath dating from 1900. The installation used 3 ADAT connections for a total of 576 independent 1-bit channels assembled as a 12×48 grid mounted on the front wall of an empty large swimming pool, into which visitors were invited to descend. When recordings and simple synthetic sounds are fed through the system, the original audio bit stream is split with each bit going to a different loudspeaker, creating interesting timbral artefacts and spatial textures:

> "The sound produced by the installation was based on textures moving in waves and geometrical trajectories over the wall. Embedded into the particular acoustics of the bath (reminding of a cathedral) the installation created a unique listening experience. Visitors could freely walk around in the bath and enter the empty pool to listen to the installation from very different perspectives." [86]

In the first example featuring a multichannel full-resolution computer-controlled system, Steve Heimbecker's *Signe* (2008) makes use of his turbulence sound matrix diffusion system. This system uses data collected by the wind array cascade machine (WACM), a network of 64 sensors deployed in a grid that records the behaviour of vertical sticks under the influence of wind, essentially capturing a time-varying vector field which is then used to modulate the frequency and amplitude of sine waves. There is a one to one correspondence between wind sensors and loudspeakers, with the diffusion system composed of 8 concave columns with 8 loudspeakers each that completely surround the listener both horizontally and vertically. In *Signe* wind data is also used to diffuse recordings of typewriting and piano sounds. The WACM captures a whole slice of the wind intensity field, when this data is applied to synthesis parameters maintaining the same grid positions between sensor and loudspeaker, the coherence of the recorded field is perceived in spatial attributes of the audio output.

Using an ingeniously simple design, *A World Beyond the Loudspeaker* (1998) by Edwin van der Heide, is a sound installation featuring a vertical grid of 4×8 full-range loudspeakers placed side to side, playing a composition created from recordings obtained with an equivalent array of microphones. The loudspeaker array performs wave field synthesis of the recorded sound field, therefore being in front of the array is as if one was in front of a window into the original space. In this specific case although the signals of each loudspeaker are somewhat decorrelated the waves from each loudspeaker actually interfere mid-air recreating the original sound field physically, the auditory system therefore only perceives what was in the original sound field. Whether the listener will perceive diffuse sources or not will depend on what was captured on the recordings. This installation is a unique case where it was possible to completely capture all the complexity of an original spatial impression[20].

We now move on to computer-music examples of the use of decorrelated bundles, where the specific technical approach is well documented in peer-reviewed literature. Wilson's spatial swarm granulation [13] technique employs boids[21] to control the location of streams of granulated sampled sound. When a grain is spawned the position

[20] The same limitations of WFS apply, spatial reproduction is only accurate up to the sampling frequency, possibly around 1000Hz.

[21] *Boids* is a computer algorithm, created in 1986 by Craig Reynolds, which simulates the spatial movements, in particular the flocking behaviour, of birds. Groups of boids can be controlled using relatively small number of global rules such as separation, cohesion, alignment and attractors.

of the corresponding boid is used to assign the grain to the closest loudspeaker (no panning is used). Wilson reports that system successfully created sources which are both diffuse and localized, and which can be moved through the listening space. With certain settings related with avoidance rules “a subjective ’liveliness’ (somewhat akin to the effect of rustling leaves)” was reported. Since summing localization was not used the effect was quite immune to the precedence effect.

In Kim-Boyle’s spectral spatialization [14], multiple output signals for loudspeakers are derived by applying different fast Fourier transform (FFT) operations to an input signal. In one approach spectral delays with different mappings were applied to each loudspeaker signal. The mapping specified how much delay to apply to each FFT bin of one output signal, and was different between loudspeakers. The spectral delay mappings were stored in buffers filled with random or user-supplied[22] data, and the different buffers could be cross-faded for dynamic changes. Each output signal also had a different spectral amplitude mapping[23]. The delay times used ranged from dozens to hundreds of milliseconds. The original signal was sent to all loudspeakers simultaneously but with different spectral components appearing first in different loudspeakers. The other approaches described by Kim-Boyle perform multi-loudspeaker summing localization of each bin, using an algorithm similar to VBAP, with data from particle systems, boids or stochastic systems used to control the location of each bin. Perceptually, the method created at times a single spatially diffuse source, or if the delays were larger, the location of individual spectral components was perceivable with “certain spectral components of the sound lagging behind others or preceding them” [87, p.142]. Constraining the boids to a small region and then allowing them to extend to the whole space created a gesture where a source would go from being localized at a single location to becoming diffuse, spreading over the whole space. Whether groups of bins merge or not into a single auditory event was influenced both by the principles of spatial hearing and of perceptual grouping outlined in section §2.3.

Spatio-Operational Spectral Synthesis (SOS) [88] operates on a similar principle, with different spectral components or harmonics, obtained from a synthesis process, assigned to different loudspeakers or trajectories. SOS theory directly references Bregman’s Auditory Scene Analysis [89], exploiting the fact that when components from the same

[22]Possibly through “drawing” the contents of the buffer with the mouse.
[23]A specification of the gain change to apply to each FFT bin.

complex harmonic sound are spatialized to different locations they can still merge into a single auditory stream and depending on the degree of locatedness, can be localized to a certain position in space. On the other hand if the components are sufficiently transformed or tampered with, multiple auditory streams will appear. Working close to the segregation threshold, with the streams merging and segregating over time, can create interesting spatial sensations.

In Lyon's image-based spatialization [90] an image with 32 pixels along the x-axis and n pixels along the y-axis is progressively scanned along the y-axis with the 32 pixels of each horizontal line mapped to synthesis parameters of sound processes panned at 32 different positions around a circle. This technique was used in the composition *Spaced Images with Noise and Lines* (2011) with the intensity of each pixel controlling at times different synthesis parameters. In the simplest mapping an input signal is sent directly to all 32 outputs with the amplitude of each output determined by the pixel intensity. In this case summing localization will take place, thin oblique lines generate easily perceivable spatial trajectories, while noise creates a more indeterminate spatial image. In another mapping the pixel intensity controls a resonant frequency[24] with amplitude controlled by a second image. Alternatively each spatial location had a fixed resonant frequency and the image would control the amplitude as in the first case which would result in a spatialized chord for a noisy image and a melodic progression for a line-based image. In another section sinusoidal tones with a percussive envelope were generated at each spatial position with a random frequency selected from a fixed range, with the pixel intensity controlling the sinusoid amplitude and envelope duration:

> The result is a constant, quiet, sort of thudding background articulation (albeit spatially complex) from the low-amplitude pixels, and a sporadic, bell-like surface melody with clear spatial articulation from the high amplitude-pixels. Although sine waves are often difficult to resolve spatially, the single pixel mapping to a fixed virtual location, combined with the percussive attack envelope, locates the individual sinusoids starkly in different regions of the space.

Finally, in Marimoto's *Lifegame* [64] (2009) a 192 loudspeaker rectangular grid[25], usu-

[24] Possibly of a bandpass filter, the paper does not specify.
[25] The Game of Life WFS system [91].

ally driven with WFS rendering software, has 192 synthesis processes running in parallel connected directly to each loudspeaker. In the final section of the piece each loudspeaker is put in correspondence with a cell of a cellular automaton which evolves according to the specified rules and the state of the neighbouring cell/loudspeaker. The state of the cell is used to mute or not mute the output of a chaotic logistic synthesis process.

Across all these examples we can see the deployment of multiple similar processes across different locations in space. The electromechanical examples tended to use percussive events whose rhythmic differences created spatial textures, with some employing a stochastic approach to the event timing and others carefully coordinating the events according to deterministic rules. Many of the electroacoustic examples divided an original signal into multiple output signals, often through spectral splitting, applying further transformations to these (spectral spatialization, image-based spatialization). In other cases such as spatial swarm granulation, the turbulence sound matrix, and Marimoto's *Lifegame,* multiple spectrally related but decorrelated outputs were spatially choreographed. Many of the described techniques apply to horizontal systems although some, such as the turbulence sound matrix, *A World Beyond the Loudspeaker,* and spatial swarm granulation, used three-dimensional systems placing sound along a surface. Several of the authors referenced that the merging and segregation of auditory streams plays an important role in this type of spatialization.

Chapter 3.

Model and Algorithm

This chapter introduces parameter field spatialization, a technique for spatialization of sound events in computer-music audio-synthesis environments through the creation of decorrelated bundles[1]. This technique is based on a mathematical model of spatial surface patterns, a perceptual spatial attribute of sources with width and height (or generally, with extent). This spatial attribute is related with the specific distribution, over an area of space, of differences in a perceivable attribute of sound. The objective is to create auditory events which have large width and height, possibly entirely surrounding the listener, and to create a pattern along the extent of the auditory event which can be precisely controlled.

The mathematical model is based on the differentiable geometry of surfaces, with perceived patterns being equated with abstract mathematical patterns, which can be thought of as moving images projected onto a smooth curved surfaces. This chapter will introduce the relevant mathematical theory, in particular, analytical[2] descriptions of surfaces, a method for calculating the distance between two points on a curved surface, and methods for selecting uniformly distributed points on a surface. It will also introduce parameter fields, which are real-valued functions[3] whose domains are surfaces. The parameter field is a key concept and as such a significant portion of the chapter is spent introducing its mathematical formulation and associated computational

[1] As introduced in chapter 1, a *decorrelated bundle* is a group of spectromorphologically-related decorrelated audio signals spatialized simultaneously at different locations in space.

[2] Synthetic geometry is the study of geometry through axioms and propositions derived through logic arguments (an example is Euclid's *Elements*), while analytical geometry is the study of particular geometric objects through coordinates, equations and functions. Analytic geometry is characterized by being constructive, that is, it makes possible the explicit calculation of geometric properties of interest, such as intersection points or surface normal vectors.

[3] A real-valued function is a function whose codomain is the real numbers: $f : A \to \mathbb{R}$. It maps every element of A to a real number.

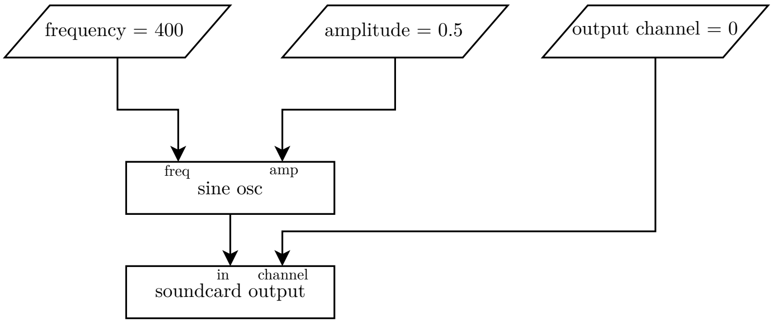


Figure 3.1.: A sound process definition composed of a unit generator graph. The rectangles are unit generators, the slanted boxes are parameters of the ugen graph, which can be set directly by the user. "freq" and "amp" are inputs of the "sine osc" ugen and "in" and "channel" are inputs of the "soundcard ouput" ugen.

algorithm.

This chapter will detail an algorithm for the generation, from a given parameter field, of multiple concurrent control signals connected to a parameter of a sound process in order to create a spatial surface pattern. Although the focus will mostly be on parameter fields, since this is the technique implemented in *ImmLib,* alternative control-signal generation methods using discrete sequences (instead of continuous functions) are also briefly examined. Finally, specific parameter fields featuring different shapes and behaviours are introduced, describing for each its mathematical formula and the parameters that can be manipulated in order to change the pattern.

3.1. Overview

We start with a general overview of the technique. Parameter field spatialization is intended for groups of loudspeakers laid out in a two-dimensional spatial configuration covering spaces such as domes, spheres, single walls or ceilings, or even the entire surface of a room[4]. It is not intended for unidimensional loudspeaker setups such as 5.1 surround systems or n loudspeakers equally spaced on a circle, as those systems can create auditory events with width but not height. This was a conscious decision on the part of the author. It is possible to apply an equivalent technique to unidimensional setups [90], creating unidimensional spatial patterns, nevertheless it was decided that the focus of this research would be bidimensional patterns.

The technique spatializes a sound process by creating multiple instances of the same

[4]Four walls plus floor and ceiling.

sound process definition, which are spatialized at points uniformly distributed on an imaginary surface, conceptualized as hovering over the loudspeaker grid.

In MUSIC-N style languages, such as *CSound* and *SuperCollider*, "the sequence of numbers corresponding to musical sounds are generated by simulated instruments built up from combinations of unit generators. Each unit generator is a small block of computer instructions performing a given operation such as that of an oscillator" [92]. A sound process is then a computer routine that generates an audio signal based on a graph of unit generators (ugens), its definition[5] (see an example in figure 3.1). A sound process definition or instrument, can be instantiated multiple times, concurrently or sequentially, using different values for its parameters. A parameter of a sound process is a node in the ugen graph which can be set directly by the user to a specific numerical value. It will also be called a *control*[6]. Typical examples of parameters of a sound process are frequency of oscillation, amplitude of the output signal, and durations of envelope stages. Summarizing, from a single sound process definition, different sound process instances are created, each one generating a different audio signal, which is sent to a different loudspeaker or virtual source (see figure 1.2). In order for an extended auditory event to be created the signals must be decorrelated, methods for achieving this have already been discussed in section §2.4.2.

The generated signals can be spatialized in two different ways, either each signal is sent directly to a loudspeaker or it is associated with a virtual source to be spatialized using an appropriate panning technique. The loudspeaker system and spatialization method used must be able to place a virtual source on this virtual surface in order for the technique to work correctly, for instance a full sphere requires loudspeakers above and below the floor level.

The spatial surface patterns are created by associating a mathematical function whose domain is this conceptual surface, with a parameter of the sound process. This function will be called a *parameter field.* As detailed above, multiple instances of the same sound process are run concurrently, each being associated with a different spatial

[5] *A* ugen graph connects inputs of one ugen to other ugens. In *SuperCollider* a sound process definition is called a `SynthDef`, in *CSound* it is called an instrument definition, part of an orchestra file. In *Max/MSP* there is essentially just one global instrument which can be reassembled during playback, while the ugens are usually called *audio objects*. `poly~` allows some modularization of the ugen graph.

[6] In *Max/MSP* these are number boxes, in CSound instrument i-variables, and in SuperCollider `NamedControl`s.

location and, since they are all created from the same definition, each having the same parameter available for changing. In order for a spatial surface pattern to appear, a different control signal[7] is generated for each instance based on its spatial position. All the control signals, one per instance, are generated from the same parameter field. This signal is then applied to the chosen parameter of the sound process. The control signal will then modulate the parameter, that is, instead of the parameter being fixed at a static numeric value it will change according to a certain function of time. Since each instance will have a different function of time modulating the chosen parameter, its sonic output will behave differently. The perceptual integration of all the behaviours caused by the modulations from the control signals can cause a spatial pattern to appear in the auditory event.

Motivating the mathematical formalism

The reader might be wondering at this point what motivates the use of mathematical functions and surfaces:

1. **Why use mathematical functions?** We are interested in bidimensional sonic spatial properties, which change along the height and width of an auditory event. We want to use a model of these properties which can be stored in a computer in order to then purposely generate the patterns through a specific spatialization technique. What form could this model take? How can bidimensional continuous data be generated and stored in a computer? The model could for instance be a bitmap image stored in a digital format from which the necessary control signals mentioned before would be derived. The image could have been captured using a digital photo camera. The disadvantage of this approach is that, first of all the image is a discrete collection of individual pixels. Since it is not continuous, when applied to a grid of loudspeakers or virtual sources with different number of lines and columns it has to be interpolated. Furthermore it is static, it will not change in time. Another approach could be to use moving images stored in a digital format: digital video. This would address the issue of lack of temporal change, but another important property would still be missing, *parametricity*. Parametricy is the ability to control the final output by changing a set of numeric

[7] A control signal is a signal which operates at lower frequencies than audio.

parameters from which the output is generated using rules. For instance, vector graphics is a parametric approach to images, and vector-graphic animations or 3D rendering is a parametric approach to digital video. Parametricity is important when fine control and real-time interaction is desired. A photographic image has considerable complexity and definition, but once it is captured it cannot easily be changed, while a vector graphic tends to be simpler but can be changed in real-time by manipulating a small set of properties. General rules for generating bidimensional output are best expressed in mathematical form as this is the language that computers easily understand. Specifically rules for generating continuous bidimensional patterns, such as a gradient, can be effectively described by functions whose domain is a plane or rectangle.

2. **Why use differentiable surfaces?** The loudspeaker arrays admissible for this technique are not restricted to planar grids, they can be curved such as with spheres or domes. Instead of working with flat patterns we need to work with patterns which are projected onto any type of smooth surface, therefore instead of working just with functions defined on a square we need to work with mathematical functions defined on arbitrary smooth curved surfaces. Smooth surfaces are surfaces without hard edges. These surfaces have the important property that there is a plane tangent to the surface at any point, the *tangent plane*. Mathematically they are differentiable surfaces or regular surfaces, as differentiability implies the lack of hard edges. Differentiable curves and surfaces embedded in the three-dimensional Euclidean space have important useful properties: the length of a differentiable curve can be calculated as the integral of the length of its velocity vector[8]; the tangent plane allows the measurement of angles between curves and the measurement of the area of a region on a surface; on a differentiable surface there exist special curves, called *geodesics*, which are the shortest path between two points and act in many ways as if they were straight lines. Several interesting patterns defined on a surface can be described in terms of distance to a point, it is therefore crucial for defining these functions to be able to determine the distance between two points on a given curved surface. Functions defined on differentiable surfaces which make use of the distance given by geodesics ac-

[8] The derivative $\alpha'(t) = (\alpha'_1(t), \alpha'_2(t), \alpha'_3(t))$ of the curve α.

company the curvature of a surface in a "natural" way, without the distortions that happen when a pattern defined on flat surface is stretched to fit on a curved surface. Such stretching, in reverse form, can be seen in world maps, which are the projection of a sphere onto a plane. A monochromatic pattern projected onto smooth curved surface can be described mathematically by a real-valued function whose domain is a differentiable surface.

3. **Why discretize surfaces?** Computational devices can only generate discrete bidimensional output. In the case of images, a digital image is outputted by lighting individual pixels in an array forming a screen. For instance, with vector graphics, when one creates a colour gradient, the colour goes continuously from a colour a at screen position A to colour b at screen position B. This pattern can be stored as a continuous function, given by a formula, which given (x, y), a coordinate pair on a rectangle, returns the colour value $\text{color}(x, y)$. In order to see the graphic on a computer screen, this continuous representation must be transformed into a discrete array of numbers, one for each pixel in the screen. In the case of sound, the bi-dimensional data is outputted through a loudspeaker array, but the process is analogous. A flat surface such as a screen, represented by a rectangle can be discretized uniformly simply by choosing a certain number of rows and columns and equally dividing the rectangle horizontally and vertically. While computer screens are flat, the surfaces used by parameter field synthesis can be curved and an equal division in rows and columns does not produce a uniform distribution of points. Due to this issue, other types of algorithms have to be employed for selecting uniformly distributed sets of points on a curved surface.

Having given a general outline of the technique and the motivation for the use of the mathematical formalism, we will now look in detail at the mathematical model, specifically at differentiable surfaces, uniform distributions of points and functions of differentiable surfaces.

3.2. Surfaces

With parameter field spatialization a sound event is spatialized using a two-dimensional loudspeaker array creating an auditory event which can span the entire length and width of the array. The technique requires that a virtual surface be defined mathematically using a parametric representation. A parametric representation of a geometric shape is a representation that traces a given shape while depending on certain parameters controlling the general properties of the shape.

The following example illustrates the difference between a parametric and a non-parametric representation. One parametric representation of a circle is given by $f_R(\theta) = R(\cos\theta, \sin\theta)$, where the circle is traced by varying a single variable, θ, from 0 to 2π. This is different from the non-parametric representation given by the equation $x^2 + y^2 = R^2$, from which it is not straightforward to draw a full circle by varying a single variable.

The technique also requires that the surface be discretized, that is, that a finite set of points on the surface be chosen according to some criteria. This set of points will be called the *sample set* of the surface. If the signals from the decorrelated bundle are to be sent directly to the loudspeakers, then the positions of the loudspeakers form the surface sample set, otherwise, if panning is used the chosen sample set determines the positions of virtual sources. If panning is used the points must be chosen such that they are uniformly distributed on the surface so that there is equal spatial resolution in any given direction. A parametric representation allows for easy discretization [93], for instance a set of points on the circle can be obtained as the image by f_R of a set of values in $[0, 2\pi]$:

$$\{0, \pi\} \to \{R(\cos 0, \sin 0), R(\cos\pi, \sin\pi)\} \to (R, 0), (-R, 0)\}.$$

We will nevertheless see later that, although a parametrization simplifies the generation of points on the surface, these points are usually not uniformly distributed, and as such this method is of limited use for creating sample sets.

The general mathematical theory describing surfaces using parametric representations can be found in the field of differential geometry. The study of surfaces has a rich history in mathematics, going back to antiquity, for instance some properties of surfaces of revolution were known to Archimedes, but the modern formulation started its development with the introduction of Cartesian coordinates by Descartes [94] and later of differential calculus by Newton, Leibnitz and others [95]. The contemporary formulation of Differential Geometry focuses on manifolds, which are generalizations of curves and surfaces in $\mathbb{R}^n$ to other more abstract spaces, such as the real projective plane, which cannot be directly embedded in three-dimensional space without intersecting itself. For this work it is sufficient to deal with smooth surfaces which are embedded in three-dimensional Euclidean space $\mathbb{R}^3$, called regular surfaces: "Roughly speaking, a regular surface in $\mathbb{R}^3$ is obtained by taking pieces of a plane, deforming them, and arranging them in such a way that the resulting figure has no sharp points, edges, or self-intersections and so that it makes sense to speak of a tangent plane at points of the figure. The idea is to define a set that is, in a certain sense, two-dimensional and that also is smooth enough so that the usual notions of calculus can be extended to it" [96, p.52].

Definitions

The mathematical notation that is used throughout this chapter is succinctly detailed in appendix A.

If S is a subset of $\mathbb{R}^3$, then S is a *(regular) surface* if, for each point p in S, there is an open set V of $\mathbb{R}^3$ and a function $\mathbf{x} : (D \subset \mathbb{R}^2) \to V \cap S \subset \mathbb{R}^3$, where D is an open set in $\mathbb{R}^2$, satisfying certain conditions [95][9]. The function $\mathbf{x} : D \to S$ is called a coordinate chart of the surface[10] (see figure 3.2). Each coordinate chart "draws" a portion of the surface, since V is not necessarily the whole of S, and certain surfaces require multiple coordinate charts in order to be completely mapped. The coordinate chart is similar to the correspondence that a flat world map makes with the actual spherical earth.

The conditions the coordinate chart must satisfy essentially state that the surface cannot have jumps, nor hard edges, and cannot auto-intersect. The conditions also state

---

[9]

1. $\mathbf{x}$ is differentiable and has Jacobian matrix with rank 2.
2. $\mathbf{x}$ is a homeomorphism, that is, $\mathbf{x}$ is continuous, one-to-one and onto $V \cap S$ and has a continuous inverse $\mathbf{x}^{-1} : V \cap S \to D$.

[10] Do not confuse the function $\mathbf{x}$ with the first coordinate in $\mathbb{R}^3$, x, although both use the letter x they are quite different mathematical objects. They are differentiated graphically by using a bold font for the chart. The use of the letter $\mathbf{x}$ for the chart is standard in differentiable surfaces textbooks [96, 95].

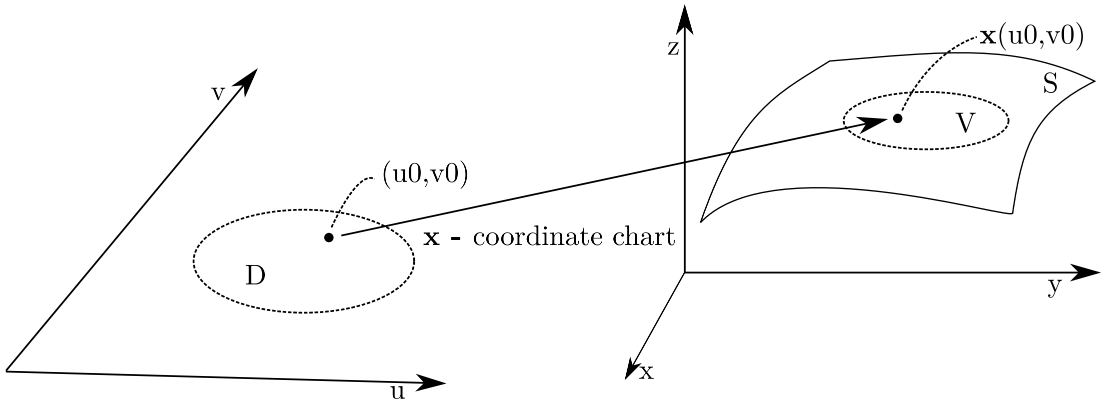


Figure 3.2.: A coordinate chart.

that to each point in D corresponds a single point in V and vice-versa, therefore the coordinate chart is a vehicle to go back and forth between a flat two-dimensional space (D) and a curved two-dimensional space (S) immersed in a larger three-dimensional space.

The sphere and cylinder are examples of regular surfaces. On the other hand, a cone is not a regular surface since the vertex cannot be mapped in a smooth manner. The pages of a book are also not a regular surface, since a point on the spine is not locally two-dimensional, given that there are always multiple pages intersecting at that location.

A coordinate chart for a surface is a function of two variables, usually notated using the letters u and v, to distinguish them from the coordinates in $\mathbb{R}^3$, so that

$$\mathbf{x}(u, v) = p = (p_x, p_y, p_z).$$

As an example,

$$\mathbf{x}(u, v) = (\cos(u)\cos(v), \sin(u)\cos(v), \sin(v))$$

with $D =]0, \pi[\times] - \frac{\pi}{2}, \frac{\pi}{2}[$ is a coordinate chart for the unit sphere (see figure 3.3). While S in this case is the whole sphere, the set V for this coordinate chart is a single hemisphere. To cover the whole sphere, 6 coordinate charts would be needed mapping to the 6 possible hemispheres which alight with the x, y and z axis.

Since the coordinate chart is bijective from D to V, any point in D is associated with a point in V and vice-versa. As such, on some occasions in this work, for abbreviation, no distinction will be made between points on the surface S and points in D, nevertheless,

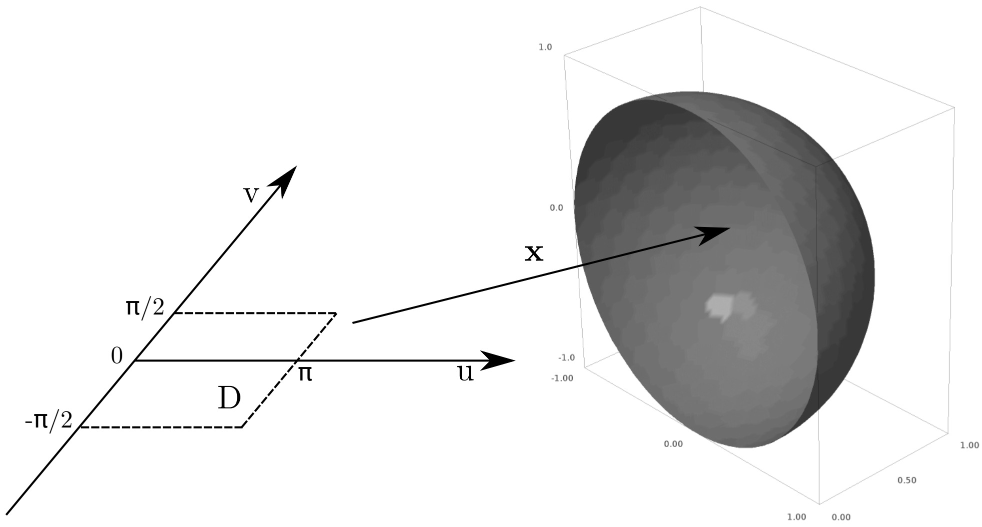


Figure 3.3.: A coordinate chart for the unit sphere.

when the distinction is important it will be made clear.

A parameter field is a real-valued function $f : S \to \mathbb{R}$ defined on a regular surface S. To each point on the surface it corresponds a real number. It can be thought of as a monochromatic image attached to a curved surface (see for instance the plot of the sphere in figure 3.32). Coordinate charts are useful for defining parameter fields since given a function $f : D \subset \mathbb{R}^2 \to \mathbb{R}$, defined on the flat two-dimensional parameter space, we can define a function $f_{\text{ext}} : S \to \mathbb{R}$ on the curved surface (see figure 3.4). Given a point in $(x, y, z) \in \mathbb{R}^3$, we can first send it back to the parameter space with $(u, v) = \mathbf{x}^{-1}(x, y, z)$, and then calculate the value of f on that point, $f(u, v)$. This motivates the following definition

$$f_{\text{ext}}(x, y, z) = f(\mathbf{x}^{-1}(x, y, z)),$$

or using function composition

$$f_{\text{ext}} = f \circ \mathbf{x}^{-1}. \tag{3.1}$$

Constructing functions on S in this way is useful as it is more natural to define functions on the two-dimensional flat parameter space than on the three-dimensional space where the surface is embedded, and for this reason in *ImmLib* parameter fields are functions of u, v, two spatial coordinates and t, time. For instance, in the case of a spherical surface,

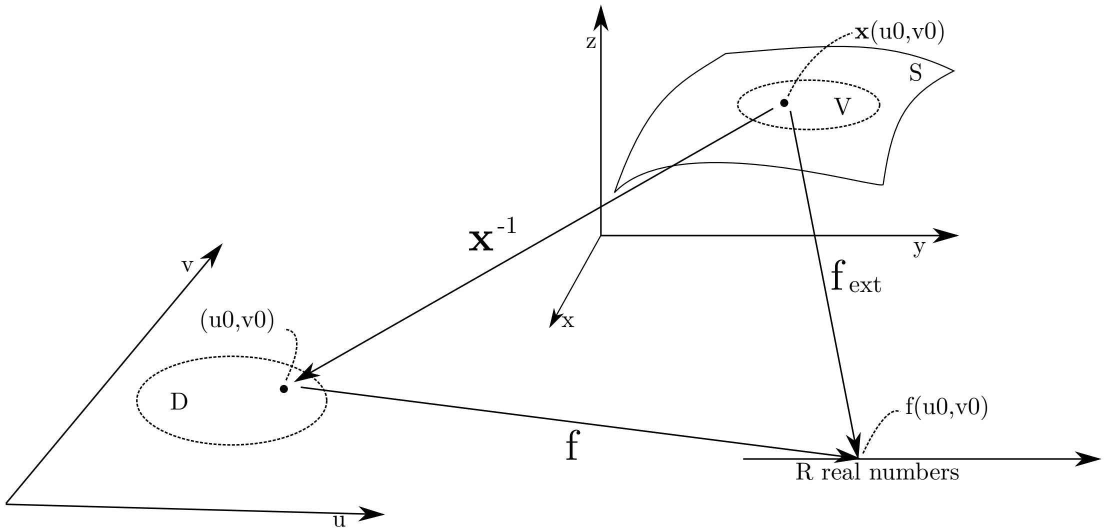


Figure 3.4.: Defining functions on a surface.

a parameter field is a function of azimuth and elevation (θ, ϕ) instead of (x, y, z).

Measuring distance on a surface

In order to create functions defined on the surface with complex spatial behaviours from simpler functions it is useful to have a way to measure distances on the surface. With a distance function on the surface it is possibly to construct a symmetrical circular pattern by "rotating" a pattern on a line around a fixed point. Multiple such patterns can be summed to create a more complex result. This is achieved by composing a given function $g : \mathbb{R} \to \mathbb{R}$ with the function that maps to the distance to a fixed point on the surface:

$$f(u, v) = g(\mathrm{dist}((u, v), (u_0, v_0))).$$

The function g can be thought of as a line with a certain monochromatic pattern which is rotated around the point (u_0, v_0) in order to form a circular pattern (as an example see figure 3.18a).

Calculating distances in $\mathbb{R}^n$ is straightforward, given $p = (p_1, ..., p_n)$ and $q = (q_1, ..., q_n)$, the distance between the two point is

$$\mathrm{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2}.$$

Defining and calculating distances on a regular surface is not as straightforward, it requires developing a rigorous treatment of geodesics, of which we give only a brief overview.

A connected surface is a surface where for any pair of points there is a piecewise regular curve[11] joining the points. Informally, it is a surface where any two points can be connected by a smooth path. Given a regular curve $\alpha : [t_0, t_1] \to S$, its length is

$$L(\alpha) = \int_{t_0}^{t_1} \sqrt{\langle \alpha'(t), \alpha'(t) \rangle} dt, \tag{3.2}$$

where $\alpha'(t) = (\alpha_1'(t), \alpha_2'(t), \alpha_3'(t))$ is the derivative of α. The distance between two points p, q on the surface S is the infimum[12] of the set of lengths for all regular curves connecting p to q:

$$\mathrm{dist}(p, q) = \inf\{L(\alpha) \,|\alpha \text{ is a regular curve, } \alpha(t_0) = p \text{ and } \alpha(t_1) = q\}.$$

The distance between two points on the surface is therefore the length of the shortest smooth path along the surface between the two points. It would be useful to have a systematic way of finding such curves, this can be done with geodesics.

A geodesic curve is a curve on a surface whose tangential component of its acceleration α'' is 0. When a curve exists between p and q whose length is equal to $\mathrm{dist}(p, q)$, it can be shown that it is a geodesic. Geodesics are therefore the curves that minimize the distance between two points on a surface. The geodesics on a surface can be calculated by solving two second-order differential equations[13], therefore in many cases given the coordinate chart of a surface it is possible to obtain a formula for its geodesics. Given a formula for a geodesic between any two points on a surface it is possible to calculate the distance between two points using equation (3.2). This is useful as it provides a direct way to calculate the distance between points for simple surfaces such as spheres or cylinders, for which formulas for the geodesics are known (see appendix B).

Coordinate chart domain

For the model being developed here, we only consider one coordinate chart per surface with domain a closed set $D = U \times V = [u_a, u_b] \times [v_a, v_b]$, which will henceforth be

[11] A regular curve is similar to a regular surface but one-dimensional.

[12] The infimum of $X \subset \mathbb{R}$ is the greatest real number that is less than or equal to all elements of X. Any bounded nonempty subset of $\mathbb{R}$ has an infimum.

[13] These equations involve the Christoffel symbols of the coordinate chart.

called the *parameter space*[14]. This corresponds to a flat rectangle. Dealing with a single coordinate chart simplifies the implementation significantly. However, because the domain is a closed set with boundary, the way in which the coordinate chart maps the boundary can cause a discontinuity to appear in the parameter field. The rigorous definition of coordinate chart requires the function to be bijective on the open set $]u_a, u_b[\times]v_a, v_b[$. By diverging from the rigorous definition and allowing a closed domain it is possible for the coordinate chart to become non-injective[15] on the boundary, in which case segments on the boundary of the domain will possibly be mapped to the same segment on the surface, overlapping (see figure 3.5).

Given a function f defined in $D = [u_a, u_b] \times [v_a, v_b]$, the parameter space, if we require its extension to S, $f_{\text{ext}} = f \circ \mathbf{x}^{-1}$ to not have discontinuities in S, then for correctness it must take the same values in the portions of the boundary that are collapsed onto the same segment on the surface, otherwise f will have discontinuities at those segments when approaching from one side of the boundary and crossing to the other side of the boundary. For instance, in the case of the sphere, if f is continuous in the open set $]u_a, u_b[\times]v_a, v_b[$, for f_{ext} not to have discontinuities in S the proposition

$$\begin{aligned}\forall \phi \in]-\frac{\pi}{2}, \frac{\pi}{2}[\exists a \in \mathbb{R}\, (f(0,\phi) = f(2\pi,\phi) = a) \wedge \\ \exists b \in \mathbb{R} \forall \theta \in]0, 2\pi[\, (f(\theta, \frac{\pi}{2}) = b) \wedge \\ \exists c \in \mathbb{R} \forall \theta \in]0, 2\pi[\, (f(\theta, -\frac{\pi}{2}) = c)\end{aligned}$$

must hold. The first line states that, for a given elevation, all points mapped to azimuth 0 or 2π, which are the same azimuth, must be mapped by f to the same value. The second line states that all points mapped to the north pole must be mapped by f to the same value, while the third line states the same for the south pole.

The simplification brought to the implementation by using only one coordinate chart is nevertheless worth the necessary care when defining continuous functions on the surface. The problem is dealt with in *ImmLib* by making sure functions always have the same value in collapsed or “glued” edges of the rectangle which is the domain of

[14] The surface parametrization should not be confused with the parameters of an audio digital synthesis process. The “parameter” in parameter space is related with the surface, while the “parameter” in parameter field is related with a sound synthesis parameter. Hopefully, these two uses of the word parameter will not confuse the reader.

[15] f is injective $\leftrightarrow \forall x, y\, x \neq y \rightarrow f(x) \neq f(y)$.

the coordinate chart.

Surface sample set

A parametric representation of a surface is one of the ingredients needed for *ImmLib* surfaces. The other ingredient is a discrete surface representation using point clouds, where the points correspond to the positions of virtual sources or physical loudspeakers. A discrete representation is obtained through a sampling procedure which arrives at a finite set of points on the surface, the *sample set*. Although generic surface sampling algorithms are available [93] *ImmLib* opts for the simpler approach of providing only specific algorithms for a predetermined set of surfaces. The sample set must have a uniform distribution, showing constant local density, in order to equally spread the sound along the surface. The simplest sampling procedure is to construct a uniform lattice on the parameter space and map it to $\mathbb{R}^3$, the problem with this approach is that the parametrization can distort the geometry of the parameter space and transform a uniform distribution in $\mathbb{R}^2$ into a highly non-uniform distribution in $\mathbb{R}^3$, therefore for most surfaces, with the exception of the plane, other more complex methods are required. As an example, if a sheet of paper with horizontal and vertical parallel lines is wrapped around a sphere the lines will converge at the north and south poles, therefore with a regular lattice more points would be mapped onto those regions than on the rest of the sphere.

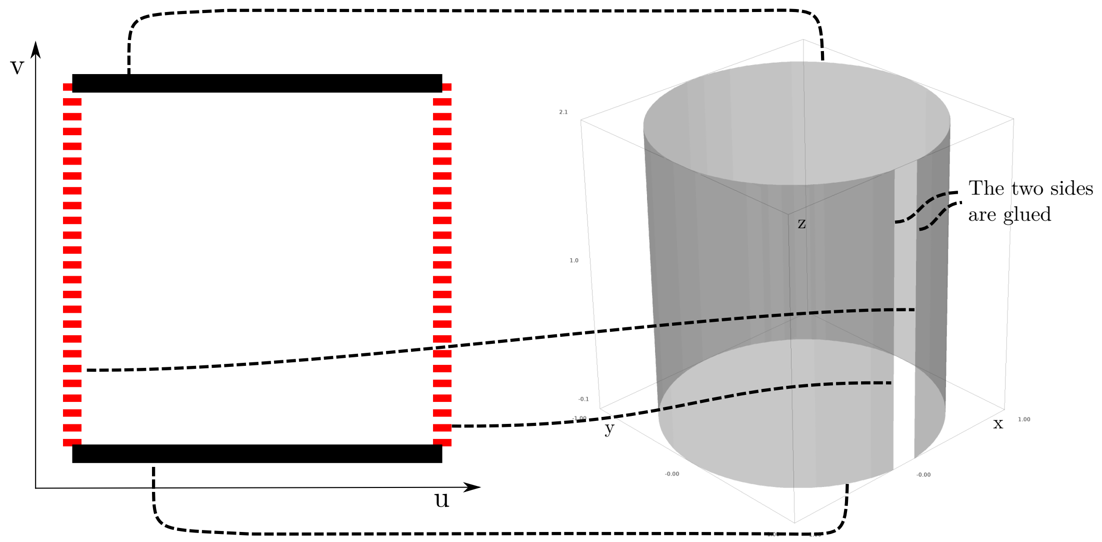
3.3. *ImmLib* surfaces

An *ImmLib* surface contains all the necessary geometric information to create the multiple modulation signals from a mathematical definition of a parameter field. It groups the continuous representation of a surface with a set of discrete points on that same surface.

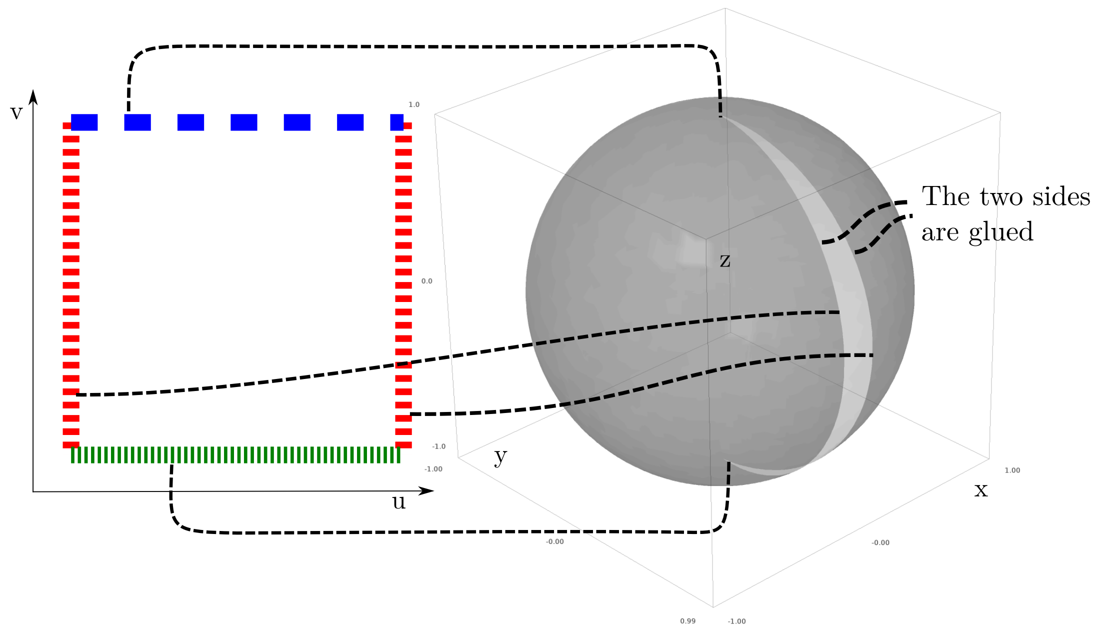
It is a group of 4 objects $(S, \mathbf{x}, \mathrm{dist}, H)$:

1. The regular surface $S \subset \mathbb{R}^3$.

2. The coordinate chart $\mathbf{x} : D \subset \mathbb{R}^2 \to S$. $\mathbf{x}$ is needed to map points in the flat rectangle into three-dimensional space and vice-versa.



(a) Cylinder parametrized from rectangle. The upper and lower borders of the rectangle correspond to the upper and lower circles of the cylinder while the left and right borders of the rectangle are glued together.



(b) Sphere parametrized from rectangle. The upper and lower borders of the rectangle are collapsed onto two points, the north and south pole. The left and right borders of the rectangle are glued together.

Figure 3.5.: A rectangle mapped to a sphere and cylinder.

3. A formula for calculating distances on the surface: dist : $D \times D \to \mathbb{R}$. The distance function is useful for generating parameter fields from symmetrical patterns.

4. A sample set of points on the surface $H = \{(u_1, v_1)..., (u_n, v_n)\} \subset D$ representing either the locations of real loudspeakers or virtual sources, with coordinates given in the parameter space.

Note that

$$\text{dist} : [u_a, u_b] \times [v_a, v_b] \times [u_a, u_b] \times [v_a, v_b] \to [0, 1],$$

the distance function of an *ImmLib* surface, is normalized:

$$\text{dist}((u, v), (u_0, v_0)) = \frac{\text{dist}'((u, v), (u_0, v_0))}{D}$$

where dist$'$ is the original distance function for the surface and D is the maximum possible distance on the surface (sphere $D = \pi$, plane $D = \sqrt{(u_b - u_a)^2 + (v_b - v_a)^2}$).

Although many different *ImmLib* surfaces can be defined, most loudspeaker setups are covered by the sphere, hemisphere, plane and cylinder. These 4 types of *ImmLib* surfaces are better suited for different types of loudspeaker systems:

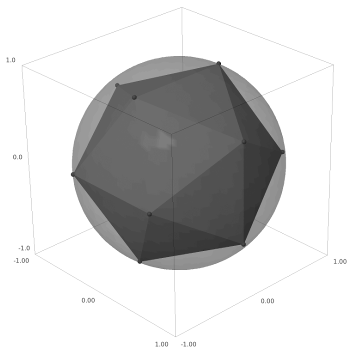
- Loudspeakers above ground at different heights $\to$ hemisphere.
- Loudspeakers above and below ground $\to$ sphere.
- Regular grid of loudspeakers placed on a flat wall or ceiling $\to$ plane.
- Regular grid of loudspeakers wrapping around the listener without loudspeakers overhead or underneath $\to$ cylinder.

Let us define these four *ImmLib* surfaces. For each surface we define the coordinate chart, the distance function and the possible sample sets. The sample set H is defined on the parameter space, the actual points on the surface can be determined by mapping the points in H into $\mathbb{R}^3$: $\mathbf{x}(H) = \{\mathbf{x}(v) \,|v \in H\}$.

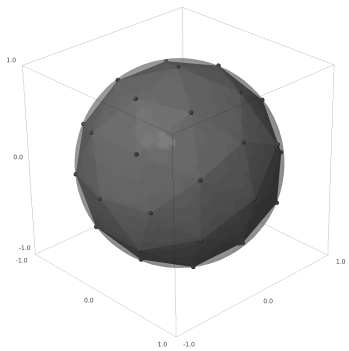
3.3.1. Sphere

The sphere with radius $R > 0$ is the set

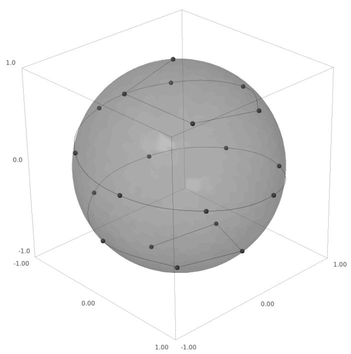
$$S = \{(x, y, z) \in \mathbb{R}^3 |\, x^2 + y^2 + z^2 = R^2\}$$



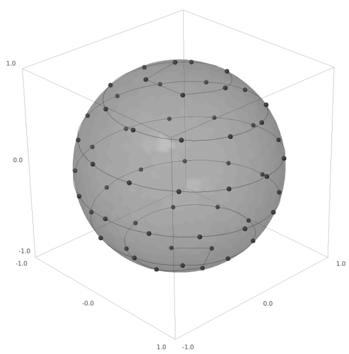
(a) 12 points on sphere corresponding to the 0th order geodesation.



(b) 42 points on sphere corresponding to the 1st order geodesation.



(c) 20 points on sphere selected using the spherical spiral algorithm.



(d) 60 points on sphere selected using the spherical spiral algorithm.

Figure 3.6.: Sphere sample sets.

with coordinate chart $\mathbf{x} : D = [0, 2\pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}] \to S$ such that

$$\mathbf{x}(\theta, \phi) = (R\cos(\theta)\cos(\phi), R\sin(\theta)\cos(\phi), R\sin(\phi)). \tag{3.3}$$

This parametrization uses two angles, azimuth (θ) and elevation (ϕ). The geodesics on the sphere are segments of great circles[16] and, with the exception of antipodal points, there is a unique great circle segment connecting any two points on the sphere. The distance between two points on the sphere is by definition the length of the segment of great circle connecting them, given by

$$\mathrm{dist}((\theta_1, \phi_1), (\theta_2, \phi_2)) = \arccos(\cos(\phi_1)\cos(\phi_2)\cos(\theta_1 - \theta_2) + \sin(\phi_1)\sin(\phi_2)) \tag{3.4}$$

(The derivation of this formula is detailed in appendix B).

Discretizing a sphere into a set of n points with a uniform distribution is a complex geometry problem with different algorithms having different trade-offs between the uniformity of the distribution and the computational cost and other practical limitations. For the sphere two different discretization algorithms are provided (see figure 3.6). The first algorithm based on geodesic domes gives the best symmetry but is limited in practice to either 12 or 42 points. The second algorithm, based on spherical spirals, is slightly less symmetrical than the previous but can generate sample sets with any number of points.

The first algorithm consists of taking the vertices of a geodesic dome, an approximation of the sphere using triangles. “A geodesic dome is a triangulation of a Platonic solid or other polyhedron to produce a close approximation to a sphere (or hemisphere). The nth order geodesation operation replaces each polygon of the polyhedron by the projection onto the circumsphere of the order-n regular tessellation of that polygon” [97]. To construct a geodesic dome we start with an icosahedron, one of the five platonic solids composed of 20 equilateral triangles, and project its vertices onto a sphere containing it. This produces 20 triangles whose vertices are on a sphere, the vertices of these triangles form the first sample set which has 12 points. The next step is to divide each triangle in 4 equal sized triangles and project the vertices of those triangles again onto the sphere, which results in 42 points equally spaced on the sphere. This process can

[16] Great circles are the intersection of the sphere with a plane passing through the centre of the sphere.

be repeated indefinitely giving sample sets of size 162, 642, 2562 and so forth. Since it is unlikely that one will have access to loudspeaker systems with loudspeakers located at vertices of geodesic domes, these sample sets are more useful for virtual sources. For our purposes only the first and second order geodesations, with 12 and 42 points respectively, are useful since 162 points is above what the computer system used in this research was able to render with regards to synthesis and spatialization.

It would be useful to have a discretization algorithm for the sphere which is able to generate any number of points. An algorithm from computational geometry by Bauer [98] which selects n points along a spherical spiral was selected for this end. This particular algorithm was selected due to its simplicity and the relative uniformity of the resulting sample set. The spherical spiral sample set with n points is

$$H_n = \{f(k)|k = 1, .., n)\}$$

where the k^{th} point can be directly computed using the formula

$$\begin{aligned} f(k) &= (\theta, \phi) = (h\sqrt{n\pi}, \frac{\pi}{2} - h) \\ h &= \arccos(\frac{2k-1}{n-1}) \end{aligned}$$

which gives

$$f(k) = (\arccos(\frac{2k-1}{n-1})\sqrt{n\pi}, \frac{\pi}{2} - \arccos(\frac{2k-1}{n-1})).$$

3.3.2. Hemisphere

The (upper) hemisphere with radius R is the set $S = \{(x, y, z) \in \mathbb{R}^3 | \, x^2 + y^2 + z^2 = R^2 \wedge z \geqq 0\}$ with coordinate chart $\mathbf{x} : D = [0, 2\pi] \times [0, \frac{\pi}{2}] \to S$. All the other elements remain equal to the previous case, the sphere, with the necessary adaptations to the different domain.

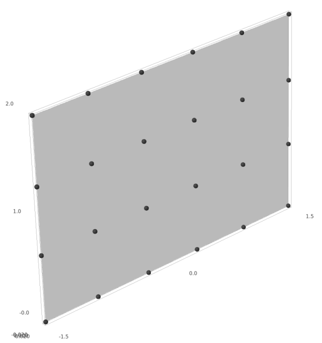


Figure 3.7.: Sample set for plane with $m = 6, n = 4$.

3.3.3. Plane

A "plane" in this context shall mean the parallelogram generated by two vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^3$ from the origin $\mathbf{o} \in \mathbb{R}^3$,

$$S_{\mathbf{w}_1,\mathbf{w}_2} = \{(\mathbf{o} + u\mathbf{w}_1 + v\mathbf{w}_2 : \, u, v \in [0,1]\} \subset \mathbb{R}^3$$

with coordinate chart $\mathbf{x} : D = [0,1] \times [0,1] \to S_{\mathbf{w}_1,\mathbf{w}_2}$, such that

$$\mathbf{x}(u, v) = \mathbf{o} + u\mathbf{w}_1 + v\mathbf{w}_2. \tag{3.5}$$

The geodesic connecting two points on the plane is the straight line that passes through the two points. The distance between two points on the plane is therefore given by

$$\begin{aligned} dist((u_1, v_1), (u_2, v_2)) &= \|\mathbf{x}(u_1, v_1) - \mathbf{x}(u_2, v_2)\| \\ &= \|u_1\mathbf{w}_1 + v_1\mathbf{w}_2 - u_2\mathbf{w}_1 - v_2\mathbf{w}_2\|. \end{aligned}$$

A sample set for the plane with uniform distribution can be obtained directly using a lattice in the parameter space with m rows and n colums since this particular parametrization does not distort the geometry of $\mathbb{R}^2$. A sample set with m horizontal points and n vertical points is given by

$$H_{m,n} = \{\frac{i}{m-1} | i \in \{0, ..., m-1\}\} \times \{\frac{i}{n-1} | i \in \{0, ..., n-1\}\}.$$

An example with $m = 6$ and $n = 4$ can be seen in figure 3.7.

3.3.4. Cylinder

The cylinder with radius R and height h is the set

$$S = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 = R^2 \wedge z \in [0, h]\}$$

with coordinate chart $\mathbf{x} : D = [0, 2\pi] \times [0, h] \to S$ such that

$$\mathbf{x}(u, v) = (R\cos(u), R\sin(u), v).$$

A geodesic $\gamma(t)$ on a cylinder going from $\mathbf{x}(u_1, v_1)$ to $\mathbf{x}(u_2, v_2)$ is a portion of a helix:

$$\gamma(t) = (R\cos((u_2 - u_1)t + u_1), R\sin((u_2 - u_1)t + u_1), (v_2 - v_1)t + v_1)$$

(see figure 3.8). The distance between two points on the cylinder is therefore given by

$$\begin{aligned} \operatorname{dist}((u_1, v_1), (u_2, v_2)) &= L(\gamma) = \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle} dt \\ &= \frac{\sqrt{(u_1^2 - 2\,u_1u_2 + u_2^2)R^2 + v_1^2 - 2\,v_1v_2 + v_2^2}u_1}{u_1 - u_2} \\ &\quad - \frac{\sqrt{(u_1^2 - 2\,u_1u_2 + u_2^2)R^2 + v_1^2 - 2\,v_1v_2 + v_2^2}u_2}{u_1 - u_2} \end{aligned}$$

(The derivation of this formula is detailed in appendix B).

A sample set for the cylinder with uniform distribution can be obtained using a lattice with m horizontal points and n vertical points contained in the parameter space:

$$H_{m,n} = \{\frac{2\pi i}{m-1} | i \in \{0, ..., m-1\}\} \times \{\frac{ih}{n-1} | i \in \{0, ..., n-1\}\}.$$

An example with $m = 10$ and $n = 5$ can be seen in figure 3.9.

The *ImmLib* software library currently only implements the sphere, hemisphere and

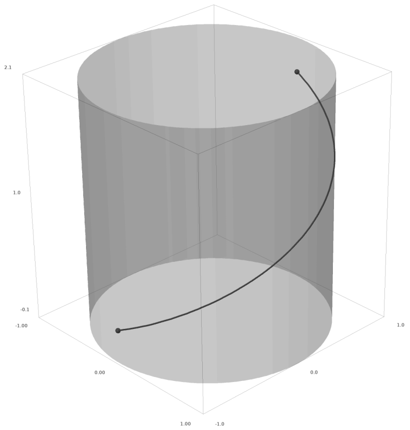


Figure 3.8.: Geodesic on a cylinder.

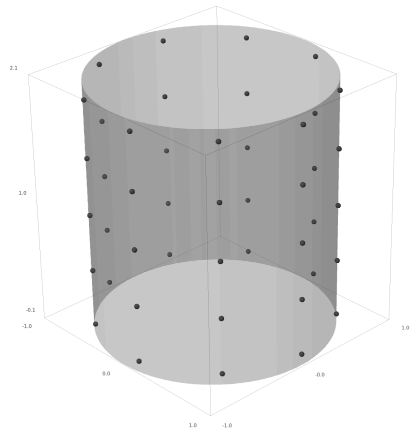


Figure 3.9.: Cylinder sample set with 10×5 grid.

plane surfaces, since only those were used in the practical work. It would be straightforward to also implement the cylinder in the software library.

3.3.5. Virtual and Direct modes

An *ImmLib* surface can be used in one of two different modes, *virtual* or *direct*. As described before, with parameter field spatialization, n instances of the same sound process definition are created for a grid with n loudspeakers. In virtual mode, the output of the instance k of the sound process associated with point $(u_k, v_k) \in H \subset \mathbb{R}^2$ is panned to $\mathbf{x}(u_k, v_k) \in \mathbb{R}^3$. It becomes a virtual source at that position in three-dimensional space. In direct mode, the sample points correspond to the actual positions of the loudspeakers in space, so that given the position $p_k \in \mathbb{R}^3$ of the kth loudspeaker the corresponding sample point is $\mathbf{x}^{-1}(p_k)$. A diagram showing both modes can be seen in figure 3.10 on the following page.

3.3.6. Loudspeaker setups used in the practical work

During the course of this research two loudspeaker systems were used. The first is installed in the Sonic Lab of the Sonic Arts Research Centre, a medium sized concert venue featuring unique architecture: the floor of the room is an acoustically-transparent metallic grid allowing sound from loudspeakers underneath to reach the audience. The loudspeaker system permanently installed in the room is composed of 4 rings of 8

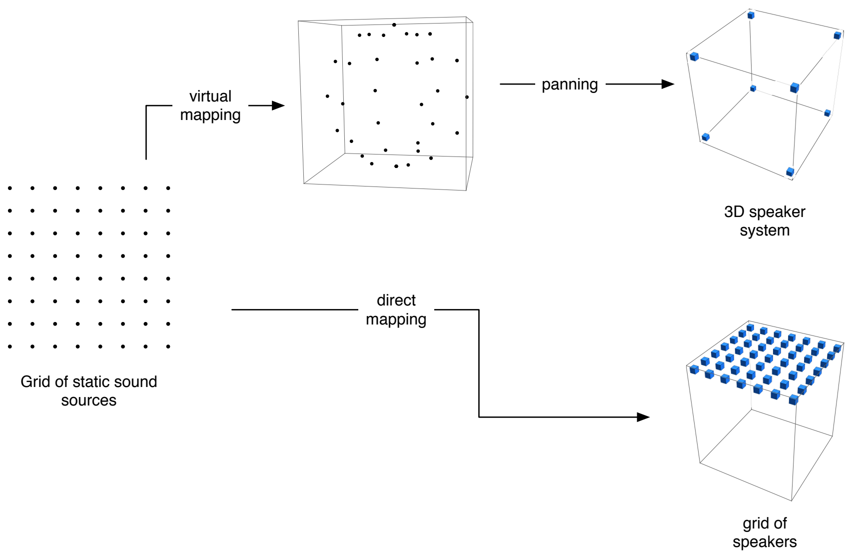


Figure 3.10.: Virtual and direct modes.

loudspeakers, with 3 rings above ground (rings 1, 2 and 3 at heights 0m, 2.8m and 5.5m) and one ring below ground (ring 4 at height $-4, 6$m). Although the loudspeaker rings at the Sonic Lab do not form a sphere, they can for all practical matters be approximated by a perfect sphere (delays are used to compensate for distance differences, although this only works for centred listeners). Although using this setup one can place a sound at any direction, looking at figure 3.11 it is clear that this system has better resolution in the upper than the lower hemisphere. The *ImmLib* surface used on this system was the sphere. Two sample sets were used in virtual mode, one with the 1st order geodesation, therefore using 42 points, the other using the spherical spiral algorithm with up to 60 points. In direct mode the sample set used the positions of the 32 loudspeakers.

The second system used was a custom-built grid of pocket loudspeakers created by Pablo Sanz Almoguera, a colleague at SARC [99]. It was configured for use with *ImmLib* into a grid with 5 rows and 5 columns with 1.5m horizontal spacing and 0.5m vertical spacing, attached to a wall of a sound studio with the lowest loudspeaker 20cm from the floor (see figure figure 3.12). This system was used from a computer equipped with a 26-channels soundcard, using the plane *ImmLib* surface in direct mode. Later a different configuration was also experimented with, using 6 columns and 4 rows, with a horizontal and vertical spacing of 50cm.

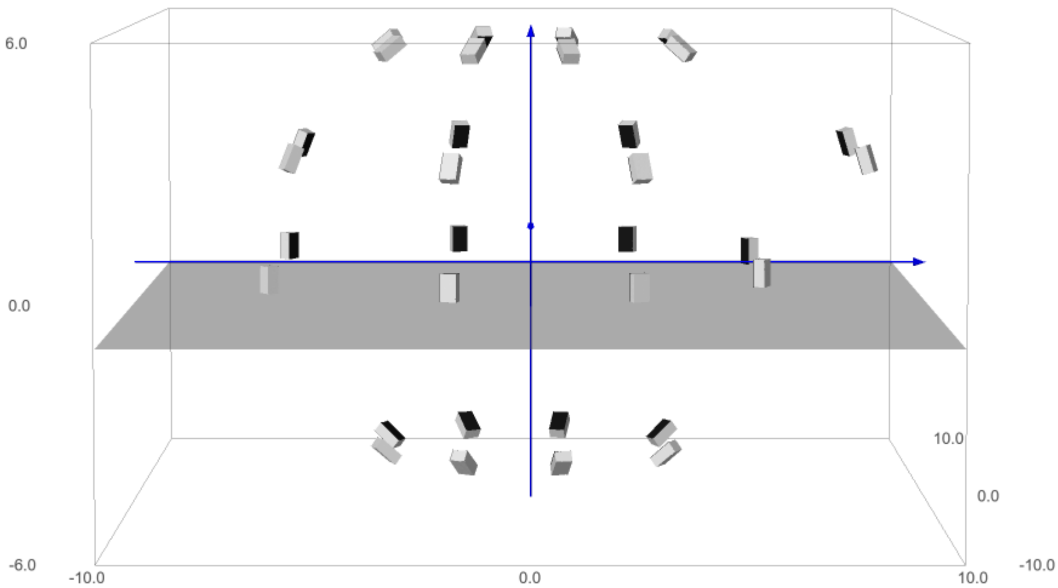


Figure 3.11.: Sonic Lab loudspeaker system. The grey area represents the metal grid floor, the arrow is at ear level.

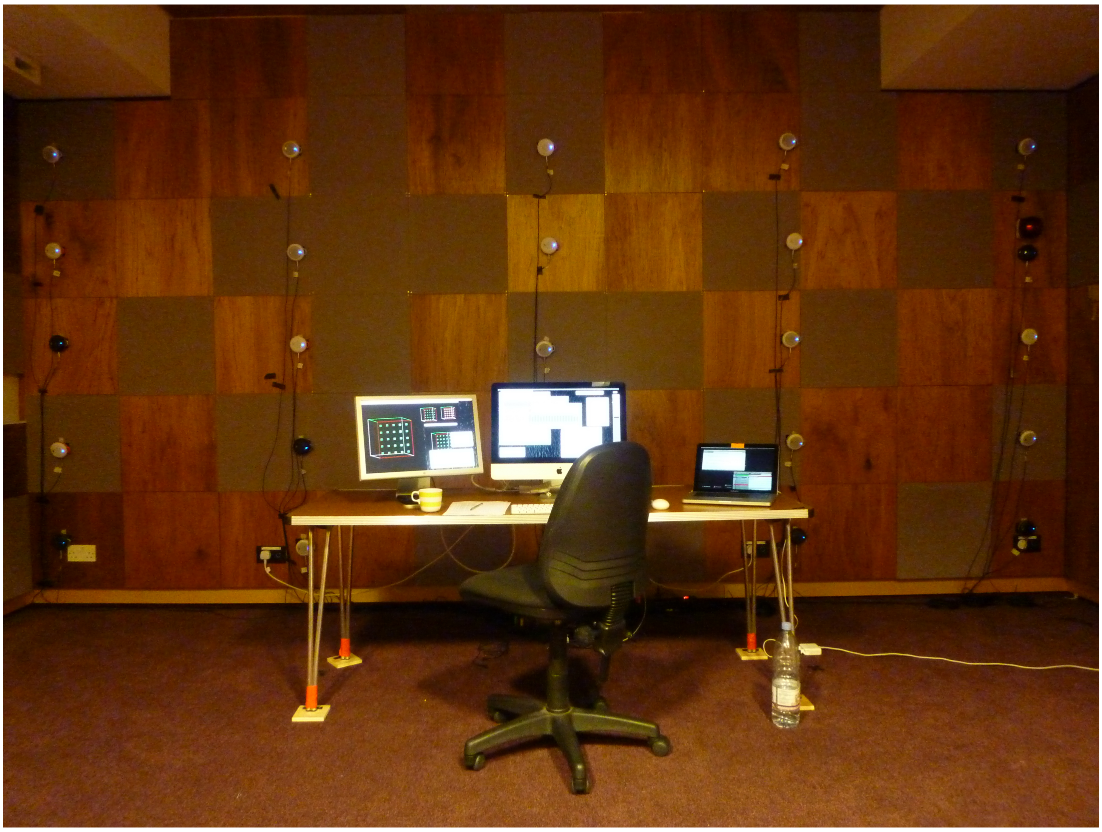


Figure 3.12.: Custom built vertical rectangular grid.

3.4. Parameter fields

Parameter fields model spatial surface patterns using real-valued functions defined on the surface S. As was already seen, using equation (3.1) it is possible to take a function defined on the parameter space $D \subset \mathbb{R}^2$ and extend it to the actual surface $S \subset \mathbb{R}^3$, therefore the parameter field is a function of two spatial dimensions and time:

$$f : (D \subset \mathbb{R}^2) \times \mathbb{R} \to \mathbb{R}.$$

For instance, $f(0.5, 0.5, 10)$ is the value of f at point $\mathbf{x}(0.5, 0.5)$ on the surface, 10 seconds into the future. In general, the value of pf, a parameter field assigned to a sound-process parameter, at time t and position $w = (u, v)$ is

$$\text{pf}(u, v, t).$$

Selecting a specific point on the parameter space (u, v) we can interpret the function of time

$$f_{u,v}(t) = \text{pf}(u, v, t)$$

as being a control signal, or envelope, that modulates the parameter of the instance of the sound process that is to be sent to location $\mathbf{x}(u, v)$ in the listening space (see figure 3.13). Ideally, the sound coming from this direction should be perceived as if the parameter of the sound process has the value $\text{pf}(u, v, t)$, although as we will see later this is not always the case.

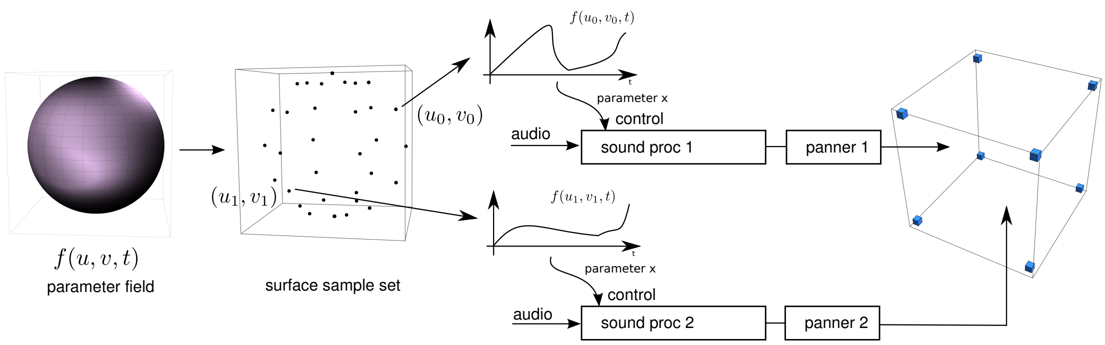


Figure 3.13.: Parameter field spatialization algorithm.

Discretization

Parameter fields are mathematical functions of continuous space ($\mathbb{R}^2$) and continuous time ($\mathbb{R}$) which will be used to generate control signals in a computer program. Given this specificity, parameter fields will be restricted to those functions which can be written down using an explicit formula which can be imported into a computer program. Such functions are those that can be expressed in *closed-form*[17].

In order to modulate a parameter of a sound process in a computer program, the output must be calculated at discrete time steps $\{t_j \in \mathbb{R} | j \in \mathbb{N}\}$ with the time delta between evaluations $T = t_{j+1} - t_j$ fixed. $\mathrm{pf}(u, v, t)$ can be directly calculated at each time step since programming languages are able to directly calculate the value of the different functions involved in closed-form expressions. The sequence of values to apply to the parameter of a sound process at point (u, v) can be thought of as a breakpoint envelope and is given by the sequence

$$\begin{aligned} s_{u,v} &: \mathbb{N} \to \mathbb{R} \\ &j \mapsto \mathrm{pf}(u, v, t_j). \end{aligned}$$

How is the size of the time step determined? The Nyquist–Shannon sampling theorem states that in order to accurately reproduce a signal with frequencies up to B, a sampling frequency of $2B$ must be used. Since the human hearing upper threshold is around 20000Hz for audio signals a sampling frequency of at least 44100Hz should be used for audio . In real-time computer audio synthesis output values are therefore usually sent to the sound card at the fixed rate of 44100Hz. There are, nevertheless, many types of audio modulation signals such as ADSR envelopes and low frequency oscillators whose bandwidth is smaller than 22050Hz and therefore can use a lower sampling frequency. In particular for the type of modulation created through the use of parameter fields a lower sampling frequency can be used. In this research generally a value of 10Hz was used since the trade-off between CPU usage and perceptual smoothness of the changes was considered acceptable.

[17] Closed-form expressions include only elementary arithmetic operations, factorials, exponents, logarithms, trigonometric and inverse trigonometric functions. They exclude functions given by infinite sums, limits or integrals such as the gamma and Bessel functions.

Main algorithm

The main algorithm of parameter field spatialization can be split in different steps (see figure 3.13):

1. A sound process is to be spatialized. A sound process definition, composed of a graph of ugens, is chosen.

2. The sound process is associated with an *ImmLib* surface $(S, \mathbf{x}, \mathrm{dist}, H)$ with sample set
$$H = \left\{(u_j, v_j) \in \mathbb{R}^2 \mid j \in \{1, ..., n\}\right\}$$
composed of n points. In direct mode a loudspeaker index array $[i_1, ..., i_n]$ is also stored. i_j is the number of the loudspeaker corresponding to point j of the sample set.

3. A subset of all the parameters of the sound process are associated with parameter fields. Let us assume that parameter p is associated with parameter field pf.

4. n instances of the sound process are created and started in the synthesis engine based on the same sound process definition chosen in 1.

   a) When in virtual mode the output of each instance is fed into a (different) panner which spatializes its input signal at the corresponding point of the sample set $\mathbf{x}(u_j, v_j) \in \mathbb{R}^3$.

   b) When in direct mode the output of instance k is sent directly to the soundcard output with index i_k, the index being retrieved from the stored loudspeaker-indexes array.

5. At a regular interval pf is evaluated at each of the points of H. The result of evaluating pf at point j and time t, $\mathrm{pf}(u_j, v_j, t)$, is assigned to the parameter p of instance j of the sound process, possibly causing a change in the sound output of this instance. We can think of $f_j(t) = \mathrm{pf}(u_j, v_j, t)$ as a control signal modulating parameter p of instance j.

Parameter fields can themselves have parameters, that is, the expression for the parameter field can involve additional variables which can be changed by the user in real-time, in which case the control signal becomes

$$f_j(t) = \mathrm{pf}(u_j, v_j, t, c_1, c_2, ...)$$

where $c_1, c_2, ...$ are the parameters of the parameter field. The parameters of the parameter field control general geometric properties of the corresponding bidimensional pattern, these parameters are what make parameter fields parametric in the sense discussed when motivating the use of mathematical functions on surfaces. Continuously changing these parameters while the sound is ongoing creates a temporal evolution of the spatial surface pattern, which can be used for compositional purposes.

The parameters of parameter fields can be modulated by time dependent functions,

$$f_j(t) = \mathrm{pf}(u_j, v_j, t, c_1(t), c_2(t), ...),$$

which can take the form of a breakpoint envelope or low frequency oscillator. They can also be changed directly by the user in real-time via a graphical user interface (GUI), a human interface device (HID), a musical instrument digital interface (MIDI) device, open sound control (OSC) messages, or through live-coding.

Example

Let us go over a concrete example of a parameter field[18]. Consider the parameter field defined by

$$\mathrm{pf}(u, v, t, u_0, v_0, r) = \begin{cases} 0 & \text{if } \mathrm{dist}((u, v), (u_0, v_0)) > r \\ 1 & \text{if } \mathrm{dist}((u, v), (u_0, v_0)) \leq r \end{cases} \tag{3.6}$$

where $p_0 = (u_0, v_0)$ is a fixed point on the surface and r is a non-negative number. This parameter field, with single parameter r, does not depend on time, only on u, v, p_0 and r. It acts like a spotlight: any points closer than r to p_0 are illuminated (value 1) and all the other points are in the dark (value 0), r is the radius of the base of the light cone. Figure 3.14 shows a visualization of this parameter field for different values of u_0, v_0 and r. When the parameter field is evaluated using the sample set on the sphere

[18] It might be helpful at this point to view the video spotlight.mkv in the companion USB flash drive, listening with headphones.

depicted in figure 3.15, the corresponding control signals generated from the parameter field for the multiple instances of the sound process as r is continuously changed from 0 to 1 can be seen in figure 3.16.

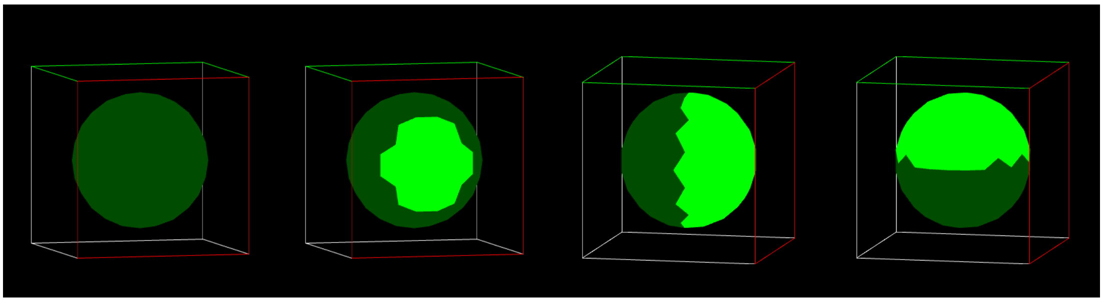


Figure 3.14.: Visualization of pf with $(u_0, v_0) = (0, 0)$, $r = 0$, $r = \frac{1}{4}$, $r = \frac{1}{2}$ and $(u_0, v_0) = (0, \frac{\pi}{2})$, $r = \frac{1}{2}$.

Let us assume the parameter field is modulating the `frequency` parameter of the sound process depicted in figure 3.17, with the output of the parameter field scaled such that a value of 0 is equivalent to 1000Hz and a value of 1 to 20000Hz. Listening to this example on a spherical setup with loudspeakers above and below the floor level, with $r = 0$ one would hear an enveloping cloud of low-passed noise coming from all directions. Each of the 20 virtual sources is playing a decorrelated noise signal which contributes to the feeling of immersion. Increasing r from 0 to 1, one would hear the sound in the frontal direction start to become brighter as the frequency increases from 1000Hz to 20000Hz, this brighter region would start to expand symmetrically until finally occupying the whole sphere. This would be perceived as a shell of high frequency noise progressively enclosing the listener.

Continuity

If the mathematical function used for a parameter field is continuous on the spatial variables[19], which is not uncommon for functions given by simple formulas, then the relationship between the values of the synthesis parameter being modulated at two points close together is not arbitrary: the closer the points the closer the values of the parameter. This gives rise to spatial continuity which contributes to the creation of spatial surface patterns. In computer music, it is not uncommon to create multiple

[19]For any t, $(u, v) \mapsto f(u, v, t)$ is continuous, that is for any t if $\lim_{n\to\infty}(u_n, v_n) = (u, v)$ then $\lim_{n\to\infty} f(u_n, v_n, t) = f(u, v, t)$.

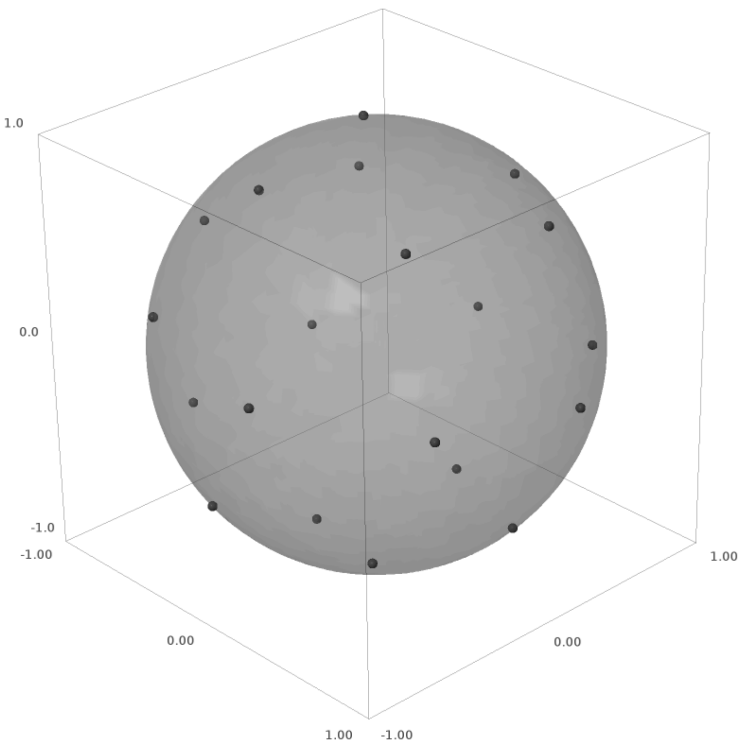


Figure 3.15.: Sphere sample set with 20 points.

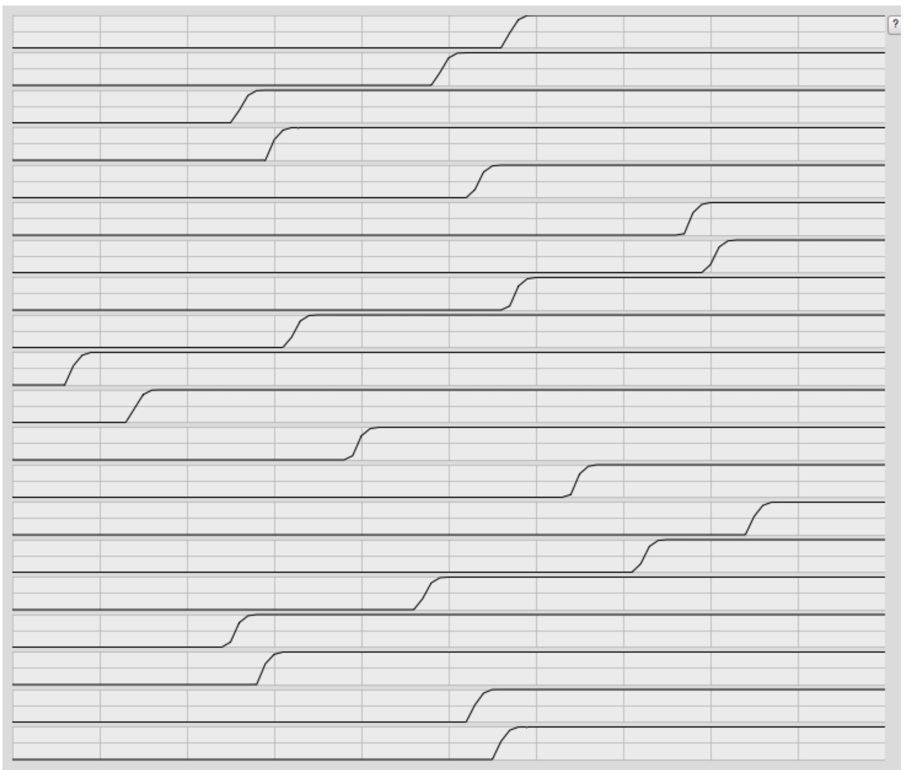


Figure 3.16.: Envelopes generated from pf for each position in the sample set as r goes from 0 to 1.

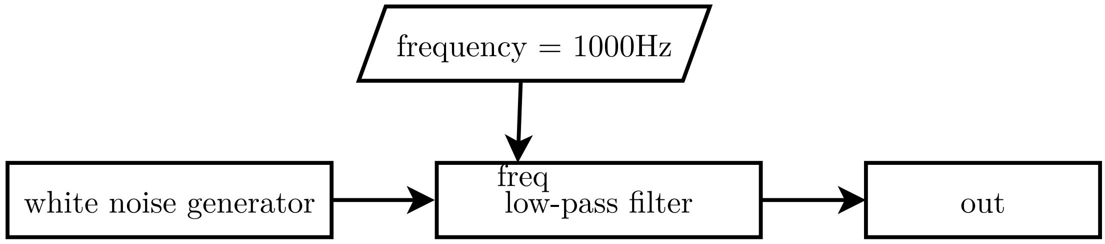


Figure 3.17.: Diagram of sound process definition consisting of a white-noise generator connected to a low-pass filter whose freq input is assigned a user control named “frequency”.

instances of the same sound process, with different settings or envelopes, in order to increase sonic density. Usually the different settings or envelopes are randomized, tweaked by hand or generated by an algorithm. It is uncommon though, for the settings applied to each sound process instance to increase in similarity with a decrease in distance of the corresponding sources in space. This property of spatial continuity which appears when using parameter fields is perhaps one of strongest motivations for the use of continuous mathematical functions in this research instead of more ad-hoc methods for generating settings for each sound process instance, such as through stochastic procedures.

Parameter fields and the distance function

Parameter fields can be defined by formulas based on the individual coordinates u and v or they can instead be defined using the distance function available from *ImmLib* surfaces. Such parameter fields will have the general form

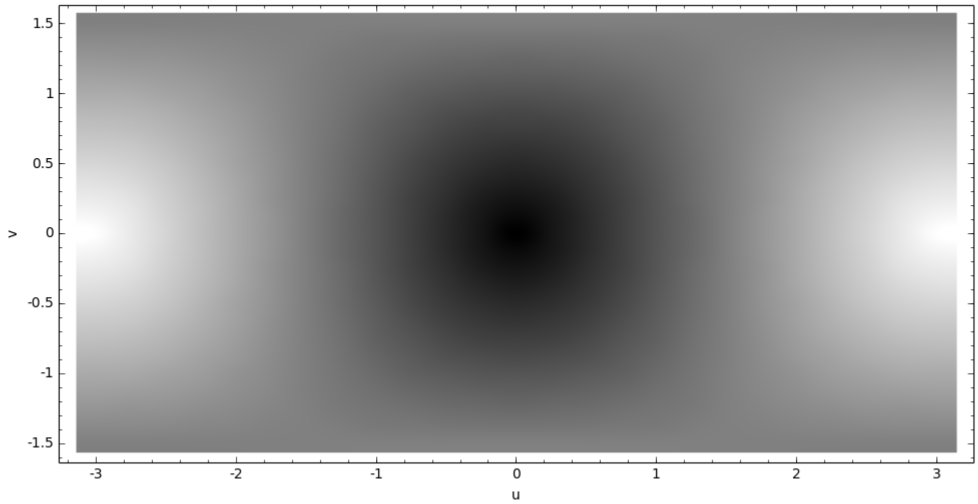
$$\mathrm{pf}(u,v,t) = f(dist(u,v,u_0,v_0))$$

where $f:\mathbb{R}\to\mathbb{R}$ and $(u_0,v_0)\in\mathbb{R}^2$ is the reference point. The reference point can be made a parameter of the parameter field

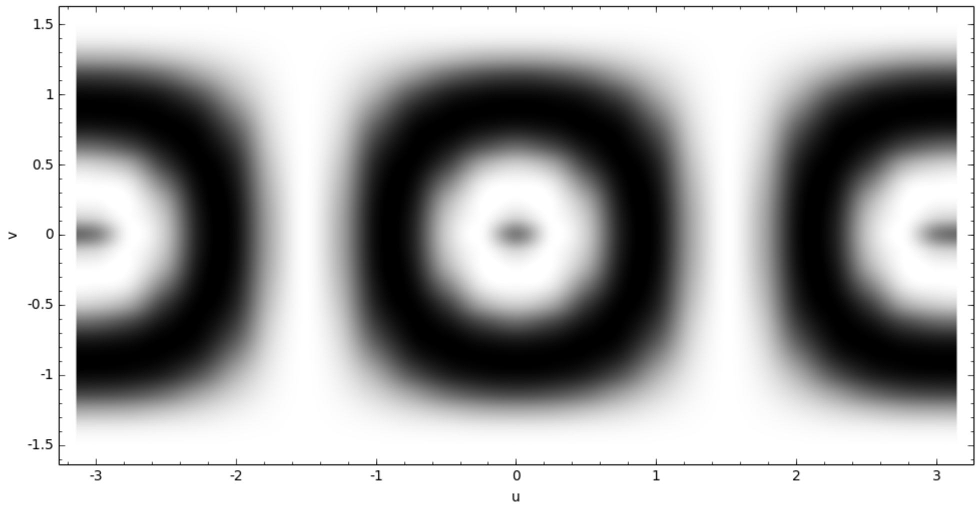
$$\mathrm{pf}(u,v,t,u_0,v_0) = f(dist(u,v,u_0,v_0))$$

and multiple reference points can be used

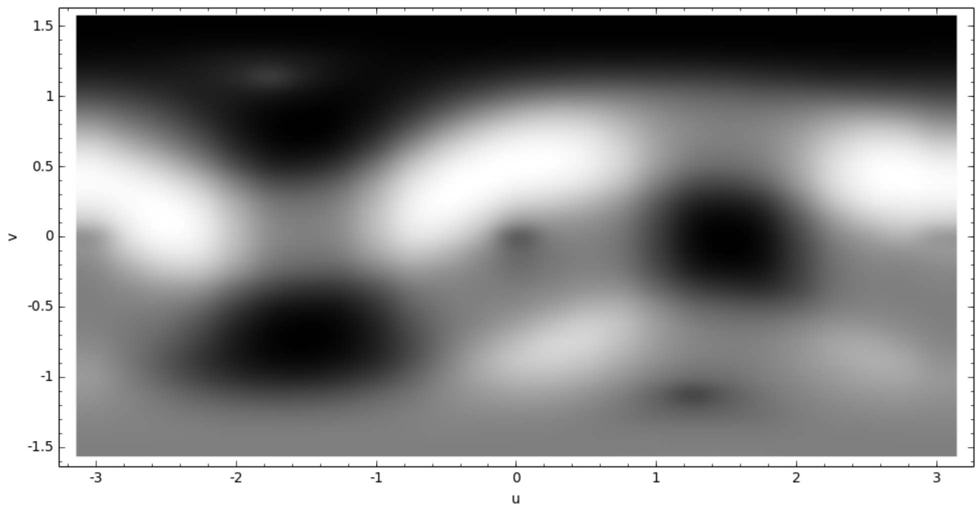
$$\mathrm{pf}(u,v,t,u_0,v_0,...,u_n,v_n) = f(dist(u,v,u_0,v_0),...,dist(u,v,u_n,v_n)).$$



(a) Plot of $f(u, v) = \mathrm{dist}(u, v, 0.0, 0.0)$ with dist the distance function for the sphere, $u \in [-\pi, \pi]$ and $v \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.



(b) Plot of $f(u, v) = atan(sin(5\,\mathrm{dist}(u, v, 0.0, 0.0)))$ with dist the distance function for the sphere, $u \in [-\pi, \pi]$ and $v \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.



(c) Plot of $f(u, v) = atan(sin(3\,\mathrm{dist}(u, v, 0.0, 0.0))) + atan(sin(4\,\mathrm{dist}(u, v, 1.3, -1.2)))$ with dist the distance function for the sphere, $u \in [-\pi, \pi]$ and $v \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

Figure 3.18.: Examples of distance-based parameter fields.

Such parameter fields will be symmetrical in relation to the reference point, since for any $r > 0$ the value of the parameter field is constant on a circle of radius r around $\mathbf{x}(u_0, v_0)$ on the surface. Whenever a symmetrical pattern is desired the use of the distance to a single point on the surface is a viable approach. Three examples of distance-based parameter fields can be seen in figure 3.18. The first parameter field is the identity, therefore the plot shows circular symmetry around a fixed point. The second parameter field uses the sine function which is periodic. The third one is composed of the sum of three different version of the second, with three different reference points.

Parameter field transformations

Having defined a parameter field it is useful to be able to rotate it in the surface. A pattern created from a particular mathematical expression might have a fixed orientation which is not the desired orientation, using a rotation it can be reoriented to the desired alignment. With multiple simultaneous sound processes using different parameter fields it might be desired that the patterns created have particular orientations in space in order to contrast the events spatially.

Rotation of a parameter field must be implemented specifically for each surface taking into account the symmetries of the surface. A pattern on the sphere can be reoriented by rotating the whole three-dimensional space around the origin since the sphere is invariant for such rotations. For the plane the rotation is done not in three-dimensional space but around an axis perpendicular to the plane and passing through the centre of the corresponding parallelogram. *ImmLib* implements rotations for the sphere and plane.

Rotations of patterns on the sphere are executed through general three-dimensional rotations around the origin. In three dimensions any orientation can be achieved from an initial orientation by rotating sequentially by three angles around three different axis. Depending on the chosen axis these angles, α, β, γ, are called Euler angles, Tait–Bryan angles, or yaw, pitch, and roll. Given the rotation function $r(w, \alpha, \beta, \gamma)$ which rotates a point $w \in \mathbb{R}^3$, the coordinate chart $\mathbf{x} : D \to S$, and a parameter field $\mathrm{pf}(u, v, t)$, we can define a rotatable version of pf with

$$\mathrm{pf}_{\mathrm{rot3D}}(u, v, t, \alpha, \beta, \gamma) = \mathrm{pf}(\mathbf{x}^{-1}(r(\mathbf{x}(u, v)), \alpha, \beta, \gamma)), t).$$

To calculate the rotated parameter field at point (u, v) in the domain of the coordinate chart, first the point is mapped to three-dimensional space, then the three-dimensional rotation is applied, subsequently the point is mapped back to the parameter space and finally the original parameter field is calculated at the new position.

In the case of the plane, if the two vectors used to define it are perpendicular, rotating the parameter space actually achieves the desired effect since in that case the mapping does not distort the geometry of the parameter space, angles between vectors in the parameter space are unaltered by mapping into the plane through the coordinate chart. Rotations in two dimensions around the origin are entirely defined by a single angle of rotation θ. Assuming that $r(u, v, \theta)$ is the function that rotates the point (u, v) by θ then a rotatable version of a parameter field is defined by

$$\mathrm{pf}_{\mathrm{rot2D}}(u, v, t, \theta) = \mathrm{pf}(r(u, v, \theta), t).$$

In the case of the plane it is also possible to scale the parameter field by s and translate it by $(u_t, v_t) \in \mathbb{R}^2$ by scaling and translating in the parameter space:

$$\mathrm{pf}_{\mathrm{rotScaleTrans2D}}(u, v, t, \theta, s, u_t, v_t) = \mathrm{pf}(r(su, sv, \theta) + (u_t, v_t), t).$$

Issues with calculating values directly from closed-form expressions

Due to the fact that the sequence of values to apply to the parameter of the sound process are calculated from the closed-form expression of the parameter field, changing a parameter of the parameter field in real-time can cause discontinuities if the parameter interacts with the time variable, for instance changing c discontinuously in $f(u, v, t, c) = sin(ct)$ will cause a jump. Because of the closed-form formula, changing the parameter of the parameter field will cause the "history" of the parameter field to be rewritten, and therefore the value at the current time step, $sin(c_{t_j} t_j)$, might be completely different from the value at previous time step, $sin(c_{t_{j-1}} t_{j-1})$. These discontinuities in the values of a parameter field when a parameter is changed can be heard and are undesirable.

Functions calculated through recurrence relations do not exhibit this behaviour since the last value is calculated in terms of the previous, nevertheless they usually require multi-step algorithms of higher complexity. For instance, the sine oscillator with frequency φ can be expressed in closed form as $f(t, \varphi) = sin(2\pi\varphi t)$ or as a recurrence relation where $s(n) = sin(\theta_n)$ and θ_n is calculated in terms of θ_{n-1} and φ ($\theta_n = \theta_{n-1} + g(\varphi)$ for a suitable function g which in general will depend on the sampling rate). This is the approach generally used for implementing ugens in computer synthesis.

The discontinuity nuisance was considered minor compared to the benefit of being able to write the parameter fields using normal mathematical syntax. The next chapter details some means of attenuating the discontinuity problem.

3.5. Parameter sequences

Parameter fields are functions of continuous time and continuous space which are directly evaluated at specific points in space and moments in time in order to generate the necessary control signals to drive a parameter of a sound process. This is the main approach proposed in this work, and the only one fully implemented in the *ImmLib* library. An alternative to this approach is the use of functions of discrete time and discrete space, that is, sequences (see figure 3.19).

A sequence is usually defined as a function of the natural numbers $s : \mathbb{N} \to A$, in particular a real-valued sequence $s : \mathbb{N} \to \mathbb{R}$ is a function from the natural numbers to the real numbers. Multidimensional sequences extend this notion to multiple variables: $s(n, m)$ with $s : \mathbb{N} \times \mathbb{N} \to A$. In discrete mathematics there are sequences that are defined not through closed-form expressions but through recurrence relations, for example $s(n) = -s(n-1)$, $s(0) = 1$, resulting in the sequence $(1, -1, 1, -1, ...)$.

There are many interesting systems in mathematics and the natural sciences defined by recurrence relations. In the area of nonlinear mathematics and chaos theory sequences such as the logistic map[20] and cellular automata are defined naturally through recurrence relations and exhibit very complex behaviour. Systems such as cellular automata are inherently discrete in both space and time, on the other hand, systems described by partial differential equations (PDEs) are defined on continuous space and time but are discretized in order to obtain a numerical approximation. For most systems described by a PDE, including chemical and physical systems demonstrating complex behaviour, no closed-form expression is known for the solution to the equation, even

[20] The logistic map is the sequence given by the non-linear recursive equation $s(n+1) = rs(n)(1-s(n))$. It was proposed by Robet May [100] as a demographic model and exhibits chaotic behaviour.

when it can be proved that the solution exists. In these cases simulating the system requires numerically finding the solution to the equation using methods such as finite-difference (FDM), which discretize the function in both time and space through recurrence relations on time and space variables.

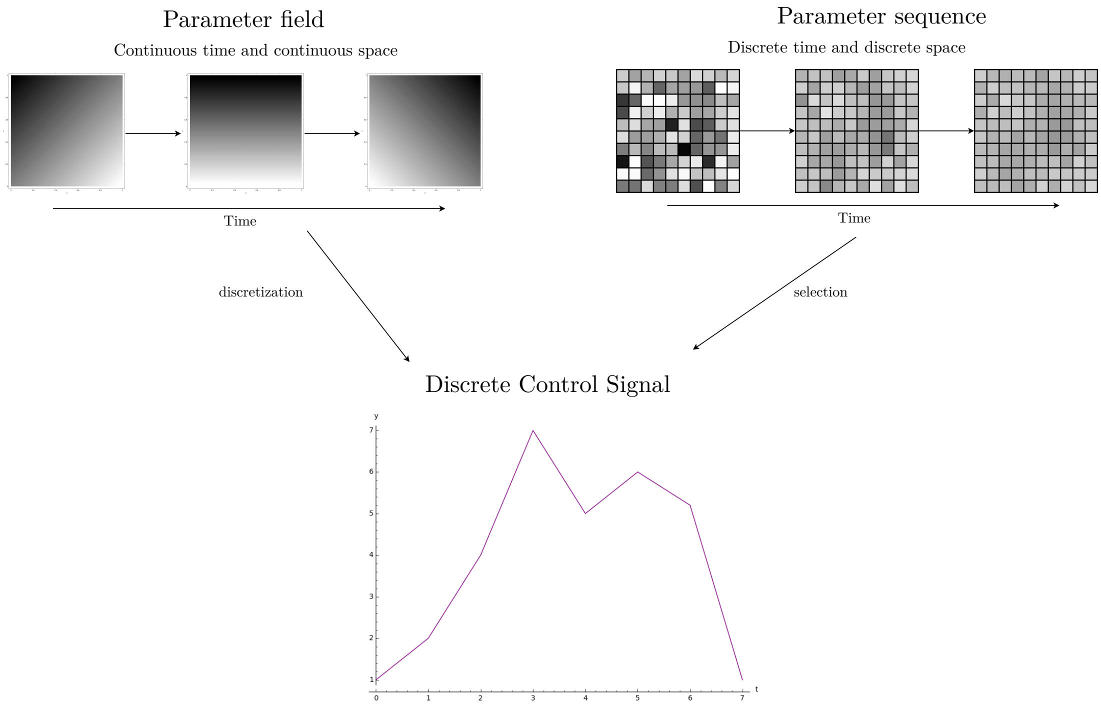


Figure 3.19.: Parameter fields and parameter sequences.

With these systems a tremendous amount of complexity seems to come from seemingly simple equations. The differential equations behind physical systems or the recurrence rules behind chaotic sequences are easy to write down and often use simple algebraic operations and simple functions such as the trigonometric functions. Small changes in the parameters or starting state of these systems can cause dramatic changes in behaviour. Using such systems could enable the creation of spatial surface patterns with complex behaviours using relatively simple equations. Also, the behaviour of the system can usually be controlled with a rather small number of parameters, with small variations giving rise to completely different behaviours.

During this research some exploratory work was conducted on the use of sequences for generating control signals. Some sequences were implemented and tested in listening sessions. Even if the results were not conclusive, and further work is needed to validate this approach, it does seem promising, therefore the mathematical formalization and some examples will be presented. Nevertheless, none of the systems experimented with were included in *ImmLib* since they were deemed too experimental and difficult to control.

Definitions

We define a parameter sequence as the analogous construction to a parameter field for a discrete space and time. Given an *ImmLib* surface with sample set

$$H = \{(u_0, v_0) = w_0, ..., (u_n, v_n) = w_n\}$$

a finite set of points on the parameter space and $T = \{t_i | i \in \mathbb{N}\} \subset \mathbb{R}$ the set of evaluation times, a parameter sequence takes the form

$$\mathrm{pf}(u_i, v_i, t_j) = \mathrm{pf}(w_i, t_j)$$

with $\mathrm{pf} : H \times T \to \mathbb{R}$. This definition is more convenient but equivalent to a definition using only the indexes:

$$\mathrm{pf}'(i, j) = \mathrm{pf}(u_i, v_i, t_j).$$

If sequence $\mathrm{pf}(w_j, t_k) = \mathrm{pf}(u_j, v_j, t_k)$ is defined by a recurrence relation, that implies that it might depend on $\mathrm{pf}(w_l, t_k)$ for $l \neq j$, and $\mathrm{pf}(w_l, t_m)$ for $m < k$ and any l. The exact indices l for w_l on which $\mathrm{pf}(w_j, t_k)$ depends will vary with different recurrence relations, although in many examples they only depend on indexes corresponding to neighbouring points of w_j. This is typical of physical systems where interactions always take some time to propagate through space. If the set of points on the surface forms a grid it is possible to index a point using two variables (w_{ij}), with the points $\{w_{(i+l)(j+p)} | l, p \in \{-1, 1\}\}$ adjacent to w_{ij} on the surface. In this case $\mathrm{pf}(w_{ij}, t_k)$ will depend on $\{\mathrm{pf}(w_{(i+l)(j+p)}, t_m) | l, p \in \{-1, 1\} \wedge m \leq k\}$. In general the sample set does not alway form a grid, in which case it must be known which points are considered neighbours of a given point. This information can be encoded as a function $w_j \mapsto X_j \subset H$, where H is the surface sample set, of which a simple example is the function $\mathrm{neigh} : H \times \mathbb{N} \to H$ that selects the n closest points in H to w_j using the distance function for the surface. When several points are at the same distance to w_j we define neigh as selecting points using lexicographical ordering based on the coordinates in $\mathbb{R}^3$: if $\text{dist}(w_i, w_j) = \text{dist}(w_l, w_j) \wedge w_i \neq w_l$ then $w_i = (a_1, b_1, c_1)$ is ordered before $w_l = (a_2, b_2, c_2)$ if the following condition holds:

$$\begin{aligned} & (a_1 < a_2) \\ \vee \quad & (a_1 = a_2 \wedge b_1 < b_2) \\ \vee \quad & (a_1 = a_2 \wedge b_1 = b_2 \wedge c_1 < c_2). \end{aligned}$$

If does not hold, w_i is positioned after w_l.

Examples

As a simple example, consider the parameter sequence which takes the average of the previous values of the four closest points:

$$c(u_i, v_i, t_n) = \frac{\sum_{j=1}^{4} c(u_{p_j}, v_{p_j}, t_{n-1})}{4}$$

with $(u_{p_j}, v_{p_j}) \in \text{neigh}((u_i, v_i), 4)$. The starting state $c(u_i, v_i, t_0)$ is generated randomly. Figure 3.20 shows a visualization of the evolution of the sequence. It is apparent that the initial spatial differences are blurred until the pattern is almost a single colour.

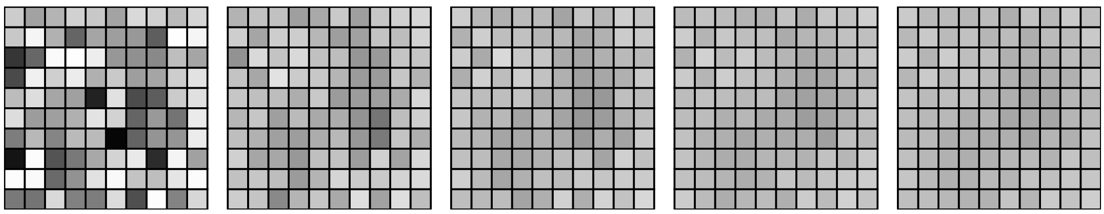


Figure 3.20.: Averaging parameter sequence, first five time steps.

Reaction–diffusion systems and continuous automata, a continuous version of cellular automata, are the systems from mathematics and the natural sciences that seemed the most promising for this research, due to the richness of the spatial patterns they generate.

Reaction–diffusion systems "are systems involving constituents locally transformed into each other by chemical reactions and transported in space by diffusion[21]. They

[21] "the process whereby particles of liquids, gases, or solids intermingle as the result of their spontaneous movement caused by thermal agitation and in dissolved substances move from a region of higher to one of lower concentration." [101]

arise, quite naturally, in chemistry and chemical engineering but also serve as a reference for the study of a wide range of phenomena encountered beyond the strict realm of chemical science such as environmental and life sciences.” [102].

One example of a chemical reaction which can be modelled as a reaction-diffusion system is the Belousov-Zhabotinsky reaction (BZ reaction). This chemical reaction “makes it possible to observe development of complex patterns in time and space by naked eye on a very convenient human time scale of dozens of seconds and space scale of several millimetres” [103] (see figure 3.21).

Alan Turing on his seminal 1952 paper “The chemical basis of morphogenesis” [104] describes a reaction-diffusion theory for morphogenesis, the biological process by which an organism develops its shape. Reaction-diffusion systems have also been used to model the stripes and spots seen in animal’s skin. The spatial patterns created by these systems have been called *Turing patterns.* Turing pattens can be very complex and highly detailed, exhibiting almost life-life behaviour. As an example of an artistic application of Turing patterns, the artist Jonathan McCabe has used computational algorithms modelling reaction-diffusion systems to create incredibly complex images and video works (see figure 3.22 and [105]).

Cellular automata is another class of systems which could be of use for creating complex spatial patterns. “A cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells” [107]. Two dimensional cellular automata could be spatialized by assigning each of the cells to one of the static sound sources. If the same spatial organization is maintained, such that the neighbours of a cell are also spatially located as neighbours in the surface sample set, then the spatial patterns generated by the cellular automata should become audible. Cellular automata only take values 0 and 1 or possibly a finite set of values, using cellular automata as parameter sequences would switch the sound-process parameter between a discrete set of values. Continuous automata [108] on the other hand take values continuously between two real numbers (the elements are still discretely arranged in space), thus being able to create smooth spatial patterns.

More work has been done with grid-based self-evolving or dynamic systems in the visual arts than in sound synthesis, due to the fact that a screen can be naturally rep-

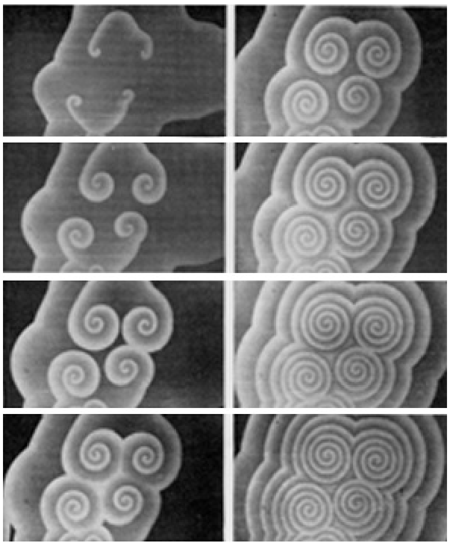


Figure 3.21.: Belousov-Zhabotinsky reaction: chemical waves and patterns. Adapted from [103].Copyright Anatol M. Zhabotinsky, used here under the Creative Commons Attribution-NonComercial-ShareAlike 3.0 Unported License.

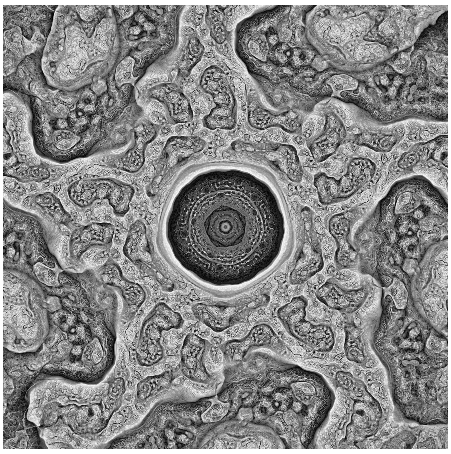


Figure 3.22.: Computer generated image featuring a multi-scale Turing pattern by visual artist Jonathan McCabe [106]. Copyright Jonathan McCabe, used here under the Creative Commons Attribution-NonComercial-ShareAlike 3.0 Unported License.

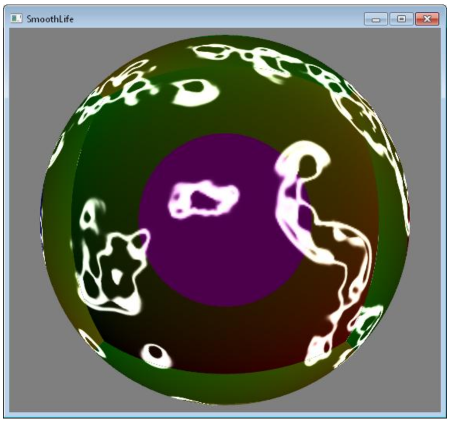


Figure 3.23.: Screen-shot of *Smooth Life,* a cellular automata simulation program [110].

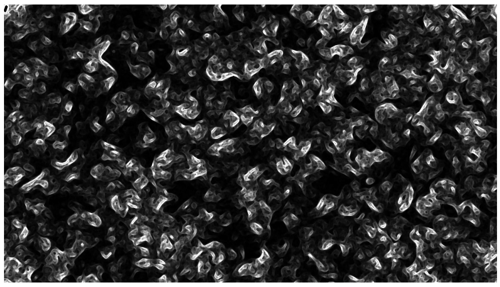


Figure 3.24.: A continuous automaton rendered in *Processing* [111].

resented as a two-dimensional grid while sound is usually modelled as a one-dimensional stream of values. There are many examples of these systems available for *Processing*, a programming language for visual arts, such as simulations of reaction-diffusion systems, visualizations of cellular automata (see figure 3.24), fluid dynamics, Ginzburg-Landau equation and others. The open source program *Smooth Life* [109, 110] simulates a version of Conway's game of life, a cellular automata system, with a continuous domain. Interestingly, one of the versions of the program uses a grid on a sphere, an example of simulating such systems outside of planar surfaces (see figure 3.23).

The BZ reaction was implemented and tested in the Sonic Lab. A description of the implementation will be given at the end of the following section, while a description of the perceptual results will be given in section §5.1.1.

3.6. Implemented parameter fields

This section describes the parameter fields which were created and experimented with during this research. Defining spatial patterns through mathematical expressions is neither easy nor intuitive. Some types of patterns can be imagined and then a suitable expression searched that gives rise to them. Another approach is to take simple functions, such as sines and cosines, and attempt to combine them in different ways, evaluating the sonic result in order to find interesting combinations. The composition of trigonometric functions like sine and cosine and distance functions on the surface can quickly create quite interesting spatial surface patterns. Another approach is to look for complex mathematical functions in the diverse areas of modern mathematics which might create useful patterns.

ImmLib ships with a set of default parameter fields, which constitute a library of generators of spatial surface patterns, which the user can explore, and alter, creating other derived parameter fields. In *ImmLib* these predefined parameter fields are accessible as class methods[22] of the `PField` *SuperCollider* class. These parameter fields were arrived at mostly by trial and error, using basic mathematical functions combined in different ways. This was the case of parameter fields such as the barU, gradient, expandContract, spotlight, and its derivatives. Others, such as the wave functions and spherical harmonics are standard mathematical functions, which can be found in text books, which were simply copied and adapted for smooth curved surfaces.

This section will describe each parameter field, stating the mathematical definition, or description of the algorithm when appropriate. A perceptual description of the application of these parameter fields to different sound processes will only be given in chapter 5, after the implementation is introduced.

Pure parameter fields

A pure parameter field is defined solely by a single mathematical function, while a pre-evaluated parameter field can change the mathematical function being used in real-time. The reason for naming them "pre-evaluated parameter fields" will become clear in the next chapter covering the implementation. In essence, pure parameter fields are evaluated calculating always the same *SuperCollider* function which corresponds directly to a mathematical function. On the other hand, pre-evaluated parameter fields use a dynamic algorithm which can decide to change the *SuperCollider* function being used for calculation based on real-time input from the user, or from a random-number generator.

The pure parameter fields are presented first, in ascending order of mathematical complexity, followed by the pre-evaluated parameter fields. On the colour plots of

[22] A class method is a method of a class which can called directly without instantiating an object of that class. These methods are usually used to implement pure functions (functions in the mathematical sense, that do not produce side-effects) or to essentially create a global object which can be accessed from any point in the code. In *SuperCollider* a class method is invoked by using the class name and a dot: `MyClass.myMethod`.

parameter fields, white is 0 and green is 1, on the grey-scale plots white is 0 and black is 1. In the plots and visualizations depicting the sphere, although the point of view is from outside the sphere, the reader should keep in mind that the actual listener would be inside the sphere, therefore hearing the pattern from the opposite perspective. The plots were done in this way since it is easier to visualize a pattern on the sphere from outside than from inside. The *ImmLib* syntax is introduced at this point since, although in the case of the pure parameter fields there is mostly a one to one correspondence of *ImmLib* syntax and mathematical formula, for the pre-evaluated ones only a description of the algorithm is given, therefore the syntax is useful to gain an understanding of which parameters can be modulated or changed in real-time.

barU and barV

These parameter fields create respectively vertical and horizontal strips of wideness w. They are defined by

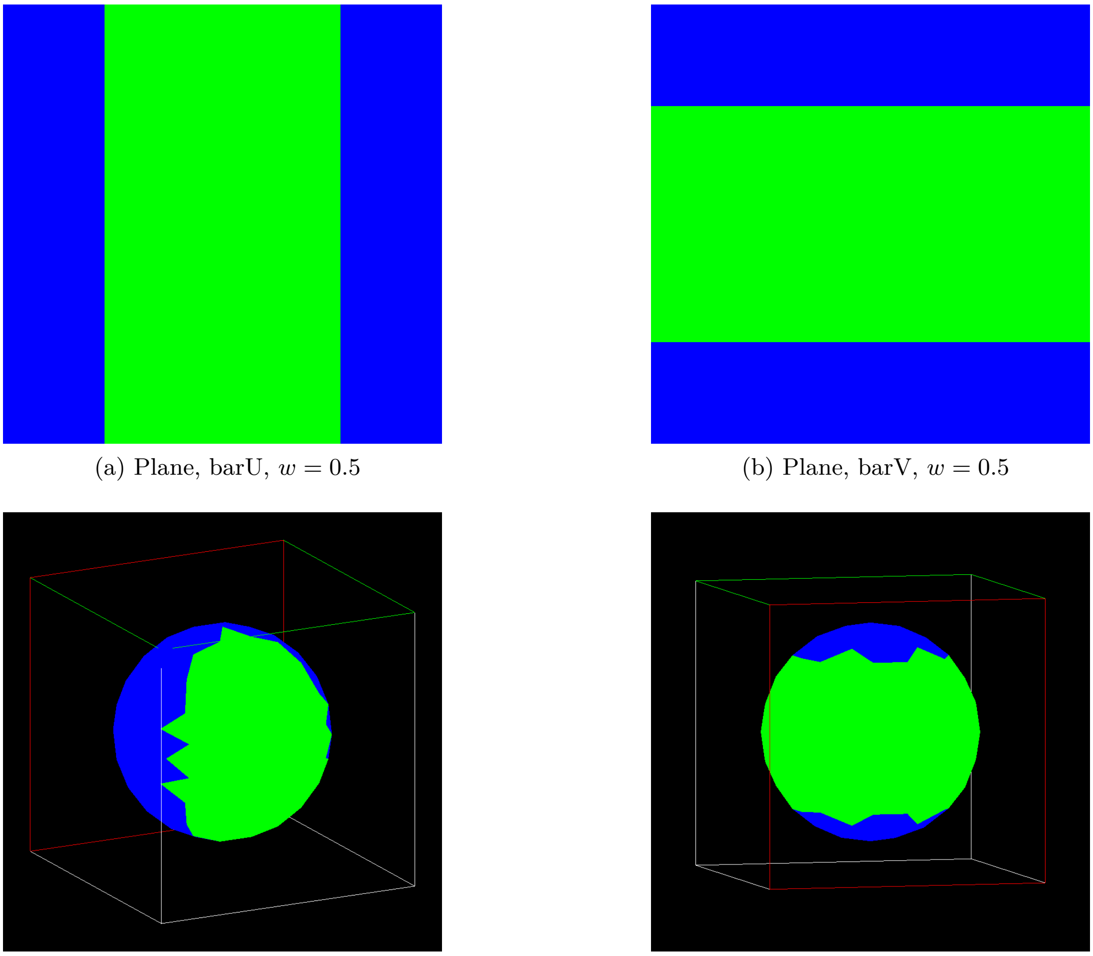
$$\begin{aligned}
\mathrm{barU}(u,v,t,w) &= \begin{cases} 0 & \text{if } |u-c| > wc \\ 1 & \text{if } |u-c| \leq wc \end{cases} \\
\text{where } c &= \frac{u_b - u_a}{2},
\end{aligned}$$

$$\begin{aligned}
\mathrm{barV}(u,v,t,w) &= \begin{cases} 0 & \text{if } |v-c| > wc \\ 1 & \text{if } |v-c| \leq wc \end{cases} \\
\text{where } c &= \frac{v_b - v_a}{2}.
\end{aligned}$$

The *ImmLib* syntax for initializing this parameter field is `PField.barU(t, w)` and `PField.barV(t, w)`. Visualisations of these parameter fields can be seen in figure 3.25.

Gradient

This parameter field creates a smooth circular gradient from a reference point. The gradient can be linear or have a non-linear shape determined by `curve`. It is defined by



(a) Plane, barU, $w = 0.5$

(b) Plane, barV, $w = 0.5$

(c) barU, $w = 0.5$

(d) barV, $w = 0.5$

Figure 3.25.: Visualisation of barU and barV parameter fields for two surfaces.

$$\begin{aligned}
\text{gradient}(u, v, t, u_0, v_0, a, b, \text{curve}) &= g(a, b, \text{curve}, x) \\
\text{where } x &= \text{dist}((u, v), (u_0, v_0)) \\
u, u_0 &\in [u_a, u_b] \\
v, v_0 &\in [v_a, v_b] \\
a, b &\in [0, 1] \\
\text{curve} &\in [-12, 12]
\end{aligned}$$

$g(a, b, \text{curve}, 0) = a$, $g(a, b, \text{curve}, 1) = b$ and $h_{a,b,\text{curve}} : [0, 1] \to [a, b]$ with $h(x) = g(a, b, \text{curve}, x)$ is monotonic and differentiable (see figure 3.26). The parameter field changes continuously from value a at point (u_0, v_0) to value b at points at the maximum distance from (u_0, v_0) with shape given by the `curve` argument. When curve $= 0$ it uses a linear shape:

$$g(a, b, 0, x) = a(1 - x) + bx.$$

Notice that the definition is in fact a family of functions, indexed by the surface used, since each surface will have a different distance function.

Figures 3.27a and 3.27b show a plot of the parameter field directly on the sphere while figures 3.27c and 3.27d show the same parameter fields plotted on the parameter space of the sphere. Figures 3.27c and 3.27d show a plot of the parameter field on the plane surface (note that for the plane plotting on the surface or on the parameter space gives essentially same result).

The *ImmLib* syntax for initializing this parameter field is `PField.gradient.(t, u0, v0, a, b, curve)`.

Gradient1D

This parameter field, which is only valid for the plane, creates a smooth gradient along the line determined by a unit vector with circular coordinates $(1, \theta)$. It is defined by

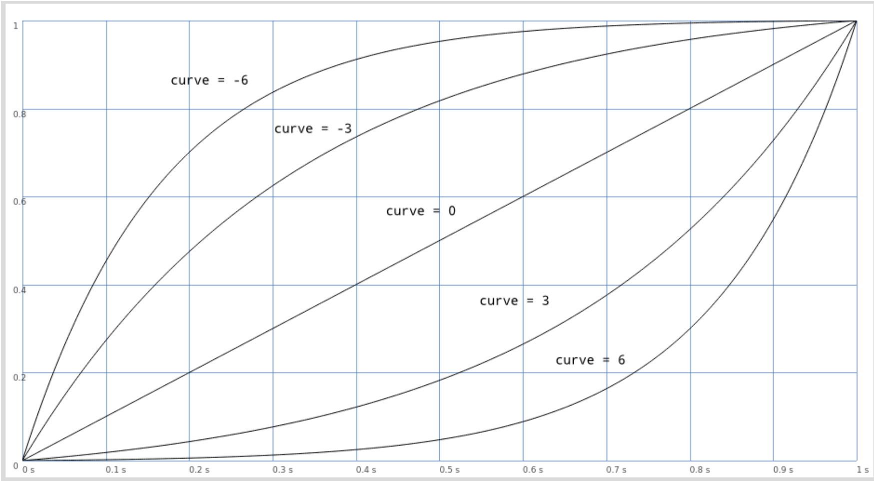


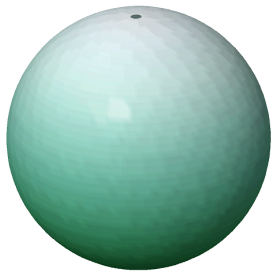
Figure 3.26.: Plot of $y = h(x)$ for different values of `curve`. The horizontal line is x, the vertical line is y.

$$
\begin{aligned}
\text{gradient1D}(u, v, \theta, a, b, \text{curve}) &= g(a, b, \text{curve}, d) \\
\text{where } \mathbf{a} &= \mathbf{o} + \frac{M}{2}(\cos\theta, \sin\theta) \\
\mathbf{n} &= (-\sin\theta, \cos\theta) \\
\mathbf{b} &= \mathbf{a} - (u, v) \\
d &= \|\mathbf{b} - \langle \mathbf{b}, \mathbf{n} \rangle \, \mathbf{n}\| \\
u &\in [u_a, u_b] \\
v &\in [v_a, v_b] \\
a, b &\in [0, 1],
\end{aligned}
$$

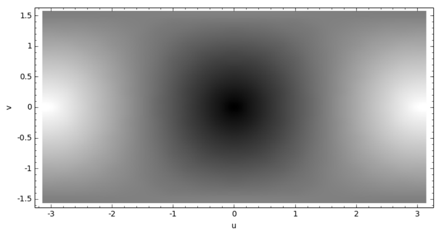
M is the maximum distance on the surface, $\mathbf{o}$ is the centre of the domain D, and g is the same function as defined for `gradient`. In the plane `PField.gradient` creates a circular gradient while this parameter field creates a linear gradient. Despite the different geometry this parameter field can be used in the same situations as gradient.

The *ImmLib* syntax for initializing this parameter field is `PField.gradient1D.(t, angle, a, b, curve)`. Figure 3.28 shows a plot of the parameter field on the plane surface.

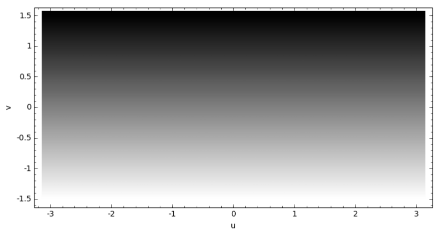
(a) $(u_0, v_0) = (0, 0)$



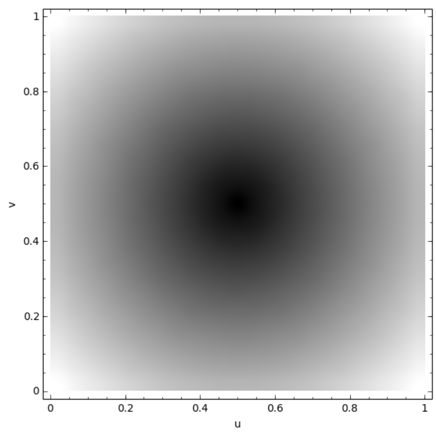
(b) $(u_0, v_0) = (0, \frac{\pi}{2})$



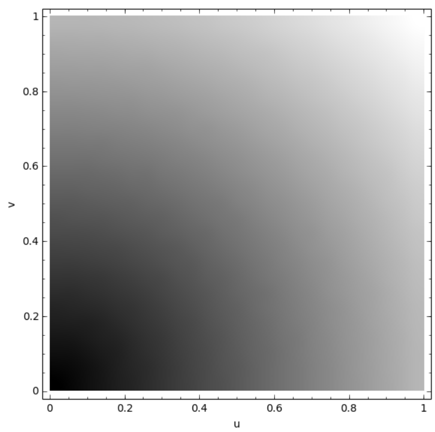
(c) Sphere, $(u_0, v_0) = (0, 0)$.



(d) Sphere, $(u_0, v_0) = (0, \frac{\pi}{2})$.

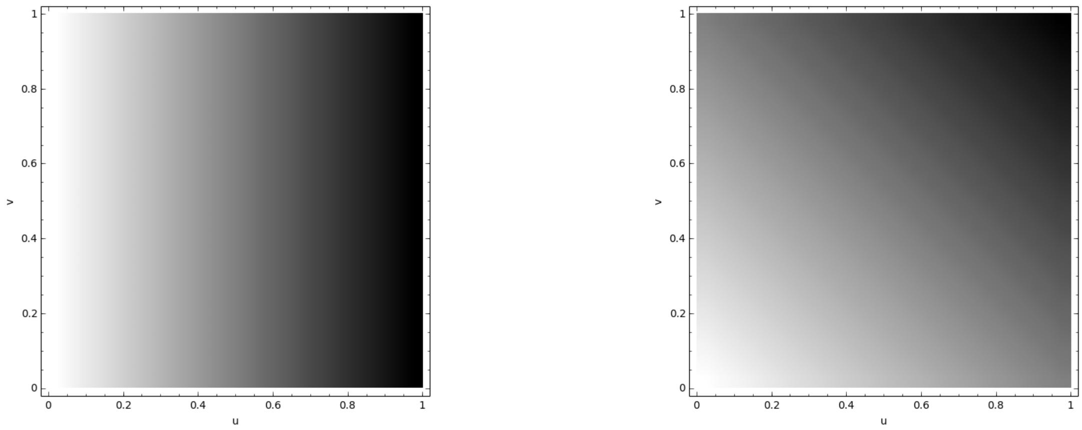


(e) Plane, $(u_0, v_0) = (\frac{1}{2}, \frac{1}{2})$.



(f) Plane, $(u_0, v_0) = (0, 0)$.

Figure 3.27.: Plots of gradient parameter field with $t = 0, a = 0, b = 1$ and curve $= 0$ for two different surfaces and various values of (u_0, v_0).



(a) $\theta = 0$ (b) $\theta = \frac{\pi}{4}$

Figure 3.28.: Plots of gradient1D parameter field with $a = 0, b = 1$ and curve $= 0$ on the plane and various values of θ.

Spotlight

As the name suggests, this parameter field acts as a spotlight where the radius of the illuminated zone is determined by c and centred around $p = (u_0, v_0)$.

To formally define the parameter field we start by looking at the function which is 1 on $B_c(p) = \{x \in D \mid \mathrm{dist}(x, p) < c\}$, the open ball around $p \in D$, and zero elsewhere. The open ball generalizes the interior of a circle to any surface. Adding a bump function with parameter d to smooth the transition from 0 to 1 happening at the border of $B_c(p)$, where d is the wideness of the smoothing area, we get

$$\begin{aligned} \mathrm{spotlight}(u, v, t, u_0, v_0, c, d) &= g(x) = \begin{cases} 0 & \text{if } x > c' + d \\ b(\frac{x - c'}{d}) & \text{if } c' < x \leq c' + d \\ 1 & \text{if } x \leq c' \end{cases} \quad (3.7) \\ \text{where } u, u_0 &\in [u_a, u_b] \\ v, v_0 &\in [v_a, v_b] \\ c, d &\in [0, 1] \\ c' &= (1 + d)c - d \\ x &= \mathrm{dist}((u, v), (u_0, v_0)) \\ b(x) &= \begin{cases} 2^{1 - \frac{1}{1 - x^2}} & \text{if } x \in [0, 1[\\ 0 & \text{if } x = 1 \end{cases}. \end{aligned}$$

This parameter field grows symmetricalally from a reference point (u_0, v_0) as c goes from 0 to 1 until occupying all the surface. It returns 1 if the distance between (u, v) to (u_0, v_0) is smaller than c and 0 otherwise. d controls the wideness of the smoothing area created by $b(x)$, a function that goes smoothly from 1 to 0 as x goes from 0 to 1. The auxiliary expression c' is constructed such that when $c = 0$ the smoothing region is not yet active (it is to the left of the $[0, 1]$ interval in the graph) and when the $c = 1$ the smoothing area has already been applied everywhere and the function is constantly equal to 1 on the surface (it is to the right of the $[0, 1]$ interval in the graph), as shown in figure 3.29.

The parameter fields randomHills and moveHills, to be described later, implement complex behaviours using multiple randomly-generated simultaneous spotlights.

The *ImmLib* syntax for initializing this parameter field is `PField.spotlight(t, u0, v0, c, d )`. Figure 3.30 shows different visualizations of the parameter field.

expandContract

A parameter field, only valid on the sphere, defined by:

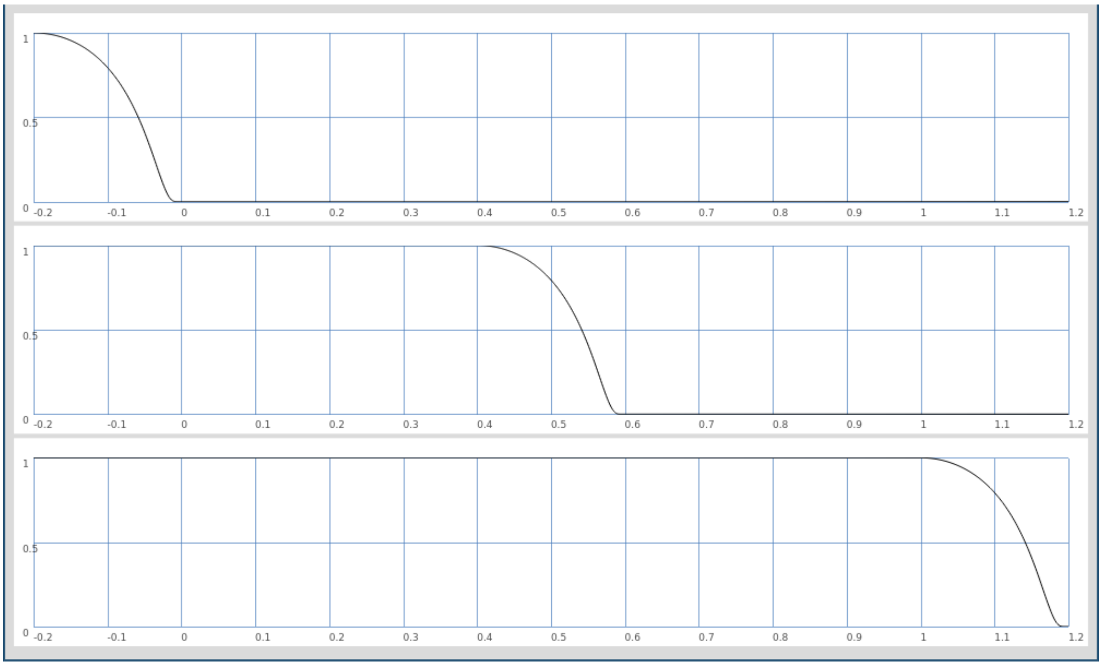
$$\text{expandContract}((u, v, t, u_0, v_0, c) =$$
$$\begin{cases} \text{spotlight}(u, v, t, u_0, v_0, 2c) & c \in [0, 0.5[\\ \text{spotlight}(u, v, t, u'_0, v'_0, 1 - 2c) & c \in [0.5, 1]. \end{cases}$$

It expands symmetricalally from a reference point until occupying the entire sphere and then shrinks into (u'_0, v'_0), the antipodal point, with the amount of expansion determined by c. It is similar to spotlight, but while with spotlight the pattern grows from, and shrinks to the same point, with this parameter field the patterns grows from one point and then shrinks into the opposite point.

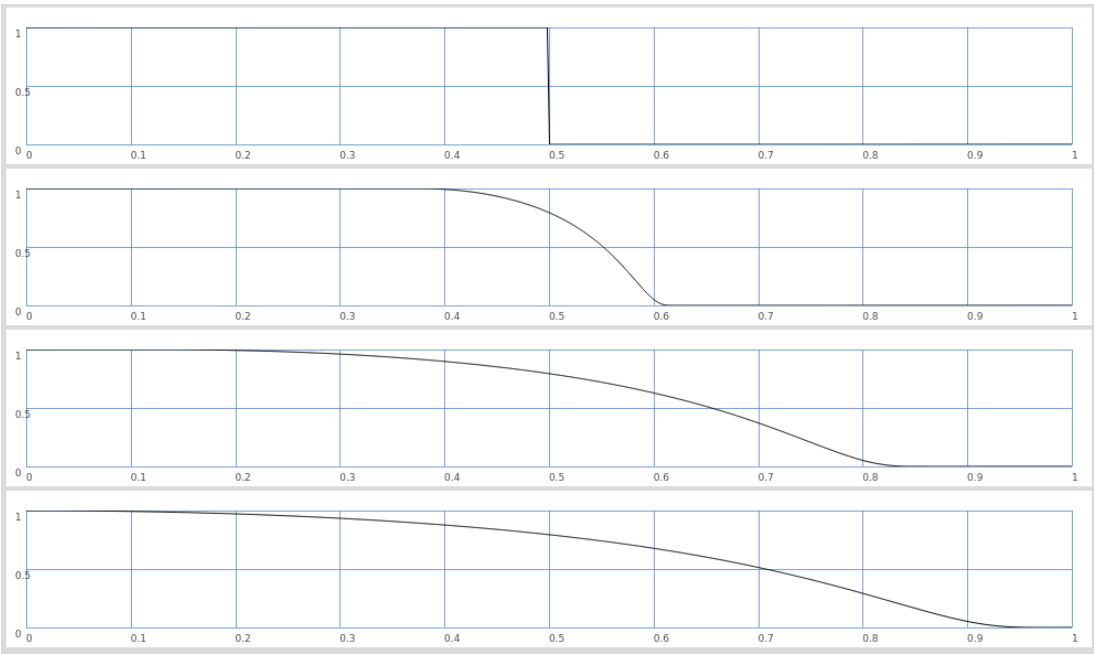
The *ImmLib* syntax for initializing this parameter field is `PField.expandContract(t, u0, v0, c)`.

sphericalHarmonic

A parameter field defined by the spherical harmonic function of degree l and order $-l \leq m \leq l$ multiplied with a sinusoidal function of time with frequency f. As Fourier series represent functions defined on the circle, spherical harmonic series of functions

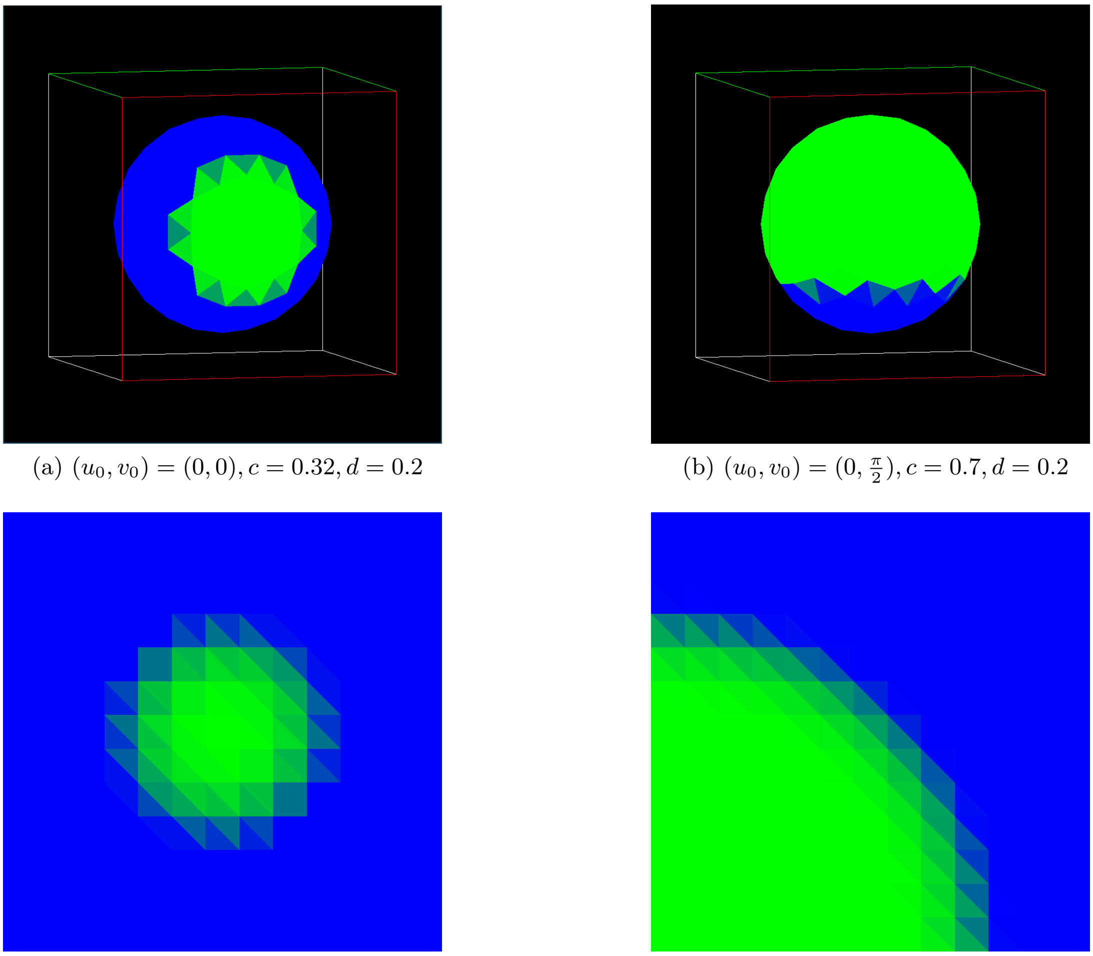


(a)



(b)

Figure 3.29.: a) Plot of $y = g(x)$ (see equation (3.7)) for $d = 0.2$,$c = 0, c = 0.5, c = 1$. The horizontal line is x, the vertical line is y. b) Plot of $y = g(x)$ for $c = 0.5$, $d = 0.0$, $d = 0.25$, $d = 0.75$, $d = 1.0$. The horizontal line is x, the vertical line is y.



(a) $(u_0, v_0) = (0, 0), c = 0.32, d = 0.2$

(b) $(u_0, v_0) = (0, \frac{\pi}{2}), c = 0.7, d = 0.2$

(c) Plane, $(u_0, v_0) = (0.5, 0.5), c = 0.35, d = 0.5$

(d) Plane, $(u_0, v_0) = (0, 0), c = 0.6, d = 0.5$

Figure 3.30.: Visualization of the spotlight parameter field for two surfaces.

represent any function on the sphere[23]. As can be seen in the plots in figure 3.31, the spherical harmonics form lobes, some positive and others negative, which are also sometimes seen in 3D microphone polar patterns. Spherical harmonics are also an important component of the ambisonics mathematical formulation although they are not used here in that sense.

The parameter field is defined by

$$pf_{l,m}(u,v,t,f) \quad = \quad \frac{Y_\ell^m(u',v')cos(2\pi t f)+1}{2}$$

where u',v' are u,v converted to $[0,2\pi]$ and $[-\frac{\pi}{2},\frac{\pi}{2}]$. The Laplace spherical harmonics Y_ℓ^m are given by[24]

$$Y_\ell^m = \begin{cases} P_\ell^{|m|}(\cos\theta)\sin|m|\varphi & \text{if } m<0 \\ P_\ell^m(\cos\theta) & \text{if } m=0 \\ P_\ell^m(\cos\theta)\cos m\varphi & \text{if } m>0 \end{cases}$$

where $P_\ell^m(x)$ are the *associated Legendre polynomials* defined by the recurrence formulas

$$\begin{aligned} P_{\ell+1}^{\ell+1}(x) &= -(2\ell+1)\sqrt{1-x^2} \\ P_\ell^\ell(x) &= (-1)^l(2\ell-1)!!(1-x^2)^{(l/2)} \\ P_{\ell+1}^\ell(x) &= x(2\ell+1)P_\ell^\ell(x) \end{aligned}$$

where !! is the double factorial.

Spherical harmonics are not time dependent, in *ImmLib* they are animated by multiplying the spherical harmonic with a sinusoidal function of time, $cos(2\pi f t)$. The parameter field creates shifting patterns where one zone of the surface goes to zero while another goes to one, and then vice-versa. The zones themselves remain fixed for each spherical harmonic (unless the parameter field is rotated in real-time).

[23] More precisely, any square-integrable function on the sphere can be expanded as linear combination of the Laplace spherical harmonics, which form an orthonormal basis of the Hilbert space of square-integrable functions.

[24] Usually the spherical harmonic functions are normalized with a constant which depends on m, l, but this implementation does not apply normalization as it is not necessary for the purposes of *ImmLib*.

The *ImmLib* syntax for initializing this parameter field is `PFieldsphericalHarmonic(m, l).( t, f )`, where `m` and `l` are initialization arguments and `t` and `f` are modulatable arguments.

Pre-evaluated parameter fields

wave2D

The parameter field defined by

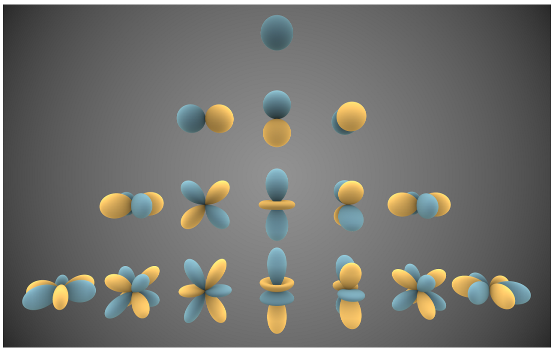
$$\begin{aligned} \mathrm{pf}(u, v, t, u_0, v_0, l, f) &= g(lr - ft) = g(\frac{1}{\lambda}(x - \lambda ft)) = g(\frac{1}{\lambda}(x - vt)) \\ \text{where } r &= \mathrm{dist}((u, v), (u_0, v_0)) \\ u, u_0 &\in [u_a, u_b] \\ v, v_0 &\in [v_a, v_b] \\ l, f &\geq 0 \\ l &= \frac{1}{\lambda} \\ & g : \mathbb{R} \to [0, 1] \end{aligned}$$

It implements a circular wave travelling through a surface caused by a point source at (u_0, v_0) emitting a signal given by function g where λ is the wavelength and f is the frequency of the source. $l = \frac{1}{\lambda}$ is the spatial frequency, the number of waves that fit in one spatial unit of measurement (such as on one meter). In the implementation l is variable and can be changed in real-time, on a physical system this would be equivalent to being able to change v, the speed of propagation in the medium, since $v = \frac{\lambda}{T} = \lambda f = \frac{f}{l}$.

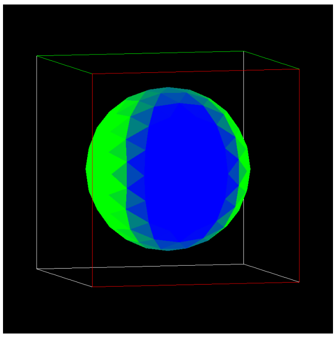
`wave2DSin` and `wave2DSaw` are specialized versions of `wave2D` with $g(x) = sin(2\pi x)$ and $g(x) = x \pmod 1$ respectively, simulating sinusoidal and saw wave signals:

$$\begin{aligned} \mathrm{pf}_{\mathrm{wave2DSin}}(u, v, t, u_0, v_0, l, f) &= sin(2\pi(lr - ft)) \\ \mathrm{pf}_{\mathrm{wave2DSaw}}(u, v, t, u_0, v_0, l, f) &= (lr - ft) \pmod 1 . \end{aligned}$$

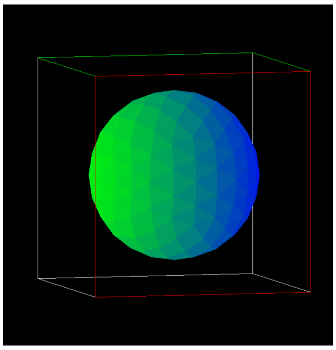
The *ImmLib* syntax for initializing these parameter fields is `PField.wave2D(t, u0,`



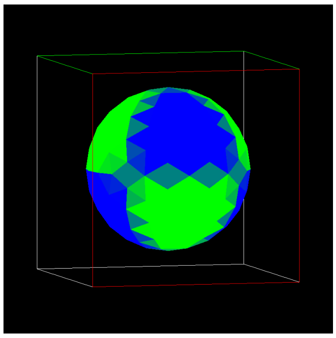
(a) Visualization of spherical harmonics with $l = 0, 1, 2, 3$. ©Iñigo Quílez, used here under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported license.



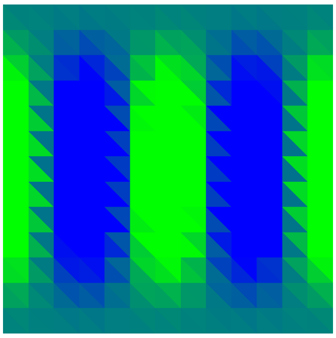
(b) $m = 2, l = 2$



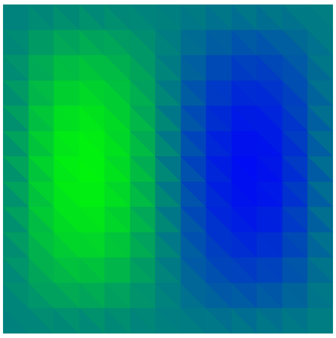
(c) $m = -1, l = 1$



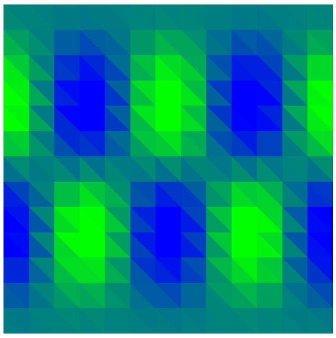
(d) $m = 2, l = 3$



(e) Plane, $m = 2, l = 2$

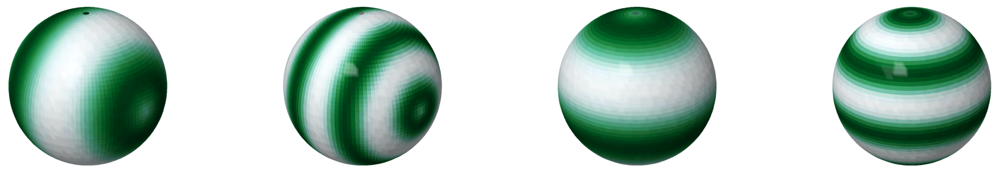


(f) Plane, $m = -1, l = 1$

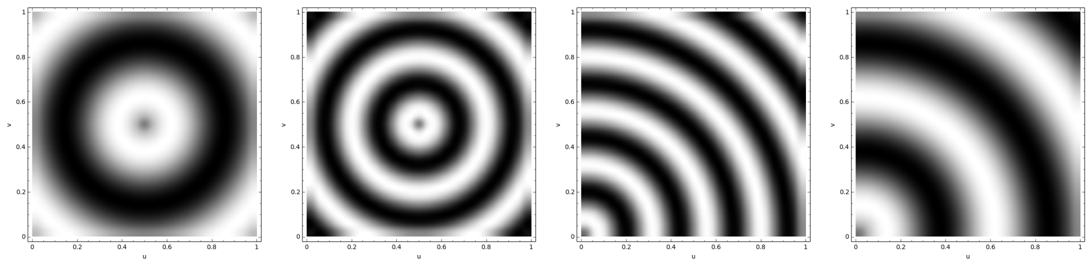


(g) Plane, $m = 2, l = 3$

Figure 3.31.: Visualizations of the sphericalHarmonic parameter field for two surfaces.



(a) $p = (0, 0)$ $l = 2$ (b) $p = (0, 0)$ $l = 4$ (c) $p = (0, \frac{\pi}{2})$ $l = 2$ (d) $p = (0, \frac{\pi}{2})$ $l = 4$



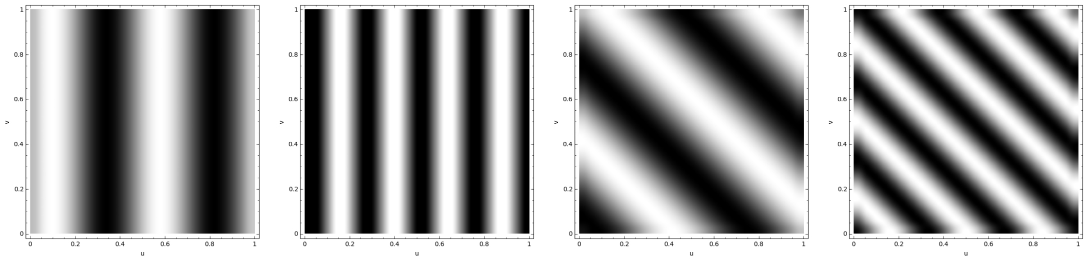
(e) Plane $p = (0.5, 0.5)$ $l = 2$ (f) Plane $p = (0.5, 0.5)$ $l = 4$ (g) Plane $p = (0.0, 0.0)$ $l = 4$ (h) Plane $p = (0.0, 0.0)$ $l = 2$

Figure 3.32.: Plots of the wave2DSin parameter field for two surfaces ($p = (u_0, v_0)$).

`v0, l, freq, g)`, `PField.wave2DSin(t, u0, v0, l, freq)`, and `PField.wave2DSaw(t, u0, v0, l, freq)`. All three parameter fields were implemented pre-evaluated in order to be able to change the frequency of the oscillation smoothly, without the discontinuities mentioned in section §3.4. The details of the implementation of the frequency-change smoothing will be given in the next chapter. Figure 3.32 shows different visualizations of this parameter field.

wave1D

This parameter field, which is only valid for the plane, implements a plane wave travelling through the surface along a direction determined by angle θ emitting a signal given by function g where λ is the wavelength and f is the frequency of the source. While in the previous parameter field the wavefront is a circle in this case the wavefront is a line. It is defined by



(a) $\theta = 0$ and $l = 2$ (b) $\theta = 0$ and $l = 4$ (c) $\theta = \frac{\pi}{4}$ and $l = 2$ (d) $\theta = \frac{\pi}{4}$ and $l = 4$

Figure 3.33.: Plots of the wave1DSin parameter field on the plane.

$$
\begin{aligned}
\mathrm{pf}(u, v, t, \theta, l, f) &= g(lu' - ft) \\
\text{where } (u', v') &= r(u, v, \theta) \\
u, u_0 &\in [u_a, u_b] \\
v, v_0 &\in [v_a, v_b] \\
l, f &\geq 0 \\
l &= \frac{1}{\lambda} \\
& g : \mathbb{R} \to [0, 1],
\end{aligned}
$$

and r is the rotation function for the plane.

Specialized versions for the sine and saw functions are also defined:

$$
\begin{aligned}
\mathrm{pf}_{\text{wave1DSin}}(u, v, t, \theta, l, f) &= sin(2\pi(u'r - ft)) \\
\mathrm{pf}_{\text{wave1DSaw}}(u, v, t, \theta, l, f) &= (u'r - ft)(\mod 1) \\
\text{where } (u', v') &= r(u, v, \theta)\,.
\end{aligned}
$$

The *ImmLib* syntax for initializing these parameter fields is `PField.wave1D(t, u0, v0, l, freq, g)`, `PField.wave1DSin(t, u0, v0, l, freq)`, and `PField.wave1DSaw(t, u0, v0, l, freq)`. Figure 3.33 shows different plots of the parameter field.

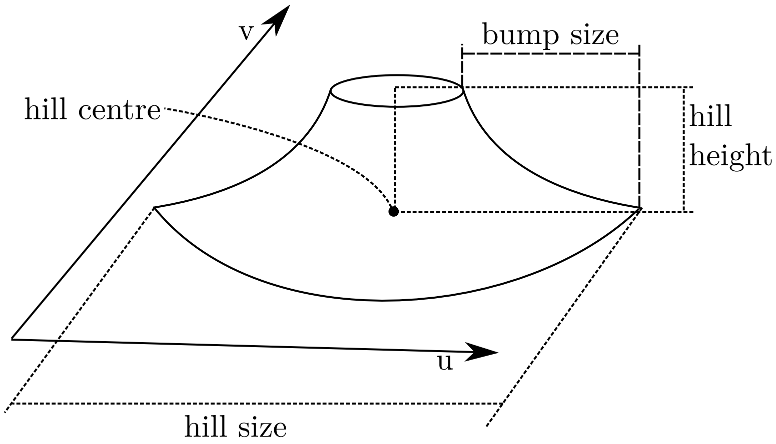


Figure 3.34.: Diagram of a single hill of the randomHills parameter field.

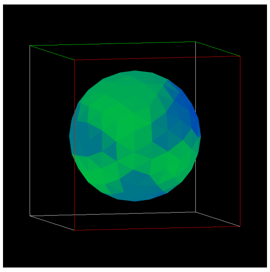
randomHills

This parameter field is implemented such that periodically a new mathematical function is created and cross-faded with the previously active function. The cross-fade time equals the period, such that functions are seamlessly changed. The created function is composed of the sum of a given number of spotlight parameter fields, each one centred around a random point, with random wideness (c parameter) and multiplied with a random weight (see figure 3.34). Each spotlight parameter field acts as a “hill” with a certain wideness and height. The sum of the various randomly-determined hills creates a pattern with hills and valleys (see figure 3.35i). The period, number of summed spotlights, range for the wideness, and range for the heights can be changed in real-time by the user, although new values only take effect when a new function is activated at the end of each cross-fade. A second version of this parameter field (randomHills2) also randomizes the cross-fade time within a selectable range.

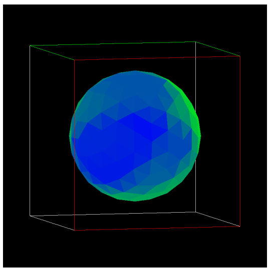
The *ImmLib* syntax for initializing this parameter field is `PField.randomHills(t, numSecs, numHills, sizeA, sizeB, bumpSize, heightA, heightB)`. Figure 3.35 shows different visualizations of the parameter field.

moveHills

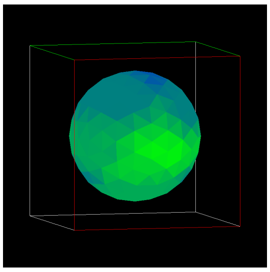
This parameter field is implemented such that periodically a set of hills, such as described for the randomHills parameter field, have their centres moved to randomly selected positions. At each cycle, the centre of each hill is budged by a certain amount in a random direction, creating Brownian movement. The period, number of summed



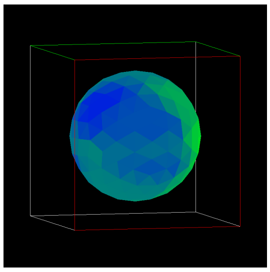
(a)



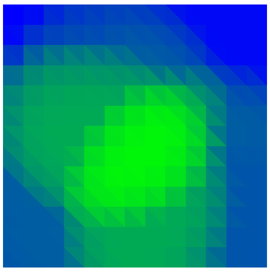
(b)



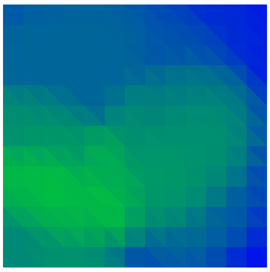
(c)



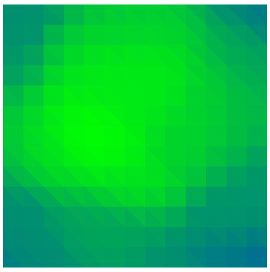
(d)



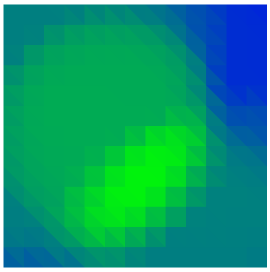
(e)



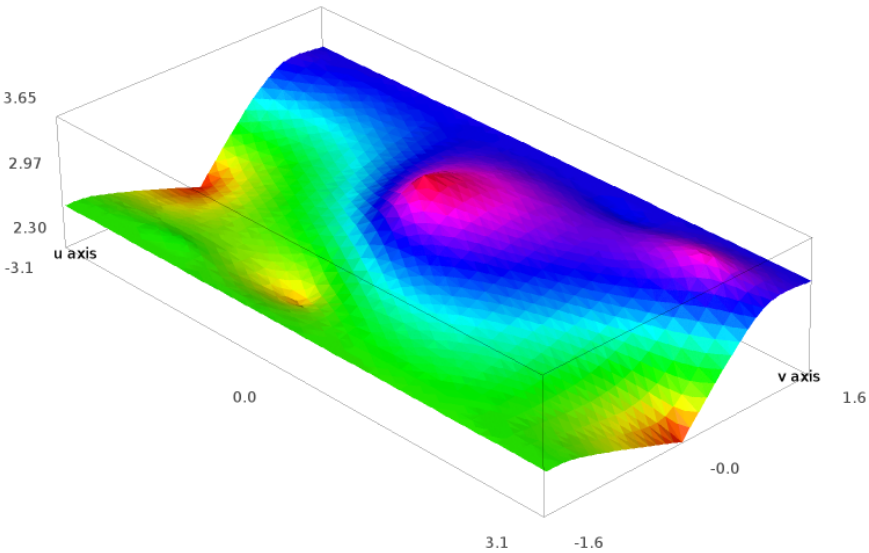
(f)



(g)



(h)



(i) Artistic rendition of a set of hills.

Figure 3.35.: Visualizations of the randomHills parameter field.

spotlights, range for the hill wideness, and random dislocation range can be changed in real-time by the user.

The *ImmLib* syntax for initializing this parameter field is `PField.moveHills(t, numSecs, numHills, size, step)`.

parameter sequences

BZ reaction

This parameter sequence implements the idealised Belousov–Zhabotinsky reaction. This is a chemical reaction which exhibits temporary oscillating behaviour, self-organizing on a two-dimensional plane, resulting in spirals. In this reaction 3 chemicals A, B, C interact according to chemical equations

$$\begin{aligned} A + B &\rightarrow 2A \\ B + C &\rightarrow 2B \\ C + A &\rightarrow 2C. \end{aligned}$$

A can be created only if B exists, B can be created only if C exists and C can be created only if A exists. Alasdair Turner [112] suggests a simple algorithmic implementation of this model. If a_t, b_t, c_t are the quantities of these chemicals at time t, and if time is discretized, then

$$\begin{aligned} a_{t+1} &= a_t + a_t(b_t - c_t) \\ b_{t+1} &= b_t + b_t(c_t - a_t) \\ c_{t+1} &= c_t + c_t(a_t - b_t). \end{aligned}$$

Spatially the chemicals are diffused by considering a two-dimensional cellular automata model, averaging each chemical for each cell and its neighbour cells before applying the reaction equations.

This parameter sequence was implemented in *ImmLib*, by calculating a function f which generates a new set of values based on the recursive relations above, for each point, on every timer tick (see appendix D.7). The averaging, simulating the diffusion process, is done considering the point plus the three closest neighbours. After averaging, the resulting value is divided by k which is a parameter of the parameter sequence.

Chapter 4.

ImmLib software library

This chapter introduces the *ImmLib* software library. It describes its syntax and user interface, detailing how surfaces and parameter fields can be defined, and how both can be associated with a given sound process. An overview of the most important aspects of the implementation is also provided. This chapter also details how the library is used in practice, when composing electronic music, and discusses some of the technical issues involved in the implementation.

4.1. Overview

ImmLib is a library, a set of classes[1], for the *SuperCollider* language [113], which implements parameter field spatialization.

SuperCollider is a programming language for real-time audio synthesis and algorithmic composition, with an efficient sound synthesis engine (*scsynth*) designed specifically for music composition and performance. Coming from the tradition of MUSIC-N, it allows the interconnection of ugens to form larger digital signal processing (DSP) networks. The *SuperCollider* language (*sclang*[2]), is a general purpose object-oriented language (OOP) with some features of functional programming. Blocks of code are evaluated in real-time by the *sclang* interpreter. The *SuperCollider* standard class library includes, besides the general purpose abstractions such as collections, powerful procedures for algorithmic creation of ugen graphs and rich abstractions for musical events (routines, patterns, etc.). The synthesis engine (*scsynth*), written in *C/C++*, combines ugens ac-

[1] Besides the *SuperCollider* classes, *ImmLib* also provides a separate program, named *pfVisualizer*, for 3D visualization of parameter fields.

[2] *sclang* is also the name of the interpreter for the *SuperCollider* language.

cording to pre-made declarative descriptions, `SynthDef`s, instantiating them as `Synth`*s*, which can be dynamically created and destroyed. *scsynth* is a separate process and communicates with *sclang* via Open Sound Control (OSC) [114] messages.

SuperCollider was chosen based on multiple factors. It is a high-level general-purpose language with modern features such as closures, objects, and coroutines which allow for a strong level of abstraction which is necessary for large-scale projects such as *ImmLib*. Since it is an object-oriented language it can easily be extended by adding new classes or overloading existing ones. At the same time it is geared towards music and sound, therefore a significant part of the low-level operations are presented through higher-level domain-specific languages or application programming interfaces (APIs), allowing for faster prototyping of musical software. The sound server is very efficient, and there is a large collection of ugens available covering most synthesis and audio processing needs. The server's ability to instantiate ugen graphs dynamically from a given definition was important for implementing the parameter field spatialization algorithm since it greatly simplifies automatic parallel creation of any number of instances of the same sound process definition. *SuperCollider* is cross-platform, therefore *ImmLib* can be used both *in Windows*, *MacOSX* and *Linux*. Finally, it has a strong community of users and developers, which follow an open-source philosophy.

ImmLib is implemented on top of several existing *SuperCollider* libraries (see figure 4.1), the most important of which is *UnitLib* [115], a high-level performance and composition system, created as part of *WFSCollider* [68], a software for WFS spatialization[3]. *UnitLib* mimics to a certain extent the model of a DAW, featuring tracks, which are called *chains*, and effects, which are called *units*, although still maintaining the algorithmic character of a programming language library. The evaluation of parameter fields is implemented using the functional reactive programming (FRP) paradigm which enables sampling-rate independent reasoning about time-varying values. For this purpose, *ImmLib* makes use of *FPLib*, a library for functional programming in *SuperCollider* developed by the author, mostly during this research. The actual spatialization is performed either directly, by sending the multiple generated audio signals to the corresponding sound card outputs, or through VBAP panning using the `VBAP` ugen from

[3] *UnitLib* was developed by Wouter Snoei and the author as a project of the Game of Life Foundation [91].

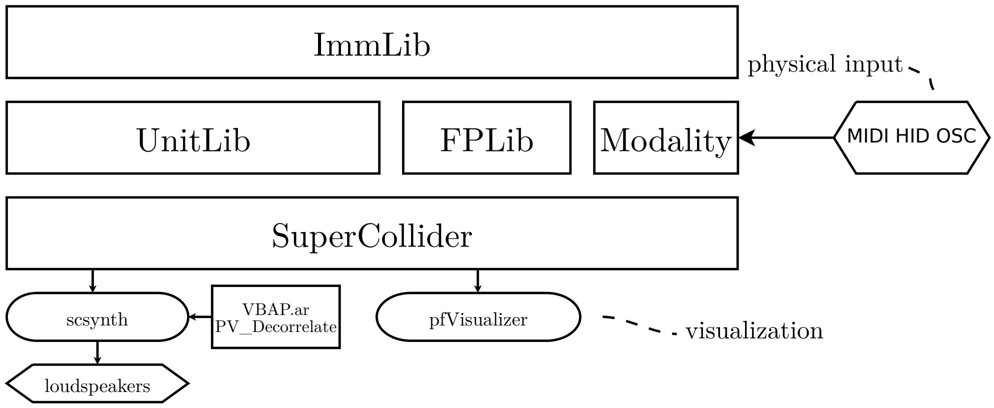


Figure 4.1.: ImmLib software architecture.

the *sc3plugins* library, originally developed for the *BEASTmulch* system [116][4]. There is also a stereo preview mode which uses binaural synthesis with virtual ambisonics, implemented using the Ambisonic Toolkit (ATK) library for *SuperCollider* [58]. For decorrelation of mono sound files, *ImmLib* uses a ugen based on a technique by Kendall [11], `PV_Decorrelate`, also from *BEASTmulch.* For real-time visualization of parameter fields, a separate *OpenGL* program was developed, which is launched when needed and controlled via OSC messages. Real-time control via physical devices is handled using the *Modality* library [117], which greatly simplifies the use of MIDI and HID devices in *SuperCollider.*

UnitLib encompasses all the components of *WFSCollider* unrelated to the WFS spatialization algorithm. It automates many of the low-level tasks that are part of the workflow in *SuperCollider*, such as loading and unloading buffers for sound file playback, ordering the synth node tree, CPU-load balancing, checking that synth arguments are within constraints or patching together the inputs and outputs of multiple synths using audio buses. It also provides a *score* abstraction, allowing the placement of events on a timeline. Scores are more limited than what the abstractions for real-time algorithmic event creation in *SuperCollider* allow, but on the other hand they present a time abstraction familiar to many computer music practitioners, allow a quick workflow when dealing with events with specific start times and durations via a GUI interface and, since they can be created and manipulated algorithmically, still maintain some of the flexibility of *SuperCollider*'s standard library.

UnitLib was selected as the basis for *ImmLib* since it provides a high-level envir-

[4]The VBAP infrastructure for UnitLib is implemented in the *SuperCollider* library of *WFSCollider*, for practical reasons. That library is therefore also needed by *ImmLib*.

onment for composition, where many of the technical steps which are part of the implementation of parameter field spatialization can be made invisible to the user. Also, the DAW-type interface allows for a fixed-medium piece to be entirely created in *SuperCollider,* without the need for additional software for sequencing and mixing.

The implementation of parameter fields in *ImmLib* was designed based on two requirements:

1. It should be possible for users to write their own parameter field definitions.

2. It should be possible to write parameter fields using syntax that is as close as possible to corresponding mathematical formulas.

The first requirement dictated that it should be relatively easy to write a parameter field definition, without for instance having to write classes and recompile the language. The second requirement suggested that parameter fields should be evaluated in the language instead of the sound server. In *sclang* it is possible to write mathematical functions in a way that is almost identical to the original mathematical formula, while in the server one cannot directly write functions, but only connect ugens. Although in some cases the translation to ugen graphs can still be similar to the mathematical formula, in other cases it might be quite different or even impossible. At the start of this research it was not known which parameter fields would be created, and what their formulas would be, therefore erring on the side of caution, it was decided that parameter fields would be evaluated in *sclang.* Finally, as is the case with synthesis definitions and temporal event patterns in *SuperCollider*, it was desired that the definitions of parameter fields be declarative[5].

4.2. *UnitLib*

The interface of *ImmLib* is essentially the same as *UnitLib*, it is therefore important to describe the syntax and semantics of the later. The main classes of *UnitLib* are depicted in figure 4.2.

SuperCollider and *UnitLib* both use a two step approach to instantiating a synth, a sound process running on the server. In SuperCollider the ugen graph is described

[5]A declarative approach in programming is one that describes the problem to be solved, instead of describing the specific steps needed to solve the problem.

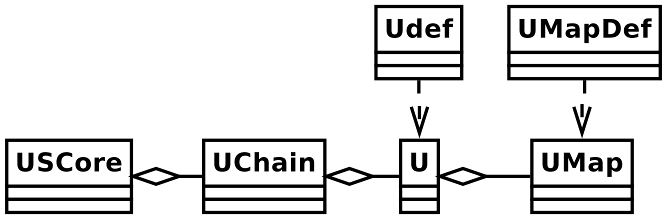


Figure 4.2.: Diagram of the most important *UnitLib* classes. A line with a rhombus connects two classes, where one contains an instance of the other. A dotted line connects two classes, where one makes use in some way of the other.

using an anonymous function, associated with a name and stored in a `SynthDef`. To start a synthesis process based on a ugen-graph description, the name of the `SynthDef` is used. In *UnitLib* a ugen graph is declared using the `Udef` class. This class stores a list of arguments which correspond to named controls, nodes of the ugen graph which can be set to a specific numerical value from the language:

```
Udef(\noise, { |amp = 1.0|
  UOut.ar(0, WhiteNoise.ar * amp )
})
```

A `Udef` is instantiated with the `U` class (which is read as “unit”), given a list of pairs of argument names and their values:

```
U(\noise, [\amp, 0.5]).prepareAndStart
```

Arguments of `U` correspond to the controls of `SynthDef`, they are the parameters of the sound process which can be changed in real-time by the user.

Units can be chained using a `UChain`:

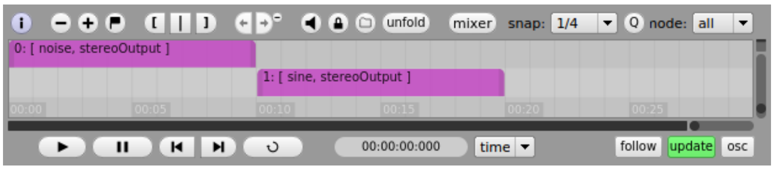
```
UChain(
  [\noise, [\amp, 0.5]],
  [\lowPass, [\freq, 1000]],
  [\stereoOutput, []]
).prepareAndStart
```

Here the output of a white noise unit is sent to a low pass unit, whose output is sent to two loudspeakers using stereo panning. Each unit can access the audio output of previous units, and send audio to subsequent units using a system of mono audio buses, referenced using a positive integer, which are internal to each chain and not accessible by other chains. This greatly simplifies the standard *SuperCollider* bus management.

The internal chain bus system together with the stored `Udef`s constitute a simple patching system. `Udef`s are defined once and saved in the user's `Udef` library. *UnitLib* includes a library of `Udef`s covering the most basic sound processes such as noise generators, oscillators, file playback, EQ, reverb or compression. While working on a chain the user can quickly add, replace or remove units to the chain, with the re-patching of the audio happening transparently without the need for explicit bus management[6].

Chains can be placed on a timeline as events with a start time, duration and fade in and fade out times using the `UScore` class:

```
UScore(
  UChain( 0.0 , 0, 10.0,  \noise , \stereoOutput ),
  UChain(10.0 , 1, 10.0, [\sine , [\freq ,440]] , \
      stereoOutput )
).gui
```



The score above would play white noise for 10 seconds, followed by 10 seconds of a sine tone at 440Hz. A `UScore` is a complete specification of the internal state of several chains together, with the corresponding time information, and can be persisted to disk.

Besides being able to send audio from one unit (`U`) to another, it is also possible to send the output of one unit to the parameter of another unit, instead of the input. These special modulator units are implemented by the `UMap` class. When working on a ugen graph it is often the case that initially some values are set to a constant, yet later they are changed into a ugen, in order increase complexity. This is also the case with parameters of a unit, which correspond to ugens that can be directly set to a value by the user. With `Umaps` it is possible to quickly set up different types of modulations for parameters of a unit without having to change the original `Udef`, re-enforcing code re-use. Using `Umaps` it is possible to assign arbitrary breakpoint envelopes to a unit argument, as well as low frequency oscillators (LFOs). Envelope and LFO `UMaps` are integrated with the timeline in such a way that starting playback mid-way through a score will set the start position of the envelopes and initial phase of the LFOs accordingly. As an example, the static frequency value 400Hz in

```
UChain([\sine, [\freq, 400]], \stereoOutput)
```

can be changed to an exponential envelope which goes from 100Hz to 1000Hz in 60s with the following change (see figure 4.3):

[6]Nevertheless, the chain must be stopped and restarted in order for the changes to take effect.

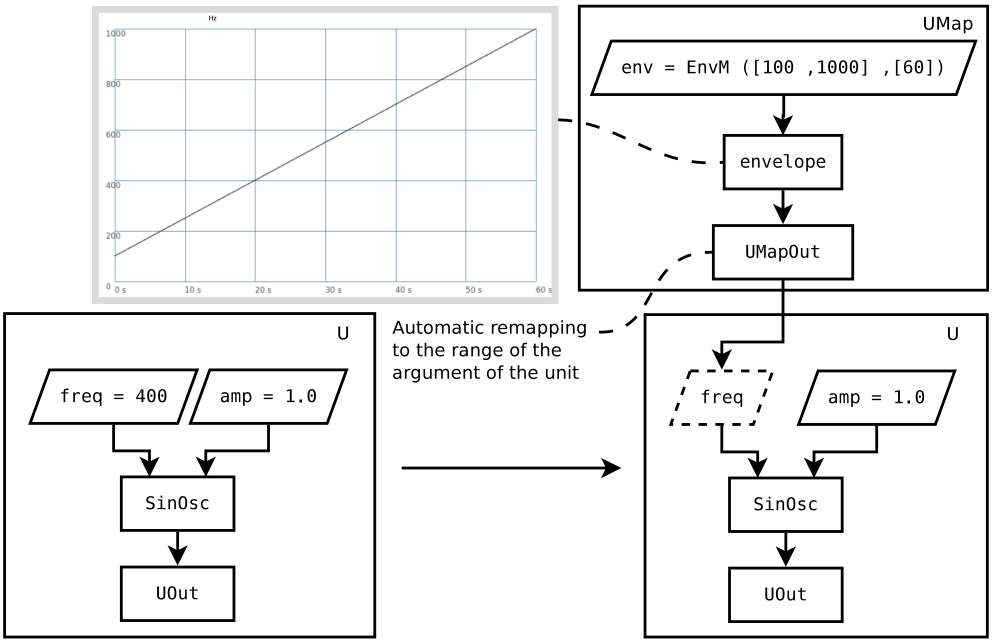


Figure 4.3.: Assignment of a `UMap` with a breakpoint envelope to the `freq` parameter of a unit.

```
var env = [\envelope, [\env, EnvM([100,1000],[60])]];
UChain([\sine, [\freq, env]], \stereoOutput)
```

Graphical User Interfaces

Most of the functionality of *UnitLib*, with the exception of defining the `Udef`s, can be controlled from GUIs. The `UChain` GUI presents all the parameters of each unit, which have widgets appropriate for the corresponding data type, such as sliders, 2D sliders, range sliders, file pickers, and EQ graphs. It is possible to drag units onto, or in between, existing units on the GUI, as well as drag umaps onto the widgets of the parameters of a unit (see figure 4.12). The GUI for `UScore` provides the usual sound event editing capabilities typical of DAWs. It is also possible to change the audio and control rate interconnections between units of a chain visually.

Object persistence as code

SuperCollider derives much of its power from being a general purpose (text) programming language, allowing straightforward and terse algorithmic creation of complex DSP graphs and time structures. On the other hand some tasks when creating a composition, such as fine-tuning start times and durations, are easier to accomplish using a GUI than programmatically. *UnitLib* objects can be seamlessly converted from code

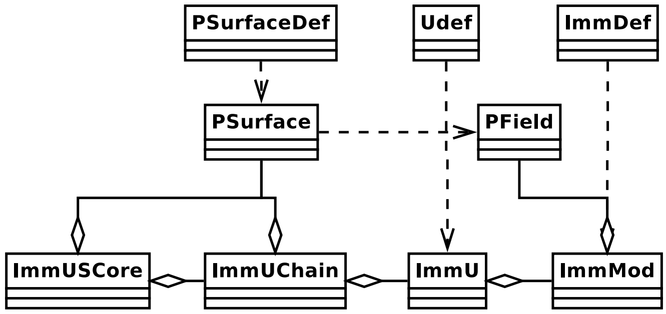


Figure 4.4.: Diagram of the most important *ImmLib* classes. A line with a rhombus connects two classes, where one contains an instance of the other. A dotted line connects two classes, where one makes use in some way of the other.

to GUI and back, making it possible to switch between the two approaches depending on the specific musical task or problem at hand. This is possible because any *UnitLib* object is capable of generating the code that when interpreted will instantiate a new object in the exact same state as the original one. Scores are therefore persisted to disk simply by generating and saving the *SuperCollider* code which when interpreted will recreate the corresponding object. Being able to tweak parameters using the GUI while listening to a sound process, to add, replace and remove units by dragging from a list of existing `Udef`s, and to copy-paste events inside a score or between scores using the GUI, allows one to joyfully and quickly explore the design space of a piece. At the same time, being able to use code to construct any of the *UnitLib* objects, allows for abstraction, avoids repetitive manual tasks and opens up the possibilities of algorithmic creation of complex structures.

4.3. Parallel synth creation

The main classes in *ImmLib* are `ImmU`, `ImmUChain` and `ImmUScore` which extend respectively `U`, `UChain` and `UScore` from *UnitLib*. They are responsible for creating and managing the n parallel synths for each unit, fetching the corresponding surface, and managing the spatialization engine. The `PSurface` class implements *ImmLib* surfaces, as specified in section §3.3 while the `PField` class is used for defining and evaluating parameter fields. The general outline of the relationship between the most important classes in *ImmLib* can be seen in figure 4.4.

The classes `ImmUChain` and `ImmU` inherit from `ParUChain` and `ParU`. One of the key steps in the parameter field spatialization algorithm is the creation of multiple decorrelated signals. `ParUChain` and `ParU` implement the automatic creation of n synths for each unit (usually each unit creates a single synth), which is necessary to create the n decorrelated signals from a single sound process definition. The syntax for `ParU` is the same as `U` except that the first argument of the constructor is an integer determining how many synths to create:

```
ParU(10, [\sine, [\freq, 400]])
```

The `ParArg` class, which wraps around an array of size n, allows setting an argument of the unit to different values at each of the n synths:

```
ParU(2, \noise, [\output, [\bus, ParArg([0, 1])]] )
```

This code will create two decorrelated white noise signals sent to the first and second outputs of the sound card. The two signals are decorrelated because *SuperCollider* automatically assigns a new random seed to each of the synths.

Surfaces are defined in *ImmLib* with the `PSurface` class, while `PSurfaceDef` associates a surface with a name. `ImmUChain` and `ImmUScore` classes reference the surface always by name. For instance,

```
PSurfaceDef(\surf1, PSphere(40))
```

associates the name `surf1` with a sphere featuring 40 points, and

```
ImmUChain( \surf1, \noise, [\lowPass, [\freq, 1000]] )
```

will automatically fetch the surface named `surf1`, detect that the surface contains 40 points, and create 40 synths for each of the two units (`noise` and `lowPass`). In order to change the surface being used on a score, only the `PSurfaceDef` statement needs to be changed, making it relatively easy to play the same score on different loudspeaker setups with different geometries.

More details about the CPU load-balancing and resource management system used by `ParUChain` can be seen in appendix section §C.1.

4.4. Surfaces and the spatialization engine

The `PSurface` class stores all the geometric information relative to an *ImmLib* surface: the coordinate chart function, the points of the sample set in several different formats, the domain of the coordinate chart and options regarding the rendering method. This

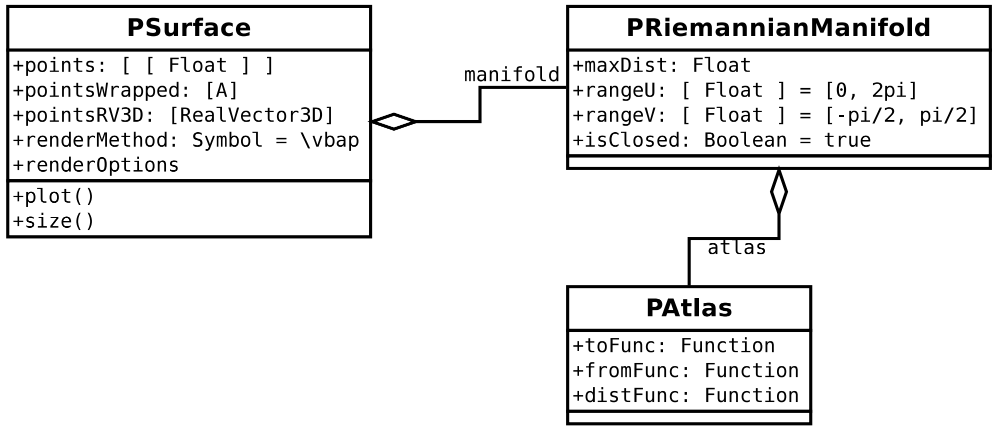


Figure 4.5.: `PSurface` UML class diagram.

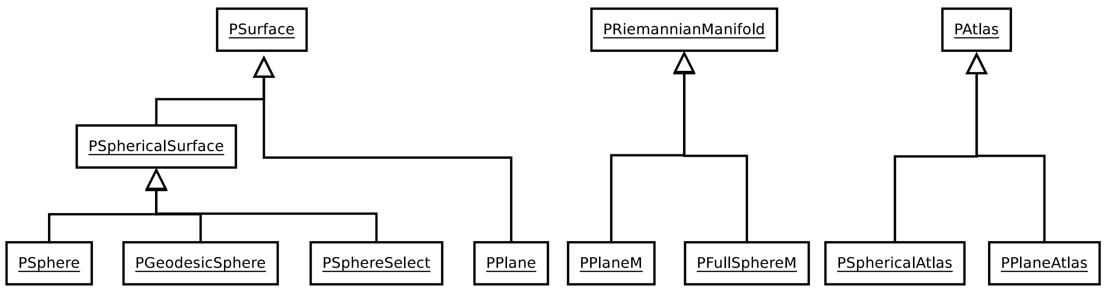


Figure 4.6.: Inheritance diagram for surface classes.

class is used when evaluating parameter fields, by the visualisation tool, for determining how many parallel synths to create, and for creating buses for routing audio from each chain to its panner. Table 4.1 together with figures 4.5 and 4.6 detail the implementation of the `PSurface` class hierarchy.

The spherical spiral discretization procedure is implemented by `PSphere(n)` where `n` is the number of points, the geodesic dome discretization algorithm is implemented by `PGeodesicSphere(n)` where `n` is the geodesation order. The `PSphereSelect(n,f)` class implements the hemisphere and similar surfaces using a test function f to select points generated using the spherical spiral algorithm. The plane surface is implemented by `PPlane(origin, dx, dy, n, m)` where origin, dx and dy are vectors and `n` and `m` the number of points along `dx` and `dy`.

Spatialization modes

Currently, *ImmLib* has two spatialization modes when used with loudspeakers: VBAP and direct. For preview over headphones, binaural synthesis using 1st order ambisonics is available, although it is of limited use since a considerable amount of definition in

Variable (Class)	Purpose
`points` (`PSurface`)	Points of the sample set with coordinates in D, the domain of the coordinate chart. For the sphere $D = [0, 2\pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ and for the plane $D = [0, 1] \times [0, 1]$. These are the values that are passed to the `u` and `v` arguments of the function stored in `PField.func`.
`pointsWrapped` (`PSurface`)	Points of the sample set encoded with a class appropriate for the panners.
`pointsRV3D` (`PSurface`)	Points of the sample set mapped to $\mathbb{R}^3$. Needed for sending to *pfVisualizer*.
`renderMethod` (`PSurface`)	Either 'vbap' or 'direct'.
`renderOptions` (`PSurface`)	Currently only used for direct mode in which case it is assigned a dictionary such as `(\spkIndxs: (0..24) )`, specifying the indexes of the loudspeakers to be associated with each point in the sample set.
`fromFunc` (`PRiemannianManifold`)	This function is the coordinate chart of the surface, taking arguments `u` (float) and `v` (float) and returning a `RealVector3D`.
`toFunc` (`PRiemannianManifold`)	The inverse of the coordinate chart. Takes one argument of type `RealVector3D` and returns an array `[u,v]` with two floats.
`distFunc` (`PRiemannianManifold`)	This function takes arguments `u1,v1,u2,v2`, all floats, corresponding to the positions of two points in the parameter space, and returns one float, their distance.
`rangeU`, `rangeV` (`PAtlas`)	Two arrays `[ua,ub]` and `[va,vb]` corresponding to the domain of the coordinate chart.
`maxDist` (`PAtlas`)	Maximum distance possible on the surface.
`isClosed` (`PAtlas`)	Whether the surface is closed, such as the sphere, or open, such as the plane.

Table 4.1.: Variables of surface related classes.

the spatial surface patterns is lost. VBAP and stereo binaural correspond to what was called virtual mode in section §3.3.5. When considering different panning techniques which could be used with loudspeakers in virtual mode, the following was taken into account.

In virtual mode the panning method chosen must be able to pan sounds on a surface in three-dimensional space, not just along a curve on a two-dimensional plane such as a circle. For reproduction over loudspeakers we have seen in the review of existing panning methods that VBAP, HOA and WFS are all capable of panning sources in three dimensions, at least on the surface of a virtual sphere whose radius is bigger than the largest loudspeaker distance to the centre. 3D WFS, using a planar or spherical array, is capable of placing a virtual source anywhere in three-dimensional space[7]. Nevertheless, when considering a vertical rectangular loudspeaker array, it should be taken into account that the distances between loudspeakers in the x and z axis must be lower than 0.15m, therefore a system covering a wall of 3 by 6 meters would need at least 800 loudspeakers [5]. Not only is using such number of loudspeakers infeasible but with this density of loudspeakers it would be preferable to use *ImmLib* in direct mode, saving the high CPU cost of the WFS rendering algorithm. A 2D WFS system is capable of placing virtual sources on a rectangle defined by 4 loudspeaker linear arrays, but to do so it must use focused sources. Informal listening tests conducted previously by the author on a rectangular 192-loudspeaker WFS system [91] determined that focused sources produced using four linear arrays forming a rectangle are not stable enough to allow good differentiation between multiple simultaneous decorrelated streams arranged in an horizontal grid. The movements created with a virtual grid of sources in this system using parameter field spatialization were vague, lacking definition (for more details see the author's master thesis [70]). For this reason only VBAP and HOA were considered suitable for use with *ImmLib* in virtual mode over loudspeaker arrays. Currently only VBAP is supported, although HOA could be added in the future.

VBAP was chosen for the implementation due to several factors, some perceptual and other practical: HOA can create quite smooth virtual trajectories but *ImmLib*

[7] Simulating free-field conditions, therefore only the wavefront-curvature and distance-attenuation cues are present, the ratio of direct to reverberated sound is missing, which is an important cue for distance perception.

only creates static virtual sources as such this feature is not needed; HOA requires separate encoding and decoding phases which complicates the implementation; Given a virtual source VBAP only sends audio to the 3 closest loudspeakers to that source. In a large listening room or concert hall this can greatly reduce the precedence effect for very off-centre listeners. A listener at the edges of the listening space will probably be quite close to one loudspeaker, VBAP will guarantee that the outputs of virtual sources at the opposite side of the room will only be played at loudspeakers at the opposite side of the room, ensuring that the precedence effect cannot cause those sources at the opposite side to be perceived as being in the wrong side of the room. On the other hand, on some decoding settings and for some source positions, HOA will play audio in all loudspeakers simultaneously. Therefore it is more prone to have the precedence effect destroy the immersive effect for highly off-centre listeners. Finally, a well maintained set of VBAP ugens was already available for SuperCollider.

When in direct mode the mono outputs of the multiple synths of the last unit in a `ImmUChain` are sent directly to the soundcard outputs, routed according to the loudspeaker index list stored in `PSurface`. The points stored in the `PSurface` must match the actual positions of the loudspeakers in space for the system to work properly. It is not necessary to obtain the coordinates of the loudspeakers with great accuracy, all that is required is that the points stored in the `PSurface` are proportional to the actual coordinates of the loudspeakers in physical space. For example, if the loudspeaker grid is a rectangle with 3:2 proportions, then the points stored in the `PSurface` must also form a rectangle with 3:2 proportions, although the actual location of the points need not match.

When in VBAP mode an extra unit is added to every `ImmUChain`. This unit sends the output of synths of the previously last unit to a hidden `ParUChain` containing the VBAP panners. The VBAP panner `Udef` uses the `VBAP.ar` ugen and does additional distance correction using a delay ugen whose delay time for a given loudspeaker is $\frac{(\text{maxDist}-\text{dist})}{c}$, where dist is the distance of the loudspeaker to the centre, maxDist is the maximum of such distances, and c the speed of sound in air. Only one `ParUChain` panner is started for each surface, therefore all `ImmUChain`s using that surface share the same panners, achieving savings in CPU usage. This optimization is possible because all `ImmUChain`s using the same surface need to send their multiple mono audio outputs

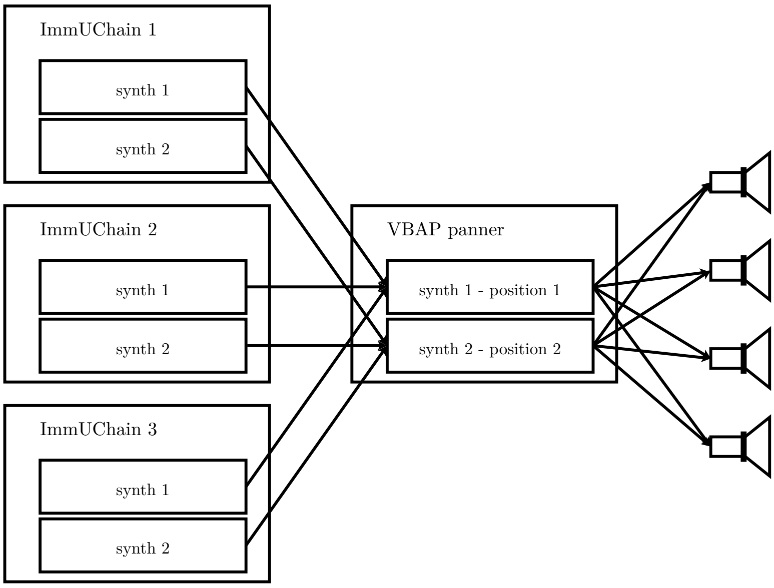


Figure 4.7.: Multiple *ImmLib* events using a set of VBAP panners associated with a specific surface.

to the same set of panners corresponding to the same set of points (see figure 4.7). In VBAP mode `ImmUChain`s associated with different surfaces must be placed in separate `ImmUScore`s, one per surface, since the VBAP panners for a surface are associated with the `ImmUScore`. Nevertheless, `ImmUScore`s can be placed inside `UScore`s so that events using different surfaces can be played simultaneously (see an example in appendix D.3). For instance, a sphere and a plane can be used simultaneously.

4.5. Functional Reactive Programming, animation and real-time interaction

There are two key reasons for the use of functional reactive programming (FRP) in *ImmLib*. First, early on it was decided that parameter fields would be evaluated in the language, and not in the sound server, in order to be able to directly write mathematical functions and simplify the implementation of the visualization tool (see section §4.1). To animate a parameter field in the language, a timer must be set that repeatedly calls the corresponding function with a new time value. Second, parameter field and unit parameters should be controllable in real-time from GUIs, MIDI or HID devices, or via OSC messages.

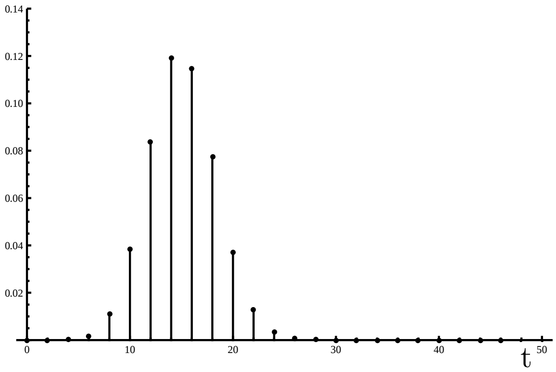
The traditional method in computer programming of dealing with both incoming events and animation is through *callback* functions. A callback function is registered and called every time an event is received or every time the animation timer requests a new frame. Callback functions, although easy to define, lack composability and require the basic patching, registering and de-registering, to be handled by the user. Functional Reactive Programming, or FRP, is a paradigm for programming dynamic and reactive systems using first-class composable abstractions that tries to address some of these issues. Both event processing and animation can be handled quite elegantly with this approach:

> "Functional Reactive Programming, or FRP, is a general framework for programming hybrid systems in a high-level, declarative manner. The key ideas in FRP are its notions of behaviors and events. Behaviors are time-varying, reactive values, while events are time-ordered sequences of discrete-time event occurrences." [...] "Behaviors and events are both first class values in FRP, and there is a rich set of operators (combinators) that the user can use to compose new behaviors and events from existing ones. An FRP program is just a set of mutually-recursive behaviors and events, each of them built up from static (non-time-varying) values and/or other behaviors and events."[118, p.242]

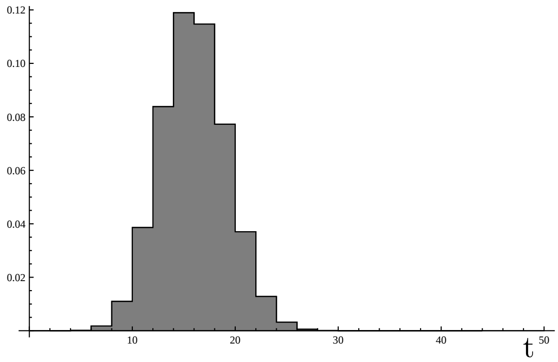
FRP is a novel programming technique still being actively researched and reworked, nevertheless there are already several mature libraries, with the most popular being geared towards web front-end development (*Elm* [119]). Most of the original work on FRP was done on the *Haskell* programming language[8]. Classic FRP was conceived by Conal Elliot and Paul Hudak and first presented on the paper "Functional Reactive Animation"[120]. Classic FRP demonstrated itself to be an elegant approach but had issues with time and space leaks[9] which were addressed in later formulations, namely Arrowized FRP [121].

An FRP implementation for *SuperCollider* was created as part of the *FPLib* library [122]. The latter makes available a number of functional programming constructs such as the *functor*, *applicative functor*, *traversable* and *monad* interfaces in *SuperCollider*. The FRP classes were initially ported from the *reactive-web [123]* library for the *Scala* language and then adapted using ideas from *reactive-banana* [124], a modern

[8]Haskell is a modern, pure, non-strict, statically-typed functional programming language.
[9]Memory grows unexpectedly or computations take unexpectedly too long to complete.



(a) Event stream represented as a list of time-value pairs.



(b) Signal represented as a step function.

Figure 4.8.: FRP: event streams and signals.

FRP library for *Haskell*.

The main entities for FRP are the `EventStream` and `FPSignal` classes. Signals are slightly different from the behaviours of Classic FRP: behaviours model any continuous time function $f : \mathbb{R} \to A$, while signals only model step functions (see figure 4.8b). An event stream can be thought of as a list of time and value pairs: $\{(t_1, x_1), (t_2, x_2), ...\}$ (see figure 4.8a). Outputs are defined in terms of inputs using *combinators* (pure functions) applied to the signals or event streams in order to construct an *event graph* (see figure 4.9). To get events into the event graph the system has to register with external sources, the inputs, and to have any effect on the outside world it must perform actions based on the outputs of the event graph. The *event graph* together with the *inputs* and *outputs* forms an *event network*. To build an *event network* the inputs, outputs and event graph are described in a definition which can be compiled into a running event network. The compiled event network can be started and stopped at any moment, stopping it disconnects it from all registered event sources and stops performing all output actions.

It is possible to create complex programs with time-dependent logic by composing event streams and signals using a small set of combinators:

- transforming event streams into signals and vice-versa:

```
sig1 = es1.hold(0.0)
es2 = sig2.changes
```

- Merging multiple event streams into one event stream by accepting events from any of the original streams. '|' should be read as 'or':

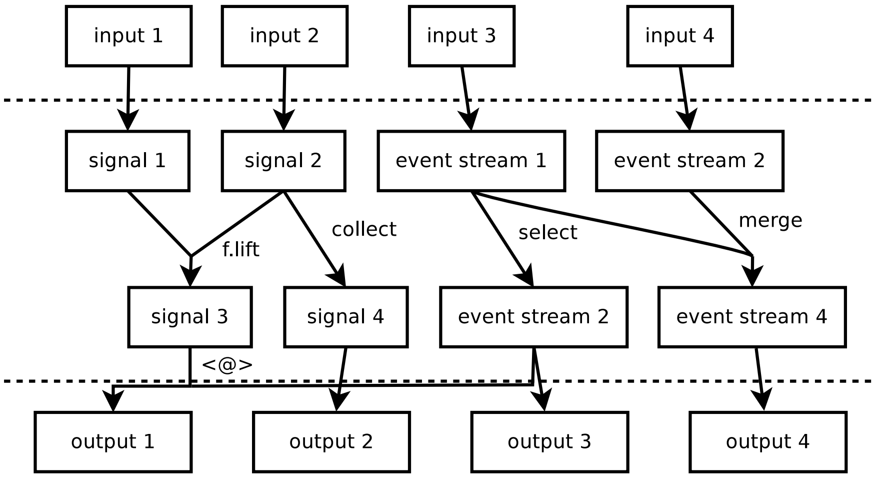


Figure 4.9.: FRP event graph.

```
mergedEs = es1 | es2 | es3 | es4
```

- Modifying events of an event stream or signal using a function:

```
es.collect( _ * 10.0 )
```

- Filtering events of an event stream using a function that returns a boolean:

```
es.filter( _ > 0.5 )
```

- Keeping state that can be influenced by several different event streams by associating each event stream with a state altering function:

```
(fes1 | fes2 | ...).injectF(initialState)
```

The state is first set to an initial value, then each time one of the event streams receives an event it generates a new function to be applied to the current state, generating a new state. It is important to keep in mind that signals and event streams can carry any type of objects, including functions, not just numeric values. For example, a counter that can be incremented by one event stream and decremented by the another:

```
(es1.collect{{|x|x+1}} | es2.collect{{|x|x-1}}).injectF(0)
```

- Merging n signals using an n-ary function:

```
f.lift.(sig1,sig2,...)
```

This is possible to do with signals but not with event streams, since only signals remember their last value. The lift method transforms any function accepting normal types into a function which accepts signals carrying those types, and returns a signal. As an example,

```
atan2(_,_).lift.(xSig, ySig)
```

converts two signals carrying x and y coordinates to one signal carrying the corresponding angle.

- Applying a time-varying function (stored in an signal) to an event stream:

```
fSig <@> es //fSig: FPSignal[Function]
f <%> sig <@> es //f: Function
```

This allows event streams to use previous values from other nodes in the graph. For example, starting a synth on pressing a button with frequency determined by the current value of a slider (which corresponds to the last value received from the slider):

```
f = { |freq,e| IO{
  Synth(\def1,[\freq,freq.linexp(0.0,1.0,100.0,2000.0)])
}};
f <%> sliderSig <@> buttonEs
```

- Dynamic event switching: changing the event graph by running a function on an incoming event that returns the new event stream or signal to use (see figure 4.10):

```
sig.switchTo({ create new FRP graph here })
```

All FRP objects created inside the function are automatically disposed of when the function is run again, preventing memory leaks.

In *FPLib*, time is represented by an event stream of floats carrying the elapsed time in seconds and updated periodically by a timer. Mathematical functions of time can be easily calculated by applying the function to this time signal:

```
{ |t| sin( 2pi*t ) }.lift.(timeSignalFromTimer)
```

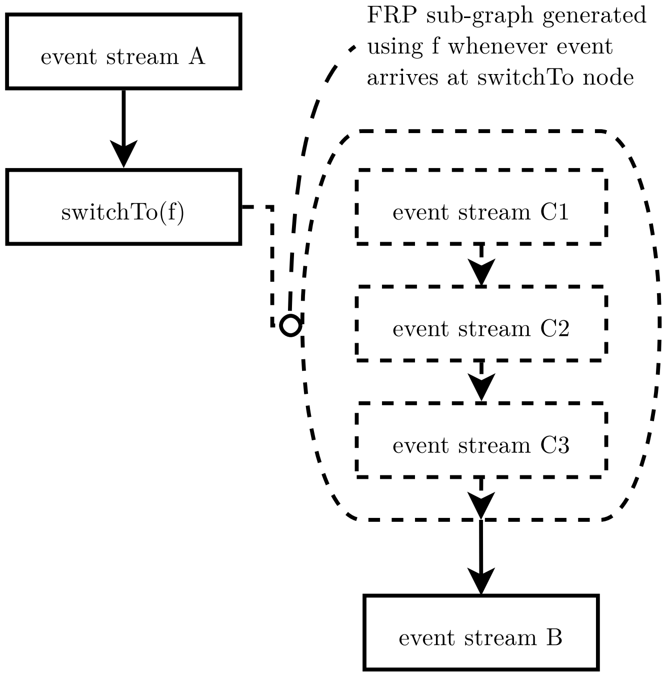


Figure 4.10.: Diagram of `switchTo` combinator.

Method of `FPSignal`	functionality
`timeSig.lfsine(freq, phase)`	Sinusoidal oscillator.
`timeSig.line(a, b, dur)`	Breakpoint envelope which goes from a to b in dur seconds.
`sig.integral(timeSig)`	Performs integration.
`timeSig.changeRate(sliderSig)`	Slowdown or speed-up time.
`sig.switchLater(timeSig, laterSig, switchTime)`	Behaves like `sig` until `switchTime` from then on behaves like `laterSig`.

Table 4.2.: Time related methods of `FPSignal`.

There are a number of methods of `FPSignal` that operate on time signals (see table 4.2). A number of oscillators were implemented, as well as integration (the reverse operation of differentiation). The `integral` method integrates an arbitrary signal carrying floats over a period of time, which is useful in *ImmLib* for controlling the rate of passage of "time" in real-time, by integrating a signal carrying the time rate. This allows smoothly changing the "speed" at which parameter fields are evaluated. When used on periodic functions it is analogous to changing the frequency of the oscillation but without the discontinuity issues described in section §3.4.

Inputs to the event graph can come from MIDI or HID (via the *Modality Toolkit*), OSC, GUIs or timers. An output of the event graph is a function that is called when a value reaches the end of the FRP graph and which performs side-effects. Functions inside an FRP graph must be side-effect free, while functions at the terminal vertices of an FRP graph can, and should, perform side-effects. See appendix D.1 for a simple example of an FRP event graph.

Discussion

FRP event graphs have similarities to the ugen graphs of MUSIC-N languages: in both cases the low-level plumbing needed to connect entities through which values are passing is abstracted away using a domain specific language. The main difference between ugen graphs and event graphs is that in the first case information flows at a constant rate, the whole graph is computed at every tick of a clock, while in FRP this need not be the case, events can come at any time. Also, an event can cause only a subset of the graph to be re-calculated, allowing for instance to disable part of the graph with an accompanying decrease in CPU usage. Also, dynamic event switching is possible in *FPLib* because event graphs are defined and evaluated using the same language, *sclang*. In *SuperCollider*, ugen graphs are defined in *sclan*g but evaluated in *scsynth* which has no knowledge of the *SuperCollider* language, this is one of the reason why in *SuperCollider* the ugen graph of a `Synth` cannot be changed after it is instantiated.

The functional approach to event processing just discussed has certain advantages. Since all the functions used to construct the graph are pure[10] it is possible to abstract a subset of them into a single function and be confident that the result will be identical[11]. Also, the event graph does not depend on the nature of the inputs or outputs, for instance MIDI inputs can be changed to GUIs or OSC messages without changing any of the event graph code. The ability to abstract parts of the event graph into self-contained functions makes FRP a highly modular approach and allows building a personal library of functions for event processing that can be re-used for different projects.

[10] *SuperCollider* is not a typed pure FP language, therefore there are no guarantees that the functions passed by the user to the to the FRP combinators are actually pure, it is up to the user to make sure this is the case.

[11] This property is called *referential transparency*.

Integrating FRP networks into *UnitLib*

An FRP graph can be used to control arguments of a *UnitLib* unit by placing an action at a terminal node of the graph: whenever a value reaches this node the action sets a specific argument of the unit, such as `freq`, to that value. This suggests that an FRP graph can be embedded inside a unit, facilitating the interaction between different inputs, such as MIDI, and the arguments of the unit.

For use with units, FRP graphs are packaged inside a function, similarly to `SynthDef`s, and assigned a global name:

```
ImmDef(\frpGraph1, { |timeSignal| ... FRP graph here ... A })
```

A terminal node of the FRP chain is tagged with the name of the argument to which it should be routed, therefore the assignment of FRP signals to unit arguments is declarative, making the whole FRP graph definition declarative:

```
A = (freq:USpecArg(signalForFreq), amp:USpecArg(signalForAmp))
```

The FRP graph is compiled and started during the initialization of the unit, therefore it is already active before sound is started. When the unit is started, creating the synths, the timer is also activated (with t = 0), and when the unit is stopped the timer is stopped.

Figure figure 4.11 shows a simple example where the frequency and amplitude of a sine oscillator unit are connected to the outputs of an FRP graph. The frequency is controlled by OSC messages of type `/freq,f`, possibly received from a tablet or smart phone, and the amplitude is set to the multiplication of the value of a slider of a MIDI controller with a sine oscillator animated using the built-in timer and remapped to the range $[lo, hi]$ which is controlled by two sliders automatically created in the GUI of the chain. The corresponding code is available in appendix D.2.

4.6. Parameter fields

Pure parameter fields are created using the `PField` class. The `PField` class constructor has only one argument, a function, which should follow the pattern

```
{ |u,v,t,c1,c2,...| ... }
```

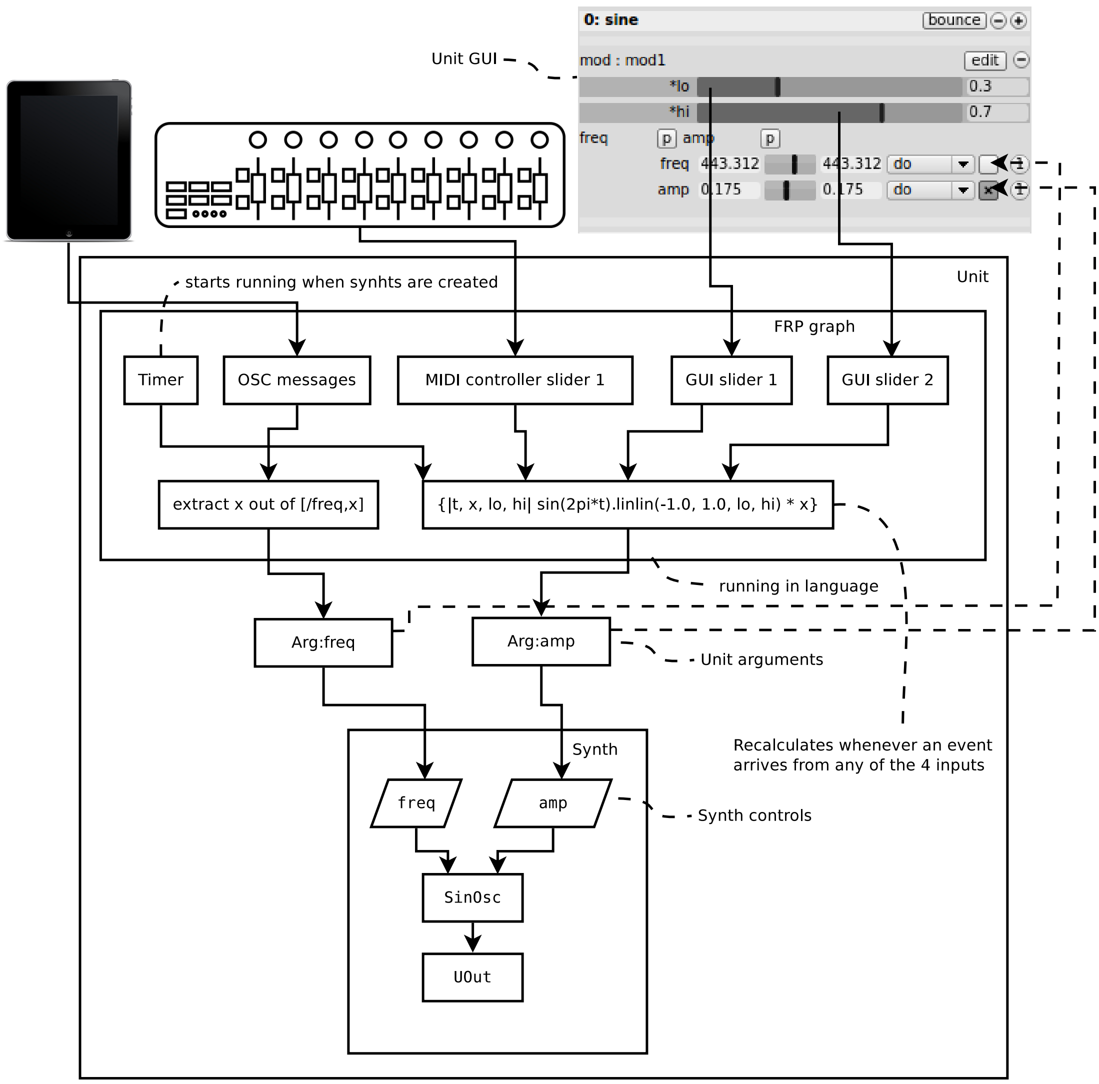


Figure 4.11.: Integration of FRP graph inside a unit. Example featuring events from network device, MIDI device and GUI.

and is stored for later evaluation or manipulation. This *SuperCollider* function corresponds to a parameter field as introduced in the model. It receives $n+3$ floats, where n is the number of parameters of the parameter field and should return a single float:

```
pf1 = PField({|u,v,t,c| sin(2pi * (t - (c*(u - v)) ) ) })
```

`PFields` can be created using any of the available numerical functions in *SuperCollider,* such as trigonometric functions or algebraic operators, and can make use of conditional statements such as the if - then statement.

A `PField` is evaluated inside an FRP graph in the presence of a `PSurface` of size n, being passed a mandatory time signal together with optional signals for additional parameters, and returns a signal carrying an array of floats of size n, the result of the evaluation of the function stored on the `PField` at each of the points of the `PSurface`. The parameter field defined above would be evaluated with:

```
sigResult = pf1.value(timeSig, cSig)
```

A number of predefined parameter fields are available as class methods of `PField`. Some of these, the *pure* parameter fields, return a `PField`, the others, the pre-evaluted, return an `FPSignal` carrying an array of floats of size n. This second group consists either of parameter fields which make use of dynamic event switching to alter the mathematical function being used on receiving specific events, or parameter fields that are pre-evaluated in order to implement smooth changes of an oscillator frequency.

`Pfields` can be transformed or composed before or after evaluation, with different results. Transformations happening before evaluation operate on a continuous-space representation of the function while those happening after evaluation operate after spatial discretization. Using the continuous-space representation, a parameter field with at least one parameter can be composed with another parameter field. Also, any parameter field can be rotated using the procedure outlined in section §3.4 and implemented in the `PField.rotate2D` and `PField.rotate3D` methods for the plane and sphere respectively. The rotation can be animated using the `FPSignal` oscillators or other time-based signals. Examples of composition and rotation can be seen in appendix D.5.

Post-evalution transformations are applied to the `FPSignal` returned from the evaluation of the parameter field which can be manipulated using any of the FRP combinators reviewed before. Mathematical operators (`* / + -`) can also be used directly on signals through polymorphism, but this is expensive as it creates a function evaluation step per operation and should be avoided except when only a single mathematical operator is to be applied (e.g. `mySig * 0.5`). The `switchInto` combinator allows changing the signal currently associated with a unit argument dynamically, when some other signal produces an event.

The `collect` or `lift` combinators used with the time signal allow entirely bypassing the parameter field mechanism, creating the final values directly. This was used to prototype and experiment with continuous automata and the BZ reaction system introduced in section §3.5 (see appendix D.7).

The evaluation of parameter fields takes place inside the FRP graph function of `ImmDef`. Besides storing and compiling the FRP graph, the `ImmDef` can also declare a set of named parameters, with corresponding numeric range:

```
ImmDef(\frpGraph2, f, 0.1,
  [\param1, [lo1,hi1], \param2, [lo2,hi2],...])
```

These parameters can be either set to specific values directly,

```
ImmMod(\frpGraph2, [\param1, value1, \param2, value2,...])
```

or changed via sliders in a GUI (see figure 4.12). When an `ImmDef` declares sliders, these are available as signals in the FRP graph, which can then be used for controlling parameters of parameter fields. It is possible to change the current `ImmDef` associated with a unit, even while the sound is playing, by dragging from a list view to the GUI of the unit[12].

ImmLib makes available a set of predefined `ImmDef`s, one per pre-defined `PField`, where all the parameters of the parameter field have already been assigned `ImmDef` parameters with the corresponding numeric range. This allows the user to quickly try any of the predefined parameter fields, being able to move the sliders in the GUI in order to try out different values. These predefined `ImmDef`s can also serve as starting points for further modifications of the FRP graph, since they can be copied to a different file and renamed.

[12] An `ImmDef` has the specific arguments of the unit it will act upon hard-coded, therefore when dragging a new `ImmDef`, it must also modulate parameters that are present in the unit. A workaround for this issue will be described later.

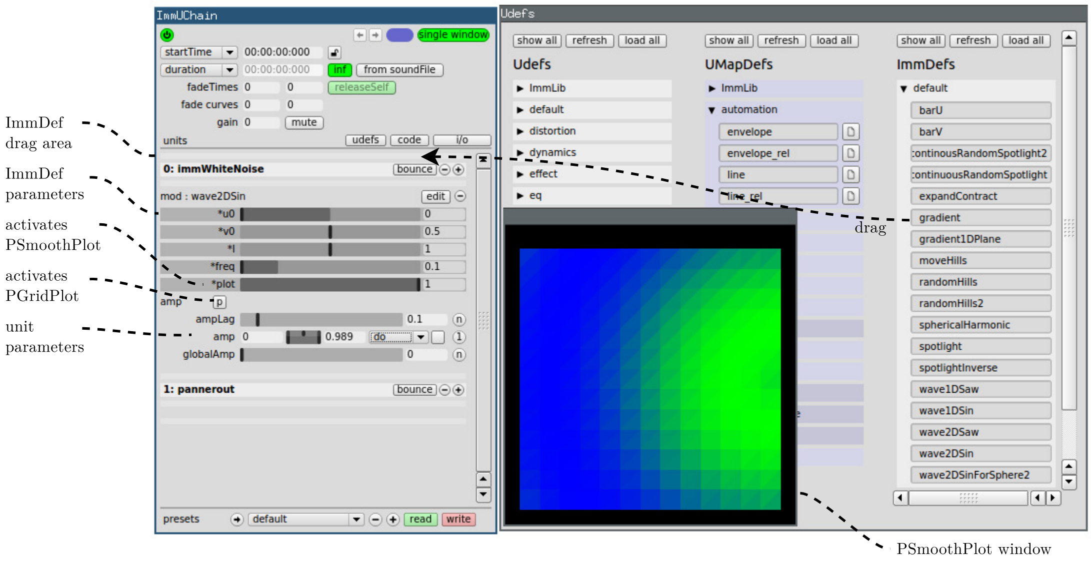
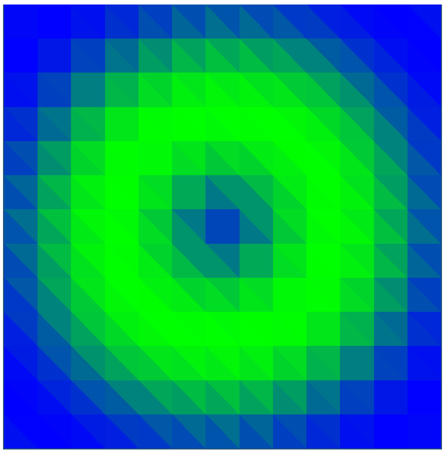


Figure 4.12.: `ImmUChain` GUI

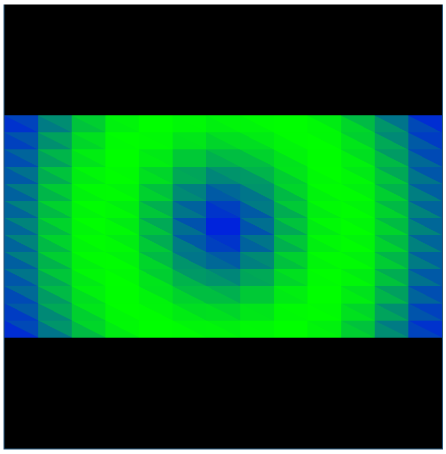
Currently, due to technical reasons a `PField` is not entirely independent from the surface being used to evaluate it, nevertheless `ImmDefs` can be constructed in order to work around this issue (see appendix C.4). Practically all the predefined `ImmDefs` are surface independent, the exceptions being gradient1D and wave1D which use an algorithm that is only defined for the plane surface, and expandContract which is only defined for the sphere. Note that the actual behaviour of the `ImmDef` will be different in different surfaces since the distance function or the domain of the coordinate chart can be different. In particular, changing the plane surface from a rectangle to a square will not "stretch" the surface pattern, since the distance is calculated on the actual surface and not on the parametric space (see figure 4.13).

Summarizing, parameter fields:

- Are evaluated inside an FRP graph whose outputs are connected to the arguments of a unit.
- Can be defined with mathematical functions using `PFields`.
- Can also be defined as FRP sub-graphs, for instance when using time-rate modulation or dynamic event switching.
- Can be animated with low-frequency oscillators and breakpoint envelopes.
- Are packaged inside `ImmDef` definitions.



(a)



(b)

Figure 4.13.: `wave2DSin` visualized with (a) a `PPlane` with 2m × 2m and (b) a `PPlane` with 2m × 1m.

4.7. Parameter field visualization

A set of tools are available in *ImmLib* for the visualization of the sometimes quite complex surface patterns created by parameter fields. Even if the movements displayed in the visualization do not necessarily match what is perceived aurally, it is nevertheless important to understand how a parameter field behave in "abstract", before interacting with the synthesis processes.

There are three plotting classes available: `PSmoothPlot` and `PGridPlot` implemented as an external process (*pfVisualizer*) and `HemiPlot` implemented in *SuperCollider* using the built-in 2D drawing primitives (see figure 4.14). *pfVisualizer* is a 3D graphics program, written in *Haskell* using the *OpenGL* API, which renders basic 3D primitives (cubes and triangles) using positions and colours received from OSC messages. The primitives are rendered inside a wire-frame cube for orientation, and the 3D scene can be freely rotated with the mouse, or set to specific perspectives using keyboard shortcuts. The scene is animated from *SuperCollider* by sending the numerical values resulting from the evaluation of the parameter field in an OSC message to the external process at every tick of the animation timer.

`PGridPlot` plots a parameter field by displaying cubes at the positions of the points in the surface sample set and interpolating the colour of the squares between two fixed colours using the values of the parameter field. This plot type works with both pure and pre-evaluated parameter fields.

`PSmoothPlot` renders the *ImmLib* surface in 3D using triangles. It only works with pure parameter fields, as it needs to perform two different spatial discretizations, one for the sound engine and another for the visualization, therefore it needs access to the original function. Currently it renders the sphere using 320 triangles generated from the 2nd geodesation of the unit sphere (see section §3.2) and the plane using 338 triangles. The values sent to *pfVisualizer* are generated in *SuperCollider* using the points at the centre of the triangles which requires evaluating the parameter field twice for each timer tick, once using the points of the surface sample set, and another time using the centre of the triangles. Each *pfVisualizer* rendering the 320/338 triangles at 10Hz takes 17% of one core of a 2.0Ghz quad core CPU[13], while *sclang* uses 7% of the CPU. Static parameter fields such as `spotlight` only update the renderer when a parameter of the parameter field is changed. In this case, when idle, *pfVisualizer* uses 2% of the CPU. On this CPU up to a maximum of 8 instances of `PSmoothPlot` can be used simultaneously.

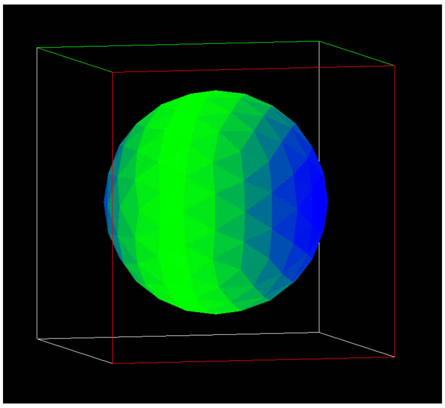
`HemiPlot` plots a parameter field which is being evaluated using a `PSphere` by projecting the points of the sample set in the north and south hemispheres into two separate circles. For those parameter fields where `PSmoothPlot` cannot be used, this plot type sometimes makes the movements clearer than `PGridPlot`.

When using the predefined `ImmDefs` it is possible to start a `PSmoothPlot` for the parameter field on demand, after the chain is already playing. Predefined `ImmDefs` have a *plot* parameter which can take values 0 or 1, acting as a switch. Setting this parameter to 1 opens a `PSmoothPlot` for the parameter field, setting it to 0 terminates the pfVisualizer process.

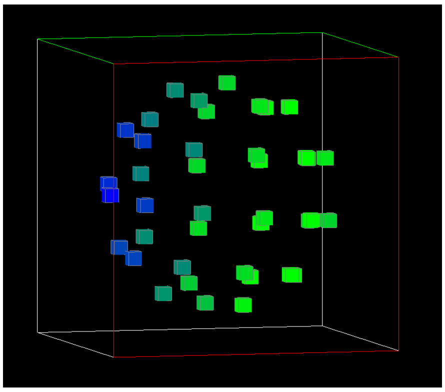
4.8. Workflow

The first step when using *ImmLib* is to choose one of the spatialization modes, VBAP, direct or binaural stereo. When one wants to use more sources than loudspeakers, or when one wants to place sources in positions distinct form the loudspeaker positions, the system should be started in VBAP mode. When one wants to create sources restricted to the positions of the loudspeakers, the system should be started in direct mode. For preview over headphones, the stereo binaural mode should be used. To change from

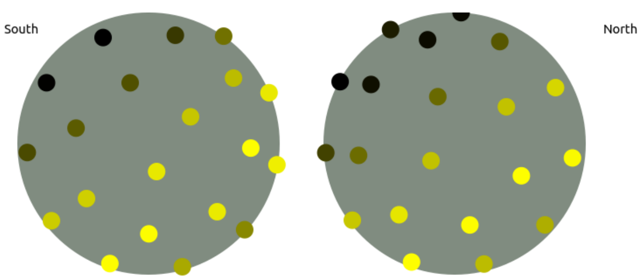
[13]Intel i7-2635QM, clock fixed at 2.0Ghz.



PSmoothPlot (a) `PSmoothPlot`



PGridPlot (b) `PGridPlot`



PHemiPlot (c) `PHemiPlot`

Figure 4.14.: The different types of visualizations of parameter fields available in *ImmLib*.

VBAP mode to direct mode, and vice-versa, the score can be left unchanged, only the surface definitions need to be changed. To preview in stereo a score created for VBAP or direct mode, no changes are needed to either the surface definitions or the score, it suffices to startup the system in stereo mode.

The second step is to create the definitions for sound processes, surfaces, and parameter fields using `Udef`, `PSurfaceDef`, and `ImmDef`. These definitions are then addressed by name in the scores. These should be saved in a text file and loaded before opening any score previously worked on. Changing these definitions will change the sonic content of a score, so some care must be taken to keep the version of the definitions and the score in sync. On the other hand it is possible to play the same score using different definitions without having to change the score itself which can be useful for quickly evaluating differences in the number of points used, in the synthesis definitions or in the parameter fields. The different definitions can be evaluated once and subsequently used in multiple events or scores.

After having created and loaded all the definitions for sound processes, surfaces and parameter fields, it is possible to use *ImmLib* entirely from a GUI. The events can be dragged in the timeline, fade in, fade outs and durations changed. The values of the `Udef` and `ImmDef` args can be changed by using the widgets in the event GUI. `Udef`s and `ImmDef`s can be changed by dragging from a list view directly to the event GUI, in the case of `ImmDef`s this will happen instantaneously, without having to stop the `ImmUChain`.

For performances, the final score can be played in real-time, or a multi-channel sound file generated for a specific loudspeaker system. When performing a given score with a different loudspeaker setup from where it was originally created, the surface definitions must be changed and the piece remixed, as most probably the balance will not be preserved from one system to another. Due to the high number of synths that are created when using *ImmLib,* it is normal to have scores which have more events than what the system can handle in real-time. In such cases, parts of the score can be rendered to a multichannel sound file and re-imported into the score as an audio file. This process is time consuming since the recording must be done in real-time, but it does present a way to realize scores which go over the limits of the system being used.

Although *ImmLib* is geared towards fixed-medium pieces, it is also possible to use the system as a digital instrument. All parameters of sound processes and parameter fields can be controlled in real-time from a wide array of devices and protocols. FRP graphs allow complex event logic, typical of a musical instrument[14], and scores and chains can be started and stopped at will, therefore a score with events with infinite duration can effectively act as an instrument.

4.9. Technical Issues

We will now review some of technical issues that arose while developing and using *ImmLib*.

Every event in a score creates n synths when paired with a surface with n points; when m events are playing, with l units each, $n \times m \times l$ synths will play simultaneously plus n panners synths. The number of concurrent synths playing can rapidly explode when working on a score, the library is therefore CPU intensive and takes *SuperCollider* to the limit.

Some tests were done on the author's laptop computer to determine the maximum number of events the system can play under different situations (see table 4.3). The laptop uses a 2011 Sandy Bridge 2.0Ghz quad core Intel CPU (i7-2635QM) and all tests were run on *Linux* with the CPU clock manually fixed at 2.0Ghz (CPU frequency scaling disabled). When testing in VBAP mode a 32 loudspeaker configuration was used, and in all tests a `Udef` containing one white noise ugen was used. On the first four tests direct mode achieves a higher number of events or points on the surface. This is because the VBAP panners use more CPU than the direct mode "panners", which just forward the audio to the soundcard outputs, applying a user controllable 4 band EQ (also applied in VBAP). In tests 4 and 5 the bottleneck is the calculation of parameter fields in *sclang*, therefore direct and VBAP modes have the same number of maximum sources. Given a surface with 20 points, in direct mode *scsynth* can render 65 events, while *sclang* can only render 36, for VBAP it is 60 vs. 36. For a surface with 36 points it is 31 (direct) and 28 (VBAP) vs. 20. Summarizing, when using a parameter field, *ImmLib* can render 36 events using a surface with 20 points, or 20 events using a surface with 36 points. It should be noted that the `Udef` used for testing

[14]FRP was originally implemented in *FPLib* for the Modality project [117], which is entirely dedicated to building complex instruments for live performance in *SuperCollider*.

Test	mode	(1) points (2-5) `UChains`	synthesis synths	panner synths	total synths
1	direct	650	650	1300	1950
	VBAP	130	130	260	390
2	direct	65	1300	1320	2620
	VBAP	60	1200	1220	2420
3	direct	31	1116	1152	2268
	VBAP	28	1008	1044	2052
4	direct	36	720	740	1460
	VBAP	36	720	740	1460
5	direct	20	720,	756	1476
	VBAP	20	720	756	1476

Test description:

1. One event, no `ImmMod`, maximum number of points in surface.
2. Surface with 20 points, no `ImmMods`, maximum number of `UChains`.
3. Surface with 36 points, no `ImmMods`, maximum number of `UChains`.
4. Surface with 20 points, 1 `ImmMod` per uchain, maximum number of `UChains`.
5. Surface with 36 points, 1 `ImmMod` per uchain, maximum number of `UChains`.

Table 4.3.: CPU usage tests.

is extremely simple, in most cases the `Udefs` will be far more complex, and *scsynth* can become a bottleneck before *sclang* does.

Besides the issues with the CPU bottleneck, it was observed that *ImmLib* can also hit a network bottleneck. Once the number of simultaneous synths becomes high enough, in certain configurations the operating system can start to drop the UDP OSC messages, in this case the sound servers should be run over TCP instead. In some cases the number of simultaneous messages which must be sent can even be higher than the maximum number of messages which *scynth* can process in a given time period. Since this number is hardcoded, overcoming this issue requires changing the constant in the source code to a sufficiently high value and recompiling the program (see appendix C.3). If OSC messages from *scsynth* to *sclang* are lost, this will quickly place *ImmLib* in an unusable state as the synth end notification messages are needed to manage the CPU-load balancing.

At the start of the section it was reported that given a high enough number of chains, or surface sample set size, *sclang* can become CPU bound, therefore becoming a bottleneck for the whole *ImmLib* system. *sclang* becomes CPU bound when it cannot calculate the values of the parameter fields for all points fast enough. *sclang*, the interpreter of the *SuperCollider* language, is extremely slow when compared to a compiled language such as *C*, *Fortran* or *Haskell*, or even to other interpreted languages such as *JavaScript*. Also, it is a single threaded program, and there is no easy way to setup communication between multiple *sclang* processes. The number and complexity of FRP graphs that can be evaluated concurrently is therefore limited. *scsynth* is several orders of magnitude more efficient than *sclang*, if the parameter fields had been implemented on the server instead of the language, parameter field evaluation would not be a bottleneck for a score since the parameter fields would be run at control rate and use only LFOs and simple mathematical operators. As an experiment, some of the simpler parameter fields were implemented as `UMaps` and it was observed that the same spatial patterns would be produced, although as was expected the movements were smoother than in the *sclang* implementation (The code is available in appendix D.6). The corresponding synthdefs were still quite similar to the mathematical formulas, although not as close as the original `PFields`.

Currently in order to use all the processing power of a computer, *ImmLib* should be used with as many *scsynth* processes[15] as CPU cores in the system, since each *scsynth* process can only use one core. There is a re-implementation of *scsynth*, named *supernova [125],* which is multi-core aware, but it currently does not perform well on OSX. Since the workstation in the Sonic Lab system uses this operating system, and for general cross-platform compatibility, it was decided that *scsynth* would be used with *ImmLib*. *UnitLib* has a CPU load-balancing algorithm which works at the level of `UChains`, distributing the chains amongst the different servers. In order for *ImmLib* to make use of this load-balancing algorithm in a way that was transparent and automatic for the user, a complex bus and group automatic management system had to be implemented. More details on this system can be found in appendix C.2.

In order to partially overcome the limitations just described, a feature could be added to *ImmLib*, allowing an event to be "frozen", that is, recorded to an n-channel sound file, where n is the size of surface sample set, and reloaded onto the *ImmLib* score as a sound-file playback event. This would allow unlimited events while still enabling

[15] Here process is used in the sense of operating system process.

re-tweaking an event by "unfreezing" it, tweaking while auditioning in real-time and then "refreezing".

`ImmDef`s once defined have the argument that they should modulate in the unit hard-coded. This implementation decision led to undesired consequences, such as requiring that the `ImmDef` code be changed in order to alter the argument of the unit to which it is assigned. This issues could be addressed in the future, by changing the implementation, in the meantime, a temporary alternative was created which allows an `ImmDef` to be assigned to units with different argument names using the `UMap` system. In this approach, first a `pfield UMap` is assigned to the argument of the unit, afterwards any `ImmDef` can be assigned to the argument, with the `UMap` taking care of automatically remapping the values from the range `[0,1]` to the correct range for that argument. Using this system `ImmDef`s can be dragged from a GUI list view to any argument of any unit.

Summarizing, the most significant technical issue with *ImmLib* is the high CPU and network usage which limits the number of events that can be played simultaneously. Creating multiple decorrelated audio signals requires the sound processes to be kept separate, having a multiplier effect on the amount of computing resources used.

Chapter 5.

Reflective listening sessions and use in electroacoustic composition

During the initial stage of the research, different parameter fields were conceived, implemented in software and assessed. Alongside the software development, examples were created, to understand how each parameter field interacted with different sound processes, and these were evaluated in reflective listening sessions where extensive notes were taken.

An important part of the practical work carried out during this research consisted of collaborations with three sound artists, who were approached and invited to use *ImmLib* to produce a work of their own.

The direct practical outcome of this research, which complements this thesis, can be divided between two contributions, the software, including all the documentation and examples, which is available in the annexed media (see appendix F) and the sound examples, sketches and complete sound works produced with the software.

This chapter details a perceptual description of the use parameter fields with different sound processes. It describes the collaboration process and the approach of the artists to composing with the library. A technical and perceptual description of the works created using *ImmLib* is presented. Finally, general observations about the use of parameter fields for specific goals are outlined.

5.1. Reflective listening sessions

During the listening sessions, the parameter fields were experimented with mostly in the Sonic Lab[1] using the sphere *ImmLib* surface (see section §3.3.6). Some experimentation was also done with the vertical rectangular loudspeaker grid using the plane *ImmLib* surface. Different sound process definitions and distinct parameter values of parameter fields were tested. The observations presented below are based on notes taken during these testing sessions, and reflect my personal subjective perception and opinion.

In these tests I would play a selected *ImmLib* patch, without anyone else present in the room, and then proceed to listen attentively, often with eyes closed and without any visualization tool opened. After several minutes of careful listening I would stop the sound and take notes of what I had perceived. The patch would then be restarted with slightly different values and the same process repeated. Usually the listening position was at the centre of the Sonic Lab, although for specific patches other listening positions were assessed. During each test I had full knowledge of which code was running, there was no randomization in this respect.

In what follows next, perceptual descriptions are given for the assignment of different parameter fields to the amplitude parameter of a simple `Udef`, consisting of a single white noise generator with an amplitude control which is applied a `Lag` ugen[2]. This `Udef` was chosen because multiple decorrelated white noise signals produce a highly enveloping auditory event featuring a single auditory stream, producing very sharply-defined spatial surface patterns. It was therefore considered ideal for an initial presentation of the perceptual properties of each parameter field. Afterwards, the perceptual results using other `Udef`s, featuring different types of synthesis are presented. Those were primarily assigned the wave2DSin parameter field, nevertheless in cases where other parameter fields give quite different results, this will be mentioned. The `Udef`s used are described in table 5.1, and the ugen graph diagrams and corresponding code is also available in appendix D.8. For the definitions of the parameter fields and corresponding parameters, the reader should refer to section §3.6. The annexed media contains videos with binaural sound of most of the parameter fields modulating the

[1] The 32 loudspeakers of the Sonic Lab were controlled from an 8-core workstation running a UNIX-based operating system with a sound card supporting no less than 32 channels.

[2] This ugen "slows" down a signal, not allowing changes in an incoming signal faster than a given threshold. It is used to smooth the staircase type signal generated by the parameter field.

white noise `Udef`, as well as sound examples in binaural stereo using some of the `Udefs` just mentioned, although again it must be stressed that the stereo binaural examples are not representative of the output with loudspeakers.

The observations in the next subsections refer to listening tests completed using the sphere *ImmLib* surface in the Sonic Lab system. Later a description of the differences between this system and the vertical rectangular grid will be given for selected parameter fields and `Udefs`. When mentioning directions the usual coordinate system of $\mathbb{R}^3$ should be assumed, where positive values along the x-axis are to the right, positive values along the y-axis are towards the front and positive values along the z-axis are overhead. Alternatively spherical coordinates (θ, ϕ) will be used where θ is azimuth and ϕ is elevation, $(0, 0)$ is front, $(\frac{\pi}{2}, 0)$ is right and $(0, \frac{\pi}{2})$ is up. The median plane is the plane defined by the y and z axes, the frontal plane is defined by the z and x axes and the horizontal plane is defined by the x and y axes.

5.1.1. Perceptual description of parameter fields

gradient

This parameter field can be used to change the focus in space from one direction to another by moving the reference point (u_0, v_0). If the value of `curve` is high (> 5, see figure 3.26) it can act as a point-source like panner, placing the sound only at a specific location. In this configuration it is similar to the spotlight parameter field. If curve is close to 0, such that the gradient increases linearly, and the difference between the extreme values of the gradient, a and b, is small, then it will emphasize slightly one hemisphere relative to the opposite hemisphere. It can also be made to evolve in time from one type of behaviour to the other by modulating a, b and `curve` with an envelope. It can be used to perform a spatial cross-fade of a parameter from x to y by setting the initial state $a = x$ and $b = x$, and using an envelope to change b from x to y in n seconds followed by a change of a from x to y in m seconds.

spotlight

This parameter field restricts the sound to a specific zone of the surface. If c is small and the reference point is moved, then it acts as a panner with a trajectory. It can also be used to create a spatial fade-in where c is assigned a $0 \to 1$ envelope. The same

`Udef` name	Parameters	Description
`whiteNoise`	`amp`	White noise generator. Since the `amp` control is meant to be modulated by a parameter field, a `globalAmp` control is added. The `Out` ugen sends the signal at its input to the output of the soundcard given by `bus`.
`periodic Impulses`	`freq`	A single periodic impulse generator, with `freq` its frequency, and controllable impulse width.
`sineDist`	`overdrive`	The output of a sinusoidal wave generator is amplified, with amplification controlled by the `overdrive` parameter and then sent through a distortion ugen. The frequency of the sinusoidal wave generator is randomly picked for each point from a selectable range at event start-time.
`noiseSaw Sweep`	`mix`	The mix between a brown noise and a saw wave generator is sent through a group of sharp band pass filters which have frequencies modulated with low-frequency sine oscillators with random periods. The `mix` parameter controls the cross-fade between the two generators. The sharp band-pass filters slowly "scan" the spectrum of the generators.
`file Playback`	`amp`, `startTime`	Playback of a sound file loaded onto a buffer in RAM with controllable start-time and playback speed.
`granulator`	`trate`, `pos`	A granulation ugen with grain creation triggered by a random impulse generator, with `trate` the impulse density and `pos` the position of the grain in the sound file. Grain size is also controlled by `trate`.

Table 5.1.: `Udefs` featuring different synthesis types.

usage of an envelope on c, with the parameter field applied to a parameter influencing timbre, creates a timbral spatial cross-fade, a gesture where a new timbral quality is introduced in one region of the space, subsequently spreading to all directions. The gradient parameter field can in many cases achieve the same cross-fade effect more smoothly. This parameter field can also be used to play different sounds at different zones of the surface, by assigning a different reference point to each sound and using a small c.

It can also be used as a "mask", restricting another parameter field to a specific zone of the surface. This is done simply by multiplying it with the other parameter field. In the region where the spotlight is zero the parameter will have a constant value, while in the region where the spotlight is one the parameter will be modulated by the second parameter field.

barU and barV

With barU a curtain of sound appears that progressively wraps around the listener starting from the front and going anti-clockwise as w goes from 0 to 1. The movement is quite clear and the aural perception matches exactly what is shown on the visualization. Sinusoidal modulation of the wideness creates a smooth movement which is clearly perceptible at up to 2Hz.

With barV and w close to 0 one perceives the sound as coming from a horizontal ring, which becomes progressively more vague as w increases, until the sound is localized in all directions. Slowly modulating w sinusoidally creates a constant perception of a ring, whose height is vague. If the parameter field is rotated such that the ring is vertical, with w close to 0 the sound is perceived as coming from two distinct points, one overhead and one below, as w increases the sound quickly fills the whole sphere. This is a good example of how the auditory system reacts differently to two *ImmLib* setups which are mathematically symmetrical, as the auditory perception in the median plane does not work in the same way as in the horizontal plane.

expandContract

With c sinusoidally modulated, there is a periodic movement along a line: the sound appears from one direction, fills the sphere, leaves from the antipodal direction, and then reappears from that direction again and the cycle repeats. The clarity of the movement along a direction depends on the oscillator frequency, the lag time and the orientation. With $f = 0.15$Hz the direction of movement can easily be determined for any orientation, although it is slightly less clear for orientations in the median plane. When f reaches 1Hz, instead of a smooth movement, alternating fade-ins and fade-outs at antipodal points on the sphere are perceived.

sphericalHarmonic

Any values of m and l create a movement that is rhythmic and periodic. This is perceived not as a shape sliding between positions along the surface, but as a cycle where a shape fades-in in one position, then fades-out at the same position, then fading-in and out again in a different position, with the cycle then restarting again in the first position.

$l = 1$ The sound very smoothly oscillates between two hemispheres along the x-axis (left-right) with $m = -1$, along the z-axis (up-down) with $m = 0$, and along the y-axis (front-back) with $m = 1$. The auditory event does not entirely occupy each hemisphere, it concentrates around the intersection of the axis of movement with the sphere.

$l = 2$ There are two lobes active simultaneously around antipodal points. The transition is still quite smooth.

$l = 3$ There are more lobes active simultaneously, and the transitions are more abrupt.

If the amplitude is restricted so that it is always non-zero, the movement becomes more vague and the pattern less sharply-defined, with a periodic change of emphasis happening along certain directions. Using the rotation methods of `PField` it is possible to orient the lobes into different directions. Two different sound events with a spherical harmonic parameter field with lobes pointing in different directions can create a spatial alternation between the two sounds.

wave2DSin

A periodic movement is established travelling through the surface in alignment with an axis going through the centre of the sphere. This movement creates a rhythm which is related to l, the spatial wavelength of the wave. Setting $f = 0.2$Hz, such that a slow rhythm is produced, as it takes 5 seconds for the wave to complete one cycle, and $u_0 = \frac{\pi}{2}$, such that the wave enters from the left and exits from the right, different values of l can be evaluated[3]:

$l = 0$ There is no spatial movement, the sound is just amplitude modulated equally everywhere in the sphere.

$l = 0.05$ The sound fills the sphere quickly from the left, for some time it is present in all the surface, then exits from the right, therefore becoming silent, then the cycle repeats. One needs to be attentive to detect that the sound is starting from the left.

$l = 0.2$ At no point does the sound fill the entire surface, sound enters from one direction fills the whole sphere and immediately exists from the opposite direction, with the direction of movement quite clear. The same is true setting $v_0 = \frac{\pi}{2}$ or $v_0 = -\frac{\pi}{2}$ (up and down).

$l = 0.3$ The sound is present only in a smaller area, the surface pattern is more sharply-defined, taking the shape of a band or stripe. The loudness appears to increase as the band passes through the middle of the surface and sometimes it is possible to perceive two peaks of amplitude halfway through the movement. It is still the case that the surface returns to silence momentarily when a trough is centred on the sphere.

$l = 1.2$ When one peak of amplitude is arriving at the exit point another peak is appearing at the entry point without silence ever being established in all the surface during the movement.

Fixing $l = 0.4$, and different values of f can be assessed:

[3] A sinusoidal wave usually oscillates around 0 going from -1 to 1, on the other hand the output of `wave2DSin` is in $[0, 1]$, such that at a fixed point in surface the output oscillates around 0.5, and the wave as it travels through the surface is composed of a single peak going from zero to one and back to zero.

$f = 0.13$Hz The movement is slow and the extent and contour of the sounding area is easily determined on the surface, occupying roughly half of the sphere at any given moment.

$f = 0.5$Hz The direction of movement is very clear but the contour of the sounding area is not as clear as before. Changing (u_0, v_0) it is easy to determine the direction of movement. Front-back and back-front movements provoke different psychological reactions, back-front seems to be more intense, perhaps because of the feeling that the source of the movement is behind oneself, triggering a defence response.

$f = 1.8$Hz The perception of a continuous movement starts to break down, one instead perceives independent out-of-phase amplitude modulation happening at antipodal points.

$f = 3.1$Hz There is practically no sense of direction, one is aware of the amplitude modulation and the modulation has a certain spatial character to it. In this last case, changes of u_0, v_0 are almost imperceivable.

If two different sound events are played at the same time, both modulated with this parameter field, with the movement happening along different axes but with the same value for f, movement is perceivable but it becomes difficult to determine the two axes of movement. The detection of the direction of movement is easier if different values of f and different types of sound process definitions are used for each sound event.

Considering three different ugen graphs, a white-noise generator, amplitude-modulated white noise with a square impulse generator or an impulse generator, when the parameter field modulates the `amp` control, the shape and direction of the movement is more or less equally clear in all three cases. For the white noise the perception is clearer when the movement is sideways (relative to the head) while the modulated white noise appears to be clearer in the front-back direction. Modulating the `index` control of the `fm Udef` creates a quite more subtle and less clear movement although if the range of modulation[4] of `index` is high enough the effect is definitely perceivable. With attentive listening it is possible to track shape and location of the increase in energy in the high frequency region of the spectrum, as well as the direction of movement.

[4]The output of the parameter field is scaled to a given range.

randomHills

This parameter field is characterized by a constantly changing behaviour, due to the intrinsic randomization, happening periodically. Assessing different values of hill size:

size < 0.3 The hills are narrow enough that none of the points of the sample set fall within the base of the hill, most of the time there is silence in all directions with sporadic appearances of noise at specific points, creating narrow, almost point-like sources.

$0.35 <$ size < 0.45 A single large or multiple smaller clouds of noise periodically appear at different locations and then dissolve back into the surface. There is a repeating cross-fade from one set of clouds to the next. It is not possible to determine the shape, and contour of these clouds, although one has a sense of their size, or wideness. A lower number of hills creates a more abrupt appearance of each cloud, a higher number of hills creates a smoother movement and a tendency towards smaller, more numerous clouds.

$0.45 <$ size < 0.6 There is no longer the sense of a set of clouds being periodically replaced by the next, instead the perception is of one sound event which moves and distributes itself along the surface, sometimes bifurcating into multiple threads, somewhat like droplets of water sliding on a curved surface that unite and segregate. Using hill sizes up to 0.6, where the sound object does not occupy the entire surface constantly, is useful for playing several sound events simultaneously, as then they can be at times separated on the surface with clouds from one event not always intersecting with clouds from the other event.

$0.6 <$ size < 0.8 The sound comes constantly from all directions although there is an undulation of the loudness, continuously changing on the surface, with regions becoming quieter as other regions become louder.

moveHills

Using 3 hills different values of size can be evaluated:

size $= 0.27$ The base of the hills only rarely and briefly touches a point of the sample set, this creates an effect of noise briefly appearing from different static point sources. In this case auditory stream merging does not occur.

size $= 0.37$ The perception is of vague trajectories which are not particularly smooth.

size $= 0.5$ A number of auditory events are perceived, occupying a somewhat large area. Sometimes it seems they are sliding on the surface, and at other times, possibly when there are overlaps, that they are appearing and disappearing, without moving.

BZ reaction

The behaviour of this parameter field is highly dependent on the value of k. The initial state of the chemicals is randomized, giving a random initial spatial sound pattern. With most values of k the parameter sequence quickly becomes either 1 or 0 everywhere on the surface and stays perpetually in that state. Assessing different values of k:

$1.0 < k < 2.5$ The spatial pattern oscillates very quickly, with the movement progressing in a certain direction (different every time, due to the random initial state), until after a while the pattern again gets “stuck”, without any further movement.

$k = 4.0$ The movement is slower with only sub-regions of the surface getting “stuck”.

$k = 5.0$ The oscillations only happen a couple of times, the pattern then becomes uniform on the surface although sometimes the oscillations appear again after a couple of seconds of staticness.

$k > 6.0$ The pattern very quickly converges to 0 in all the surface.

The spatial sound patterns that emerge are hard to control and sustain, but quite dynamic and interesting. Although promising, this parameter sequence lacked controllability and it was not included with *ImmLib*. More work could be done in the future to find more parameters of control for this parameter sequence.

5.1.2. Perceptual description of different sound processes

variable-width periodic impulse generator (`periodicImpulses Udef`)

If `freq` is assigned a parameter field, then the clicks at each point will become desynchronized, and therefore the signals become decorrelated. With the wave2DSin parameter field assigned to `freq`, if the output is scaled to a very-low-frequency range[5], then merging of the audio signals does not occur, and clicks are perceived individually. There is a rhythmic increase and decrease of the density of clicks, whose direction can only be detected through attentive listening. With the output scaled to a higher frequency range[6], at the peaks of the surface wave the click density increases enough to cause a textural auditory event to appear, which is shaped as a ring, featuring higher loudness and high-frequency content,which moves over a static background of individual clicks. In this case the sharpness of the pattern is high and comparable to decorrelated white noise.

With the gradient parameter field it is possible to choose values[7] such that the impulses fuse into a single auditory event in one hemisphere, while remaining segregated in the opposing hemisphere.

Sinusoidal wave generator with distortion (`sineDist Udef`)

With `overdrive` = 0 the sine waves are not distorted and are summed into a single complex non-localizable tone. The distance of the auditory event is hard to gauge: at different moments during one period of the tone it seems to be far away or quite close.

If the wave2DSin parameter field is assigned to the `overdrive` parameter, regions of the surface where the output of the field goes above a certain threshold will distort the sine waves, causing additional harmonics to appear. With the frequencies of the sine waves covering a whole tone[8], the event is perceived as at least two separate auditory streams[9], a low-frequency complex tone with low locatedness and a group of high-frequency tones, forming one or two extended sources with high locatedness. The direction of movement of the zones of high frequencies can be detected through

[5] 0.12Hz $\leq$ `freq` $\leq$ 2.8Hz.
[6] 1.812Hz $\leq$ `freq` $\leq$ 29Hz.
[7] $u = 0,\ v = 0,\ a = 0.34\text{Hz},\ \text{and } b = 50\text{Hz}$.
[8] 150Hz $\leq$ `freq` $\leq$ 170Hz.
[9] If the spatial frequency of the surface wave is higher, more auditory events can be perceived forming separate rings.

attentive listening, being easier to detect when it is perpendicular to the median plane.

If the range of frequencies of the sine tones covers a full octave, then the added harmonics will no longer fuse into a single auditory stream, in this case, multiple tones with distinct fundamental frequencies are perceived.

Mix of noise generator with saw wave generator filtered by a bank of parallel sweeping band-pass filters (`noiseSawSweep` `Udef`)

Assigning the wave2DSin to the `mix` parameter two auditory streams are perceived, the first a constant drone (the saw wave), the other a ring of noise travelling along a very well defined direction. Although both auditory streams are perceived as having the exact same overall spectral contour (due to being filtered by the exact same set of band pass filters) they are clearly perceived as two different streams. Interestingly, it is the noise that is perceived as moving, and having a sharply-defined pattern, while the saw wave is mostly perceived as static and shapeless. The saw wave stream also appears to be "ducked"[10] by the noise stream, decreasing in level every time the noise passes by a region of the sphere. Concentrating, it is possible to perceive a change of emphasis of the saw wave in different regions of the surface, although it is subtle.

Sound file playback

When playing a mono sound file without a parameter field being set, not using the decorrelation `Udef`[11], and with the same per-point values for all parameters, all audio signals at each virtual source are identical. Walking around the Sonic Lab away from the centre, one can perceive a point source located statically at the closest loudspeaker. Undoubtedly this is due to the precedence effect, which was mentioned in section §2.3. While walking, as the distance between the two closest loudspeakers becomes almost the same, the sound becomes unlocalizable, and without any sense of envelopment or extent, then moving away from this location the sound is quickly localized again in the closest loudspeaker. Applying the randomHills parameter field to the amplitude parameter (still without decorrelation) one perceives a point source moving through the room, with a very vague trajectory, with recurrent jumps.

[10] Ducking is the reduction in gain of one audio signal when another signal increases in gain.
[11] `\decorrelate` `Udef` which uses the `PV_Decorrelate` ugen from *BEASTMulch*.

Placing the decorrelation unit after the playback unit, in any position in the room the sound is no longer perceived as coming from the closest loudspeaker, and the localization becomes very vague. Standing at the centre, the sound is concentrated in the direction where the head is pointing, and moving the head seems to also move the sound. Unlike with decorrelated white noise, where the sound is clearly localized in the whole sphere, in this case, there is very little sense of sound coming from the back or sides. When the decorrelation unit is deactivated, immediately a point source with a clear position is perceived. At the precise moment when the decorrelation unit is turned back on it does appear as if one can hear sound from the back and sides, although that perception quickly subsides.

With the decorrelation activated, applying the randomHills parameter field to the amplitude results in the perception of the sound travelling in the surface as a "blob" with extent, whose precise size or shape is indeterminate, and the movement only seems to occur in the hemisphere in front of the listener, going at most 90 degrees away from the median plane. Listening from an off-centre position, the sound moves in the same fashion as just described for a central listening position, but the movement appears to be restricted to the hemisphere to which one is closer, even if one's back is facing the loudspeakers, in which case the sound will seem to come either from the back or sides but not from the front. We can conclude that with this setup the spatialization is not stable and independent of listening position and head direction, although a sense of envelopment and a weak perception of a spatial pattern is produced.

With just file playback (decorrelation unit deactivated) and randomized per-point start time in $[0\text{s}, 5\text{s}]$ the sound is clearly localized in the entire sphere, and the envelopment is much stronger than in the previous case. This technique works specially well in textural recordings without silences where a randomized start time in a range of one minute still guarantees playback of spectrally-similar material. In such a setup the level of immersion is comparable to decorrelated white noise.

Granulation

Assigning the wave2DSin parameter field to `trate`[12] there is a periodic increase and decrease of grain size, translated into the sound periodically becoming more textural.

[12]Output of the parameter field scaled to $[13, 32]$, and a low value of spatial frequency (`1`).

The direction of movement is hard to determine and the spatial pattern is blurred.

Applying wave2DSin to the grain position, if the peak of the wave hits upon an area of the buffer with sound while the trough hits upon a more silent area, then the movement is very clear since it is similar to assigning the field to the amplitude parameter. With the peak and trough hitting upon zones of the buffer with different spectral characteristics the direction of the movement is again very perceivable. The resulting effect can be described as follows: one wave carrying the first sound is pushed out of the room by a second wave that appears with the second sound, with the cycle repeating. The multiple audio streams do not completely merge, they form a texture and with attentive listening one can still perceive the individual streams.

5.1.3. Vertical rectangular grid

The perceptual descriptions presented so far are specific to the spherical system of the Sonic Lab. As was mentioned already, a second system, a rectangular grid of portable loudspeakers attached to a wall was also used during this research (see section §3.3.6) . I would now like to comment on the differences between this setup and the Sonic Lab, where the loudspeakers are roughly arranged in a sphere. Comparing a typical listening position for the rectangular grid positioned 4 meters away from the grid to the listening position at the centre of the Sonic Lab, the angular density[13] of loudspeakers was an order of magnitude higher in the first case. Assuming these two listening positions, and virtually transporting the rectangular grid to the Sonic Lab, the entire grid would fit between 4 loudspeakers of the Sonic Lab. It is therefore not surprising that the perceptual results when listening to *ImmLib* output with the rectangular grid were characterized by significantly higher definition of the spatial patterns and movements[14].

What follows presents listening tests conducted with the rectangular grid setup. The listening position was centred relative to the rectangle and at a distance of 4 meters, facing the centre of the array. The procedure was the same as described at the start of the chapter.

Applying the wave1DSin parameter field with `angle` = 0 to the amplitude in the `white-noise Udef`, a downward movement can be perceived. With this setup it is

[13]Number of loudspeakers per radian.

[14]The fact that the Sonic Lab is more reverberant was possibly also playing a role in the diminished resolution

possible to evaluate different values of l, where l is the spatial frequency of the wave:

$l = 1$, A rectangle of noise, of height comparable to the height of the grid sweeps across the grid from top to bottom.

$l = 0.4$ A line of noise spanning the width of the array sweeps the rectangle from top to bottom.

$l > 0.5$ One starts to hear the individual lines of loudspeakers of the array activate and deactivate one by one, since in this case one cycle of the wave is narrow enough to fit between two lines of the array.

$l > 1$ Two lines of the array play concurrently, as now the wavelength is small enough for two cycles to fit in the height of the grid.

The direction of the movement is extremely clear, and relatively small variations of angle are detectable.

Comparing this parameter field with wave2DSin, which creates a circular instead of plane wave, with the centre of the wave at the mid-point of the upper segment of the rectangle one perceives again a downward movement, but this time the sound does not start from a whole line of loudspeakers but just from the two centre loudspeakers of the top line and then spreads outward from that point, ending noticeably in only two loudspeakers at the two bottom corners. The difference between the shape created by the plane and circular wave can therefore be detected, which is quite remarkable since both movements share the same general direction and their shapes are not too dissimilar.

In general the surface patterns are more clear when standing up, and located at a point equidistant from the left and right borders of the grid, since in this position the head is exactly at the centre of the grid. The general perception for each parameter field is similar to the Sonic Lab except that the patterns are quite sharper. The precision with which one can hear the shape of the parameter fields on the rectangular grid for simple materials such as white noise approached that of the visual domain, it felt almost as if one was seeing the patterns, a quite peculiar and unusual experience. When using more complex sources and multiple layers the effect was not as pronounced.

When positioned very close to the wall (< 1m) segregation of the audio signals starts to happen although there is also more envelopment and immersion. In this case beside angular movements, approaching and receding movements are also perceived.

5.2. Electroacoustic composition - Collaboration with sound artists

As part of this research different sound artists were approached and invited to use *ImmLib* to produce new sound works. The objective was to uncover how the tool would be used in the production of these works, specifically, how different composers with different styles and interests would make use of the possibilities afforded by the tool, and to what extent the tool would influence their compositional approach. The collaborations would also provide valuable feedback on the practicalities of using the software, feedback that could be used to correct problems and to make the software more intuitive. In total three composers collaborated in this project, Michael Dzjaparidze, Pablo Sanz and Justin Yang. As an additional contribution I also created a composition using the library, which is analysed in this section. These collaborations, involving extensive interaction and close contact between myself and the subjects over long periods of time and featuring open-ended goals, differ clearly from a systematic controlled experiment with large sample size for the study of pre-defined variables. One of the objectives of this research was to uncover the possibilities of the specific technique described here, and as such it was not preoccupied with obtaining validated scientific results in the field of psychoacoustics but was instead geared towards musical discovery and exploration, through practical work. Since this section will deal extensively with these three composers, and since there is no danger of ambiguity, I will refer to them by their given names.

The collaborations began with a period of familiarization with the library, with individual training sessions, followed by a practical work period where each composer worked on their own with the tool, with sporadic meetings taking place to resolve technical issues and to exchange information. While Michael, Justin and myself presented finished works in public realized using *ImmLib*, Pablo having started the collaboration later, did not have time to complete a work at the time of writing.

Justin presented an acousmatic theatre performance in March 2014 at the Sonic

Lab under the *Theatre Lab* project, containing some sounds created with *ImmLib*. Two works, *Extase 2, 3 & 4* by Michael and *Unexpected Criticality* by myself, were presented in a concert dedicated to *ImmLib*, in November 2014 at the Sonic Lab. In both works practically all sound events were spatialized using parameter fields and no other software was used besides *SuperCollider*[15] and *ImmLib*. The three works are available for listening in binaural stereo format in the annexed media. The binaural versions allow the reader to gain some sense of the spatialization, nevertheless they constitute a very poor documentation of the real experience since a considerable amount of spatial detail is lost.

The composers were asked to fill out a survey[16] in order to gain more understanding of their background, specially regarding their practices in spatialization and electroacoustic music, to obtain insights into their use of the tool, which features were most exercised, and to what purpose in the compositions, and finally to obtain feedback regarding the user experience and personal evaluation of the tool's impact on the composition practice. In the case of Michael and Pablo, informal interviews were also conducted in order to answer some questions that arose after a preliminary analysis of the work done and the data from the survey.

The following subsections analyse the outcome of these collaborations taking into account the works produced, how *ImmLib* was used in the production of the works, the answers to the survey, the informal interviews conducted with the collaborators and the notes taken during this process.

5.2.1. Michael Dzjaparidze

Background

Michael Dzjaparidze is a composer of electronic music, his work has two threads, electroacoustic works leaning towards the ambient aesthetic and electronic dance music. During this collaboration he was a Ph.D. candidate at SARC [126], developing a *SuperCollider* library for physical-modelling (PM) sound synthesis[17], and working on a portfolio of electroacoustic pieces. His main tool for generating material for his com-

[15] Michael's piece did use his own *SuperCollider* library *PMLib* [126].
[16] Justin declined to fill this survey.
[17] Part of library is written in *python*. The library focus on the creative possibilities for electronic music rather than on accurately simulating existing acoustic instruments.

positions is *SuperCollider*, hence being well prepared for using *ImmLib*[18].

Michael reports that most of hist past work was stereophonic, although since arriving at SARC he has composed 8-channels, 24-channels and 32-channels pieces. Michael's approach to spatialization prior to contact with *ImmLib* has been to create the same number of decorrelated signals as loudspeakers from a single synthesis definition. When working on his synthesis definitions, the use of stochastic ugens which are automatically given different random seeds in *SuperCollider*, ensures that the multiple signals are decorrelated. Also to ensure decorrelation some parameters of the unit are manually or algorithmically given different values for each stream. Michael feels that this approach to spatialization works very well for ambient and drone music as it immerses the listener in the sound creating a powerful experience. On the other hand, he finds that trajectory based spatialization, featuring very precise movements of single mono sources, does not work well on his pieces. Michael's approach has similarities with parameter field spatialization, and he reports this as a reason for becoming interested in using the library.

The collaboration was initiated with the explicit objective of Michael realizing the spatialization of a specific composition for the Sonic Lab's 32 loudspeaker system using *ImmLib*, which was to be added to his portfolio. After one or two introductory sessions in November 2013, where Michael gained an understanding of the capabilities of the tool and how it could be used for his piece, he decided on a strategy for realizing the piece. He would first create and sequence all the sounds in stereo (headphones), using *UnitLib*, and afterwards the score would be converted to the *ImmLib* syntax and the parameter fields and surfaces added. This approach was chosen since he prefers to start to work on a piece in stereo, using headphones, in order to fine-tune the timbral aspects of individual sounds.

He worked independently on the piece, with sporadic technical input from myself relating to *SuperCollider* and *ImmLib*[19]. The stereo version of the piece was completed in February 2014 and Michael started working on the piece in the Sonic Lab in May 2014. The piece was finally presented in November 2014.

---

[18] He is also proficient in *Processing* and *Python*, and has used many other languages such as *MATLAB*, *C*, *C++* and *JavaScript*. He also had previous experience with different computer-music synthesis environments besides *SuperCollider* such as *Max/MSP*, *PD* and the *Web Audio API*.

[19] The authorship of the code for the piece is entirely his.

Extase 2, 3 & 4

The piece *Extase 2, 3 & 4*, lasting 19′40″, is composed from different sections which, except for an initial abrupt transition, slide seamlessly into each other. In each section several layers of complex textural harmonic and sometimes stochastic sounds undergo a slow spectral evolution. All sounds present in the piece are synthetic, created with different `Udefs`, and the entire piece consists of a single *ImmLib* `UScore`, the piece being therefore entirely created in *SuperCollider*. The score was created using partial algorithmic structure generation followed by extensive manual tweaking of parameters (from event duration, fade-in and fade-out times to numeric values or break point envelopes for synthesis parameters). The `Udef`s used on the piece can be divided in three groups:

1. physical modelling `Udef`s, simulating coupled objects such as strings, plates and resonators, created using second-order digital filters whose coefficients are calculated using *PMLib*.

2. `Udef`s built out of stochastic ugens used mostly as excitation signals for the PM objects.

3. Subtractive synthesis `Udef`s, sometimes used to simulate natural sounds such as wind and water.

The piece is composed of a large number of short duration events, using the excitation and subtractive synthesis `Udef`s, and a small number of long duration events, assigned the PM `Udefs`. Given that all `Udef`s, but specially the PM `Udef`s, were quite CPU intensive, the system was used in direct mode (no VBAP panners), and the number of points in the sample set of the surfaces had to be limited to around 24 points, therefore less than the total number of loudspeakers in the system[20]. All the parameter fields used in the piece were assigned to the `amp` parameter of the `Udef`s.

Perceptual description of the piece

The piece opens with a set of short duration events which are reminiscent of waves crashing on a beach created using filtered broadband noise with breakpoint envel-

[20]The score used a large number of surfaces, each with a different selection of around 24 loudspeakers, therefore, when the output of all events are taken into account, all the 32 loudspeakers were used.

opes. The envelopes were finely tweaked in order to create a highly realistic rendition. Several individuals reported after the concert their belief that these sounds were not synthetic but instead recorded. Michael reported that when spatializing these events with *ImmLib* his intention was to re-enforce as much as possible the realism of the beach wave sounds. Each sonic gesture, a wave crashing, is perceived as a single auditory stream occupying an evolving region of the surface. The overall spatial and sonic effect, even if it was not realistic to the point that a closed-eyes listening experience would literally transport oneself to a beach, was very immersive and evocative of the type of soundscape one hears in that situation, creating a feeling of “spaciousness”, envelopment, of being “in” the sound.

The events of this section were divided into two groups, each sharing a spherical harmonic parameter field[21], with the fields rotated such that the lobes pointed in different directions. The parameter fields of the two groups were very effective at creating slow movements across the room with clear direction of movement. Because the start of the event was percussive (the first impact of the wave), it sounded as if the wave crashed on one side of the room and then traversed the room through the sides. Michael reported that when experimenting, of all the parameter fields, sphericalHarmonic was the smoothest, and the one which best matched the nature of the wave sound, with low spherical harmonic orders being more adequate for his goal.

The next sections consist mostly of different PM chains (strings and plates) being excited by different stochastic or impulsive signals. An excitation chain sends its audio output via buses to the input of the PM chain and a parameter field is applied only to a PM chain and not to the excitation chain. The excitation chains were nevertheless using the parallel expansion in order to create the same number of decorrelated signals as points in the surface sample set, therefore, although at each point of the surface the PM chain was excited by the same excitation `Udef` with the same parameters, due to the randomness of the stochastic ugens in the excitation udef, at each point the PM object received a different audio signal.

The PM `Udef`s required a different approach from what has been described so far in order to be used in *ImmLib*. These `Udef`s create a virtual model of a physical object by sending excitation signals into a set of second order resonators, which correspond

[21] This was achieved by sending the output of the parameter field to a bus which was read concurrently by all events in one group. The sphericalHarmonic parameter field was used with `m` = 1 and `l` = 1.

to the normal modes of vibration of the object [126]. It would have been too CPU intensive to have all the modes play at each point, therefore each PM chain had its modes distributed amongst different points of the surface sample set, such that different harmonics or partials of the sound created by the `Udef` were distributed to different points on the sphere. The *ImmLib* spatialization strategy was essentially the same for all PM chains: a randomHills parameter field was applied to the amp parameter, with settings[22] generating smooth changes, and ensuring that the sound always covers the whole sphere

The output of the PM objects can sound quite different depending on the nature of the excitation signal. Percussive excitation signals in some instances sound like a solid object hitting the string/plate and at other times like a bow or scrape like sound, and were localized to different positions depending on the listening position. When one listens in a position close to the centre of the room the localization is ambiguous and fuzzy, paying close attention, each attack is localized at a different position. When closer to the edges of the room the percussive events very clearly localize at the closest loudspeaker. The output when using non-percussive excitation signals, is localized in the whole sphere enveloping the listener, although specific frequencies (modes) seem to come from specific locations in the room. With attention it is possible to detect the location of some of the frequencies of the sound, although the perceived location can change by altering listening position or moving the head. Some of the amplitude crescendos intrinsic to the sound suggest it is starting from a point in the sphere and spreading to the whole upper hemisphere. The movement of the random hills causes specific string/plate modes to appear and disappear, this causes the set of active modes to evolve during this section. As the modes fade-in and fade-out, slightly different chords are formed, therefore the spatialization also interacts with the harmonic content of these events.

At around 3:30 one event starts which evokes the sound of wind, created by sending a complex broadband-noise excitation signal to a string which has a wave2D parameter field with spatial frequency controlled with an envelope. The parameter field creates a sense of motion with unclear direction of movement. Michael reports that this event in the stereo version was acting as a static background layer to the rain-drop sounds,

[22]Usually 3 hills, hill-change period 3.5s, and hill wideness in $[0.4, 0.65]$.

and that when this parameter field was added in the *ImmLib* version, the event became more dynamic and rich, without becoming overtly gestural. This event was pointed out by Michael as a case where assigning a parameter field unexpectedly created an interesting spatial and sonic effect. It altered the nature of the event relative to the stereo version, creating a "circular wave like motion, which sounded more like an echo of the very first-part wave", something which was not present in the stereo version.

During the middle sections of the piece the excitations sometimes become dense enough that only a single auditory event is perceived for a specific event, at other times, with less dense excitations, many small duration auditory events appear all over the sphere, sometimes suggesting a soft percussion or rattling.

The end of the piece is signalled with a group of events producing what sounds like a beach wave traversing the listening space, in a very clear spatial gesture, which echoes the start of the piece. It is interesting that even if these events use a randomHills parameter field the movement is perceived as vectorial along a direction, probably due to the short attack of the sounds.

Observations on the use of ***ImmLib*** on the piece

Having analysed the piece it is now possible to make some remarks regarding the use of *ImmLib*. The spatial nature of the events can roughly be separated into 3 categories:

1. Broadband-noise-based sounds: the multiple audio signals merge completely into a single auditory stream creating a unified spatial image. They are enveloping, being perceived as occupying (without a parameter field) the entire surface constantly. The spatial surface patterns generated using parameter fields are very sharply defined.

2. Sustained harmonic PM sounds: also enveloping but there is less auditory stream merging than in the first case, in some situations each audio signal can be perceived as an entirely separate entity. Whether merging happens or not seems to depend on the density of the stochastic events, the duration of each percussive event and on the level of concentration of the listener. Spatial surface patterns are more blurry.

3. Percussive, textural, micro-event sounds: their spatial properties range from very

clear localization of each percussive event, in which case each audio signal is perceived separately, to a texture with mostly vague localization although still with perceivable individual events when listening attentively .

With events of categories 2 and 3, the segregation also seemed to depend on whether multiple *ImmLib* events were playing concurrently, in which case if one such event (of type 2 or 3) was in the background, since less attention would be drawn to it, the chances of merging appeared to be higher.

The surface patterns created can be divided between those featuring vague spatial movements and blurry surface patterns, still with a sense of envelopment and immersion, and those with well-defined gestural movements and sharply-defined patterns and possibly also a very clear vector or direction of movement. Of course, any gestural movements mentioned above were still more diffuse and vague than trajectories of point sources. Some of the percussive events, with a sharp onset, at times created gestural movements due to the abrupt beginning of the sound calling attention to the initial position or shape of the pattern, which then evolved spatially during the sustain and decay phases.

Reflecting on the collaboration and composition process

Michael reported satisfaction with the final result of the piece, particularly regarding the spatialization produced using *ImmLib*. Nevertheless, he did find that not every element was enhanced by the use of *ImmLib*, specifically he felt the ending section sounded better in the stereo version[23]

Michael also remarked that it was unfortunate that due to CPU usage limitations, the system was not capable of rendering 32 parallel audio signals or more when used on his piece, as having more points would probably have enhanced the spatial effect, specifically, he felt the string events sounded somewhat sparse and that at times he could feel there were "holes" in the surface: regions without perceivable output for these events.

[23]For the final segment of the piece the original output was recorded and then played back with the original layer, pitch-shifted one and two octaves lower. Michael felt the added recorded layers worked better in the stereo version although he couldn't pinpoint the cause of this. Also, he mentions that if he had had more time he would have modified some parameters further, for instance adjusting somewhat parameters impacting on the density of textural sounds, to compensate the change from stereo to 32 channels.

If CPU were not a limiting factor, then the PM objects could have been spatialized in *ImmLib* differently. Instead of distributing the modes amongst the points of the sample set, each mode could play concurrently at all points of the surface, and be assigned a separate parameter field. As to the holes mentioned above, it is possible that besides the low number of points, another factor was at play. It has been shown a complex periodic sound with components spatialized at different locations can still be perceived as a single auditory stream, which is localized at a single location [127, p. 297].

Michael gave the following justification for using parameter fields only on the amplitude parameter. The piece had a large number of events with an already well defined timbral structure, to reverse engineer that structure to make use of parameter fields while keeping the same spectral gesture intact would have been laborious, and not feasible in the time available. Lack of time was also the reason mentioned for not doing more experimentation with different synthesis parameters. Finally, he mentioned that in order to explore modulation of other parameters a good strategy might be to build the piece around the intended spatialization instead of adding a spatialization layer to an already existing piece.

Similarly, when discussing why he did not create a new parameter field, only using the ones predefined in *ImmLib*, he again remarked that the time available to finish the piece was not enough to enter into that type of exploration.

Regarding the question of what did *ImmLib* bring to the composition, Michael reports that if he had not had access to *ImmLib* he would have probably spatialized the piece by creating 32 concurrent decorrelated signals assigned to each loudspeaker of the Sonic Lab which would probably also have created a quite immersive and enveloping spatial sensation. He feels therefore that what *ImmLib* brought to the piece, with the movements and patterns created by parameter fields, was a more dynamic spatialization. He observed that even if using just decorrelated signals is “very effective”[24], using *ImmLib* “definitely adds another dimension”. As specific examples, he mentions the static background layer to the rain-drop sounds on first half of the piece, where the movement induced by the parameter field contributed to a more dynamic and engaging event, and the sea wave sounds at the start of the piece which became more realistic due to the movement from the parameter field re-enforcing the intended illusion.

[24]The quotes in this chapter are from the survey responses available in appendix E or from the interviews conducted with the composers.

Michael reports that using *ImmLib* did not significantly change his understanding of spatialization strategies in electroacoustic music, as his approach was already quite close to the one presented here, but it did extend it, pointing to ways of making the spatialization of decorrelated signals more dynamic. As to whether he would be interested in using *ImmLib* in the future, the answer is affirmative, specially taking into account that since he worked on *Extase 2, 3 & 4* the library has become easier to use. He would be particularly interested in working on a piece where *ImmLib* would be used from the start, designed to take advantage of the possibilities of the library, exploring some of the features of the library that he did not have time to explore fully, such as using parameter fields to modulate synthesis parameters other than amplitude or using multiple parameter fields per unit.

5.2.2. Miguel Negrão

In September 2014 I started composing a piece centred around *ImmLib* with the goal of presenting it in a concert dedicated to *ImmLib*-based compositions in the Sonic Arts Research Centre on 13th November. The starting point for the piece was a group of examples or demos which I had developed since 2012 to explore the possibilities of the library and gain insight into what types of ugens and parameters could be of interest.

Unexpected Criticality

The piece, named *Unexpected Criticality*[25] with a a duration of 15'30" is entirely composed of synthetic material assembled directly in an *ImmLib* score for real-time playback[26].

In total three `Udef`s were used in the piece. One featured a non-periodic random impulse generator with controllable density, triggering an envelope with controllable shape multiplied with pink noise and filtered with low-pass and high-pass filters. This `Udef` creates clicks of different timbre, sharpness, density and frequency range. Another `Udef` featured the sum of several sinusoidal oscillators with random frequencies in a given range (fixed at synth instantiation time), which are slightly deviated by summing

[25] A reference to *criticality accidents*, uncontrolled nuclear chain reactions.

[26] While all events were tweaked in real-time during the composition of the piece, for the final playback version, specific events were bounced to 32-channel files separately and imported back into the score in order to be able to play the whole score without CPU issues. The code for generating the piece is available in the annexed media.

with a noise generator . The output of each sine wave is multiplied with an envelope with a random start time. This `Udef` was used in the piece with 15 sine waves over a range of high frequencies (e.g. 10000Hz to 16000Hz), resulting in a buzzing sound. The envelopes with random start-times contribute to internal change as different sine tones progressively appear and disappear, changing the timbre of the sound over time. For some events a variation of this `Udef` was used where the sine waves were multiplied with a low-frequency square wave, creating a more localizable texture.

The third `Udef` appearing in the piece is a feedback system, using a triangle wave generator, a sinusoidal oscillator and a pitch-shifter which is based on code found at the `sccode.org` website by xfff [128]. This `Udef` although quite simple can create very complex and drastically different outputs, from almost pure tones to harsh noise, passing through gestural events which sound almost humanly-composed. The timbral evolution often goes on for several minutes, even without changing or modulating any parameters. Usually the number of harmonics increases until finally bifurcating and becoming noise-like. Although its output is at times perceived as unpredictable, chaotic or random, the `Udef` is entirely deterministic, therefore using the exact same settings gives the exact same audio output, even after large periods of time have elapsed[27].

The ugen diagrams and corresponding code is available in appendix D.9.

Regarding the spatialization, all events in the piece were spatialized in *ImmLib* by applying a parameter field to the amplitude parameter. Unless otherwise stated, the surface used was the sphere with the spiral discretization algorithm with a sample set containing 20 points, instead of 32, in order to be able to work on the piece in real-time. One specific event that benefited considerably from a higher number of points was rendered separately using 40 points and then re-imported into the score as a soundfile.

Perceptual description of the piece

The piece starts with a loud percussive burst of noise coming from a relatively well defined direction towards the top of the sphere which quickly crumbles into a stream of noisy particles spreading down along the surface in multiple individual threads, with the threads then moving along the sphere. This initial sound is followed by other

[27]It should be noted that the algorithms used to generate noise in a digital synthesis system make use of pseudo-random number generators and are also fully deterministic, depending only on the random seed. In the case of this `Udef`, since no stochastic ugens are involved, no random seeds are involved either, and the same setting always produce the same output.

events with similar envelopes, timbres and spatialization. All these events had the randomHillsDeterministic parameter field assigned to the amplitude parameter, in order to choose a specific spatial gesture for each event. This parameter field was created by modifying randomHills such that the sequence of random hills is calculated beforehand and stored. Using the same stored sequence produces the same spatial movement. These events have a lot of internal spectromorphological movement which together with the rapid changes from the randomHills parameter field creates the perception of different threads of sound moving through the sphere, sometimes spatially merging and then separating again.

While these events are ongoing another event starts with a high-frequency abrupt percussive onset, which then quickly stabilizes into a low-frequency growl aided by the use of the subwoofers. The `Udef` for this event contains a low-pass filter with frequency starting at 20000Hz, and progressing down to 100Hz in 6 seconds. The envelope had a different start time for each of the 20 parallel audio signals, depending on the height of the associated point on the surface, with the lower points starting the envelope first and the points in the ceiling being the last[28]. This was done in order to create a spatial effect where it seemed that the high frequencies were leaving the room through the ceiling. The effect was quite subtle, but added a spatial element to the spectral gesture.

At this point in the piece an abrupt change happens when the previous event is suddenly replaced by a new event which will last 6'50", almost half the duration of the piece. The event, which also uses the feedback `Udef`, starts with a loud, violent percussive sound, and settles quickly into a complex sustained distorted harmonic tone with texture. The drone undergoes a continuous timbral evolution created by applying envelopes (identical at all points) to the feedback parameters. Towards the end of the event the tone becomes progressively noisier with energy in the high frequencies progressively increasing. The event has the amplitude modulated by a randomHills [29] creating a smooth set of slowly-moving hills constrained such that the amplitude level is between 0.6 and 1.0, with a single auditory stream always occupying the whole sphere.

[28]This procedure is equivalent to making the start-time of the envelope a parameter, and applying a gradient parameter field to the start-time. That was not done because at the time it was not possible to apply parameter fields to parameters of `UMaps`.

[29]The settings of the randomHills were `'numSecs', 6.1, 'numHills', 3.0, 'sizeA', 0.50, 'sizeB', 0.62`.

The motion of the spatial pattern becomes more perceptible as the high-frequency content of the drone increases. The piece was played purposely quite loud, this caused the small changes in level due to the field to be easily perceptible. This event, consisting of a dense noisy drone, played at a high loudness level and coming from all directions, creates a sense of violent envelopment as if compressing the listener with a sphere of sound. After a while one grows accustomed to the "compression", relaxing, and starts to feel more as if "inside" the sound. When this event is played at a high loudness level it seems to cause the sound to be localized closer to the listener than the distance of the loudspeakers, something which does not happen when playing at a lower loudness level, contributing to its immersiveness.

The event becomes progressively more sonically unstable and noisier, until the drone decays in hiccups into broadband noise together with pitched "blips", which segregate into multiple threads. At this point it is abruptly replaced by a group of high-frequency sine tones.

The sine waves are separated into three events which start within more or less a minute of each other. Each event contains a group of increasingly high-frequency tones. Each sine oscillator has an envelope with fixed duration but variable start-time. The envelope's start-times and the frequencies of the oscillators are randomly selected at synth instantiation time[30], and are therefore different for each point of the surface. The amplitude parameter of the unit is assigned a randomHills parameter field similarly to the previous events, but this time between 0 and 1 creating silence at times in regions of the sphere. The change of emphasis on the surface caused by the randomHills causes the spectral content to also change at the same time, since each point contains a different group of sine tones with different envelopes. The first event has very low locatedness, it does not form a pattern occupying a specific area, although the locatedness oscillates during the duration of the event. In those moments when the sound feels more localizable, the movements of the head greatly influence localization. The third group of events, containing higher frequencies than the previous two, feels slightly more localizable.

Around this time another group of sine tones start which are amplitude modulated with a low-frequency square wave, resulting in significantly more localizable events.

[30]Using the `LinRand` and `ExpRand` ugens.

In these events a moveHills parameter field is assigned to the amplitude parameter, creating a very clear sense of movement, almost like trajectories, as if events with duration of a few seconds would start at one position and make a trajectory to another point and disappear. The trajectories were nevertheless very vague. The multiple simultaneous trajectory-like movements create a sense of quick movements happening all around the sphere in quite precise locations, in contrast to the hazy localization of the other three groups of sine tones.

At 9:40 an event using the clicks `Udef` starts, creating very sharp clicks (from 2ms to 0.02ms), with very low density, such that each click is a separate auditory stream localized very precisely in one of several static positions. For this event a sample set with 40 points was used, to ensure adequate spatial density, since the individual clicks were very localizable. As the density increases and the sharpness decreases, the clicks from different points start to happen almost simultaneously and the event progressively morphs into a noisy texture becoming a single auditory stream with extent, that moves and spreads through the surface. With this event it is possible to visualize with great clarity the actual shape of the sphere where the sources appear to be positioned. Interestingly, the earlier drone even if using the same set of loudspeakers, is perceived as emanating from a quite smaller sphere, thus the spectral and temporal nature of the sound material clearly influences the perception of the size and distance of the virtual surface.

Finally, the last two events of the piece make their appearance. They are created with the same "clicks" `Udef` but with a longer impulse duration resulting in a low-frequency noise texture which reminds somewhat of wind and muffled rain drops, although the suggestion is vague and not particularly realistic. These two events dynamically distribute themselves through the sphere, changing quite slowly, with somewhat vague localization, being perceived as coming more from the ground and underground levels. The two events did at times sound as if they were traversing the sphere under the floor, possibly since the loudspeakers above, in rings 2 and 3, had difficulty outputting these low-frequencies. The last event, the one with the lowest frequencies, has quite a long fade-out which together with the low-pass filtered noise creates the sensation of the space opening up, of the object becoming progressively further away while still enveloping the listener.

Observations on the use of *ImmLib* on the piece

The first events of the piece present very clear spatial gestures, composed of "threads" that split and merge into a "blob", which are quite different from point source trajectories. It is a good example of the type of spatial gestures that are possible with *ImmLib* but difficult to achieve using nothing but uncoordinated individual modulation of the amplitude at each point on the surface, for instance using stochastic ugens.

The goal of the spatialization for the drone section was to create, first a sense of compression, and later of entering "into" the sound, while maintaining spatial interest during the event by continuously changing the spatial emphasis of regions of the sphere. The isotropic nature of the sphere sample set ensured that the sound was coming from all directions, and given that a single spectrally dense auditory stream occupying the sphere was perceived, it created a quite immersive experience.

The sine wave groups present a case of interesting interaction between the amplitude parameter field and the spectral content along the surface, since timbral changes are created from amplitude changes. The internal envelopes of each sine wave, also cause the beating patterns to progressively change, adding to the spectral change. This made the event more dynamic, contributing to continued interest throughout.

Regarding the use of different perceptions of distance, the sine wave events give very little sense of distance. On the other hand the clicks, which start as the sine waves are playing, produce an extremely clear sense of distance and direction, with the two overlapping spatial sensations contrasting with each other.

The only parameter fields used on the piece were randomHills, moveHills and gradient, while the only parameters modulated were the amplitude except for the feedback events where the feedback-network controls were also applied the gradient parameter field, in order to create timbral differences. Since this was my first composition with *ImmLib*, I decided to focus on a small set of features of the library, and leave further experimentation for future work.

5.2.3. Pablo Sanz

Background

Pablo Sanz is a sound artist and composer who works mostly with environmental-sound, recorded using a diverse array of listening devices from different types of microphones to hydrophones, ultrasound sensors and accelerometers. He has presented his work in systems with 8, 16, 32 (Sonic Lab) and 43 (ZKM sound dome) loudspeakers.

His approach to spatialization is eclectic but the use of multiple decorrelated signals appears often throughout his work and he very rarely resorts to trajectories applied to virtual point sources. His recordings are usually performed with multiple sensing devices simultaneously placed at different locations so as to obtain several decorrelated perspectives of the same aural event. Being reliant on multiple decorrelated audio signals to create an immersive situation, he has developed strategies for audio decorrelation. For situations where multiple decorrelated recordings of the same event are not available, or in order to contrast even more each audio signal, he applies different per-channel time shifts. Other approaches used include differences in playback speed and spectral spatialization.

In *Transient Lapse* (2012) he created a public site-specific installation featuring 8 loudspeakers placed linearly along a horizontal distance of 100 meters controlled by custom software which allowed not only to create the usual linear *a* to *b* trajectories but also other spatial patterns such as growing from the middle to the sides and from the sides to the middle. This approach is somewhat similar to that of *ImmLib*, although reduced to one dimension.

In 2013 he presented another installation featuring a custom-built grid of 24 portable battery-powered loudspeakers covering the ceiling of a room. The content was created using granular spatialization of recorded material featuring the sound of a propeller from a wind turbine. This piece, developed independently from the research being described in here, suggested that a collaboration using the portable loudspeaker grid and *ImmLib* could have interesting results.

field::grid

Given Pablo's interest in *ImmLib* and our aesthetic affinities we decided to collaborate on a work for this portable loudspeaker grid, tentatively named *field::grid* with the understanding that both of us would contribute sound material and create different layers or different sections of a piece. Pablo would be working mostly with recorded material while I would use mostly synthetic sound, the work would be presented as an installation, with the grid placed in a wall of a medium-sized darkened room, possibly hidden under acoustically transparent fabric.

Given that Pablo lacked experience with *ImmLib* and that both Pablo and myself were unfamiliar with this particular geometric configuration, a test system was set up in order to explore sonic and spatial possibilities. The portable grid system in a 5×5 configuration was placed in a wall of a studio of SARC, with the 25 channels of audio being generated from *ImmLib* running on a personal computer. The physical configuration of this setup, including a photograph, was already presented in section §3.3.6. A second testing period took place subsequently with a different configuration, a grid of 6×4 loudspeakers with a horizontal and vertical inter-loudspeaker distance of 0.5m. While experimenting with both test configurations different sketches were created, simple experiments combining a small number of layers of different materials. Unfortunately it was not possible to arrive at a finished piece within the time constraints of this research project, therefore only the sketches created during the testing sessions will be commented upon.

The main reason behind the choice of a rectangular grid placed on a single wall was the desire to explore a setup with high spatial density and the necessity of keeping the physical support structure as simple as possible. An installation situation was deemed appropriate since it allows listeners to explore different spatial perspectives in their own time.

Since Pablo is not familiar with the *SuperCollider* language he used the system through the graphical interface, making use of the predefined `Udefs` and `ImmDefs`.

The sketches created by Pablo consisted almost entirely of several layers of recorded material with, in some cases, additional filtering. He used different parameter fields applied primarily to the amplitude parameter, although he also experimented with the frequency of the filter and playback rate.

Different approaches to achieve signal decorrelation were used. In the cases where the objective was to create a textural sound, the start times of the file playback for each synth were randomized within a large range (minutes) or different playback speeds per synth were set. An example of such a texture was created from a recording of a laptop using an ultrasonic heterodyne device which created a drone reminiscent of radio static. In the cases where a recording contained a sound event with a well defined internal envelope which should be preserved, the range for the start times was much narrower, around 20ms, in order to achieve some decorrelation while still maintaining the temporal envelope of the sound. An example featured contact microphone recordings of a metal flag pole actuated by wind turbulence, which besides different per-point time shifts also featured different per-point playback speeds. The first type of sketches (textural sounds) was mostly used together with the randomHills or moveHills parameter field while the second type was used mostly with the wave2DSin parameter field. Another set of sketches experimented with multiple layers of filtered white noise.

My contribution to this collaborative work also consisted, so far, of a set of sketches, some of which I will now describe. One such sketch used the feedback `Udef` already mentioned when discussing *Unexpected Criticality.* The sketch consists of four events, each with a feedback unit followed by a band-pass filter unit, used in order to better separate the events spectrally. The events all use the same `ImmDef` (see appendix D.10) where two separate gradient parameter fields are applied to the `a` and `b` parameters of the feedback `Udef`, creating a continuous spatial distribution of different timbres and behaviours. The ranges of the parameter fields controlling the two feedback parameters are picked so as to have different enough timbres across the grid rectangle while still maintaining a similar sonic behaviour. Additionally one sphericalHarmonic parameter field, with controls for rotation, translation and scale, is applied to the amplitude of the feedback `Udef`. The result is a rhytmic motion, with emphasis placed periodically on different regions of the rectangle. The scale and rotation controls are important as they change the original pattern of the spherical harmonic to better match this specific `Udef`, creating more movement. Another sketch is similar to the previous one but using wave1D instead of sphericalHarmonic: the score contained five events, each with the feedback unit tuned to give quite different timbres from low to very high frequencies, with the wave1D parameter field set to a different angle such that each sound travels quickly along a different direction. As the wave progresses across the rectangle the timbre also changes due to the gradient parameter field that is applied to the feedback parameters[31].

Another scene contains five events, all with a sound file unit set to the same recording of rice grains falling onto a ceramic dish, each event with a different playback speed (equal per-point) and different per-point randomized start times in $[0\mathrm{s}, 4\mathrm{s}]$, with the randomHills parameter field applied to the amplitude[32]. This creates an evolving soundscape and a spatial texture, where the real-world nature of the sound is still somewhat recognizable, with the sound constantly shifting about in the rectangle. There is a lot of spatial movement which doesn't appear periodic, constantly changing in a chaotic way.

The strategies used in these sketches are similar to those already presented when using the sphere, the biggest difference is that due to the increased sharpness of the patterns it is possible to fine-tune the positions and and shapes of the patterns with greater accuracy.

5.2.4. Justin Yang

Justin Yang is a "composer, improviser, instrument builder, theorist and developer of music technology" [129].

Justin, who has experience with *SuperCollider*, showed interest in using *ImmLib* for a composition and live performance. We started collaborating in November 2013, with a number of training sessions taking place where the syntax and semantics of *ImmLib* were explained. While Michael decided to first work on his piece in stereo using *UnitLib*, Justin started working right away with *ImmLib* in the Sonic Lab. This meant that he was exposed to an early version of *ImmLib* which was lacking a significant number of features that eventually became part of the release presented here, some of which were motivated by usability feedback provided by him.

After an initial exploration Justin decided to use *ImmLib* for some of the sound design of an audio drama. This project, the *Sonic Arts Theatre Lab*, was a collab-

[31] A recording in stereo binaural is included in the annexed media with filename `feedbackMultipleLayers.aiff`. It must be stressed that the spatial patterns perceivable with vertical rectangular array are quite less sharp in this recording.

[32] A recording from a version of this patch using `PSphere` in stereo binaural is included in the annexed media with filename `soundFile-randomHills.aiff`.

oration between Tinderbox Theatre Company and the SARC. The performance took place in March 2014 in the Sonic Lab consisting of a 32-channel tape piece. The story revolved around loyalist teenagers who participated in flag protests in Belfast being "rehabilitated" through the use of an experimental therapy. The sonic content of the piece was divided between the voices of the actors which were sent to individual loudspeakers and background, ambient sounds such as aeroplane noise and bands marching, which were spatialized either using just the decorrelation ugen outside *of ImmLib* (`PV_Decorrelate` [77]) or by being assigned a parameter field in *ImmLib*. The sounds were sequenced in a commercial DAW running on a separate computer which sent each individual track to the Sonic Lab workstation running *SuperCollider* for spatialization. On events which were sent to *ImmLib*, either the randomHills or wave2DSin parameter fields were used, assigned to the amplitude. All the sounds sent to *ImmLib* were mono recordings which were sent through the decorrelation unit.

Listening to the piece in the Sonic Lab, the voices localized very clearly at specific loudspeakers, being perceived as point sources. The ambient sounds were somewhat enveloping although, as a comparison, not on the level of multiple decorrelated white noise. It was quite hard to pick up any movement on the ambient sounds, although some of the water sounds near 4:30 and street sounds near 16:30 did appear to have a degree of movement, although very vague.

In this particular project the use of *ImmLib* did not seem to bring significant advantages over using just multiple decorrelated signals via the decorrelation ugen mentioned above. It seems that the level of decorrelation achieved by this decorrelation ugen is not sufficient for *ImmLib* to be able to create resilient spatial surface patterns.

5.3. Insights

Having detailed in this chapter a perceptual description of the use of parameter field spatialization, it is now possible to outline different strategies when using this technique. In the different pieces and examples it is possible to identify some patterns, which appear recurrently, regarding the use of parameter fields, together with specific sound materials, to achieve particular perceptual and compositional goals. As a word of caution I should say these are not absolute rules and the final perceptual result can depend on many factors.

This section also summarizes how the composers made use of the tool, issues regarding usability, and how the tool influenced the works created.

Observations and strategies

Decorrelation, segregation and source extent Parameter fields do not function properly without multiple parallel decorrelated signals. Decorrelation seems to be relatively easy to create when using pure synthesis. It can be achieved using stochastic ugens, such as noise or random impulse generators, as well as with chaotic ugens, in which case slightly different per-point values for the equation parameters should be used. Deterministic ugens, such as waveform oscillators or impulse generators, can also be employed, provided that the per-point settings used are different enough, and influence the output to the necessary extent to cause effective decorrelation. Decorrelation with file playback is less straightforward. If the internal envelope of the recorded sound should be preserved, then a decorrelation plugin based on the technique detailed by Kendall [11] can be used, although the level of envelopment obtained can be less than satisfactory, the spatial patterns quite vague and it creates audible artifacts. If the internal envelope of the sound can be obfuscated, for instance in order to create a texture, a random per-point start-time or pitch-shift can be used, obtaining a higher level of envelopment.

In the Sonic Lab, whilst experimenting with different `Udef`s producing different types of signals I found that sound signals completely unrelated in any way, temporally or spectrally, played at different locations are perceived as different auditory streams and produce a low level of envelopment. At the other end of the relatedness spectrum, the exact same signal played at different locations, due to the precedence effect, is perceived as a single auditory stream with no envelopment. The `Udef`s which produce the most envelopment are the ones which produce not only completely decorrelated signals at each point but also spectrally similar outputs. It was also noted that those producing the most envelopment seemed to have less tendency to cause segregation of the auditory streams, with often a single all enveloping auditory event being perceived. Using stochastic ugens such as white noise seems to be the most straightforward way to achieve high levels of envelopment, a single auditory object with extent and sharply-defined surface patterns.

Movement, boundary, and shapes in the pattern Assigning a parameter field to the final amplitude control of a `Udef` tends to cause the perception of movement along the surface which can have different characteristics.

If the event appears to occupy the whole surface constantly, for instance because the amplitude was scaled such that it is never zero, the movement is perceived as a change of emphasis, with possibly also a change of timbre, presenting specific dynamic spatial patterns, whose sharpness can vary. One way to achieve this effect is to use the randomHills parameter field in an `ImmDef` with a `linlin` scaling function ensuring the minimum amplitude value is positive or with hills with large bases (> 0.6), such that no region of the surface falls silent. The drone event in the first half of *Unexpected Criticality* is an example of this technique.

Besides the internal surface pattern, if the event does not occupy the whole surface, then it can have a shape defined by a boundary which can evolve in time. The amoeba-like movements that can be created with randomHills, where a shape extends to other regions, sometimes splitting and then merging, are a good example of this effect. Some parameter fields, such as `wave2D` or sphericalHarmonics , create movements which clearly happen along a specific direction with clear-cut patterns, while others such as randomHills create more diffuse movements, without any clear direction of movement. The first type can be considered more "geometric" while the second more "organic". In both *Unexpected Criticality* and *Extase 2, 3 & 4*, only "organic" patterns are perceived, either due to the use of randomHills or because certain types of sound material, even when pared with a "geometric" parameter field, still create an "organic" pattern. This might be indicative that "organic" patterns are more desirable from a compositional point of view, although more works by different composers would have to be analysed in order to clarify this issue.

Not all `Udef`s give rise to a clear sense of shape and contour, some `Udef`s, such as those using sine-wave generators, create a very vague spatial sensation, not localizing to a specific region and also not creating a high level of envelopment. This was the case with the sine wave events in *Unexpected Criticality.* Noisy `Udef`s seem to produce more clearly defined shapes than more harmonic ones. The moveHills parameter field produces the result closest to typical point source trajectories, somewhat like particle systems. It was also noted that when a parameter field configured to produce quick movements is used on a percussive, relatively short duration event or with rapid onset, a spatial gesture tends to be produced. An example of this technique can be heard in the starting sections of both *Unexpected Criticality* and *Extase 2, 3 & 4*.

Segregation/merging When the output of the `Udef` is not continuous and the silences are large enough such that there is little temporal overlap between signals for all the points, no auditory merging seems to occur and the position of each virtual source or loudspeaker, is perceived very clearly. If there are enough points in the surface sample set, a feeling of envelopment can still be felt, but it seems to require a higher number of points than with continuous signals which merge, perhaps because in that case it is easier for the auditory system to "complete" the holes in the surface.

Unexpected Criticality features multiple instances of creative use of segregation and merging. The drone event, which during its almost 7 minutes is a single auditory stream occupying the whole sphere, at its end decays into multiple threads which travel through the surface. On the other hand, the very sharp and low-density clicks on the second part, which are at first perceived as individual auditory events, slowly transform into a single auditory event, a texture, occupying large regions of the surface. In the middle sections *of Extase 2, 3 & 4*, there are also events where merging happens, for instance at one point the density of organic waterish sounds increases until they form a single textural auditory event.

Timbral changes with a parameter field assigned to the amplitude parameter When a parameter field is assigned to the amplitude parameter and the `Udef` presents significant timbral differences along the surface, timbral changes will be associated with the movement of the sound through the surface. If the movement is periodic, then the timbral changes are also periodic. The timbral differences along the surface can be produced from different random per-point values for a synthesis parameter or by assigning a parameter field to a parameter of synthesis (other than amplitude) which influences the timbre of the output. The use of a gradient parameter field on the parameters of the feedback `Udef` in *Unexpected Criticality* and the *field::grid* sketches, as well as the sine tone events in *Unexpected Criticality*, are examples of this technique.

When a parameter field modulates a parameter other than amplitude which causes spectral changes, it is not uncommon for two different auditory streams to be perceived, one associated with the part of the spectrum which is kept relatively constant and the other with the part of the spectrum that is changing. The first one is perceived as a spatially static background layer on top of which the second auditory stream seems to be moving. This was perceived for instance in the examples with the `sineDist` and `noiseSawSweep Udefs` in section §5.1.2.

Perceived distance When listening to events in the Sonic Lab, in most cases each event seemed to move along a sphere with fixed radius. Different sound materials appeared to be at different radii. For instance, in *Unexpected Criticality*, the sharp clicks seemed to occupy a sphere with larger radius than the drone event. Nevertheless, in all the pieces and examples, there were only two cases where there was the sensation that the pattern's depth was changing whilst it moved along the surface. One was with the complex tone created out of the sum of many sine tones, presented in section §5.1.2 (`sineDist Udef`). The other was the last event of *Unexpected Criticality* which featured a very long fade-out, which given that it had mostly low frequencies, gave the impression of the event receding from the centre, increasing in radius. In either case the changes in distance were not controllable and not related to the parameter field

It can be therefore concluded that the spatial attribute of depth was not significantly perceived in the scenes constructed during the research, although the possibility that certain types of materials can enhance the sense of depth in surface patterns shouldn't be excluded. Thinking of the cinema screen as a metaphor for the *ImmLib* surfaces, the output of *ImmLib* seems to be analogous to 2D cinematography. To create a 3D effect, where objects "come out" of the screen, a different technique is needed. Most probably it will be easier to produce spatial patterns which recede from the surface, since the spatial cues for distant sources are easier to produce than for very proximate sources.

Non-amplitude modulation It is quite remarkable that the direction of movement and in some instances the spatial pattern can be so clearly perceived when the modulated parameter is not the amplitude, and as such the event does not fall silent anywhere on the surface. In fact, in some of the examples of this type presented before, the sound level is more or less constant along the surface, therefore the pattern appears entirely due to the spectral changes.

Loudspeaker density Increasing the angular density of loudspeakers[33] seems to increase the definition of the surface patterns substantially. It would be interesting to determine if there is a loudspeaker density beyond which no further increase of surface pattern definition is detected by a listener.

off-centre listening Regarding off-centre and non-stationary listening positions, the listening experiments performed in the Sonic Lab showed that, due to the decorrelated content of each virtual source created by the software, even when standing quite close to a loudspeaker the precedence effect did not collapse the global spatial image and in fact it was possible to have different perspectives of the virtual stage by walking around the room.

Plotting From what was described so far in this chapter it should already be clear that what is shown in the 3D plot tool of *ImmLib*, which follows exactly the mathematical description of each parameter field, can be quite different from what is actually heard. The merging or segregation of auditory streams, being a perceptual phenomenon, will not be displayed in the plot tool. Neither will the level of locatedness. In the cases where the interaction of the `Udef` with the parameter field produces a vague spatial sensation, this will not be in accord with the plot tool, which always shows very defined spatial patterns. The plot tool should therefore be used only for indication and not be relied upon for predicting the actual perceptual result, which can only be determined experientially.

Assessing the sound artists' interaction with *ImmLib*

Regarding usability, the different collaborators had access to the tool at different states of maturity, it is therefore difficult to present a single picture. In the current version of *ImmLib* (v0.1.5), it is possible to use the library entirely from the GUI, although restricted to using only predefined `Udef`s, parameter fields and `ImmDef`s. A user with no knowledge of *SuperCollider* can use the library to play recorded material (field

[33]Supposing for simplification that all loudspeakers are at the same distance from the central listening position.

recordings or from other digital synthesis software) and assign parameter fields to the amplitude and playback speed. The user also has available a large set effects such as filters, dynamics processors, reverb, chorus and delays, as well as simple waveform and noise generators, again being able to assign parameter fields to any parameter of these `Udef`s via a GUI. When using the system in this way the usability is not too far from that of commercial digital audio workstations such as *Reaper* or *Ardour*. Pablo, having no knowledge of *SuperCollider*, was able to create quite interesting sketches using only the GUI, most of which used recorded material. It remains to be seen if complex purely synthetic scores could also be done entirely using the GUI, although that question is outside the scope of this study.

For a *SuperCollider* user, extending the workflow described above to also include custom-created `Udef`s is quite straightforward as the syntax for `Udef`s is essentially the same as for `SynthDef`s, and custom `Udef`s once defined on a file can be used from a GUI just like the bundled ones. Nevertheless, *SuperCollider* users tend to not settle for using just GUIs to compose. Using *ImmLib* from code requires at least familiarization with the *UnitLib* code interface, and subsequently with *ImmLib*'s. This learning process requires some effort and time, specially if the user wants to make use of advanced features such as bus management and inter-chain audio exchange. On top of this, in order to write `ImmDef`s the user must become familiar with the FRP concepts and the interface of the specific implementation used with *ImmLib*. Justin and Michael, having both used the library through code, had an average difficulty in learning the *UnitLib*/*ImmLib* interface, however they did struggle particularly with the FRP syntax and semantics, reporting that it was somewhat difficult to learn. This is undoubtedly due to the fact that pure functional programming and FRP is a very different paradigm from the mostly object-oriented style of *SuperCollider* and *UnitLib*, and requires a change of mindset, which is quite harder than learning an interface based on familiar concepts. The feedback provided by the composers regarding `ImmDef`s prompted changes: the syntax was made more similar to what is usual in *SuperCollider* and the monadic implementation was hidden behind an imperative interface. I believe that the syntax of the `ImmDef`s should now be quite more approachable to *SuperCollider* users, and all the power of abstraction for reasoning about time-dependent relations that FRP affords is still available.

In his composition Michael wrote a large number of `ImmDef`s, and although most were quite simple, some made use of FRP combinators to create more complex networks, sometimes low-frequency modulating or applying an envelope to a parameter of the parameter field, demonstrating that he was able to leverage some capabilities afforded by FRP for specific compositional purposes. Justin also created his own `Udef`s, and at the time the auto-generated GUI widgets for `ImmDef` parameters were not available, so he had to create his own GUIs and connect them to the FRP network, which he was capable of doing.

Regarding the explicit feedback provided by the artists regarding usability and learning curve, in the survey Michael considered *ImmLib* “moderately hard” to learn and “extremely hard” to use, clarifying that this was due to the overwhelming amount of technical problems that he faced, some of which were already described in section §4.9. It should be noted that Michael’s piece probably used every single feature of *ImmLib* and *UnitLib*, with some having been introduced just for that piece, which was quite technically challenging. Regarding the difficulties he was confronted with, besides the *SuperCollider* and *UnitLib* issues, he reported that “the fact that when I started using the library, spatial gestures had to be defined for each individual sounding event became rather unwieldy as my work consisted of lots of smaller sub-events”, an issue which was resolved later by adding the `ImmDef`/`ImmMod` division. Justin also reported difficulty with the learning curve, specially regarding FRP.

The plot tool appeared to be a useful resource for composers while working on the piece. Pablo reported using it constantly while working with the system, admitting that when he becomes more familiar with the library the use frequency might decrease. Michael reported using it from time to time, with frequency of use decreasing as he became more familiar with the tool.

It should be noted that none of the composers wrote their own parameter fields. At the outset I imagined that some of them might write their own parameter fields, but in hindsight that was not a realistic expectation. Writing a parameter field requires having some understanding of differential geometry, as one must write functions defined on a surface, and an equally thorough understanding of FRP in order to augment the purely mathematical functions with more dynamic behaviour. These are skills that most *SuperCollider* users do not have. The situation is comparable to *SuperCollider* ugens, the vast majority of *SuperCollider* users does not write their own ugens, and do not have the expertise to do so[34], with the writing of ugens being done by "developers". Likewise, I predict that new parameter fields will only be written by *ImmLib* developers, which will be a fraction of the users.

The first step when using *ImmLib* is to create a `Udef` from which multiple decorrelated signals can be generated. Michael reported that this process was of "average difficulty", having used "differences in filtering, trigger times and independent noise sources (see appendix E)" to ensure decorrelation. Pablo was also able to create multiple decorrelated signals using different start positions or playback rate for file playback and differences in filtering, while Justin used the decorrelation plugin. Both Pablo and Michael reported that the constraint imposed by *ImmLib* that every event must be composed of multiple copies of the same sound process was "not at all limiting" (see appendix E). This is perhaps due to the fact that in both cases it is already part of their workflow to use multiple layers of the same material in continuous sound events spanning long durations.

Looking at how the composers approached incorporating *ImmLib* into their composition workflow, Michael realized first a version of his piece in stereo in *UnitLib*, afterwards converting it to *ImmLib* and working on the spatialization in the Sonic Lab. Justin worked at least partially on the Sonic Lab, using a hybrid approach with a commercial DAW for sequencing, *SuperCollider* with VBAP for spatializing some events, and *ImmLib* for the remaining events. Pablo worked almost entirely on site, using the rectangular 25 loudspeaker grid, utilizing exclusively *ImmLib*. I worked almost entirely in the Sonic Lab, with some preparation sessions in stereo when it was not available, and used solely *ImmLib* for realizing the piece. From this one can conclude that it is possible to realize a piece, from start to finish using solely *ImmLib*, since it provides most of what is expected of a DAW in terms of sequencing and timeline manipulation. All composers spent extensive amounts of time working on site, using the final target sound system, and this seems to be necessary in order to take full advantage of *ImmLib*. The type of spatialization produced using parameter fields is novel, and therefore users of the library will not be able to use previous experience to predict the actual aural result when using the system. It takes time to build up knowledge of how an action,

[34]Knowledge of C++ and the *SuperCollider* ugen API.

approach or technique translates to an aural result, and possibly this knowledge will always be partial as there is always an element of experience that transcends our predictions. During the duration of the collaborations, the only preview system available, for stereo playback, provided essentially no useful spatial information, it was provided only for timeline organization and crude timbral preview. The current version does provide an Ambisonics-based binaural rendering mode which is still of limited use[35]. For pieces being created for a spherical system, the binaural rendering can give a crude preview of the level of envelopment and the type of movements present in the piece, on the other hand, in the case of the vertical rectangular grid, the binaural rendering has quite a lower resolution, and most of the detail which is perceivable on the real system is not perceivable in the binaural preview.

Examining the role of *ImmLib* in the works that were produced, in the case of Michael the tool was mostly used to enhance the spatial experience of the piece, to create an immersive environment which was dynamic and, in specific cases, to create spatial gestures enhancing the intended realism of an event. At the outset and before having used *ImmLib,* Michael had already a quite defined idea of how the piece should be constructed, the tool was therefore used to complement a compositional idea which was already in existence. In the case of Justin, the tool was mostly used as an "effect", to increase the level of immersion produced by certain sounds in a narrative sound piece. Regarding my piece, its departure point was the use of sketches which were created when exploring the possibilities of *ImmLib* and the piece relied heavily on the perceptual effects created by the library to an extent that a stereo version might present an experience so different from what was intended as to be senseless. *ImmLib* was an integral part of the compositional process, although the piece did not intend to be a "showcase" of what the tool can do, it was hopped that piece would stand on its own, without any knowledge of how it was produced. The way Pablo used the tool and planned to create his work was to first experiment with material in conjunction with the parameter fields, determining what worked best and then use the output of the experimentation phase to build a composition. It might be relevant to mention at this point that in all the works presented, only the amplitude parameter was modulated. It seems that this approach is the departure point for using *ImmLib* and that modulating

[35]None of the composers had access to it.

other parameters requires perhaps more experience with the library.

Both Pablo and Michael answered in the survey that the tool was “not at all similar” to other spatialization tools they had used in the past. On the other hand both reported that some of the ideas, such as the use of multiple decorrelated streams, were already part of their practice. Both had already used techniques such as Ambisonics to pan virtual sources and were not satisfied with the results. Also, neither showed much interest in the practice of sound diffusion. In both cases their approach to spatialization, before having contact with *ImmLib*, was based on assigning multiple decorrelated tracks to individual loudspeakers, it is therefore apparent that *ImmLib* was an extension of their previous approach to spatialization, and not necessarily a complete departure.

Chapter 6.

Conclusions

The research set out to explore new possibilities for spatialization in electroacoustic music. Specifically, it sought to determine if a specific technique could be developed to generate decorrelated bundles in such a way that spatial surface patterns are perceived and controllable. Also, whether such a technique can successfully be used for spatial composition of computer music works.

Some existing strategies for spatialization using decorrelated bundles identified in the literature and in current artistic practice[1] do create extended sources, a higher level of envelopment and sometimes spatial patterns, but they are either entirely specific to one sound work[2], or in the case of a software tool, are tied to one specific synthesis or processing technique[3], or do not offer detailed control of the generated spatial pattern. The research presented here advanced the field by developing a technique, namely parameter field spatialization, implemented as a software tool, which is entirely generic regarding sound synthesis and signal processing, offering precise control of spatial patterns. The study also clarified how the use of different sound synthesis and signal processing algorithms, generating different spectromorphologies, interacts with the spatialization, affecting the perceived spatial patterns, specially in terms of spatial attributes such as spatial merging/segregation, envelopment, locatedness and pattern definition.

Parameter field spatialization generates a decorrelated bundle[4] from a given sound

[1] See section §2.5.

[2] For instance van der Heide's *Pneumatic sound field* [85] and Lang and Zimoun's *Untitled Sound Objects* [83].

[3] For instance Wilson's spatial swarm granulation [13] and Kim Boyle's spectral spatialization [14].

[4] Whether the multiple signals composing the bundle are decorrelated or not will depend on the nature of the sound process as was detailed in section §5.3.

process definition and modulates parameters of the sound process in order to create the spatial surface patterns. The algorithm used to generate the control signals which modulate the parameter is based on a mathematical model of spatial surface patterns, that is, it assumes there is a link between an abstract mathematical pattern, as defined by a function with domain a (virtual) surface, and the auditory spatial pattern that is perceived by a listener. The use of a model is not predictive so much as constructive, in other words, it is more important to create new and interesting perceptions than it is to predict what will be perceived. Like any model, it is a simplification of the underlying phenomenon, and the link can be stronger, weaker or can completely break down depending on a number of factors such as the nature of the sound process or the number of virtual sources. The model assumes a non-trivial surface encompassing all the loudspeakers and describes abstract patterns in this surface through mathematical functions of time and surface coordinates (parameter fields). The control signals are generated from a parameter field by associating the surface with a set of uniformly distributed points and calculating the evolution of the function for each point of the sample set. This algorithm was implemented in *ImmLib*, a software library for the *SuperCollider* audio-synthesis environment. With the library it is possible to create both fixed timeline-based sound works and interactive instruments with the spatialization being defined by associating a parameter field with a parameter of a sound process.

6.1. Research questions

Taking into account all the work presented so far, I will now address the original research questions that motivated this study.

Can parameter field spatialization successfully create and precisely control spatial surface patterns? This study showed that under certain conditions, the auditory event created by an *ImmLib* event has extent which can encompass an entire sphere with the listener at the centre, completely surrounding the listener, and along its extent (width and height) it is possible to distinguish different features, that is, it is possible to identify a changing pattern. Based on the listening tests and sound works described in chapter 5 it is possible to conclude the following.

In theory the modulation of any parameter of a sound process using a parameter field can create a spatial surface pattern, although in practice amplitude was often used as it was more familiar and results could be obtained fairly quickly. Continuous harmonic sounds, such as the first group of sine waves in *Unexpected Criticality*, and the physical modelling sounds of *Extase 2, 3 & 4*, tend to create a less sharply-defined pattern, while stochastic signals such as those from simple noise generators seem to create the sharpest patterns. When the perceived pattern was sharply defined it was possible to distinguish spatial features of the pattern in an almost visual way, for instance, for geometric patterns it was possible to determine direction and width of stripes (valleys and troughs) and with "organic" patterns it was possible to perceive the boundary of the pattern and the way in which it evolved, expanded, spatially segregated and merged. Using a system with high loudspeaker density it was even possible to distinguish between two very similar patterns, a spherical wave and a plane wave, where the direction, frequency and spacing was the same, only the shape of the wavefront is different. Given that it was possible to resolve this amount of detail and that features such as direction, spacing or shape can be precisely controlled in *ImmLib*, it was shown that under certain circumstances it is indeed possible to have precise control over the patterns.

Parameter field spatialization was the only technique implemented in *ImmLib*; nevertheless, throughout this research another possibility was explored briefly, which showed promise. Section 3.5 showed that recurrence relations obtained using numerical methods for differential equations of physical systems can also be used to generate control signals for a parameter of a sound process in order to create spatial patterns. One specific example, the idealised Belousov–Zhabotinsky reaction, was implemented and evaluated. Although it was capable of generating complex spatial patterns and behaviours it was not as easy to control as with parameter fields. It was nevertheless shown that this is a promising avenue for further exploration.

How can spatial surface patterns be made into a compositional parameter in computer music? The first compositional decision regarding the use of parameter field spatialization in order to create spatial surface patterns concerns whether or not to use this technique on a piece, or whether to use it in all sound events or just a subset. Use of parameter field spatialization for electroacoustic composition will tend to create broad and enveloping sources. This in itself is an aesthetic choice which must be in accord with the intended spatial environment for the piece. Although no spatial sound reproduction technique is entirely neutral, parameter field spatialization can be considered less neutral than point-source based techniques, since point sources can be added to form more complex sources as is the case with this technique, and as such parameter field spatialization probably works better for certain types of electroacoustic music. In the works and sketches created so far the technique seems to work quite well with long duration sustained harmonic or stochastic textural sounds.

When working with parameter field spatialization the most important compositional choices are: selecting which events (possibly all) to apply the technique to, selecting the parameter field to apply, selecting the parameter of the sound process to apply it to, and finally choosing the values for the parameters of the parameter field. Different parameter fields and different values can create widely different behaviours and spatial patterns, therefore decisions taken in this domain are similar to decisions relating to timbre or rhythm.

Parameter fields were used in different ways in the works created with *ImmLib* documented in this study. One use of parameter fields was to determine where a sound would play, if on the whole surface, or parts of the surface, and to establish the spatial nature of the event, the pattern or spatial movement in the surface. If the auditory event permanently occupied the whole surface, then it had no border, and changing the parameter field or its settings changed the pattern and its movement. If the event did not occupy the whole surface, the pattern would have a boundary and the parameter field could also generate movement of a different type, from one location to another, as well as expansion, contraction, segregation, merging and other types of boundary movements. In some cases, usually events of shorter duration, the movement of the spatial pattern was coordinated with the attack of the event in order to create an expressive spatial gesture. Finally, in some cases, parameters of the parameter fields were themselves modulated with low frequency oscillators or breakpoint envelopes, in order to change the behaviour of the parameter field while the event was ongoing.

It was therefore demonstrated that the selection of parameter fields, and settings of those, is an integral part of the composition process when working with parameter field spatialization. The technique can be used to spatially enhance already composed sound events; or the sound and its spatialization can be composed together influencing each other. The latter approach seems to be the best way to use parameter field spatialization with parameters other than amplitude, and to explore the possibilities which are unique to this technique.

Does the aforementioned technique represent a novel approach to spatialization in electroacoustic music? In what way does it differentiate itself from other approaches?

Section 2.5 reviewed different approaches from various backgrounds making use of decorrelated bundles where in some cases spatial surface patterns were created. It was shown that the technique presented here constitutes a new approach to spatialization with decorrelated bundles. Its novelty can be summarized in two key points. First, the technique is entirely independent from the synthesis technique used, since parameter fields can be applied to any parameter of any sound processes. This gives total freedom to the composer to select which synthesis or signal processing techniques to use. Second, the technique uses mathematical functions defined on a surface evaluated on an FRP system for the creation of spatial surface patterns. This approach allows for very precise definition and control of the patterns.

Does the use of parameter fields bring advantages over direct use of decorrelated bundles? The direct use of decorrelated bundles, where each signal is totally independent[5] can easily create very enveloping extended sources, nevertheless there is little control of the spatial pattern created. Furthermore, the type of patterns created with parameter field spatialization are more dynamic, possibly helping to maintain interest in the sound event, as well as allowing for predictable (both deterministic and stochastic) movements across the pattern. It was also demonstrated that the use of this technique allows for sharper patterns than would otherwise be possible. It should also be noted that *ImmLib* greatly simplifies the creation of decorrelated bundles using its automatic parallel sound-process-expansion mechanism.

[5]Being independent is stronger than being decorrelated, it means there is also no high-level connection between features of each signal.

Is it possible to find strategies that are particularly effective[6] in terms of combining specific types of sound materials with specific uses of the technique ? In chapter 5 it was established that different sound materials and synthesis techniques generate different spatial sound percepts. Synthesis definitions based around stochastic generators such as white-noise generators seem to create the sharpest patterns, although impulse generators also generate well-defined patterns. It seems that sources with the most envelopment and the sharpest patterns were also the ones where segregation of auditory streams did not happen, instead a single auditory event was created extending to the entire surface. As is to be expected, higher density of loudspeakers also creates sharper patterns, as was shown when comparing the differences between the Sonic Lab system and the rectangular grid system. It was demonstrated that with sound processes presenting significant timbral differences along the surface, when the amplitude is modulated with a parameter field, timbral changes will be associated with the movement of the sound through the surface. This was the case, for instance, with the feedback events in *Unexpected Criticality.* Applying a parameter field to decorrelated signals generated from a mono signal using a decorrelation technique based on phase randomization [11] creates very vague spatial surface patterns and lower envelopment than with stochastic ugens, as was demonstrated by listening tests and the use of *ImmLib* in the *Sonic Arts Theatre Lab.* Therefore it appears that parameter field spatialization is not particularly effective for faithful sound reproduction.

6.2. Contributions

At the start of this thesis I defined the main contributions of this research to knowledge. I will now review them summarizing what was achieved with this study:

A model of spatial surface patterns constructed through the use of mathematical functions on differentiable surfaces together with a computer algorithm for spatialization based on this model. I detailed formulas for calculating geodesic distance and methods for generating homogeneously distributed sets of points in different surfaces. I presented an algorithm to generate control signals from a

[6]One would consider more effective those combinations that create patterns with well-defined shapes and movements or more vague but nevertheless enveloping and dynamic spatial sensations, and less effective those that cannot be distinguished from direct use of decorrelated bundles.

function whose domain is a surface, given a point sample set, with the aim of applying such signals to a parameter of a sound process in order to create a spatial surface pattern. This algorithm is generic and can be implemented in any capable sound-synthesis programming language.

A set of specific mathematically defined abstract spatial patterns, showcasing different spatial and temporal behaviours, which can be applied to parameters of sound processes. A set of parameter fields was developed using different mathematical functions. The functions used ranged from very simple (sine function, step function) to more complex (spherical harmonics). On some parameter fields FRP logic was used for switching between different mathematical functions over time (randomHills). The parameter fields developed during this study show different behaviours, some featuring geometric movements (wave2DSin, sphericalHarmonics) and others more organic movements (randomHills, moveHills).

A spatialization software library implementing the aforementioned algorithm. I introduced *ImmLib*, a software library for the *SuperCollider* audio synthesis environment, which can spatialize sound processes using two-dimensional grids of loudspeakers. This library presents a complete environment, with user-definable ugen graphs, extensive scripting capabilities, GUIs for most functionality and a 3D visualization tool for parameter fields.

Perceptual descriptions of a range of different outputs generated by the technique under different configurations. For each parameter field a perceptual description was given of the pattern created with different settings. This was usually done by applying the parameter field to the amplitude parameter of a white noise generator. The application of parameter fields to other sound processes such as granulation, impulse generator or additive synthesis was also analysed. I also presented a detailed perceptual description of the use of parameter field spatialization in publicly presented sound works created by different artists.

An exploration of possible uses of the tool for electroacoustic music and sound art through collaborations with artists. During this study I collaborated with three composers who used *ImmLib* for their own artistic practice, with the tool being adapted based on their feedback and suggestions, obtained through informal exchanges, surveys and interviews. In total three pieces were created and publicly presented making use of parameter field spatialization, including a composition of my own.

A development of strategies for using parameter field spatialization for artistic purposes in the context of computer music. I described how different sound materials, such as harmonic complex tones, impulses or broad-band noise, interact with different parameter fields, and different settings thereof, specifying when a vague or sharply-defined spatial surface pattern is created and when single or multiple auditory events appear. These principles can be taken into account when using parameter field spatialization in a composition, helping to identify what type of spatial percepts are possible with a given sound material, or alternatively, what type of sound material and parameter field should be used to create a specific spatial surface pattern.

6.3. Future research

The work developed here focused on loudspeaker arrays forming surfaces such as domes and rectangles, mostly for practical reasons. The model could be extended to the full three-dimensional space using three-dimensional loudspeaker arrays, for instance with loudspeakers equally spaced in all three dimensions in $n \times m \times l$ type grids, with the listeners either placed at the centre of the grid, or able to move amongst the loudspeakers. Switching from surfaces to volumes, functions defined on all euclidean space $\mathbb{R}^3$ would be used instead of functions defined on a surface[7]. Some of the parameter fields such as wave2DSin and spotlight could be reused, changing the distance function from the geodesic distance on the surface to the usual distance in $\mathbb{R}^3$, given by the Pythagorean theorem, although in any case, other parameter fields should be developed taking into account the added dimension. The construction of three-dimensional grids is obviously more complex but has already been shown to be possible [130].

It would be interesting to use *ImmLib* in a periphonic system with the same loudspeaker density as in the rectangle array presented in this study. This would unfortu-

[7] $f(x, y, z, t)$ instead of $f(u, v, t)$.

nately require hundreds of loudspeakers, nevertheless, as WFS systems have demonstrated, it is possible to manage such large numbers of loudspeakers. I can predict that the definition of spatial patterns in such a system would be quite impressive, while at same time being more immersive than a planar array.

Working with recurrence formulas obtained from differential equations instead of closed-form mathematical expressions was shown to be a promising avenue for further research. There are many differential equations in physics, chemistry or biology which give rise to quite interesting spatial patterns. It would be worthwhile to experiment with such systems in order to find configurations which continuously generate interesting new behaviour while at the same time being stable and manageable.

A perceptual study of the use of parameter field spatialization could deepen the understanding of the type of auditory events created with this technique. In particular, a systematic study with a higher number of subjects, both in a controlled setting and with sound works produced by artists, could help illuminate some of the psychoacoustic phenomena described in this study as well as some of the implications for electroacoustic composition. Also, if the user base of *ImmLib* increases, it would be interesting to study more extensively how the tool is being used in the creation of electroacoustic works, drawing from a wider range of compositional approaches.

As the number of loudspeakers used in sound systems of music and sound art research institutes continues to increase, and given the current interest in enveloping extended sources, techniques such as parameter field spatialization will become increasingly relevant.

Appendix

Appendix A.

Mathematical notation

Although contemporary mathematical notation is fairly standardized, for the benefit of the reader, this appendix will briefly review some key concepts and notation necessary to understand the mathematical expressions present in this work.

Most of contemporary mathematics is based on a relatively small number of axioms[1] or first principles, the axioms of propositional calculus, the axioms of predicate calculus and the axioms of Zermelo-Fraenkel set theory (ZFC)[2]. Propositional calculus is concerned with statements that can be true or false and logical combinations of such statements. Propositions are built using connectives: $\varphi \rightarrow \psi$ is read as "φ implies ψ", $\varphi \leftrightarrow \psi$ is read as "φ is equivalent to ψ", $\varphi \wedge \psi$ is read as "φ and ψ", $\varphi \vee \psi$ is read as "φ or ψ" and $\neg\varphi$ is read as "not φ".

Predicate calculus allows the construction of more refined statements by introducing variables (x, y, etc.) and quantifiers ($\forall$, $\exists$). $\forall x$ is read as "for all x", $\exists x$ is read as "there exists an x such that".

Set theory deals with whether an object belongs to another object, with membership notated using $\in$. Almost any conceivable mathematical object can be encoded in set theory, therefore in ZFC objects such as functions, numbers or lines are all sets. $x \in A$ is read as "x is a member of A". $A \subset B$ is read as "A is contained in B", $A \supset B$ is read as "A contains B" and $\emptyset$ is the empty set. A set can be notated explicitly by listing its members: $\{a, b, c\}$, $\{0, 1, ..., 10\}$, or $\{0, 1, 2, ...\}$. Set-builder notation describes a set by stating its properties: $\{x|\Phi(x)\}$ or $\{x : \Phi(x)\}$ where Φ is a predicate, is the

[1] Some proof systems require only 17 axioms to encode almost all of contemporary mathematics [131].

[2] In the field studying the foundation of mathematics there are many different set theory systems, with different properties but a variant of Zermelo-Fraenkel set theory known as ZFC is considered the standard choice.

set of xs such that $\Phi(x)$ is true. The set of real numbers between zero and two is $\{x|x \in \mathbb{R} \wedge x \geq 0 \wedge x \leq 2\}$. Set-builder notation is sometimes extended to allow setting the domain of the variable on the left side using the equivalence $\{x \in A|\Phi(x)\} \leftrightarrow \{x|x \in A \wedge \Phi(x)\}$. The previous example can be rewritten $\{x \in \mathbb{R}|x \geq 0 \wedge x \leq 2\}$. Another extension is the use of terms on the left side: $\{T(x, y, z)|\Phi(x, y, z)\} \leftrightarrow \{w|\exists x \exists y \exists z\, w = T(x, y, z) \wedge \Phi(x, y, z)\}$. For instance the set of squares of real numbers is $\{x^2|x \in \mathbb{R}\}$.

The product of sets A and B is $A \times B = \{(a, b)|a \in A \wedge b \in B\}$, it is the set composed of all pairs with first element in A and second element in B. The Euclidean three-dimensional space can be defined as the product $\mathbb{R} \times \mathbb{R} \times \mathbb{R} = \mathbb{R}^3 = \{(x, y, z)|x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in \mathbb{R}\}$, the set of triples of real numbers. Points in two or three-dimensional Euclidean space, $\mathbb{R}^2$ and $\mathbb{R}^3$ respectively, are usually notated either with a lower-case Latin letter such as p, or indicating the actual triple, for instance (x, y, z). Points in three-dimensional space are usually notated using the letters x, y and z for the coordinates. Points in two-dimensional space, when working with surfaces, usually use letters u and v for coordinates, such as in (u, v). When using points and scalars (real numbers) together in the same expression, points use a different font, for instance $a\mathbf{w}$ is read as "number a multiplying vector $\mathbf{w}$". The interval of real numbers between a and b, a and b inclusive, is $[a, b] = \{x|x \leq b \wedge x \geq a\}$.

Functions typically use lower-case Latin letters, $f : A \to B$ has domain A and codomain B, and for every $x \in A$, $f(x)$ is in B. The set of all $f(x)$ for x in A is the image of f, sometimes notated as $f(A) \subset B$. Technically a function f is a subset of the Cartesian product of its domain and codomain, $f \subset A \times B$ and $(x, y) \in f \leftrightarrow f(x) = y$. f applied to x is notated $f(x)$, with multiple variables $f((x, y, z))$ is abbreviated to $f(x, y, z)$. The composition of f and g is $f \circ g$ and $f \circ g(x) = f(g(x))$. A formula for a function is notated $x \mapsto x^2$ and is read as "x maps to x squared", which is equivalent to $f(x) = x^2$. The function domain, codomain and formula can be notated simultaneously:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto x^2.$$

Appendix B.

Derivation of geodesic length for *ImmLib* surfaces

Sphere

The formula for the length of a geodesic on the unit sphere can be obtained easily using basic spherical trigonometry, in particular the spherical law of cosines. If $\triangle ABC$ is a spherical triangle on the unit sphere, A, B, C are its vertices, a, b, c are the lengths of the the sides and $\angle C$ the interior angle at C then

$$\cos(c) = \cos(a)\cos(b) + \sin(a)\sin(b)\cos(\angle C).$$

Given two points P and Q on the surface with coordinates (θ_p, ϕ_p) and (θ_q, ϕ_q) in a spherical coordinate system using the polar angle, the geodesic connecting them is the segment of great circle passing at P and Q. Consider the spherical triangle $\triangle CPQ$ where C is the north pole with spherical coordinates $(0, 0)$. Its sides are

$$\begin{aligned} p &= \phi_q \\ q &= \phi_p \\ c &= \text{length of the arc of geodesic connecting } P \text{ to } Q \end{aligned}$$

and

$$\angle C = |\theta_p - \theta_q|,$$

therefore by the spherical law of cosines

$$\cos(c) = \cos(\phi_p)\cos(\phi_q) + \sin(\phi_p)\sin(\phi_q)\cos(\theta_p - \theta_q).$$

Considering that the arc of geodesic connecting P to Q is always equal or smaller then π then we can consider arccos as having principal value in $[0, \pi]$ and

$$c = \arccos(\cos(\phi_p)\cos(\phi_q) + \sin(\phi_p)\sin(\phi_q)\cos(\theta_p - \theta_q)).$$

The spherical coordinate system of ImmLib uses elevation instead of polar angle, the formula above is therefore translated to

$$\begin{aligned}
\phi'_p &= \frac{\pi}{2} - \phi_p \\
\phi'_q &= \frac{\pi}{2} - \phi_q \\
c &= \arccos(\sin(\phi'_p)\sin(\phi'_q) + \cos(\phi'_p)\cos(\phi'_q)\cos(\theta_p - \theta_q)).
\end{aligned}$$

Cylinder

The formula for the length of a geodesic on the cylinder was calculated by measuring the length of a parametrized curve known to be a geodesic using the *sagemath* software:

```
#formula for length of geodesic on cylinder
u1, v1, u2, v2, R,t = var('u1,v1,u2,v2,R,t')
assume(R > 0)

# parametrization of geodesic on cylinder
fs=(R*cos((u2-u1)*t+u1),R*sin((u2-u1)*t+u1),(v2-v1)*t+v1)

# derivative of fs
fsd=map(lambda x: derivative(x,t),fs)

# calculate integral
d(u1, v1, u2, v2)=integrate( sqrt( vector(fsd).dot_product(vector(fsd)) ), t, 0, 1)

# general formula
case0 = d(u1, v1, u2, v2).maxima_methods().rootscontract().simplify()
print("d(u1, v1, u2, v2):\n")
print( case0 )

# check simple case
case1=d(0, 0, 2*pi, 0).maxima_methods().rootscontract().simplify()
print("\nd(0, 0, 2*pi, 0):\n")
print( case1 )

#evaluation result:
#d(u1, v1, u2, v2):
```

```
#
#sqrt((u1^2 - 2*u1*u2 + u2^2)*R^2 + v1^2 - 2*v1*v2 + v2^2)*u1/(u1 - u2)
#- sqrt((u1^2 - 2*u1*u2 + u2^2)*R^2 + v1^2 - 2*v1*v2 + v2^2)*u2/(u1 - u2)
#
#d(0, 0, 2*pi, 0):
#
#2*sqrt(pi^2*R^2)
```

Appendix C.

ImmLib implementation details

C.1. `ParUChain` and resource management

`ParUChain` uses *UnitLib*'s load balancing algorithm to distribute the n synths of each unit, and the associated DSP load, amongst all the servers. It plays synth i of each unit on the same server, so that they can exchange audio, but synths i and j for $i \neq j$ possibly in different servers. This is feature is essential in order to play scores with considerable amounts of events.

Some care was necessary when dealing with buffers and soundfiles in `ParU`. When loading a sound file to a buffer only one buffer will be created for each server . Multiple synths, belonging to the same `ParU` and playing in the same server, will reuse the same buffer. When playing a `ParU` of size 16 on a system with 8 servers, 2 synths will run on every server but only 8 buffers will be created, one per server. This process is completely automatic and hidden from the user.

The case of playing a soundfile directly from disk must be dealt with differently. In *SuperCollider* to play a soundfile from disk using the `DiskIn` ugen, a small buffer is pre-filled with audio from the soundfile, and then during playback the audio is sent from the soundfile to the buffer continuously. This buffer must be unique to each synth and cannot be shared. Soundfile playback from disk using a `ParU` is achieved by passing a `ParArg([DiskSndFile(),...])` instead of a `DiskSndFile()` to the `soundFile` argument of the unit which will cause n independent buffers to be created for file reading. In either case it is possible to set the start position in the soundfile and the play rate to a different value for each stream.

C.2. Multiple servers and bus management

The management of the audio buses used by `ImmUChain`s to send audio to the panners is completely automatic and invisible to the user. It was implemented by creating the `UBus` class which tags each audio bus with a name and uses the usual *SuperCollider* bus allocation classes to allocate the audio bus at the moment that it is needed by a chain. When the first chain that needs the bus is started the bus is allocated using the *SuperCollider* allocator (`Bus.audio`), when subsequent chains also using the same bus are started they will just reuse the already allocated bus and when all chains that need a bus are finished the bus is automatically deallocated. When a `PSurface` is instantiated it is allocated a unique global index, this number is then used to create a set of n mono audio `UBus`es with tags `psurface_<surface index>_<point index>` for that surface. When instantiated, `ImmUChain` and `ImmUScore` fetch the audio bus tags from the `PSurface` and pass them on to the bus arguments of the chains.

Sending audio between synths via audio buses requires all synths to be on the same server. In order to satisfy this constraint all synths from `ImmUChain`s sending audio to the same panner synth must be running on the same server. This was dealt with by adapting the `UGroup` class from *UnitLib*. `UGroup` allows specifying that two `UChain`s must run inside a specific *SuperCollider* group. When a chain does not send audio to another chain it is not necessary to specify a group for playback, it will be automatically started on the server with the lowest CPU usage, but if the chain must send audio to another chain then it is necessary to use an `UGroup` to guarantee that the two chains run on the same server and to control their order of execution, for example, to place one chain after another in the node tree. A `UGroup` has a tag which `UChain`s use to refer to it and the corresponding *SuperCollider* group is allocated and deallocated automatically as needed. At instantiation time each surface creates a set of n `UGroup`s with tags `immGroup<surface index>`. This array of `UGroup`s is wrapped in a `ParArg` and passed to the `ImmUChains` which start synth i using *SuperCollider* group i obtained from `UGroup` i of the array. This algorithm distributes the CPU load relative to one `ImmUChain` evenly amongst all servers and therefore all cores of the computer.

C.3. Increasing *scsynth*'s message buffers size

This patch was used to increase the number of messages that *scsynth* can send in a control block. This was done to ensure that all notification messages relative to terminating synths are sent.

```
---
 include/server/SC_HiddenWorld.h | 8 ++++----
 1 file changed, 4 insertions(+), 4 deletions(-)

diff --git a/include/server/SC_HiddenWorld.h b/include/server/
   SC_HiddenWorld.h
index 4988b9e..853c97d 100644
--- a/include/server/SC_HiddenWorld.h
+++ b/include/server/SC_HiddenWorld.h
@@ -81,10 +81,10 @@ struct DeleteGraphDefMsg {
 };


-typedef MsgFifoNoFree<TriggerMsg, 1024> TriggersFifo;
-typedef MsgFifoNoFree<NodeReplyMsg, 1024> NodeReplyFifo;
-typedef MsgFifoNoFree<NodeEndMsg, 1024> NodeEndsFifo;
-typedef MsgFifoNoFree<DeleteGraphDefMsg, 512>
   DeleteGraphDefsFifo;
+typedef MsgFifoNoFree<TriggerMsg, 4*1024> TriggersFifo;
+typedef MsgFifoNoFree<NodeReplyMsg, 4*1024> NodeReplyFifo;
+typedef MsgFifoNoFree<NodeEndMsg, 4*1024> NodeEndsFifo;
+typedef MsgFifoNoFree<DeleteGraphDefMsg, 4*512>
   DeleteGraphDefsFifo;
 typedef HashTable<struct GraphDef, Malloc> GrafDefTable;

 struct HiddenWorld
--
```

C.4. Surface independent `ImmDefs`

An `ImmDef` is instantiated using `ImmMod(\defName, [\slider1, val1, ...])` and, like `Udef`, can be defined once and used in multiple instances of `ImmMod` simultaneously. The `ImmDef` does not store the surface to be used, this information only becomes available at instantiation time when `ImmUChain` sets the global variable `ImmDef.currentSurface` to the surface that was passed to it, so that the parameter fields can access the distance function and coordinate chart domain ranges of the surface (`distFunc`, `rangeU`, `rangeV`) .

`PFields` were implemented in such a way that the arguments `u` and `v` of the corresponding function are passed values in the domain of the coordinate chart, which is different for each surface. On the other hand, the `dist` function supplied by `PSurface` expects values also in the domain of the coordinate chart, therefore parameter fields which pass the `u` anv `v` variable to the distance function are almost independent of the surface used, except that usually the parameter field is such that the distance function compares the point `(u,v)` from the arguments of the `PField` function to a point given by the parameters `u0` and `v0` of the parameter field (`dist(u,v,u0,v0)`). It is therefore possible to write `ImmDef`s that are independent of the surface being used. To do so, the `u0` and `v0` parameters of the `ImmDef` are specified as being in $[0, 1] \times [0, 1]$ and are converted to $[u_a, u_b] \times [v_a, v_b]$ in the FRP network, so as to be in the range expected by `dist`. Since this computation only happens when the sliders for the reference point are moved, it is negligible CPU wise.

Appendix D.

SuperCollider code and ugen graphs

D.1. FRP GUI example

In this example a window with two number boxes is created. The values input into the first number box are continuously added and printed in the second number box.

```
//sum with last value
(
//Setup GUI
a = NumberBox();
b = NumberBox();
w = Window(bounds:Rect(200,200,200,100))
.layout_( HLayout(
  StaticText().string_("In:"), a,
  StaticText().string_("Out:"), b
) );
w.front;
)

//start Event Network
(
ENdef(\sum,{
  var sumSig, sum, sumSig2;

  //event stream with values from
  //first number box
  var nb1S = a.enInES;

  //recursive relation
  sum = { |a| a + sumSig.now } <%> nb1S;
  sumSig = sum.hold(0);

  //send values of sumSig to the
  //second number box
  b.enSinkValue( sumSig );

}).start;
)
```

```
//now type numbers into the first number box

//stop event network
ENDef(\sum).stop;
//close window
w.close;
```

D.2. FRP and UnitLib example

```
(
var ktl = MKtl('nnkn0');
//if the controller is not available create a virtual GUI.
//ktl.gui;
PSurfaceDef(\sphere, PSphere(10));
f = { |t, loSliderSig, hiSliderSig|

  var sig1 = OSCFunc.enIn('/freq', initialValue:[\freq,0.0]);

  //get first slider of first page of Korg Nanocontroller
  var sig2 = ktl.elements[\sl][0][0].enIn;

  //extract first float from OCS message.
  var freqSig = sig1.collect(_[1]);

  var ampSig = { |t, x, lo, hi|
    sin(2pi*t).linlin(-1.0, 1.0, lo, hi) * x
  }.lift.(t, sig2, loSliderSig, hiSliderSig);

  ( freq: USpecArg(freqSig.collect(_.dup(10))) , amp: USpecArg
     (ampSig.collect(_.dup(10))) )
};
x = ImmDef(\mod1, f, 0.1, [\lo, [0,1], \hi, [0,1] ]);
y = ImmUChain(\sphere,
  [\sine, [\freq, 400], ImmMod(\mod1, [\lo, 0.3, \hi, 0.7]) ],
  \stereoOutput
);
y.prepareAndStart
)

y.gui
```

D.3. Simultaneous ImmUScores with different surfaces

```
(
PSurfaceDef(\sphere1, PSphere(20) );
PSurfaceDef(\sphere2, PSphere(10) );
~chain1 = ImmUChain(\sphere1,
  [\immWhiteNoise, [\globalAmp, 0.3], ImmMod(\wave2DSin, [\u0,
       0.25, \l,1, \freq, 0.2])],
  [\lowPass, [\freq, 500]]
);
~score1 = ImmUScore(\sphere1, 0,0,~chain1);
~chain2 = ImmUChain(\sphere2,
```

```
  [\immWhiteNoise, [\globalAmp, 0.3], ImmMod(\wave2DSin, [\u0,
      0.5, \l,0.5, \freq, 0.1])],
  [\highPass, [\freq, 5000]]
);
~score2 = ImmUScore(\sphere2, 0,1,~chain2);
~mainScore = UScore(~score1, ~score2);
~mainScore.gui
)
```

D.4. Example of *ImmLib* code

The code below shows a complete example of typical *ImmLib* usage: defining a surface, associating one parameter field with one argument of a unit and creating one score with one chain containing the unit.

```
(
Udef(\mynoise, {
  var in = WhiteNoise.ar * 0.5 * \amp.kr(1, lag:0.3)
    * \globalAmp.ukr(0.1, \amp);
  var eq = LPF.ar(in, \freq.kr(1000) );
  UOut.ar(0, eq )
})
.setSpec(\amp, [0.0,1.0,\amp])
.setSpec(\freq, \freq);

//define one ImmDef with one pfield connected to the amp
    parameter
ImmDef(\mywave, { |t, u0, v0, l, freq, plot|

  var pf1 = PField.wave2DSin(t, u0, v0, l, freq, plot);

  ( amp: USpecArg( pf1 ) )

}, 0.1, [
    //define parameters of the ImmDef
  \u0, ControlSpec(0,2pi),
  \v0, [-pi,pi],
  \l, ControlSpec(0.0, 2.0,default:1),
  \freq, ControlSpec(1/10,2,default:0.5),
  \plot, ControlSpec(0,1,step:1,default:0)
] );

//define a spherical surface with 20 points
PSurfaceDef(\surface1, PSphere(20) );
)

(
//instantiate an ImmMod using the ImmDef defined above
//set values of pfield parameters
~mod = ImmMod(\mywave,
  [\u0, pi/2, \v0, 0, \l, 0.1, \freq, 0.5]);

~chain = ImmUChain(\surface1,
  [\mynoise, [\globalAmp, 0.1, \freq, 5000], ~mod] );
```

```
~score = ImmUScore(\surface1, ~chain);

~score.gui
)
```

D.5. PField rotation

The rotation methods of `PField` return a `PField` whose function is transformed such that the rotation angles are added as arguments of the function after the time argument and before the parameter arguments.

```
//evaluating original pfield
pf1.(timeSig, p1Sig, p2Sig)

//evaluating rotated pfield
pf1.rotate3D.(timeSig, thetaSig, phiSig, psiSig, p1Sig, p2Sig)

//with modulated rotation angles
pf1.rotate3D.(timeSig, t*0.2, t*0.3, t*0.4, p1Sig, p2Sig)
```

D.6. UMap versions of parameter fields

```
(
UMapDef(\spotlight, {

    var u = \u.ukr(0,0,2pi);
    var v = \v.ukr(0,-pi/2,pi/2);
    var u2 = \u0.ukr(0,0,2pi);
    var v2 = \v0.ukr(0,-pi/2,pi/2);
    var c = \c.kr(0.0);
    var maxDist = pi;
    var bump = { |x| 2**((1-x.squared).reciprocal.neg)*2 };
    var distFunc = { |theta1, phi1, theta2, phi2|
        acos( cos(phi1)*cos(phi2)*cos(theta1-theta2) + (sin(
            phi1)*sin(phi2) ) )
    };
    var dist = distFunc.(u, v, u2, v2);
    var cpi = c*maxDist;
    var out = if( dist < cpi,
        DC.kr(1.0),
        DC.kr(-1.0)
    );
    UMapOut.kr( Lag.kr(out, \lag.ukr(0,0.0,3.0) ).range(*\
       range.kr([0.0,1.0])) )
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

UMapDef(\expandContract, {
```

```
    var u = \u.ukr(0,0,2pi);
    var v = \v.ukr(0,-pi/2,pi/2);
    var u2 = \u0.ukr(0,0,2pi);
    var v2 = \v0.ukr(0,-pi/2,pi/2);
    var c = \c.kr(0.0);
    var maxDist = pi;
    var bump = { |x| 2**((1-x.squared).reciprocal.neg)*2 };
    var distFunc = { |theta1, phi1, theta2, phi2|
        acos( cos(phi1)*cos(phi2)*cos(theta1-theta2) + (sin(
           phi1)*sin(phi2) ) )
    };
    var f = { |u,v,u2,v2, c|
        var dist = distFunc.(u, v, u2, v2);
        var cpi = c*maxDist;
        if( dist < cpi,
            DC.kr(1.0),
            DC.kr(-1.0)
        )
    };
    var out = if(c < 0.5,
        f.(u, v, u2, v2, c*2),
        f.(u, v, u2+pi, v2.neg, 1 - (2*(c-0.5)) )
    );
    UMapOut.kr( Lag.kr(out, \lag.ukr(0,0.0,3.0) ).range(*\
       range.kr([0.0,1.0])) )
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

UMapDef(\gradient, {

    var u = \u.ukrArgSpec(0, 0, [0,2pi].asSpec, true);
    var v = \v.ukrArgSpec(0, 0, [0,2pi].asSpec, true);
    var u2 = \u0.ukr(0,0,2pi);
    var v2 = \v0.ukr(0,0, 2pi);
    var maxDist = pi;
    var distFunc = { |theta1, phi1, theta2, phi2|
        acos( cos(phi1)*cos(phi2)*cos(theta1-theta2) + (sin(
           phi1)*sin(phi2) ) )
    };
    var x = distFunc.(u, v, u2, v2)/maxDist;
    //let's try, always output [-1,1] and use range to convert
    var range = \range.kr([0.0,1.0]);
    UMapOut.kr( Lag.kr( x, \lag.ukr(0.0, 0.0, 2.0) ).linlin
       (0.0,1.0,range[0],range[1]) )
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

f = { |g|
    var u = \u.ukrArgSpec(0, 0, [0,2pi].asSpec, true);
    var v = \v.ukrArgSpec(0, 0, [-pi/2,pi/2].asSpec, true);
    var wideness = \wideness.ukr(0.0,0,1.0);
    var out = g.(u,v,wideness);
    var out2 = out.linlin(0.0,1.0,-1.0,1.0).range(*\range.kr
       ([0.0,1.0]));
    UMapOut.kr( out2 )
```

```
};

UMapDef(\barU, {
    f.( { |u,v, wideness| u < (wideness * 2pi) } )
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

UMapDef(\barV, {
    f.( { |u,v, wideness|  v.abs < (wideness * pi/2) } )
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

f = { |g, b|

    var u = \u.ukrArgSpec(0, 0, [0,2pi].asSpec, true);
    var v = \v.ukrArgSpec(0, 0, [-pi/2,pi/2].asSpec, true);
    var u2 = \u0.ukr(0,0,2pi);
    var v2 = \v0.ukr(0,-pi/2,pi/2);
    var l = \l.ukr(0.0,0,2.0);
    var maxDist = pi;
    var distFunc = { |theta1, phi1, theta2, phi2|
        acos( cos(phi1)*cos(phi2)*cos(theta1-theta2) + (sin(
            phi1)*sin(phi2) ) )
    };
    var dist = distFunc.(u, v, u2, v2);
    var freq = \freq.ukr(1/20,1/20,5,\exp);
    var phase = (l*distFunc.(u,v,u2,v2)/maxDist)*b + \phase.
       ukr(0,0,b);
    var out = g.( freq, phase);
    UMapOut.kr( out )

};

UMapDef(\wave2DSin, {
    f.({ |freq, phase, range| SinOsc.kr(freq, phase).range(*\
       range.kr([0.0,1.0])) }, 2pi);
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

UMapDef(\wave2DSaw, {
    f.({ |freq, phase|
        DelayC.kr( LFSaw.kr(freq).range(*\range.kr([0.0,1.0]))
            .lag(\lag.ukr(0,0,1)), 1, 1/freq*phase )
     }, 1);
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );

UMapDef(\wave2DPulse, {
    f.({ |freq, phase|
        var a = LFPulse.kr(freq, 0, \width.ukr(0.5, \unipolar)
            )
        .range(*\range.kr([0.0,1.0]))
        .lag(\lag.ukr(0,0,1));
        DelayC.kr(a , 1, 1/freq*phase )
```

```
    }, 1)
})
.mappedArgs_( [ \range ] )
.category_( 'immlib' );
)
```

D.7. BZ reaction implementation

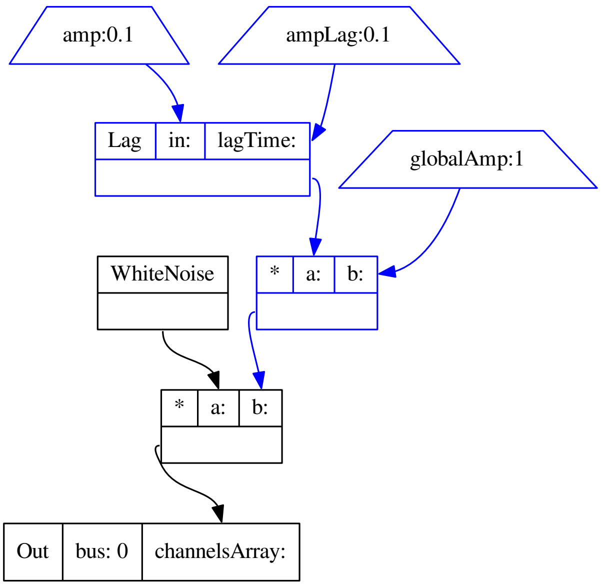
```
(
~randomizeBZR = {
 a = 2.collect{ m.collect{ rrand(0.0, 1.0) } };
 b = 2.collect{ m.collect{ rrand(0.0, 1.0) } };
 c = 2.collect{ m.collect{ rrand(0.0, 1.0) } };
}
)

(
Udef(\noise2, {
 UOut.ar(0,  Dust.ar(50) * \p1.kr(0.0)* 0.3  )
});
)

(
var indexes, p1=0,p2=1,c_a=0,c_b=0,c_c=0,  factor = 5.0;
m = 50;
q = ();

//30
q.m = m;
q.surface = PSphere(q.m);
PSurfaceDef(\bzr, q.surface );

//get 3 closest points
q.indexes = q.surface
.pointsRV3D
.collect({ |v|
  q.surface.pointsRV3D
 .collect{ |u,i| T(u,i) }
 .select{ |tup| tup.at1 != v}
 .collect{ |tup, i|

  T(tup.at1, tup.at2, tup.at1.dist(v) )

 }.sort{ |a,b|
  a.at3 < b.at3
 }
 .takeN(3)
 .collect(_.at2)
});

//one less lookup
indexes = q.indexes;

//start state
a = 2.collect{ m.collect{ rrand(0.0, 1.0) } };
b = 2.collect{ m.collect{ rrand(0.0, 1.0) } };
```

```
c = 2.collect{ m.collect{ rrand(0.0, 1.0) } };

//with interpolation
~bzrCount = 0;
~bzrLast = a[0];
~bzrNext = a[0];
~bzr =
 { |t, factor, steps, ra, rb|
     steps = steps.asInteger;
  if( ~bzrCount == (steps-1) ) {
   ~bzrLast = a[p1];
   m.do{ |i|
    i;
    c_a = 0.0;
    c_b = 0.0;
    c_c = 0.0;
    indexes[i].add(i).do{ |k|
     c_a = c_a + a[p1][k];
     c_b = c_b + b[p1][k];
     c_c = c_c + c[p1][k];
    };
    c_a = c_a / factor;
    c_b = c_b / factor;
    c_c = c_c / factor;
    a[p2][i] = ( c_a + (c_a * (c_b - c_c)) ).clip(0.0,1.0)+
       rrand(0.001,0.1);
    b[p2][i] = ( c_b + (c_b * (c_c - c_a)) ).clip(0.0,1.0)+
       rrand(0.001,0.1);
    c[p2][i] = ( c_c + (c_c * (c_a - c_b)) ).clip(0.0,1.0)+
       rrand(0.001,0.1);
   };


   if (p1 == 0) {
    p1 = 1; p2 = 0;
   } {
    p1 = 0; p2 = 1;
   };

   ~bzrNext = a[p1];
   ~bzrCount = 0;
   ~bzrLast.collect(_.linlin(ra,rb,0.0,1.0))
  } {
   ~bzrCount = (~bzrCount + 1).mod(steps);
   m.collect{ |i|
    ~bzrCount.linlin(0,steps-1,~bzrLast[i], ~bzrNext[i]).
       linlin(ra,rb,0.0,1.0)
   }
  }

 }.lift;


q[\f] = { |t, k, n, a, b|
 var but = Button();
 var w;
 var cm = CmdPeriod.add({ w.close });
```

```
 var freq = ~bzr.(t, k, n, a, b);

 k.enIn.linlin(0.0,1.0,1.0,7.0).enDebug("slider");

 PGridPlot( freq );
 PHemiPlot( freq );

 but.enIn.collect{ |v| IO(~randomizeBZR) }.enOut;

 w = Window().layout_(HLayout(but)).front;

 //freq.enDebug("freq");

 ( \p1: USpecArg( freq ) )

};

ImmDef(\bzr, q[\f], 0.1, [\k,[1.0,7.0], \n, [1,20], \a,\
   unipolar,\b,\unipolar]);

q.setting =
[ 'k', 6.4628099173554, 'n', 11.18044077135, 'a',
   0.20523415977961, 'b', 0.25619834710744 ];
//another slow changer
//[ 'k', 7.0, 'n', 11.18044077135, 'a', 0.19972451790634, 'b',
    0.24931129476584 ];
//almost always silence
//[ 'k', 7.0, 'n', 11.18044077135, 'a', 0.19972451790634, 'b',
    0.24931129476584 ]

q.chain1 = ImmUChain(\bzr,
 [\noise2, nil , ImmMod(\bzr,q.setting)]
 //[\tone1, nil , ImmMod(ImmDef(q[\f], q.surface, 0.1, [\k
    ,[1.0,7.0], \n, [1,20], \a,\unipolar,\b,\unipolar]),q.
    setting)]
);

q.score = ImmUScore(\bzr, q.chain1);

q.score.prepareAndStart;
q.chain1.gui

)
```

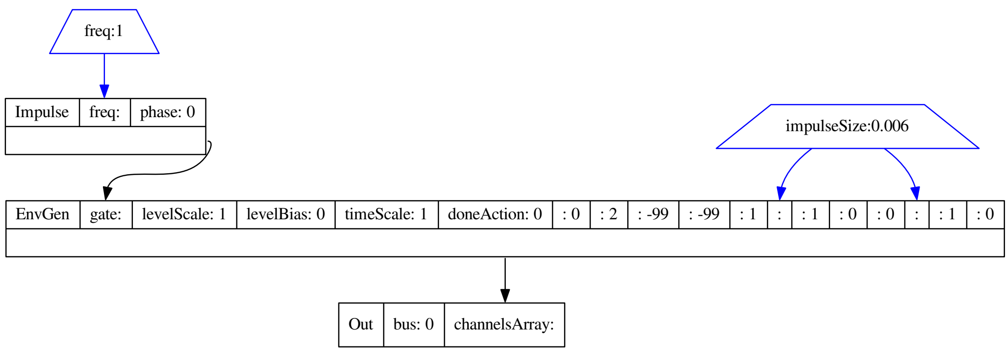
D.8. Perceptual tests `Udefs`

D.8.1. White noise




```
Udef(\whiteNoise, {

  var amp = \amp.kr(0.1, \ampLag.kr(0.1) );

  var gAmp = \globalAmp.ukr(1.0, \amp);

  var out = WhiteNoise.ar( amp * gAmp );

  UOut.ar(0,  out)
})
```

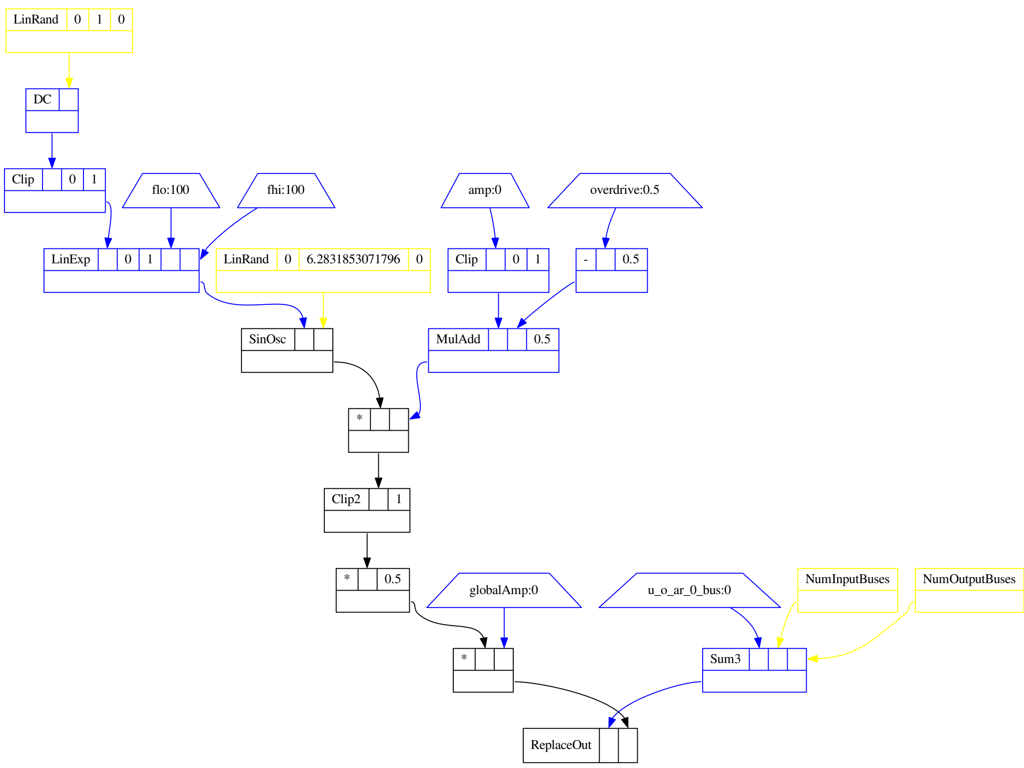
D.8.2. Periodic Impulse generator udef




```
(
Udef(\periodicImpulses,{
  var impulseSize = \impulseSize.ukr(
```

```
    default: 0.006,
    minval: 0.0001,
    maxval: 0.04,
    warp:\exp,
    lag: 0.1
  );
  var freq = \freq.ukr(
    default: 1.0,
    minval: 0.1,
    maxval: 50,
    warp: \exp,
    lag: 0.1
  );
  var trig = Impulse.ar( freq );
  var env = Env(
    levels: [0.0,1.0,0.0],
    times: [impulseSize, impulseSize]
  );
  var out = EnvGen.ar(envelope: env, gate: trig);
  UOut.ar(0, out * \globalAmp.ukr(0.0, \amp.asSpec))
})
)
```

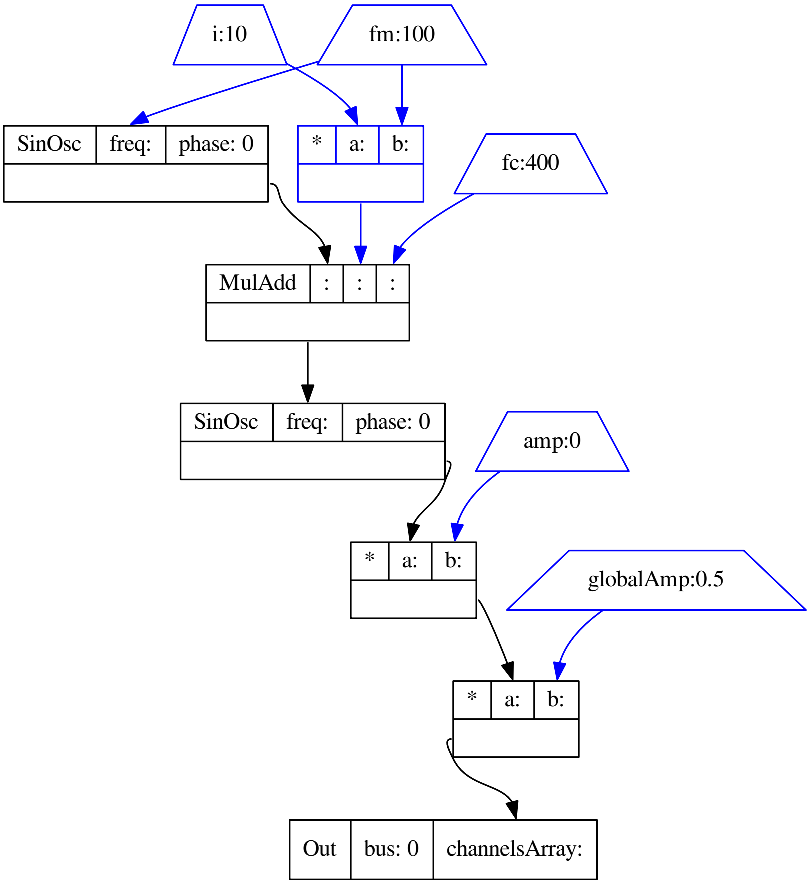
D.8.3. Distorted sine tones




```
Udef(\sineDist, {

  var lo = \flo.ukr(100,100,1000,\exp);
  var hi = \fhi.ukr(100,100,1000,\exp);
  var freq = DC.kr(LinRand(0.0,1.0)).linexp(0.0,1.0,lo,hi);
```

```
  var out = SinOsc.ar( freq, LinRand(0, 2*pi) );

  var amp = \amp.ukr(lag: 0.1);

  out = ( out * amp.linlin(0.0, 1.0, 0.5, \overdrive.ukr(0.5,
     0.5, 3.0, \exp) ) ).clip2 * 0.5;

  UOut.ar(0, out * \globalAmp.ukr(0.0, \amp.asSpec) )

})
```

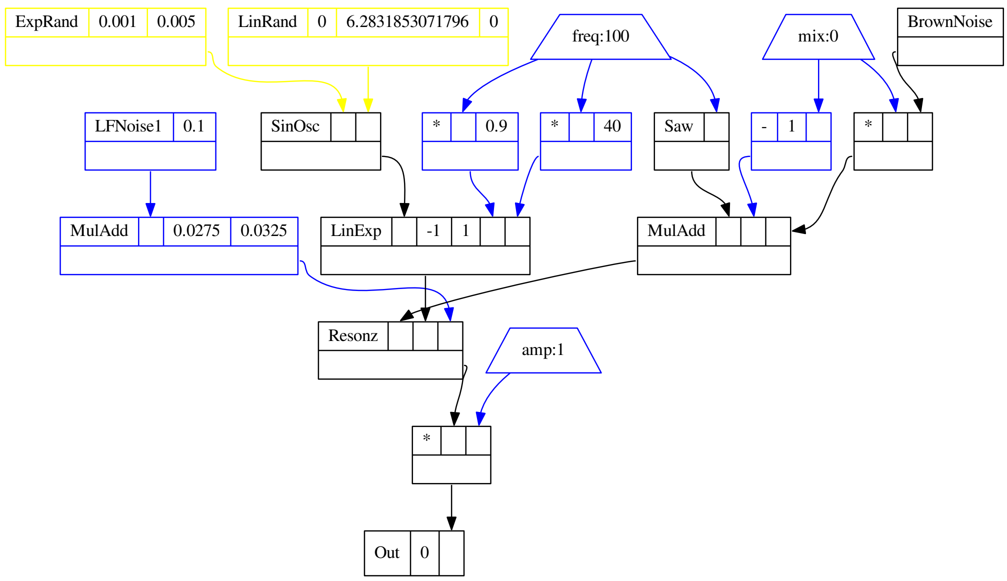
D.8.4. FM synthesis




```
Udef(\fm, {

  //index of modulation
  var i = \i.ukr(10, minval:0, maxval:50, lag:0.2);

  //modulation frequency
  var fm = \fm.ukr(100, minval:1, maxval:200, lag:0.2);

  var out = SinOsc.ar(
    \fc.ukr( 400, \freq, lag: 0.2) + SinOsc.ar( fm, mul: i*fm
       )
  );
```

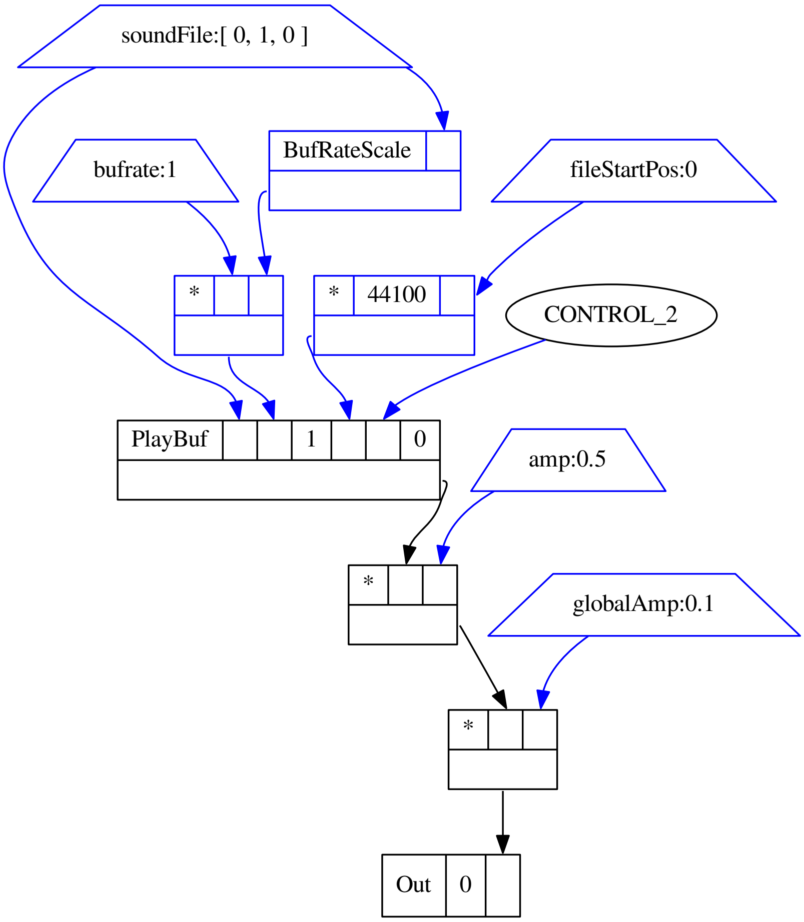
```
  var g = \globalAmp.ukr(0.0, \amp);

  UOut.ar(0, out * \amp.kr * g)
})
)
```

D.8.5. Weighted sum of noise generator with saw wave generator filtered by a parallel set of moving band-pass filters



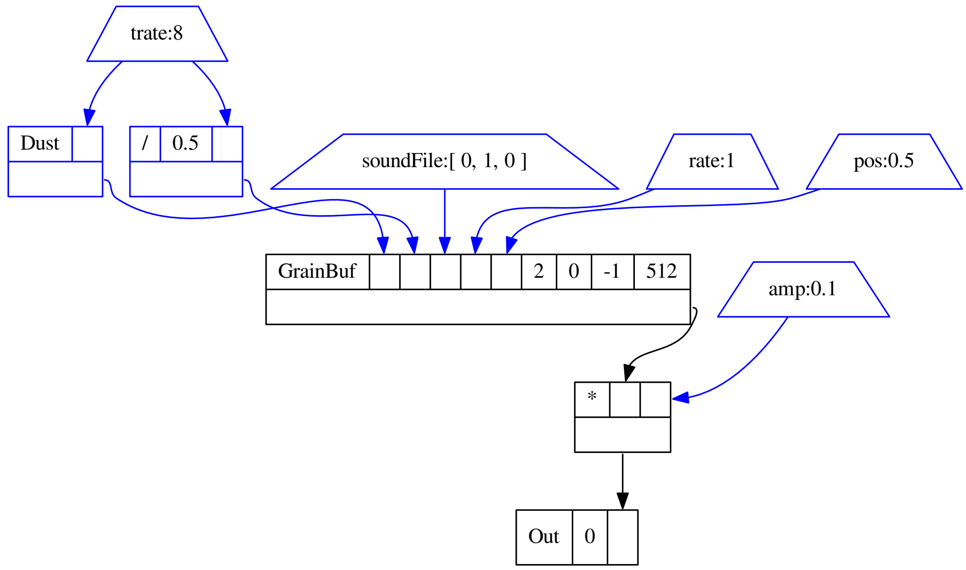

```
(
Udef(\noiseSawSweep, {
  var b = LFNoise1.kr(0.1).range(0.005, 0.06);

  var mix = \mix.ukr(0.0, \unipolar);

  var freq = \freq.kr(100);

  var source = ( Saw.ar(freq) * (1-mix)) + ( mix * BrownNoise.
     ar );

  var reson = Resonz.ar( source,
    freq: SinOsc.ar(
      ExpRand(0.001,0.005),
      LinRand(0.0, 2*pi)
    ).exprange(freq*0.9,freq*40),
    bwr: b
  );

  var out = reson  * \amp.kr(1.0) * \globalAmp.ukr(0.0, \amp.
     asSpec);

  UOut.ar(0, out )
})
)
```

D.8.6. Sound file playback




```
(
Udef(\filePlayback, {
  var sf = \soundFile.kr( [ 0, 1, 0 ] );
  var bufrate = \bufrate.ukr(1,1/4,4) * BufRateScale.kr(sf[0])
     ;
  var startPos = 44100 * \fileStartPos.ukr(0,0,60*60);
  var bufp = PlayBuf.ar( 1, sf[0], bufrate, 1, startPos, sf[2]
      );
  UOut.ar( 0, bufp * \amp.kr(0.5) * \globalAmp.ukr(0.0, \amp.
     asSpec) )
})
.setSpec(\soundFile, BufSndFileSpec() );
)
```

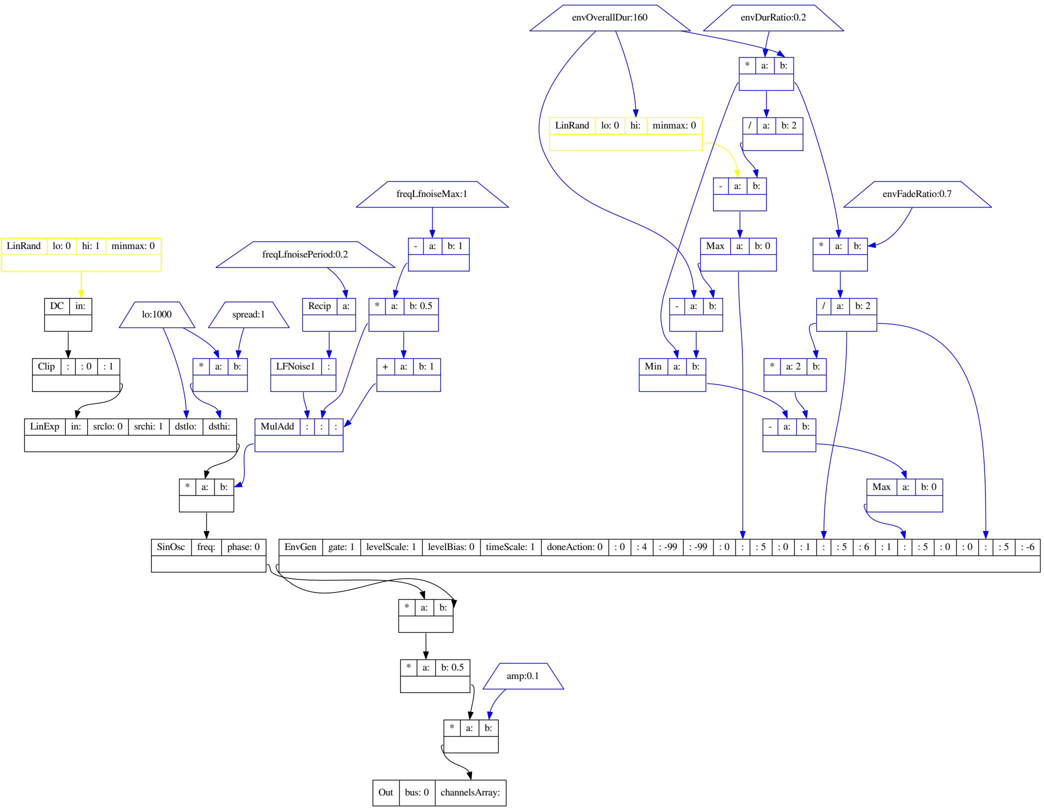
D.8.7. Granulation




```
(
Udef(\granulation, {

    var in = UIn.ar(0);
  var sndfile = \soundFile.kr([0,1,0]);
  var buf = sndfile.at(0);
  var trate, dur, clk, pos, out;
  trate = \trate.ukr(8,8,120);
  dur = 0.5 / trate;

  clk = Dust.kr( trate );

  pos = \pos.kr(0.5);

  out = GrainBuf.ar(1, clk, dur, buf, \rate.kr(1.0), pos);

  UOut.ar(0,  out * \amp.kr(0.1) * \globalAmp.ukr(0.0, \amp.
      asSpec))
})
.setSpec(\soundFile,  BufSndFileSpec() );
)
```

D.9. Unexpected Criticality `Udefs` and `ImmDefs`.

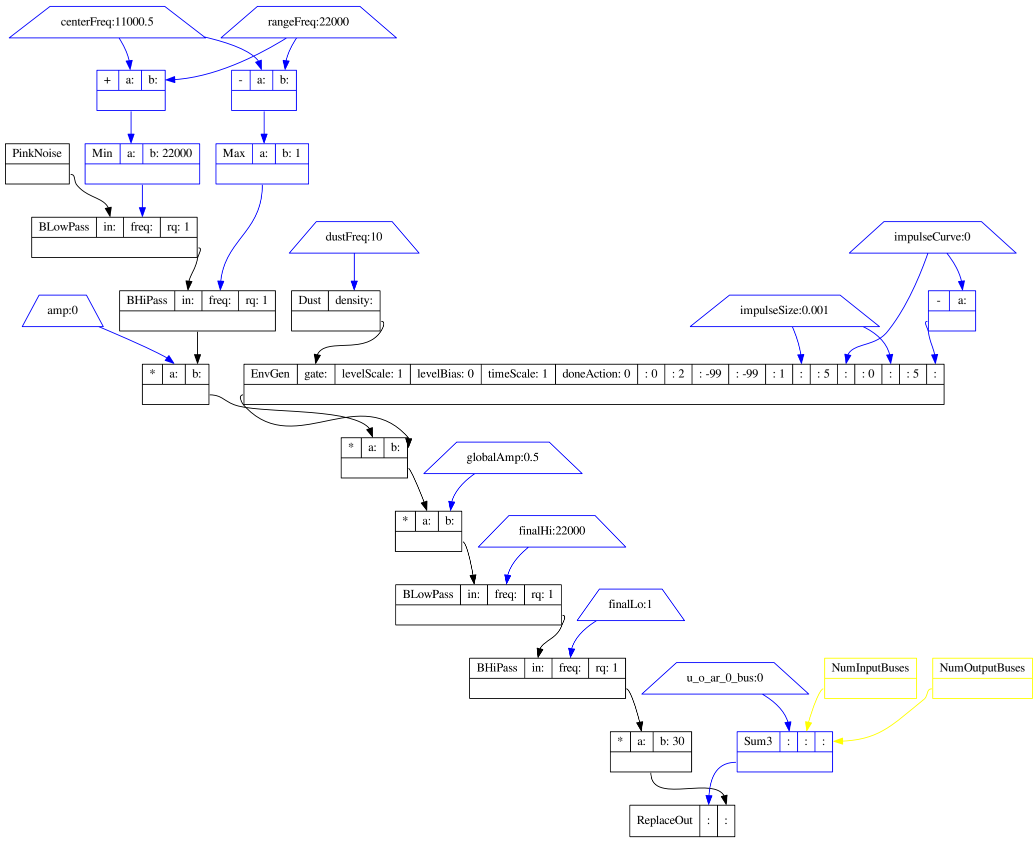
D.9.1. *Unexpected criticality* sine waves `Udef`




```
(
~hibuzzDefs = Namespace();

~hibuzzDefs.g = { |n=40| {
  var spec = [1000,16000,\exp].asSpec;
  var a = \lo.ukr(1000,spec);
  var b = \spread.ukr(1,[1,2,\exp].asSpec);

  var freqLfnoisePeriod = \freqLfnoisePeriod.ukr(1/5,[1/5,20].
     asSpec);
  var freqLfnoiseMax = \freqLfnoiseMax.ukr(1.0,[1.0,2.0,\exp].
     asSpec);

  var overallDur = \envOverallDur.ukr(160.0, [0, 10000] );
  var envFadeRatio = \envFadeRatio.ukr(0.7, \unipolar);
  var envDurRatio = \envDurRatio.ukr(0.2,\unipolar);

  var dur = envDurRatio * overallDur;
  var halfdur = dur / 2;
  var fade = (dur*envFadeRatio) / 2;

  n.collect{
    var middleTime = LinRand(0.0, overallDur);
    var startTime =  (middleTime-halfdur).max(0);
    var actualDur = dur.min(overallDur-startTime);
    var env = Env([0,0,1,1,0],[startTime,fade,(actualDur-(2*
       fade)).max(0),fade],[0,6,0,-6]).ar;
```

```
    var freq = DC.ar(LinRand(0.0,1.0)).linexp(0.0,1.0,a, a*b)
       *
    LFNoise1.kr(freqLfnoisePeriod.reciprocal).range(1.0,
       freqLfnoiseMax);
    SinOsc.ar( freq ) * env
  }.sum / n
}};

Udef(\mixedSinesEnv,{
  UOut.ar(0, ~hibuzzDefs.g.(15) * 0.5 * \amp.ukr(0.1,\amp) )
});

Udef(\mixedSinesLFPulse,{
  var n = 40;
  var spec = [1000,16000,\exp].asSpec;
  var amp = \amp.ukr(0.1,\amp);

  var a = \lo.ukr(1000,spec);
  var b = \spread.ukr(1,[1,2,\exp].asSpec);

  var lfnoisePeriod = \lfnoisePeriod.ukr(1/5,[1/5,20].asSpec);
  var lfnoiseMax = \lfnoiseMax.ukr(1.0,[1.0,2.0,\exp].asSpec);


  var sines = n.collect{
    var freq = DC.ar(LinRand(0.0,1.0)).linexp(0.0,1.0,a, a*b)
       *
    LFNoise1.kr(lfnoisePeriod.reciprocal).range(1.0,lfnoiseMax
       );
    SinOsc.ar( freq )
  }.sum / n;

  var lfpulseFreq = \lfpulseFreq.ukr(1, [1,60,\exp].asSpec);
  var out = sines * Lag.ar( LFPulse.ar( lfpulseFreq ),  0.003
     ); //base 0.001

  UOut.ar(0, out * amp* \globalAmp.ukr(0.5,\amp));
})
)
```

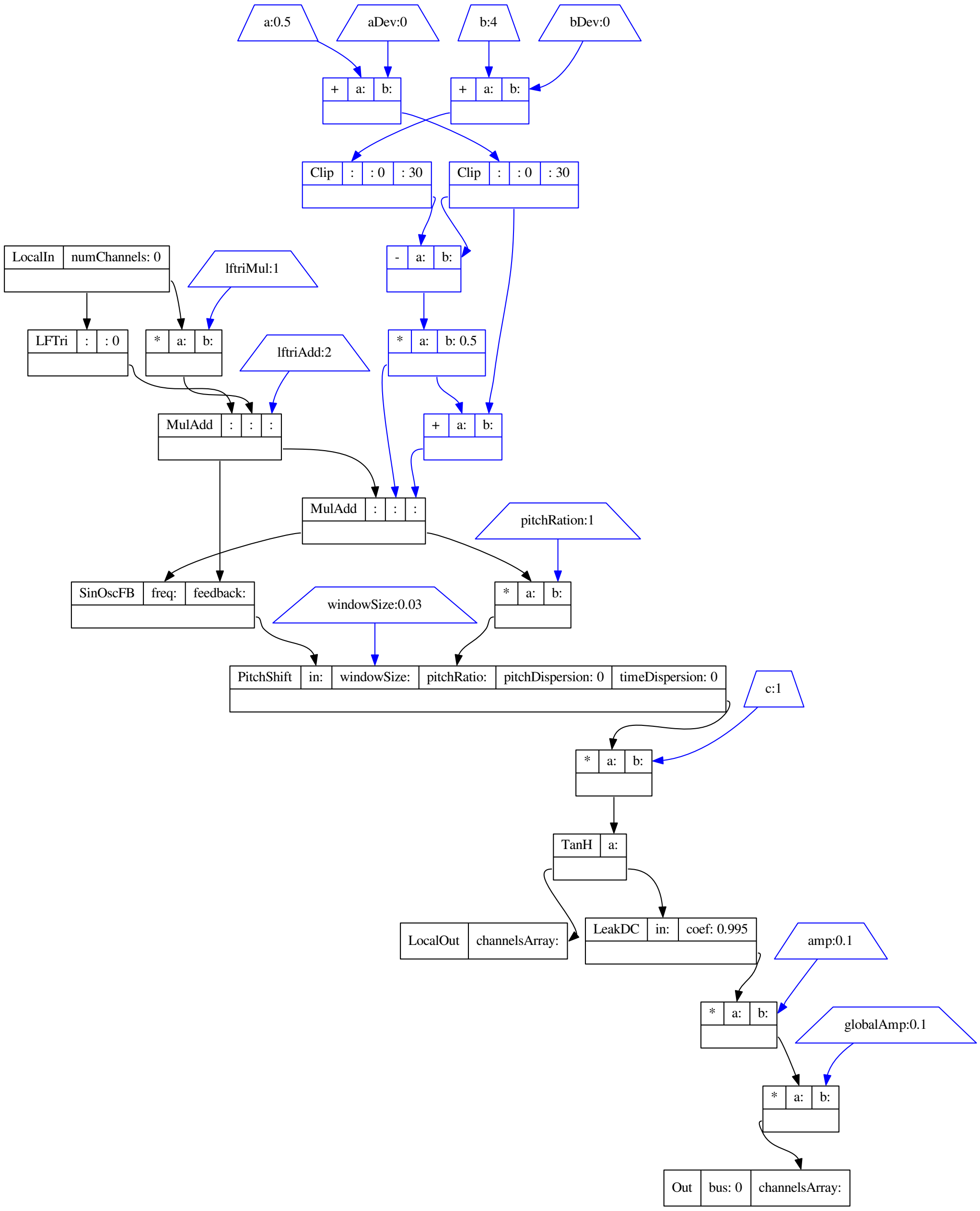
D.9.2. *Unexpected criticality* clicks `Udef`




```
(
q = Namespace();

q.impulse = {
  var impulseSize = \impulseSize.ukr(0.001, 0.00001, 0.01);
  var curve = \impulseCurve.ukr(0, -6, 6);
  var trig = Dust.ar( \dustFreq.ukr(10, 0.1, 200, \exp) );
  EnvGen.ar( Env([0.0,1.0,0.0], impulseSize ! 2, [curve, curve
      .neg]), trig);
};

q.noise = {
  var frangeHi = 22000;
  var frangeLo = 1;
  var centerFreq = \centerFreq.ukr((frangeLo+frangeHi)/2,
      frangeLo, frangeHi, \exp );
  var rangeFreq = \rangeFreq.ukr( frangeHi, frangeLo, frangeHi
      , 6 );
  var hiFreq = (centerFreq + rangeFreq).min(frangeHi);
  var loFreq = (centerFreq - rangeFreq).max(frangeLo);
  BHiPass.ar( BLowPass.ar( PinkNoise.ar(), hiFreq), loFreq);
};

q.postEQ = { |in|
  var finalLo = \finalLo.ukr(1, 1, 22000, \exp,lag:1);
  var finalHi = \finalHi.ukr(22000, 1, 22000, \exp, lag:1);
  BHiPass.ar( BLowPass.ar( in, finalHi), finalLo);
};
```

```
q.simple = { |sourceF|
  var env = q.impulse.();
  var source = sourceF.();
  var out = \amp.kr * source * env * \globalAmp.ukr(0.5, \amp)
      ;
  var out2 = q.postEQ.(out);
  UOut.ar(0, out2 * 30 )
};

Udef(\clicksNoiseSimple,{ q.simple.(q.noise) });

)
```

D.9.3. *Unexpected criticality* feedback `Udef`




```
//based on code from on http://sccode.org/1-4QC
Udef(\fb3_2,
  {
    var l,k,j,s,o;
    l = LocalIn.ar(1);
    k = LFTri.ar(freq:1, iphase:0, mul:l*\lftriMul.ukr
       (1,1/2,2), add: \lftriAdd.ukr(2,0,10));
    j = k.range(
      (\a.ukr(0.5,0,30)+\aDev.ukr(0,-5.0,5.0)).clip(0.0,30.0),
      (\b.ukr(4,0,30) + \bDev.ukr(0,-5.0,5.0)).clip(0.0,30.0)
         );
    s = PitchShift.ar(
      SinOscFB.ar( j, k ),
      windowSize:\windowSize.ukr(0.03,0.001,0.1,\exp),
```

```
      pitchRatio:j*\pitchRation.ukr(1,1/20,1)
    );
    s = (s* \c.ukr(1.0, 1, 10.0 )).tanh;
    LocalOut.ar(s);
    o = LeakDC.ar(s,0.995/*smaller less low freqs*/)  * \amp.
       ukr(0.1,\amp) * \globalAmp.ukr(0.1,\amp);
    UOut.ar(0, o)
})
```

D.10. field::grid udefs and ImmDefs.

```
ImmDef(\field_grid_1, { |t, shAngle, shScale, shU, shV, m, l,
   f, gAngle, gCurve,
  aLo, bLo, aHi, bHi|
  var sh = PField.sphericalHarmonicNormalizedFunc;
backgroundcolor={\color{white}} language=SuperCollider
  var pf1 = (T(_,_) <%> m <*> l).switchTo({ |tup|
    var l = tup.at2;
    var m = tup.at1.min(l).max(l.neg);
    sh.(m,l).rotTransScale2D.(t, shAngle.nlin(0,2pi), shScale.
       collect(_.reciprocal), shU, shV, f);
  });

  var a = PField.gradient1D.(t, gAngle, aLo, aHi, gCurve);
  var b = PField.gradient1D.(t, gAngle,bLo, bHi, gCurve);

  (
    amp: USpecArg( pf1 ),
    a: UArg( a ),
    b: UArg( b )
  )

}, 0.1, [
  'sh_angle', ControlSpec(0, 1, 'linear', 0.0, 0, "frac of 2pi
     "),
  'sh_scale', ControlSpec(1/5, 5, 'exp', 0.0, 1, ""),
  'sh_u', ControlSpec(-0.5, 0.5, 'linear', 0.0, 0, ""),
  'sh_v', ControlSpec(-0.5, 0.5, 'linear', 0.0, 0, ""),
  'sh_m', ControlSpec(-4, 4, \lin, 1, 2 ),
  'sh_l', ControlSpec(0, 4, \lin, 1, 2 ),
  'sh_freq', ControlSpec(1/10, 4, \lin, 0, 0.5 ),
  'gAngle', ControlSpec(0.0, 1.0, 'linear', 0.0, 0.0, "frac of
      2pi"),
  'gCurve', ControlSpec(-6.0, 6.0, 'linear', 0.0, 0.0, ""),
  \aSide1, ControlSpec(0, 30.0, 'linear', 0.0, 0, ""),
  \bSide1, ControlSpec(0, 30.0, 'linear', 0.0, 0, ""),
  \aSide2, ControlSpec(0, 30.0, 'linear', 0.0, 0, ""),
  \bSide2, ControlSpec(0, 30.0, 'linear', 0.0, 0, "")
]);
```

Appendix E.

Survey answers

E.0.1. Michael Dzjaparidze

```
ImmLib Library survey
=====================
February 2015
Miguel Negrão

In this survey we would like to obtain some informtion
   regarding the use of the ImmLib library.
Questions are marked with '*'.

# INTRODUCTION
==============

* State your full name.

Michael Dzjaparidze


* Specify your previous academic qualifications.

2004 - Bachelor of Communication & Media Management
2010 - Bachelor of Art & Technology - Audio Design
2010 - European Media Master of Arts - Sound & Music
   Technology


* Which computer programming languages have you used before ?

SuperCollider
Processing
Python
JavaScript/jQuery
PHP
MATLAB
Objective-C
C
C++
Java


* Which computer programming languages are you proficient in ?

SuperCollider
Processing
Python
```

```
* Which environments/systems/languages for computer music
   synthesis have you used before  ?

SuperCollider
Max/MSP
PD
Web Audio API


# Previous approaches to spatialization:

* Do the majority of your sound works use the 'stereo' sound
   reproduction method ?

Yes


* If your previous answer was 'no', which types of sound
   reproduction do you primarily use (e.g. quad, octo, WFS,
   live diffusion, etc) ?

* Have you performed live diffusion (the practice of
   controlling in real-time the routing of a stereo or four
   channels piece to a large number of heterogeneous speakers
   placed in the performance space) before ? Do you often
   perform live diffusion of your own works ?

No


* Which of these panning or sound field reproduction
   techniques have you used before ?

  * stereo panning.
    Yes
  * direct routing to individual speakers.
    Yes
  * Ambisonics 2D.
    Yes
  * Ambisonics 3D.
    No
  * Vector Based Panning 2D.
    No
  * Vector Based Panning 3D.
    No
  * Distance Based Panning.
    Yes
  * Wave Field Synthesis.
    No

* Have you used trajectory or positional based software
   systems allowing placement of virtual point sources in a
   virtual sound stage ? If so, which ? (name the library,
   plugin or software).

No
```

```
* When working on the spatialization of your sound works, have
    you ever used similar sounding but nevertheless
   decorrelated material for each event or stream in a group,
   with each stream placed at a different spatial location (e.
   g. making two version of the same sound placing one on the
   left speaker and the other on right speaker) ? If yes,
   please describe your approach to this technique (before
   using ImmLib).

Yes, for all my electroacoustic works (wether stereo or multi-
   channel) I practically do this always.


* When performing works publicly have you used public address
   systems with large numbers of speakers before ? Which ?

Prior to the Sonic Lab, no.


* Have you used speaker systems with 3D sound reproduction
   capabilities (speakers above or below horizontal level)
   before arriving at SARC ? Where ?

No




# Spatial effects:
=================

* In case you had imagined a particular spatial effect or
   experience before using ImmLib, how difficult was it to
   achieve it with ImmLib ?

Moderately hard


* How easy was it to create an immersive sound event with
   ImmLib ?

Moderately easy


* While working on your piece and using ImmLib did you
   discover an unexpected spatial effect or experience ? If so
   , please describe it and how it came about.

Not sure how to answer this... I'm not sure if unexpected is
   the right word, but it was quite interesting to hear the
   perceptual differences the
different surface functions imposed on different types of
   sound materials.


* How limiting did you find the constraint imposed by ImmLib
   that every sound must be made out of n copies of the same
   sound process ?
```

```
Not at all limiting


* How similar did you find ImmLib relative to other
   spatialization tools you have used in the past ?

Not at all similar






# Practical evaluation
======================

* Overall, how easy was it learn ImmLib ?

Moderately hard


* Overall, how easy was it use ImmLib ?

Extremely hard (but this was mainly due to technical
   complications related to UnitLib and SuperCollider itself)


* Which difficulties were you confronted with when learning
   and using ImmLib ?

Apart from the syntax learning curve, I bumped into quite a
   lot of limitations / bugs not so much related
to ImmLib, but rather related to SuperCollider and UnitLib.
   Furthermore, the fact that when I started using the library
   ,
spatial gestures had to be defined for each individual
   sounding event became rather unwieldy as my work consisted
   of lots
of smaller sub-events.





# Use in composition / performance
==================================

The following questions regard the work done on the piece you
   created with ImmLib.

* Describe which type of synths/processes/materials did you
   use for your piece.

All materials were synthesised, using a combination of
   physical modelling and
abstract synthesis techniques (e.g. subtractive, FOF, granular
   )
```

```
* Which parameter fields did you use in the composition ?
   Assigned to which parameters ?

amplitude only: Random Hills, Gradient, BarU,
   SphericalHarmonic, Wave2D, MoveHills


* For which type of materials did you use the random pfields (
   randomHills, moveHills, etc) and the deterministic ones (
   wave2D, spotlight, etc) ?

Not really possible to definitively link certain fields to
   specific materials.
I've used all fields both in conjunction with impulsive
   materials as well as with more smooth,
dense materials as often these were transformed into each
   other.


* Were some of the sounds created or selected on purpose in
   such a way that their internal properties would contribute
   to a specific spatial effect ? If so, can you describe one
   such sound, the spatial effect associated with it and what
   were the sound's internal characteristics that contributed
   to the spatial effect.

The more impulsive sounds I wanted to be as wide/diffuse as
   possible.
I found the RandomHills algorithm to work quite effective for
   this. Much more effective
than the SphericalHarmonic algorithm for instance. The
   SphericalHarmonic algorithm
in turn I found to work most effectively for sounding events
   with wave-like timbral qualities.


* Were some sounds first tested in mono/stereo before being
   used with ImmLib ?

All sounds were tested in stereo before using them with ImmLib
   ,
although using the same basis technique, using de-correlated
   signals for both speakers.


* If you answered yes in the previous question, How much did
   you change those sounds, specially regarding the timbral
   qualities, after they were imported into ImmLib and
   listened to in the Sonic lab ?

Some changed only slightly, most not changed at all


* Picking one specific sound that was changed, describe which
   changes you performed and why you changed it.

In particular with the impulsive sounds I had to modify the
```

```
   density a little,
as the density of sounds was noticeably different due to the
   larger number of de-correlated output signals.


* How important was it to create a sense of movement ?

Moderately important


* Was a sense of movement created ?

Yes


* If not, what was lacking ?

* Give an example of one movement, describe what was moving
   and how ?

Circular/rotational motion of a sustained, noisey background
   layer using the BarU field
Wave-like movement of synthesised wave sounds using the
   SphericalHarmonic field


* By placing a decorrelated process in ImmLib in some intances
    an immersive sound will be created even without using
   parameter fields. In your piece, how much of the spatial
   effects created were due to events using just multiple
   decorrelated copies without any parameter fields and how
   much as due to events that made use of parameter fields.

50% just multiple copies  -  50% parameter fields


* How easy was it to achieve decorrelation between the output
   of each synth in an ImmUChain ?

Average difficulty


* Which techniques or methods did you use to achieve
   decorrelation ?

Synthesising de-correlated signals per channel, by differences
    in filtering,
trigger times and independent noise sources.


* How was the number of points for the surface(s) chosen ?

Chosen based on the max number of points which allowed the
   piece to be played in real-time.
This had to be made dynamic as some sections of the piece are
   much more computationally demanding than others.
```

```
* How did the computational limitations (max number of points,
     max number of events, etc) affect the final result of the
    piece ?

I had to do major changes or leave parts out.


* How often did you use the plot tool to visualize the
    parameter fields ?

from time to time


* How did the need to use the plot tool change after spending
    more time with the library ?

Decreased.


* Which approach best describes the compositional approach to
    your piece regarding the use of ImmLib:

I essentially did the whole piece first in stereo (or another
    reduced format) and
then worked on the spatialization in the Sonic Lab using
    ImmLib.


By returning this form you agree that the content submitted
    will be used for research purposes.

thank you
```

E.0.2. Pablo Sanz

```
ImmLib Library survey
=====================
February 2015
Miguel Negrão

In this survey we would like to obtain some informtion
    regarding the use of the ImmLib library.
Questions are marked with '*'.

# INTRODUCTION
==============

* State your full name.

Pablo Sanz


* Specify your previous academic qualifications.

BFA ArtScience Interfaculty, University of the Arts, Royal
    Conservatoire, The Hague, NL, 2008-12
Postgraduate International Course, Institute of Sonology,
    Royal Conservatoire, The Hague, NL, 2007-08
Sound Design & Music Technology Degree, Puerta Bonita
```

```
   Audiovisual Institute, Madrid, ES, 2002-04


* Which computer programming languages have you used before ?

Max/MSP
SuperCollider
CSound
Pascal
Assembly
Basic


* Which computer programming languages are you proficient in ?

Max/MSP

* Which environments/systems/languages for computer music
   synthesis have you used before  ?

Max/MSP
SuperCollider
CSound

# Previous approaches to spatialization:

stereo panning
binaural
direct routing to individual speakers
Vector Based Panning 2D

* Do the majority of your sound works use the 'stereo' sound
   reproduction method ?

No

* If your previous answer was 'no', which types of sound
   reproduction do you primarily use (e.g. quad, octo, WFS,
   live diffusion, etc) ?

I use stereo, quad, octo and ad-hoc non-standard spatial
   configurations using different kinds of sound media
   technologies (tactile transducers and speakers with
   different characteristics).


* Have you performed live diffusion (the practice of
   controlling in real-time the routing of a stereo or four
   channels piece to a large number of heterogeneous speakers
   placed in the performance space) before ? Do you often
   perform live diffusion of your own works ?

A couple of times. No, its not something I do often.


* Which of these panning or sound field reproduction
   techniques have you used before ?
```

```
  * stereo panning.  X
  * direct routing to individual speakers.  X
  * Ambisonics 2D.  X
  * Ambisonics 3D.
  * Vector Based Panning 2D.   X
  * Vector Based Panning 3D.
  * Distance Based Panning.  not sure
  * Wave Field Synthesis.   X

* Have you used trajectory or positional based software
   systems allowing placement of virtual point sources in a
   virtual sound stage ? If so, which ? (name the library,
   plugin or software).

ReaSurround
ICST Ambisonics for Max/MSP

* When working on the spatialization of your sound works, have
    you ever used similar sounding but nevertheless
   decorrelated material for each event or stream in a group,
   with each stream placed at a different spatial location (e.
   g. making two version of the same sound placing one on the
   left speaker and the other on right speaker) ? If yes,
   please describe your approach to this technique (before
   using ImmLib).

Yes, I use that very often. Usually directly routing
   decorrelated streams of the same material to individual
   speakers.


* When performing works publicly have you used public address
   systems with large numbers of speakers before ? Which ?

Audio Kultur Festival, ZKM  Centre for Art and Media
   Technology, Karlsruhe, Germany.
WFS-GOL Scheltema, Leiden, Netherlands - 192 WFS
Hydrophones III,LEstartit, Girona, Spain  - 16-channel



* Have you used speaker systems with 3D sound reproduction
   capabilities (speakers above or below horizontal level)
   before arriving at SARC ? Where ?

Klangdom (Sound dome) in the ZKM_Kubus (ZKM_Cube)at the ZKM |
   Institute for Music und Acoustics


# Spatial effects:
=================

* In case you had imagined a particular spatial effect or
   experience before using ImmLib, how difficult was it to
   achieve it with ImmLib ?

Extremely easy
Moderately easy
```

```
Average difficulty
Moderately hard
Couldn't be done
I didn't imagine a spatial effect beforehand  X

* How easy was it to create an immersive sound event with
   ImmLib ?

Extremely easy
Moderately easy
Average difficulty
Moderately hard
Extremely hard

N/A

* While working on your piece and using ImmLib did you
   discover an unexpected spatial effect or experience ? If so
   , please describe it and how it came about.

N/A

* How limiting did you find the constraint imposed by ImmLib
   that every sound must be made out of n copies of the same
   sound process ?

Extremely limiting
Very limiting
Moderately limiting
Slightly limiting
Not at all limiting -- X

* How similar did you find ImmLib relative to other
   spatialization tools you have used in the past ?

Extremely similar
Very similar
Moderately similar
Slightly similar
Not at all similar  -- X




# Practical evaluation
======================

* Overall, how easy was it learn ImmLib ?

Extremely easy
Moderately easy
Moderately hard
Extremely hard

N.A.

* Overall, how easy was it use ImmLib ?
```

```
Extremely easy
Moderately easy
Moderately hard
Extremely hard

N.A.

* Which difficulties were you confronted with when learning
   and using ImmLib ?

N.A.

# Use in composition / performance
==================================

The following questions regard the work done on the piece you
   created with ImmLib.

* Describe which type of synths/processes/materials did you
   use for your piece.

Environmental recordings of different provenances and basic
   synth generators (noise).


* Which parameter fields did you use in the composition ?
   Assigned to which parameters ?

So far amplitudes, freq in filters, buf-rate.


* For which type of materials did you use the random pfields (
   randomHills, moveHills, etc) and the deterministic ones (
   wave2D, spotlight, etc) ?

Still experimenting.


* Were some of the sounds created or selected on purpose in
   such a way that their internal properties would contribute
   to a specific spatial effect ? If so, can you describe one
   such sound, the spatial effect associated with it and what
   were the sound's internal characteristics that contributed
   to the spatial effect.

Still experimenting.


* Were some sounds first tested in mono/stereo before being
   used with ImmLib ?

All source recordings edited, tested and sometimes pre-
   processed in mono as preparation for ImmLib use.


* If you answered yes in the previous question, How much did
   you change those sounds, specially regarding the timbral
   qualities, after they were imported into ImmLib and
```

```
   listened to in the Sonic lab ?

Changed them almost completely
Changed them a lot
Changed them moderately
Changed them only slightly
Did not change them at all

Still experimenting... various degrees of changes.

* Picking one specific sound that was changed, describe which
   changes you performed and why you changed it.

N.A.

* How important was it to create a sense of movement ?  There
   is always movement

Extremely important
Very important
Moderately important
Slightly important
Not at all important

* Was a sense of movement created ?

--

* If not, what was lacking ?

Still experimenting.

* Give an example of one movement, describe what was moving
   and how ?

Wind-like sound generated with filtered noise. Using PField to
    move amps, creates spatial effect of movement in the array
   .


* By placing a decorrelated process in ImmLib in some intances
    an immersive sound will be created even without using
   parameter fields. In your piece, how much of the spatial
   effects created were due to events using just multiple
   decorrelated copies without any parameter fields and how
   much as due to events that made use of parameter fields.

Still experimenting.

* How easy was it to achieve decorrelation between the output
   of each synth in an ImmUChain ?

N.A.

* Which techniques or methods did you use to achieve
   decorrelation ?

Different StartPos, filtering or bufRate.
```

```
* How was the number of points for the surface(s) chosen ?

N.A.

* How did the computational limitations (max number of points,
    max number of events, etc) affect the final result of the
   piece ?

Still experimenting...  no piece.

* How often did you use the plot tool to visualize the
   parameter fields ?

All the time --- X
Quite often
from time to time
Rarely
Never

* How did the need to use the plot tool change after spending
   more time with the library ?

Still experimenting...  didt reach that second stage.

* Which approach best describes the compositional approach to
   your piece regarding the use of ImmLib:

Still experimenting, back and forth between 2 and 3 .

I essentially did the whole piece first in stereo (or another
   reduced format) and then worked on the spatialization in
   the Sonic Lab using ImmLib.
It was a back and forth process of working on the sounds in
   mono/stereo then listening and changing them with ImmLib in
    the Sonic Lab and then reworking them in mono/stereo again
   .  -- X
All the sounds were created in the Sonic Lab using ImmLib and
   working at the same time on the spatialization, timing and
   spectrum.  -- X

None of the above (Please describe your approach).


By returning this form you agree that the content submitted
   will be used for research purposes.

thank you
```

Appendix F.

Contents of the companion USB flash drive to this thesis

The accompanying USB flash drive contains:

- The *ImmLib* library and necessary dependencies.
- The *ImmLib* library documentation in separate HTML files.
- Sound recordings in 32 channels and binaural stereo of:
  - *Extase 2, 3 & 4.*
  - *Unexpected Criticality.*
  - *Sonic Arts Theatre Lab.*
- Videos of several `PField`s in use with binaural stereo sound.
- A pdf version of this document.

Bibliography

[1] J. Chowning, "Turenas: the realization of a dream," *Proc. of the 17es Journées d'Informatique Musicale*, 2011. [Online]. Available: http://jim2011.univ-st-etienne.fr/html/actes/05_johnturenas2b.pdf

[2] M. A. Baalman, "Spatial composition techniques and sound spatialisation technologies," *Organised Sound*, vol. 15, no. 03, pp. 209–218, 2010.

[3] J. Daniel, R. Nicol, and S. Moreau, "Further investigations of high order ambisonics and wavefield synthesis for holophonic sound imaging," in *114th Audio Eng. Soc. Conv.*, 2003. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12567

[4] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *J. Audio Eng. Soc.*, vol. 45, no. 6, 1997. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=7853

[5] S. Spors, R. Rabenstein, and J. Ahrens, "The theory of wave field synthesis revisited," in *124th Audio Eng. Soc. Conv.*, 2008, pp. 17–20. [Online]. Available: http://www.deutsche-telekom-laboratories.de/~sporssas/publications/2008/AES124_Spors_WFS_Theory.pdf

[6] F. Rumsey, "Subjective assessment of the spatial attributes of reproduced sound," in *15th Int. Audio Eng. Soc. Conf.*, Oct. 1998. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=8096

[7] G. S. Kendall, "Spatial perception and cognition in multichannel audio for electroacoustic music," *Organised Sound*, vol. 15, no. 03, pp. 228–238, 2010.

[8] J. Blauert, *Spatial Hearing.* Cambridge, MA: MIT Press, 1997.

[9] G. Plenge, “Über das problem der im-kopf-lokalisation,” *Acustica*, vol. 26, no. 5, pp. 241–252, 1972.

[10] L. Danilenko, “Binaural hearing in non-stationary diffuse sound field,” *Kybernetik*, vol. 6, no. 2, p. 50, 1969. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/5795369

[11] G. S. Kendall, “The decorrelation of audio signals and its impact on spatial imagery,” *Computer Music J.*, vol. 19, no. 4, pp. 71–87, Dec. 1995. [Online]. Available: http://www.jstor.org/stable/3680992

[12] A. Cabrera and G. Kendall, “Multichannel control of spatial extent through sinusoidal partial modulation,” 2013. [Online]. Available: http://smcnetwork.org/system/files/MULTICHANNEL%20CONTROL%20OF%20SPATIAL%20EXTENTTHROUGH%20SINUSOIDAL%20PARTIAL%20MODULATION(SPM).pdf

[13] S. Wilson, “Spatial swarm granulation,” in *Proc. Int. Computer Music Conf.*, Belfast, N. Ireland, 2008. [Online]. Available: http://eprints.bham.ac.uk/237/1/cr1690.pdf

[14] D. Kim-Boyle, “Spectral spatialization - an overview,” in *Proc. Int. Computer Music Conf*, Belfast, 2008. [Online]. Available: http://quod.lib.umich.edu/i/icmc/bbp2372.2008.086?rgn=main;view=fulltext

[15] O. Warusfel, “IRCAM 's listen HRTF database,” Sep. 2015. [Online]. Available: http://recherche.ircam.fr/equipes/salles/listen/index.html

[16] M. V. Mathews, “The digital computer as a musical instrument,” *Science*, vol. 142, no. 3592, pp. 553–557, 1963. [Online]. Available: http://www.jaimeoliver.pe/courses/ci/pdf/mathews-1963.pdf

[17] P. Manning, *Electronic and Computer Music.* Oxford: Oxford University Press, Feb. 2013.

[18] R. Zvonar, “A history of spatial music,” *eContact!*, vol. 7.4. [Online]. Available: http://econtact.ca/7_4/zvonar_spatialmusic.html

[19] J. Harrison, "Sound, space, sculpture: some thoughts on the 'what','how'and 'why' of sound diffusion," *Organised Sound*, vol. 3, no. 02, pp. 117–127, 1998. [Online]. Available: http://journals.cambridge.org/abstract_S1355771898002040

[20] R. Priemer, *Introductory Signal Processing.* Singapore: World Scientific Company P., 1991.

[21] B. Truax, "Composition and diffusion: space in sound in space," *Organised Sound*, vol. 3, no. 2, pp. 141–146, 1998. [Online]. Available: http://journals.cambridge.org/production/action/cjoGetFulltext?fulltextid=50104

[22] S. Wilson and J. Harrison, "Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre," *Organised Sound*, vol. 15, no. 03, pp. 239–250, 2010. [Online]. Available: http://journals.cambridge.org/production/action/cjoGetFulltext?fulltextid=7916439

[23] K. Tazelaar, *On the Threshold of Beauty: Philips and the Origins of Electronic Music in the Netherlands 1925–1965.* Rotterdam: V2_Institute for the Unstable Media / NAi Publishers, 2013.

[24] J. M. Chowning, "The simulation of moving sound sources," *Computer Music J.*, pp. 48–52, 1977. [Online]. Available: http://www.jstor.org/stable/10.2307/3679609

[25] M. Geier, J. Ahrens, and S. Spors, "Object-based audio reproduction and the audio scene description format," *Organised Sound*, vol. 15, no. 03, pp. 219–227, 2010.

[26] N. Barrett, "Spatio-musical composition strategies," *Organised Sound*, vol. 7, no. 03, pp. 313–323, Dec. 2002. [Online]. Available: http://journals.cambridge.org/article_S1355771802003114

[27] D. Smalley, "Spectromorphology: explaining sound-shapes," *Organised Sound*, vol. 2, no. 02, pp. 107–126, Aug. 1997. [Online]. Available: http://journals.cambridge.org/article_S1355771897009059

[28] H. Vaggione, "Composing musical spaces by means of decorrelation of

audio signals," in *Proc. DAFx Conf.*, 2001. [Online]. Available: http://www.csis.ul.ie/dafx01/proceedings/papers/Addendum1%20-%20Vaggione.pdf

[29] F. Rumsey, "Spatial quality evaluation for reproduced sound: Terminology, meaning, and a scene-based paradigm," *J. Audio Eng. Soc.*, vol. 50, no. 9, pp. 651–666, 2002. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=11067

[30] M. Droumeva, "Understanding immersive audio: a historical and socio-cultural exploration of auditory displays," 2005. [Online]. Available: https://smartech.gatech.edu/handle/1853/50196

[31] D. Smalley, "Space-form and the acousmatic image," *Organised Sound*, vol. 12, no. 01, pp. 35–58, 2007. [Online]. Available: http://journals.cambridge.org/abstract_S1355771807001665

[32] B. Girod, R. Rabenstein, and A. Stenger, *Signals and systems.* Wiley, May 2001.

[33] M. Morimoto and K. Iida, "A practical evaluation method of auditory source width in concert halls." *Journal of the Acoustical Society of Japan (E)*, vol. 16, no. 2, pp. 59–69, 1995. [Online]. Available: http://jlc.jst.go.jp/DN/JALC/00031400992?from=Google

[34] R. I. Chernyak and N. A. Dubrovsky, "Pattern of the noise images and the binaural summation of loudness for the different interaural correlation of noise," *Proc. 6th Int. Congr. on Acoustics Tokyo*, vol. 1, pp. 3–12, 1968.

[35] J. Blauert and W. Lindemann, "Spatial mapping of intracranial auditory events for various degrees of interaural coherence," *The Journal of the Acoustical Society of America*, vol. 79, no. 3, pp. 806–813, 1986. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/79/3/10.1121/1.393471

[36] T. Hidaka, L. L. Beranek, and T. Okano, "Interaural cross-correlation, lateral fraction, and low-and high-frequency sound levels as measures of acoustical quality in concert halls," *The Journal of the Acoustical Society of America*, vol. 98, no. 2, pp. 988–1007, 1995. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/98/2/10.1121/1.414451

[37] R. Mason, T. Brookes, and F. Rumsey, "Creation and verification of a controlled experimental stimulus for investigating selected perceived spatial attributes," in *114th AES Convention*, Mar. 2003, pp. 22–25. [Online]. Available: http://epubs.surrey.ac.uk/553/

[38] H. Haas, "Über den Einflub eines Einfachechos auf die Hörsamkeit von Sprache," *Acta Acustica united with Acustica*, vol. 1, no. 2, pp. 49–58, 1951. [Online]. Available: http://www.ingentaconnect.com/content/dav/aaua/1951/00000001/00000002/art00003

[39] P. Damaske, "Subjektive untersuchung von schallfeldern," *Acustica*, vol. 19, no. 4, pp. 199–213, 1967.

[40] A. S. Bregman and S. Pinker, "Auditory streaming and the building of timbre." *Canadian J. Psychology*, vol. 32, no. 1, p. 19, 1978. [Online]. Available: http://psycnet.apa.org/journals/cep/32/1/19/

[41] M. Kubovy, J. E. Cutting, and R. M. McGuire, "Hearing with the third ear: Dichotic perception of a melody without monaural familiarity cues," *Science*, vol. 186, no. 4160, pp. 272–274, 1974. [Online]. Available: http://people.psych.cornell.edu/~jec7/pubs/archives/third_ear.pdf

[42] J. F. Culling, "Auditory motion segregation: A limited analogy with vision." *J. Experimental Psychology: Human Perception and Performance*, vol. 26, no. 6, p. 1760, 2000. [Online]. Available: http://psycnet.apa.org/journals/xhp/26/6/1760/

[43] C. J. Darwin and V. Ciocca, "Grouping in pitch perception: Effects of onset asynchrony and ear of presentation of a mistuned component," *J. Acoust. Soc. of Amer.*, vol. 91, no. 6, pp. 3381–3390, 1992. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/91/6/10.1121/1.402828

[44] V. Pulkki, "Localization of amplitude-panned virtual sources II: Two-and three-dimensional panning," *J. Audio Eng. Soc.*, vol. 49, no. 9, pp. 753–767, 2001. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=10179

[45] V. Pulkki and M. Karjalainen, "Multichannel audio rendering using amplitude panning [DSP applications]," *IEEE Signal Processing Mag.*, vol. 25, no. 3, pp. 118–122, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4490207

[46] V. Pulkki, "Generic panning tools for MAX/MSP," *Int. Computer Music Conf. Proc.*, vol. 2000, 2000. [Online]. Available: http://hdl.handle.net/2027/spo.bbp2372.2000.210

[47] "VBAP - Csound." [Online]. Available: http://www.csounds.com/manual/html/vbap.html

[48] "sc3-plugins." [Online]. Available: https://github.com/supercollider/sc3-plugins

[49] M. A. Gerzon, "General metatheory of auditory localisation," in *Audio Eng. Soc. Conv. 92*, 1992. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=6827

[50] J. Daniel, "Spatial sound encoding including near field effect: introducing distance coding filters and a viable, new ambisonic format." Audio Eng. Soc., May 2003. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12321

[51] J. Ahrens and S. Spors, "Focusing of virtual sound sources in higher order ambisonics," May 2008. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=14508

[52] ——, "Rendering of virtual sound sources with arbitrary directivity in higher order ambisonics," Oct. 2007. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=14301

[53] S. Bertet, J. Daniel, and S. Moreau, "3d sound field recording with higher order ambisonics - objective measurements and validation of spherical microphone," May 2006. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=13661

[54] P. Stitt, S. Bertet, and M. van Walstijn, "Off-centre localisation performance of ambisonics and HOA for large and small loudspeaker array radii," *Acta Acustica united with Acustica*, vol. 100, no. 5, pp. 937–944, Sep. 2014.

[Online]. Available: http://openurl.ingenta.com/content/xref?genre=article&issn=1610-1928&volume=100&issue=5&spage=937

[55] M. Kronlachner, "ambix VST, LV2 plug-ins." [Online]. Available: https://github.com/kronihias/ambix

[56] J. C. Schacher and P. Kocher, "Ambisonics spatialization tools for Max/MSP," *Int. Computer Music Conf. Proc.*, vol. 2006, 2006. [Online]. Available: http://hdl.handle.net/2027/spo.bbp2372.2006.057

[57] D. G. Malham and A. Myatt, "3-D sound spatialization using Ambisonic techniques," *Computer Music J.*, vol. 19, no. 4, pp. 58–70, Dec. 1995. [Online]. Available: http://www.jstor.org/stable/3680991

[58] J. Anderson, "Introducing... the Ambisonic toolkit," 2009. [Online]. Available: http://www.researchgate.net/profile/Joseph_Anderson12/publication/273379852_Introducing..._the_Ambisonic_Toolkit/links/54ff47160cf2eaf210b8a614.pdf

[59] M. Geier and S. Spors, "Spatial audio with the SoundScape Renderer," 2012. [Online]. Available: https://www.int.uni-rostock.de/fileadmin/user_upload/publications/spors/2012/Geier_TMT2012_SSR.pdf

[60] "Spat - Ircam website," Sep. 2015. [Online]. Available: http://forumnet.ircam.fr/product/spat-en/

[61] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *J. Acoust. Soc. Amer.*, vol. 93, no. 5, pp. 2764–2778, 1993. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/93/5/10.1121/1.405852

[62] E. N. G. Verheijen, "Sound reproduction by wave field synthesis," Ph.D. dissertation, TU Delft, Delft University of Technology, 1998. [Online]. Available: http://repository.tudelft.nl/assets/uuid:9a35b281-f19d-4f08-bec7-64f6920a3821/as_verheijen_19980119.PDF

[63] E. Corteel, "Synthesis of directional sources using wave field synthesis,

possibilities, and limitations," 2007. [Online]. Available: http://articles.ircam.fr/textes/Corteel07a/index.pdf

[64] Y. Marimoto, "Mapping complex systems," MS Thesis, 2009.

[65] A. Sauer and W. Snoei, "WFSCollider manual." [Online]. Available: http://sourceforge.net/projects/wfscollider/files/WFSCollider%20Manual/Working%20with%20WFSCollider%20v2.2.pdf/download

[66] "Four Audio | Wave-field synthesis." [Online]. Available: http://www.four-audio.de/en/references/wave-field-synthesis.html

[67] M. A. J. Baalman, "On Wave Field Synthesis and electro-acoustic music, with a particular focus on the reproduction of arbitrarily shaped sound sources, Wellenfeldsynthese und elektro-akustische Musik, mit einem speziellen Fokus auf die Reproduktion von willkürlich geformten Schallquellen." [Online]. Available: https://opus4.kobv.de/opus4-tuberlin/frontdoor/index/index/docId/1801

[68] "GameOfLife - WFSCollider." [Online]. Available: https://github.com/GameOfLife/WFSCollider

[69] M. Negrão, "The challenges and possibilities of real-time wave field synthesis," 2010.

[70] ——, "Strategies in diffuse spatialization," MS Thesis, Den Haag, Netherlands, 2010. [Online]. Available: http://kc.koncon.nl/sonology/NL/thesis-pdf/Negrao-Strategies%20in%20Diffuse%20Spatialization.pdf

[71] F. L. Wightman and D. J. Kistler, "Headphone simulation of free-field listening. I: Stimulus synthesis," *J. Acoust. Soc. Amer.*, vol. 85, no. 2, pp. 858–867, 1989. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/85/2/10.1121/1.397557

[72] ——, "Headphone simulation of free-field listening. II: Psychophysical," *J. Acoust. Soc. Amer.*, vol. 85, no. 2, pp. 868–878, 1989. [Online]. Available: http://public.vrac.iastate.edu/~charding/audio/Headphone%20simulation%20of%20free-field%20listening.%20II-%20Psychophysical%20validation%20-%20J%20Acoust%20Soc%20Am%201989%20-%20Wightman.pdf

[73] D. R. Begault, E. M. Wenzel, and M. R. Anderson, “Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source,” *J. Audio Eng. Soc.*, vol. 49, no. 10, pp. 904–916, 2001. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=10175

[74] E. M. Wenzel, S. S. Fisher, P. K. Stone, and S. H. Foster, “A system for three-dimensional acoustic visualization in a virtual environment workstation.” IEEE Computer Society Press, 1990, pp. 329–337. [Online]. Available: http://dl.acm.org/citation.cfm?id=949583

[75] M. Noisternig, A. Sontacchi, T. Musil, and R. Höldrich, “A 3d ambisonic based binaural sound reproduction system,” 2003. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12314

[76] D. Menzies and M. Al-Akaidi, “Nearfield binaural synthesis and ambisonics,” *J. Acoust. Soc. Amer.*, vol. 121, no. 3, pp. 1559–1563, 2007. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/121/3/10.1121/1.2434761

[77] “BEASTmulch - Research - BEAST - University of Birmingham,” 2013. [Online]. Available: http://www.birmingham.ac.uk/facilities/BEAST/research/mulch.aspx

[78] C. Roads, “Automated granular synthesis of sound,” *Computer Music J.*, pp. 61–62, 1978.

[79] B. Truax, “Real-time granular synthesis with a digital signal processor,” *Computer Music J.*, pp. 14–26, 1988. [Online]. Available: http://www.jstor.org/stable/3679938

[80] D. Schwarz, “Data-driven concatenative sound synthesis,” Ph.D. thesis, Université Paris 6, Paris, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.418.7880

[81] G. C. Covell, “New and Essential Xenakis, Part 2,” Dec. 2006. [Online]. Available: http://www.lafolia.com/new-and-essential-xenakis-part-2/

[82] B. Hofman, “Spatial aspects in Xenakis’ instrumental works,” Athens, Oct. 2006. [Online]. Available: http://www.iannis-xenakis.org/Articles/Hofmann(Boris).pdf

[83] P. Lang, “pe lang.” [Online]. Available: http://www.pelang.ch/works.html

[84] A. Licht, “Organizing the Unpredictable: Tim Knowles and Pe Lang + Zimoun at bitforms gallery,” Feb. 2009. [Online]. Available: http://rhizome.org/editorial/2365

[85] E. v. d. Heide, *Edwin van der Heide Website*. [Online]. Available: http://www.evdh.net/

[86] R. Minard, *S0UNDB1TS*. [Online]. Available: http://s0undb1ts.wordpress.com/2002/06/01/inventionen-2002/

[87] D. Kim-Boyle, “Spectral and granular spatialization with boids,” in *Proc. Int. 2006 Computer Music Conf.*, New Orleans, USA, pp. 139–142.

[88] D. Topper, M. Burtner, and S. Serafin, “Spatio-operational spectral (SOS) synthesis,” Hamburg, Germany, 2002. [Online]. Available: http://www2.hsuhh.de/ant/dafx2002/papers/DAFX02_Topper_Burtner_Serafin_SOS_synthesis.pdf

[89] A. S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound.* Cambridge, MA: MIT press, 1994.

[90] E. Lyon, “Image-based spatialization,” Ljubljana, Slovenia, 2012, pp. 200–203.

[91] “The Game Of Life Foundation website.” [Online]. Available: http://gameoflife.nl/en/

[92] M. V. Mathews and J. E. Miller, *Music IV Programmer’s Manual.* Bell Telephone Labs, 1967.

[93] M. A. Frank, “Fast hierarchical discretization of parametric surfaces,” MS Thesis, 2009. [Online]. Available: http://people.idsia.ch/~kail/kailfrank_thesis.pdf

[94] E. Abbena, S. Salamon, and A. Gray, *Modern Differential Geometry of Curves and Surfaces with Mathematica.* CRC press, 2006.

[95] J. McCleary, *Geometry from a Differentiable Viewpoint.* Cambridge University Press, Oct. 2012.

[96] M. P. d. Carmo, *Differential Geometry of Curves and Surfaces.* Englewood Cliffs, N.J.: Prentice-Hall, 1976.

[97] E. W. Weisstein, "Geodesic dome." [Online]. Available: http://mathworld.wolfram.com/GeodesicDome.html

[98] R. Bauer, "Distribution of points on a sphere with application to star catalogs," *J. Guidance and Control*, vol. 23, no. 1, 2000. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/2.4497

[99] P. Sanz, "Pablo Sanz - website," Sep. 2015. [Online]. Available: http://www.pablosanz.info/

[100] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976. [Online]. Available: http://isites.harvard.edu/fs/docs/icb.topic1313269.files/Lecture22/May_Simple%20mathematical%20models%20with%20very_76.pdf

[101] "Diffusion - Merriam-Webster Dictionary." [Online]. Available: http://www.merriam-webster.com/dictionary/diffusion

[102] G. Nicolis and A. Wit, "Reaction-diffusion systems," *Scholarpedia*, vol. 2, no. 9, p. 1475, 2007. [Online]. Available: http://www.scholarpedia.org/article/Reaction-diffusion_systems

[103] A. Zhabotinsky, "Belousov-Zhabotinsky reaction," *Scholarpedia*, vol. 2, no. 9, p. 1435, 2007. [Online]. Available: http://www.scholarpedia.org/article/Belousov-Zhabotinsky_reaction

[104] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions Royal Soc. London*, vol. 237, no. 641, pp. 37–72, 1952. [Online]. Available: http://rstb.royalsocietypublishing.org/content/roypth/237/641/37.full.pdf

[105] "Vimeo - Jonathan McCabe," Sep. 2015. [Online]. Available: http://vimeo.com/jonathanmccabe

[106] J. McCabe, “jonathanmccabe.com,” Sep. 2015. [Online]. Available: http://jonathanmccabe.com/

[107] E. W. Weisstein, “Cellular Automaton – from Wolfram MathWorld,” Sep. 2015. [Online]. Available: http://mathworld.wolfram.com/CellularAutomaton.html

[108] T. Yang, “Computational verb cellular networks: Part I–A new paradigm of human social pattern formation,” *Int. J. Computational Cognition*, vol. 7, no. 1, pp. 1–34, 2009.

[109] S. Rafler, “Generalization of Conway’s "Game of Life" to a continuous domain - SmoothLife,” *arXiv:1111.1567 [nlin]*, Nov. 2011, arXiv: 1111.1567. [Online]. Available: http://arxiv.org/abs/1111.1567

[110] ——, “SmoothLife,” Sep. 2015. [Online]. Available: http://sourceforge.net/projects/smoothlife/

[111] M. Conroy, “continuous automata,” Sep. 2015. [Online]. Available: https://vimeo.com/56306508

[112] A. Turner, “A simple model of the Belousov-Zhabotinsky reaction from first principles,” Bartlett School of Graduate Studies, UCL, London, UK, Report, Mar. 2009. [Online]. Available: http://discovery.ucl.ac.uk/17241/1/17241.pdf

[113] J. McCartney, “Rethinking the computer music language: SuperCollider,” *Computer Music J.*, vol. 26, no. 4, pp. 61–68, Dec. 2002. [Online]. Available: http://www.jstor.org/stable/3681770

[114] M. Wright, A. Freed, and others, “Open sound control: A new protocol for communicating with sound synthesizers,” in *Proceedings of the 1997 International Computer Music Conference*, vol. 2013. International Computer Music Association San Francisco, 1997, p. 10. [Online]. Available: http://archive.cnmat.berkeley.edu/ICMC97/papers-html/OpenSoundControl.html

[115] “Unit-Lib.” [Online]. Available: https://github.com/GameOfLife/Unit-Lib

[116] S. Wilson, J. Harrison, and S. L. Ancona, “Beast-mulch.” [Online]. Available: http://www.beast.bham.ac.uk/research/mulch.shtml

[117] M. Baalman, T. Bovermann, A. de Campo, and M. Negrão, "Modality," *Int. Computer Music Conf. Proc.*, vol. 2014, 2014. [Online]. Available: http://quod.lib.umich.edu/i/icmc/bbp2372.2014.165/1

[118] Z. Wan and P. Hudak, "Functional reactive programming from first principles," in *ACM SIGPLAN Notices*, vol. 35, 2000, pp. 242–252. [Online]. Available: http://dl.acm.org/citation.cfm?id=349331

[119] E. Czaplicki, "Elm: concurrent FRP for functional GUIs," Ph.D. dissertation, 2012. [Online]. Available: http://elm-lang.org/papers/concurrent-frp.pdf

[120] C. Elliott and P. Hudak, "Functional reactive animation," in *ACM SIGPLAN Notices*, vol. 32, 1997, pp. 263–273. [Online]. Available: http://dl.acm.org/citation.cfm?id=258973

[121] P. Hudak, A. Courtney, H. Nilsson, and J. Peterson, "Arrows, Robots, and Functional Reactive Programming," in *Summer School on Advanced Functional Programming 2002, Oxford University*, ser. Lecture Notes in Computer Science, vol. 2638. Oxford, UK: Springer-Verlag, 2003, pp. 159–187. [Online]. Available: http://haskell.cs.yale.edu/?post_type=publication&p=175

[122] "FPLib." [Online]. Available: https://github.com/miguel-negrao/FPLib

[123] "reactive-web," Sep. 2015. [Online]. Available: https://github.com/nafg/reactive

[124] "Reactive-banana - HaskellWiki," Sep. 2015. [Online]. Available: https://wiki.haskell.org/Reactive-banana

[125] T. Blechmann, "Supernova - A scalable parallel audio synthesis server for SuperCollider," *Int. Comp. Music Conf. Proc.*, vol. 2011, 2011. [Online]. Available: http://quod.lib.umich.edu/i/icmc/bbp2372.2011.128/1/--supernova-a-scalable-parallel-audio-synthesis-server-for?rgn=full+text;view=image;q1=supernova

[126] M. Dzjaparidze, "Exploring the Creative Potential of Physically Inspired Sound Synthesis," Ph.D. dissertation, Queen's University Belfast, 2015.

[127] B. C. Moore, *An Introduction to the Psychology of Hearing.* UK: Emerald Group Publishing, 2012.

[128] xfff, "more tweets (pitchshift feedback + sinoscfb)." [Online]. Available: http://sccode.org/1-4QC

[129] J. Yang, "Justin Yang." [Online]. Available: http://www.somasa.qub.ac.uk/~comedia/blog/styled-2/index.html

[130] P. Batchelor, "The intimate and the immersive in grids: multichannel sound installations," *Leonardo Music J.*, vol. 23, no. 1, pp. 6–7, 2013. [Online]. Available: https://muse.jhu.edu/journals/leonardo_music_journal/v023/23.batchelor.html

[131] N. D. Megill, *Metamath: A Computer Language for Pure Mathematics.* Morrisville, North Carolina: Lulu Press, 2007, http://us.metamath.org/downloads/metamath.pdf.