Computational composition strategies in audiovisual laptop performance

Alo Allik

The University of Hull

This thesis is submitted for the degree of Doctor of Philosophy to accompany a portfolio of audiovisual performances and the software systems used in the creation of these works

PhD supervisors: Dr. Robert Mackay and Dr. Joseph Anderson

This project was supported by the University of Hull 80th Anniversary Scholarship.

April 2014

Acknowledgements

I would like to take this opportunity to thank everyone who has supported this project throughout its duration. I am very grateful to my supervisor Rob Mackay who not only has been a very supportive and knowledgeable mentor, but also an amazingly accommodating friend during my time at the Creative Music Technology program at the University Hull Scarborough campus. I am also indebted to my first supervisor Jo Anderson for encouragement, lengthy conversations and in-depth knowledge, particularly in Ambisonic spatialization, allowing me to test the early versions of the Ambisonic Toolkit.

This project would not have been possible without the University of Hull funding this project over the course of 3 years and providing means to attend the ISEA symposium in Istanbul. I am grateful for the financial support from the School of Arts and New Media for attending conferences - NIME in Oslo, IFIMPAC in Leeds, and ICMC in Ljubljana - which has provided opportunities to present aspects of this research in the UK and abroad. These experiences proved to be instrumental in shaping the final outcome of this project.

I owe a great deal to the dedication and brilliance of the performers and collaborators Andrea Young, Satoshi Shiraishi, and Yota Morimoto, the latter two also for sowing the seeds for my interest and passion for audiovisual performances while working together on *ibitsu*.

I am also very grateful for the support of many people at the School of Arts and New Media. I would particularly like to mention Helen Mitchell and Andrew King for kindly offering me the opportunity to teach, Maria X for providing valuable study advice, Ellie Maughan for financial advice and support, fellow doctorate students Matthew Barnard and Nick Del Nero for good company while organizing and performing at concerts in Scarborough and beyond, and the technical support from Chris Jones, Alan Young and Neill Walhurst.

I would also like to thank the IT department, particularly Philip Booth, for their last minute emergency solution for the network performance between

California and Scarborough during frantic preparations for my final performance.

This project would have been unthinkable without the amazing work by countless participants and contributors in the SuperCollider community.

Finally I would like to thank my son Thorfinn Laagus and wife Bailey Grey for their support and patience. I would particularly like to mention my gratitude to Thorfinn for providing invaluable help coding the systems and converting drum rhythm patterns from box notation into SuperCollider arrays.

Abstract

We live in a cultural environment in which computer based musical performances have become ubiquitous. Particularly the use of laptops as instruments is a thriving practice in many genres and subcultures. The opportunity to command the most intricate level of control on the smallest of time scales in music composition and computer graphics introduces a number of complexities and dilemmas for the performer working with algorithms. Writing computer code to create audiovisuals offers abundant opportunities for discovering new ways of expression in live performance while simultaneously introducing challenges and presenting the user with difficult choices. There are a host of computational strategies that can be employed in live situations to assist the performer, including artificially intelligent performance agents who operate according to predefined algorithmic rules. This thesis describes four software systems for real time multimodal improvisation and composition in which a number of computational strategies for audiovisual laptop performances is explored and which were used in creation of a portfolio of accompanying audiovisual compositions.

Contents

List of Figures

1 | Introduction

> "The effective complexity of the universe is the length of a concise description of its regularities. Like the algorithmic information content, the effective complexity receives only a small contribution from the fundamental laws. The rest comes from the numerous regularities resulting from "frozen accidents." Those are chance events of which the particular outcomes have a multiplicity of long-term consequences, all related by their common ancestry."
>
> from *The Quark And The Jaguar* by Murray Gell-Mann

1.1 Project definition

The objective of this research project is to develop a multimodal autonomous improvisation system that will be able to interact in real time with human musicians. The general problem that laptop artists - who are very often programmers, composers and performers in one person - encounter is how to specify levels of granularity for musical and visual parameter control. The general problem that this project is striving to solve is how to define parameter control on multiple time scales and levels of compositional hierarchy. The motivations behind this project are complex and manifold, ranging from technological to aesthetic in nature. For one, the current state and the potential of open source software and programming tools offer a host of enticing possibilities for artists working with interactive digital media. The proliferation of interactive digital art developed with these free and open technologies might be implying shifting social attitudes towards the role of technology and their impact on society and culture. The open source aesthetic in itself has come to signify a very different social attitude in the context of the intellectual property rules of the dominant corporate socioeconomic model. In the electronic music domain it may also be considered as signaling an adjustment from the academic tradition of electroacoustic tape music towards real time and interactive models of musical composition.

The software tools themselves impact the results of the compositional process to transform from fixed and sequential towards flexible and concurrent. For example, the inherently real time architecture of the SuperCollider programming environment encourages a different method and aesthetic of composition compared to its non-real-time precursors. The idea of an electronic composition, formerly understood as a fixed, static recording achieved by deterministic accumulation of sequences of non-realtime operations, acquires a different meaning; the decision-making of the composer becomes inherently an interactive real time process and the result ceases to be fixed. Instead, the focus of technical and aesthetic interest shifts to the process of immediate interaction more characteristic of the tradition of improvisation than composition in the classical sense. Since the architecture of the technology is acknowledged to be an inevitable influence on the most basic level in the creative process and the aesthetics of the results, evaluation of the nature and purpose of technology used becomes an essential part of the project.

During this practice-based project, a portfolio of audiovisual works has been produced, evidenced by performances, presentations, the computer code written to create the content and various recordings, in both video and audio formats.

1.2 The portfolio of audiovisual works

The portfolio of creative works that this written commentary serves to support essentially consists of 4 computational performance frameworks:

- **f(x)** is an interactive performance environment that provides a real time interface to controlling emergent behavior in generative computational systems and facilitates a performance as an exploration of mapping computational concepts to sonic and visual parameters. The software and hardware used to implement and deploy f(x): SuperCollider[1] for audio synthesis and system management, Objective-C, Liblo C library for Open Sound Control (OSC) communication protocol and OpenGL for the precompiled graphics application, Korg NanoKONTROL MIDI interface and TouchOSC application on iPhone for external control. There are 6 recordings included on the accompanying medium:

[1] http://supercollider.sourceforge.net/

- zone 00
- zone 01
- zone 02
- zone 03
- zone 04
- zone 05

- **mikro** embodies an investigation into the extent to which the compositional process can be automated using techniques from machine learning, complex systems and artificial creativity. The system has been implemented in three different versions as the search for the suitable graphics environment saw the progression from Objective-C to raw C++ to the Cinder[2] C++ graphics library. The audio and all management functions were implemented in SuperCollider. There are 4 recordings included:
  - mikro:strukt - eCLAMBONE performance by Satoshi Shiraishi with machine improvisation
  - mikro:skism - eCLAMBONE performance by Satoshi Shiraishi with machine improvisation
  - bocca/mikro - voice performance by Andrea Young with machine improvisation
  - mikro:geen - solo live coding performance using the gene expression programming synthesis definition library

- **sparsematrix** explores the idea of interfacing with a composition system of predefined event loops of musical structure by writing computer code live on stage, a practice known as live coding. Two slightly different systems have been developed to achieve this. The first incorporates sparsematrix functionality into a collaborative performance with Japanese composer and performer Yota Morimoto. The graphics functions are merged into Yota's graphics application which uses the OpenFrameworks[3] C++ libraries. The audio systems exchange information about event onsets and tempo using OSC. The second system is intended for solo live coding performances and

[2] http://libcinder.org/
[3] http://www.openframeworks.cc/

is implemented in SuperCollider and Cinder. There are two recordings of sparsematrix:

  - ◊ sparsematrix:2D
  - ◊ sparsematrix:3D

- **lambda** enables all the above concepts to be combined into an integrated performance environment and used in combination with each other. The environment was deployed for the final concert at the University of Hull Scarborough Campus in December 2012 and consisted of four sections which were presented in a continuous performance of approximately 30 minutes:
  - ◊ λ:A (a network performance with the mikro system with singer Andrea Young using jacktrip software for sending live audio over the Internet between Scarborough and Calarts in Valencia, California, USA)
  - ◊ λ:T (live coding algorhythmic structures as an interface to audiovisual performance)
  - ◊ λ:G (collaborative performance with the mikro system and Satoshi Shiraishi on eCLAMBONE)
  - ◊ λ:C (real time control of emergent audiovisual processes)

During the project, all the described performance systems were put to practice at a number of concerts and presented to audiences at festivals, conferences and symposia, both within the UK and internationally. A full list of all these is included in the Appendix A.

The practice-based research degree guidelines (included in Appendix D) define a programme of research in which musical composition forms the basis of the work in the form of a "substantial folio of compositions". However, it is not defined what exactly constitutes a musical composition. Only the total duration - in this case 90 minutes - has been specified by the guidelines, which leaves the rest open to interpretation. Traditionally musical score has been considered as the physical artifact to represent a musical composition, which usually implies a defined duration. This is even more the case in the electroacoustic context where the audio file often replaces the score. However, the approach taken in this project cultivates a rather more complex situation and does not fit into the narrow confines of these guidelines. This is partly due to the nature of digital technology and its state of evolvement which inevitably

has had a major impact on the nature and function of musical composition, but also due to the cultural context and artistic premises that have shaped the development of the work presented in this thesis. The duration as a quantitative measure of the compositions is less relevant in this context as any of the 4 environments can easily fulfill this requirement in a single performance or recording session. Rather, the emphasis is on the relevance of computational concepts to the current artistic and social environment that are actively explored in each performance in an improvisatory setting. There are a few physical artifacts that, in combination with the video recordings, help to represent the compositional processes and the resultant multimodal environments: source code files, precompiled graphics applications, audio files, web pages, articles, descriptions in different event brochures. Due to the complex nature of these systems, the accurate documentation is nearly impossible as there is a strive towards creating fully immersive environments in which the graphics projection provides the only light source, thus making video recording extremely challenging, in combination with multichannel Ambisonic audio projection. The video recordings included with this written account may go some way to representing the intentions and ideas expressed in the performances, but in no way can these fully convey the immersive first-hand experience of being present in the audience or on stage as a performer.

1.3 Methodology

This project has been carried out using two major methodological frameworks from arts and sciences, both involving creation of artifacts from different, mutually augmenting perspectives:

- *Practice-based research method* is used when some of the resulting knowledge is embodied in an artifact. It involves an original investigation to gain knowledge partly by means of practice and the outcomes of that practice [16]. The portfolio of original audiovisual works constitutes a collection of artifacts in the form of digital music and computer graphics which have been deployed at performances, presentations and exhibitions. While the significance and context of these works are described in this thesis, a full understanding can only be obtained with direct reference to the software systems used in creation of these works and the performances and video

recordings themselves. Practice-based research method emphasizes the aesthetic, cultural, artistic and social aspects of the project.

- *Design science research method* is commonly used in information sciences and historically grew out of the necessity to distinguish the scientific study of the artificial from that of the natural using the scientific method. It is essentially a paradigm for problem-solving which guides the creation of innovative information systems artifacts. It addresses the twofold nature of design science as both a process and a product [44]. The principles of design science have been followed during the design and implementation of distributed software systems that enable the creation of the audiovisual environments as cultural artifacts. The design science method provides the technical framework and methodology for the project.

It was necessary to bring these two methods together considering the interdisciplinary nature of the project which combines research objects from both artistic and scientific disciplines. The methods have been thoroughly integrated and therefore it is not necessarily possible or advisable to draw any boundaries, but there are general guiding principles that outline the two approaches. The design science method guides the building of the information systems from the software engineering perspective, while essential high-level design features of these systems are informed by practice-based artistic requirements, including knowledge representation from the domains of music composition and visual arts, human-machine interface development from performance perspective, considerations for audience experience of immersive multimedia environments, real time signal processing and statistical analysis. Design evaluation assists in determining the utility, quality and effectiveness of software systems. There are five primary methodological categories suggested by Hevner *et al*: observational, analytical, experimental, testing, and descriptive. Observational and descriptive evaluations in this thesis are included in the discussion of each performance environment in turn, while a general analytical evaluation follows in the conclusions. Practice-based evaluation methodology, on the other hand, places the research and design outcomes in the wider scholarly and cultural context.

1.4 Primary contributions

The two major contributions of this project are:

- A collection of audiovisual artifacts in the shape of public performances, presentations and recordings.
- A distributed software framework facilitating improvised audiovisual collaborative performances.

The major contributions can be broken down into a number of components which are categorized according to the general dual nature of the methodology into practice-based (or compositional) and design research based (or technical). Contributions to the methods and knowledge base of the interdisciplinary field of computer music performance and composition include:

- A method of establishing emergent complex computational behavior as an interface for improvised performances.
- Autonomous and interactive algorithmic composition methods based on statistical analysis of spectral dynamics that adopt spectral dynamics and sound field composition with the Ambisonic Toolkit as significant and primary musical parameters in composition.
- Gene Expression Synthesis (GES) as a novel method for sound synthesis.

The technical contributions are predominantly in the shape of software systems and algorithms:

- 4 distributed open source software systems for real-time audiovisual composition publicly available as source code repositories on Github[4].
- A software framework for Gene Expression Synthesis for evolving SuperCollider unit generator graph functions for sound synthesis.
- A real-time modular music analysis system which utilizes machine listening algorithms to analyze a musical performance with an emphasis on the spectral dynamics of the input audio, including a musical analysis database and the algorithms designed to store, retrieve and manage musical data.

[4] https://github.com/darkjazz

- An artificially intelligent autonomous music composition system which can be subjected to supervised training using the musical analysis database.
- Interactive OpenGL computer graphics systems based on cellular automata, self-organizing maps and swarming algorithms.

1.5 Outline

The structure of this written commentary follows from the contents of the portfolio. Following this introductory chapter, in Chapter 2 the project is placed into a wider context of scientific and artistic inquiry, introducing the background, motivations and main topics of the thesis. Chapter 3 discusses the computational and artistic principles behind *f(x)* - a performance system based on three-dimensional world of cellular automata which is controlled during live performances by customizing proprietary external hardware controllers to suit the compositional and performative needs of the environment. Chapter 4 introduces the basic principles and methods used in developing the *mikro* autonomous performance environment which is developed to analyze a live musical performance by a human performer and to generate an audiovisual response using machine listening and learning methods. Chapter 5 explores the background and techniques used in *sparsematrix*, a live coding environment based on the extraction of the most essential structural components of traditional rhythm patterns and using these as basic building blocks for spontaneous composition of rhythmic loops. Another approach to reactive computer graphics is discussed. Chapter 6 discusses a novel synthesis paradigm - Gene Expression Synthesis (GES) - based on evolutionary programming principles known as gene expression programming to evolve SuperCollider unit generator graph functions. In Chapter **??** a general discussion and conclusions of this thesis are presented.

2 | Computational strategies for audiovisual performance

This chapter introduces the motivations for and basic principles and architecture of audiovisual performance systems which seek to explore emergent musical forms in immersive interactive multimodal environments.

2.1 Introduction

The advances in computing speed and power are constantly shifting the boundaries of what is possible to achieve with sound synthesis, algorithmic composition and performance interfaces in the context of real time interactive music systems. Arguably this is the main factor behind the proliferation of real time information systems used in musical performances while the use of interactive computer graphics in live electronic music has become more of a prominent feature. Artists and musicians are constantly adapting new software tools and adopting scientific methods into their creative workflows which in turn inform, guide and influence their practices, the extent of which often unbeknown to the practitioners themselves. This project explores different computational strategies to musical composition in the real time environment, utilizing concepts and methods from different scientific disciplines, including mathematics, complexity science, artificial intelligence, machine learning, and evolutionary computation among others. The multimodal environments described in this thesis have been developed in order to investigate different computational strategies and algorithmic systems in improvised audiovisual performances.

2.2 Background

Interactively improvising computer music systems are by no means a novel idea. A number of systems have been created since personal computers have

become fast enough to handle real time analysis and synthesis, and purpose-built software packages and programming tools have become available for composers and computer music researchers. Beginning with the earliest MIDI-based systems, most notably George Lewis' Voyager [50] and Robert Rowe's Cypher [76], there has been a steady accumulation of algorithms for the myriad of tasks necessary for real time music analysis and composition. Currently the thriving areas of interactive intelligent music systems, autonomous musical agents and machine listening algorithms are attracting a steadily increasing body of researchers across the world, which means there are a variety of ongoing projects exploring the different aspects of interactive machine musicianship. In the context of the current project, the previous work of a number of researchers, most notably the autonomous agent based systems developed by Nicholas Collins [21], is invaluable. Since this project explores the notion of improvisation in the context of computer-mediated performance and the strategies for algorithmic creativity, previous accounts of computer-mediated improvisation [19] and machine learning techniques in similar settings [22] have in many ways helped inform the methodology implemented in the systems described here. The concept of expanding the interactive musical foundation into a multimodal presentation has been around just as long. Many computer music systems have been adapted for multimedia purposes, one of the earliest and most notable being the HARP system by Antonio Camurri and his colleagues [15].

The interest in interactive multimodal performance environments has naturally evolved from my involvement in a number of previous musical endeavors. There has been a gradual transformation in my personal practice from electroacoustic composition to live performance while trying to reconcile divisions between different musical traditions, specifically the tradition of art music (electroacoustic, Western classical music) and popular musics (jazz, improvisation, different forms of electronic dance music). The elevation of Western classical above any other kind of music, for example typified by the attitude towards jazz in *Philosophy of Modern Music* [1], has served to artificially and unnecessarily isolate musical traditions and practices. A more recent illustrative example of this division was the exchange of critique between Karlheinz Stockhausen and electronic music producers Aphex Twin, Plastikman, Scanner, and Daniel Pemberton, known as Stockhausen vs the "Technocrats",

which first appeared in 1995 in the Wire magazine [24]. Similar kinds of attitudes have prevailed for long periods and to a certain extent unfortunately continue. One of the arguments often voiced seems to stem from what can perhaps be described in the context of listening modes, implying a division between an intellectual and impulsive activity. The analytic or *allocentric* listening mode [81] is considered a better, higher form of music making, but this division from autocentric - a mode centered in emotions - is artificial and the mistake is to cast the musical process in absolutes.

Figure 2.1: Audiovisual performance by *ibitsu* at STEIM, Amsterdam in 2006.

The main musical inspiration behind the multimodal environments discussed in this thesis comes from a few sources and is directly related to personal experiences and education. My preceding involvement in the world of electroacoustic composition and computer music has provided a foundation for a systematic approach to algorithmic composition and sound synthesis. Studying with composers Richard Karpen, Juan Pampin and Bret Battey at the University of Washington in Seattle provided an opportunity to learn the fundamental aspects of computer music and electroacoustic composition particularly the practices and traditions of computer music in the United States. The main influences that have endured from that period include appreciation for the

contributions of most notably Jean Claude Risset, John Chowning, Paul Lansky to name just a few, and the emergence of SuperCollider as a preferred environment in which to research, compose and perform music.

The gradual shift towards live performance was primarily inspired by the fact that real time processing is the main strength of the SuperCollider environment, which encourages the user to adopt a more interactive strategy to composition, compared to the previous generation of programming tools such as Csound or Common Lisp Music in which rendering an audio file to disk was still the prevailing method. This coincided with my active involvement with the Share digital arts community[1] in New York and various projects at the Institute of Sonology in the Netherlands, both of which in their own way emphasized the live aspect of electronic music and audiovisual interactive art. Performances as a member of multimedia collective *ibitsu* shaped in many ways the approach of building up sonic material from a single acoustic source by processing the instrumental input in different ways and then sampling the processed output during a performance. These samples are then used to develop the sonic material of the performance in cycles of re-recording and replaying, while the visual content is entirely derived in an analogous way by capturing and processing the live video feed which becomes a continuous feedback cycle. Figure 2.1 shows a still from a performance recorded at STEIM in Amsterdam with Satoshi Shiraishi performing on e-CLAMBONE and Yota Morimoto in charge of live visuals.

Liveness in electronic and computer music has been an ever-present topic of discussion over the course of a musical career which includes electroacoustic and computer music composition, producing and DJ-ing electronic dance music, free improvisation in traditional jazz settings as well as more exploratory electronic environments, and audiovisual laptop performances and installations among others. Since computers enable intricate control over different levels and time scales of composition, the number of musical parameters to be controlled in a real time situation can become unmanageable unless some sort of a strategy is adopted for parameter grouping or delegation of control to algorithmic agents. Before discussing the particular strategies chosen in this project, the basic architecture of audiovisual performance systems should be described.

[1] http://share.dj

2.3 Basic architecture

The basic architecture of the systems discussed in this thesis and thereby the nature of the audiovisual performances these systems facilitate derives from the way the programming environments used in this project function. The SuperCollider programming environment [55] provides the foundation for all the systems built during this project. The architecture of this environment is the result of evolution of music programming languages beginning with the Music I language developed by Max Mathews in late 1950s [54] and represents a particular strategy towards encapsulating digital signal processing and higher level musical concepts. Naturally there are many alternative ways to represent different levels of signal processing, synthesis and composition. There are two main factors that have shaped the development of audiovisual performance environments described in this account. First is the modular structure caused by the physical separation of different components. The SuperCollider environment consists of two applications: the audio server and the programming language, which in turn includes the language interpreter and the interpreter as a client to the server. The communication between server and client happens through the Open Sound Control (OSC) protocol for exchanging musical information over a TCP or UDP network. This architecture enables the server and client to be used separately although normally they still reside on the same physical machine.

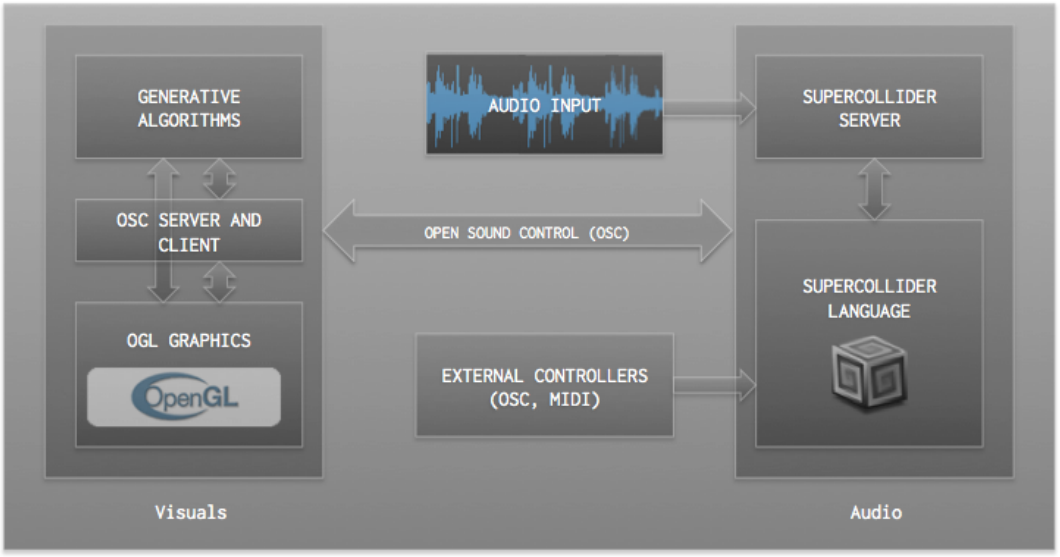


Figure 2.2: Basic architecture of an audiovisual performance system.

The second factor is the way in which audio synthesis and signal processing is represented and calculated on the audio server. The Gene Expression Synthesis (GES) method introduced in Chapter 6 explores the language level representation of signal processing unit generators. It is an evolutionary programming paradigm which evolves candidate unit generator graph solutions in 4 stages: first the unit generators and variables are encoded as linear chromosomes, then decoded into graph structures as objects, third step involves converting the graph structures into text strings which are then interpreted by the SuperCollider language interpreter and then sent to the server for further analysis. In a similar way to Trigger Feedback Networks (TFN) [3], the GES method explores the inherent structure of the SuperCollider environment simultaneously to implementing a more general genetic programming paradigm. TFN, for example, make use of the routing system, feedback and the way triggers function in real time on the SuperCollider server while GES relies on the structural idiosyncrasies of SuperCollider UGen graphs. Neither system would exist in their current shape without the specific structural features of the environment and thereby are both good examples of how the structure and idiosyncrasies of a software tool shape computational and aesthetic choices of the user. In a very similar manner, any user of Max/MSP, ProTools or Ableton Live, for example, are confined to the particular interfaces and musical concepts that the creators of the respective software tools have chosen to implement.

The third factor for the distributed architecture is the lack of efficient graphics algorithms available to use in an integrated way in the SuperCollider environment. Open Graphics Library (OGL) has been arguably the fastest and most advanced graphics tool freely available for many years and is widely used as the foundation for a number of applications ranging from scientific data visualization to 3D animation in the games industry. Despite there being plenty of alternatives for developing real time rendering applications for computer graphics (e.g. Processing, GEM, Jitter, Quartz Composer), none offer the same level of flexibility, computational efficiency and control as OGL.

The modular structure utilizing network technologies enables deployment of the performance environment on physically separate computers. This enables implementing computationally more expensive strategies and algorithms for performance and thus higher complexity. Another feature that emerges

from the distributed nature is the non-linear way of mappings between the audio and the visuals as the user is encouraged to consider the modalities as independent from each other due to the physical separation. An alternative avenue for exploration is deploying the system distributed between distant geographical locations for network performances during which participants send musical and graphics control data to remote performance spaces over the Internet.

2.4 Compositional strategies

In the increasingly inter-disciplinary and technology driven world of musical performance, computational strategies for composition and machine interaction constitute an essential ingredient in artistic exploration as artists and composers adapt and incorporate computational methods together with technological innovations into their creative workflows. This thesis explores a number of such strategies in the context of live performance with laptop computers and explores the practice as a form of virtual reality generation. Musical composition is considered in the broadest context possible of sonic art where a composer-performer acts more in the frame of Trevor Wishart's sonic designer [92], than the archaic and increasingly redundant notion of notatability of a small subset of musical parameters. In this context the traditional division between composers and performers is no longer clearly defined. Musical composition forms the fundamental corner stone of immersive audiovisual environments the performances described in this thesis produce.

The role of the composer/performer or an artist-programmer, to borrow a term from Alex McLean [58], using this architecture to create music and graphics by creating, controlling and modifying computer algorithms during a performance is distinct from the traditional paradigm of composer and performer as separate entities. An artist-programmer operating in the realm of these audiovisual systems explores the multi-layered parameter space through algorithms in a manner more akin to an improviser than a traditional composer or a performer who follows predetermined instructions. Even though many of the musical ideas as well as signal processing concepts embedded in these systems have been shaped by the electroacoustic and computer music traditions, there is an underlying principle which strives for the immediacy of real

time decision making. This is as much a feature of the SuperCollider environment as it is an aesthetic of improvisers.

Computational strategies for audiovisual performance with the latest digital technologies in the context of our current understanding of our environment are complex and manifold. The composer-performer has such a vast selection of software and hardware tools to choose from while having to consider the implications from many fields of research to the performance paradigm: cognitive, perceptual, evolutionary, physical and many more. Algorithmic strategies can assist in the decision making process of the performer faced with the exceedingly complex environment of choices. One of the strategies explored in the *f(x)* performance environment, described in more detail in Chapter 3, delegates control of groups of synthesis parameters to a three-dimensional lattice of cellular automata, known as a "world". This cellular automata algorithm defines local rules of interaction between neighboring items for a number of adjacent cells on a 3D lattice. These local rules from the perspective of each individual cell do not have any direct influence on other cells which are not immediately adjacent. When the rule is applied to all cells simultaneously, in the same defined time frame, a global pattern of behaviors results as an emergent phenomenon over a number of such discrete time steps. Each cell has a state assigned to it at any given time step, most commonly a numerical value, which is modified depending on the numerical values of adjacent cells according to a uniform rule across the entire automata world. These laws of interaction in the virtual automata universe can spawn complex patterns of behavior, some of which may even exhibit computational universality (the concept of universality is discussed in more detail in Chapter 7).

The automata values can be mapped to sound synthesis or graphics parameters and this strategy seems to fit situations where there is a need for controlling large numbers of parameters dynamically changing over small time frames. Granular synthesis is one such paradigm where individual control of each parameter of every sonic particle is overly cumbersome, time-consuming and downright impossible in a real time performance context. Yet the various microsonal techniques, extensively covered by Curtis Roads in *Microsound* [72], enable detailed control on the smallest time scale of synthesis and composition. Granular synthesis is used in some shape or form virtually in every synthesis process of this project. The different strategies employed in com-

bining this synthesis technique with a selection of generative algorithms originate from and expand on the ideas of evolving sonic screens by stochastic rules outlined by Iannis Xenakis in *Formalized Music* [96]. Cellular automata provide the source material for audiovisual explorations of Japanese artists Keiichiro Shibuya and Takeshi Ikegami [78] as well as Yota Morimoto's synthesis unit generators for SuperCollider [65]. Cellular automata have been used in different configurations and definitions throughout the project for both audio synthesis on micro level as well as composition of sonic structures on larger time scales, while non-linear mappings to 3D graphics functions emphasize the discreteness of the two worlds.

Evolutionary algorithms offer another computational method for exploration of problem spaces by evolving candidate solutions and evaluating their fitness according to how well that problem is solved. This is achieved by using techniques that are inspired by genetic operations, including inheritance, mutation, selection and crossover. Biologist Richard Dawkins discusses an evolutionary algorithm that evolves virtual creatures he calls biomorphs in his book The Blind Watchmaker [28] to illustrate the power of cumulative selection in evolution. It serves well as an explanatory device, however, one must bear in mind the unimaginably higher complexity of natural processes operating on vastly larger time scales when alluding to similarities to this artificial form of simplified imitation. The most widely used methods of evolutionary computation are genetic algorithms (GA), which typically consist of two parts: an encoded string or genotype and a decoded solution or phenotype. Genotype is often represented as a string of 0s and 1s, which translates to a computer program that calculates a solution to a problem after the decoding process [61]. Each solution in a population of individuals or chromosomes is assigned a fitness score according to how well the resulting program solves a particular problem. Genetic algorithms have been widely experimented with in the context of sound synthesis and music compositions for a few decades already. Some of the examples include Gary Lee Nelson' Sonomorphs [66], coevolution linking artificial critics and composers as an example of automating the fitness function in the context of musical creativity [85], MutaSynth for genetic sound synthesis [26] and John Biles' GenJam system for generating jazz phrases in real time interaction [14]. This is by no means even close to a comprehensive list, but rather a brief acknowledgement to the volume of exciting research

that has been undertaken.

Cellular automata, genetic algorithms and other complex systems can be employed purposefully in real time audiovisual composition by embedding them in autonomous intelligent agents as compositional strategies. According to the definition in Russell and Norvig [77], an intelligent agent is an autonomous entity that observes certain aspects of its environment, may learn about the observations by analysis and responds to achieve a predefined goal. The extent of complexity of an agent can vary from a simple reflex machine (for example the *f(x)* environment can be considered a complex reflex machine) to an autonomous composer agent that learns musical structure from examples, stores and analyzes the data and uses it to compose phrases of music in response to real time analysis of an incoming audio stream (just like the improvising agents built into the *mikro* environment).

This thesis explores three ways of real-time interaction during a performance with a computational system which consists of two physically separate applications, one for audio synthesis and composition and the other for computer graphics:

- external parameter controllers including MIDI interfaces and Open Sound Control (OSC) enabled mobile devices
- autonomous agent based approach using machine listening, statistical signal analysis and machine learning methods.
- exploring the audiovisual parameter space by writing computer code live on stage known as live coding

Ultimately these three approaches have been integrated into a single environment *lambda* so they can be used and overlapped in the same performance. The three original prototypes - *f(x)*, *mikro*, and *sparsematrix* - have served the purpose of addressing different strategies of composition and interaction individually and enabling undivided focus on specific strategies. The interactive OpenGL graphics applications incorporated into these performance environments are discussed in the respective chapters in the context of computational strategies and algorithmic visualization rather than from the aesthetic and historical perspective of audiovisual composition. This is primarily due to realization that the topic deserves a significantly more rigorous investigation than the duration and scope of this project have afforded. The emphasis

has been on the development of the sonic aspect of the multimodal environments. Before discussing each environment in detail, sound spatialization as an essential element of all the sonic environments described here warrants a brief introduction.

2.5 Virtual acoustic spaces

Multi-channel sound projection enables designing virtual sound spaces and provides the artist-programmer an entire new world to be explored. Space is just as an important a musical parameter as time, spectral dynamics and more traditional concerns with harmony. Ambisonics surround sound spatialization technology provides a high quality and yet practical solution for performance systems to enable authoring immersive sound fields in real time as an essential component of a performance. This section explores the sonic space in the audiovisual context with particular consideration for soundfield imaging with surround sound tools for creative practices.

A handful of tentative experiments aside, sound space as an essential compositional element first emerged in Karlheinz Stockhausen's piece *Gesäng der Jünglinge* which premiered at Westdeutscher Rundfunk (West German Broadcasting) in Cologne in 1956. This momentous occasion was followed by the showcasing of Edgar Varese's *Poeme Electronique* over 400 loudspeakers in the Philips pavilion at the 1958 Brussles World's Fair. Since then, composers mainly associated with electroacoustic and computer music have explored many techniques and technologies for sound spatialization, often involving an impressive array of loudspeakers and necessary supporting technology to reproduce and modify previously recorded soundfields or to synthesize virtual ones. There are many specialized high quality systems for spatialization, for example, the Sonic Lab at Queen's University in Belfast[2], BEAST at the University of Birmingham[3], or the Wavefield Synthesis (WFS) system in Leiden, the Netherlands[4]. However, these kinds of systems are often either immobile and thus site-specific or excessively cumbersome to transport. At the same time, interfacing with these systems for performance purposes can be complicated and require considerable setup and rehearsal time, especially in the context

[2] http://www.sarc.qub.ac.uk/sites/sarc/AboutUs/TheSARCBuildingandFacilities/TheSonicLab/

[3] http://www.birmingham.ac.uk/facilities/BEAST/index.aspx

[4] http://gameoflife.nl/

of live performance. Ambisonics provides a convenient yet powerful alternative for sound spatialization. One of the main advantages of this technique is the encoding of the audio signal in a way that encapsulates the spatialization information independent of the replay system. As a consequence, Ambisonic technologies facilitate focussing on the spatial aspects of sound without unnecessary concern for technological details such as loudspeaker configuration.

Ambisonics, meaning "surround sound", was first developed by a group of researchers, lead by Michael A. Gerzon in the 1970s as a recording and playback system [40]. The Ambisonic signal is encoded in four channels in the same way the Ambisonic array of microphones would record it, regardless of whether it is actually recorded or synthesized: three figure of eight microphones capturing the sound energy along the 3 spatial axes of front-rear (X), left-right (Y), top-bottom (Z) and an omni-directional microphone for overall sound energy. This most basic encoding, known as first-order Ambisonics, has been used throughout the project.

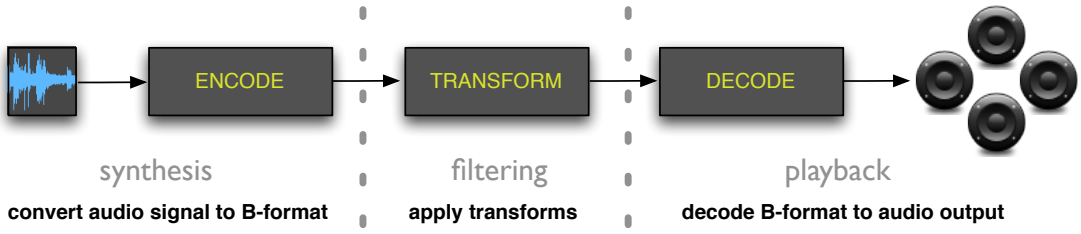


Figure 2.3: The three phases of Ambisonic spatialization

The Ambisonic Toolkit[5] (ATK), a plugin system for SuperCollider authored and developed by Joseph Anderson, provides one of the most comprehensive set of Ambisonic tools available. It is designed to provide user-friendly access to a comprehensive set of Ambisonic encoders, transforms and decoders. The toolkit emphasizes the concept of creating and shaping whole soundfields rather than placing individual sounds in virtual space. There are three phases in the process of working with synthesized soundfields in ATK, illustrated in Figure 2.3: conversion of monophonic signal to Ambisonic B-format, spatial filtering, and decoding.

The compositional concern with spatial experience derives from the practice of acousmatic music, but is no longer confined to that tradition alone. Acousmatic once believed to be the only sonic medium that considers space

[5] http://www.ambisonictoolkit.net/

as aesthetically central [82] is arguably no longer in this forsaken predicament. Ambisonic technologies like the ATK provide a laptop performer with an exhaustive array of creative tools regardless of the performance context. Nowadays space as a musical parameter is being explored at free improvisation gigs in jazz clubs as well as a live coding dance music nights in the club environment. Admittedly some of the performances during this project have taken place in institutions and concert spaces associated with the tradition of acousmatic music. However, the improvisatory nature combined with the visual element sets them apart from the acousmatic.

3 | Interactive composition: *f*(x)

f(x) is an audiovisual performance environment to enable exploration of continuous spatial functions derived from the model of continuous spatial cellular automata. The time-varying functions provide the basis for sound synthesis and computer graphics parameter mapping. The audio and the visuals are independent from each other both physically and conceptually and the reciprocal influence flows in both ways in a non-linear manner. The segmentation of the visual space and acoustic time is controlled in the performance by affecting the behavior of the automata world in real time in an attempt to reveal the complex and organic behavioral patterns in three dimensions and modifying the mapping space in response to them. The audio synthesis, computer graphics and the performance interface have all been developed in open source software. Audio synthesis and performance interfaces are implemented in the SuperCollider programming environment, graphics functions in OpenGL and Open Sound Control (OSC) protocol is used for communication between the audio and visual applications.

3.1 Introduction

Digital technology has provided an incredible variety of opportunities for artistic exploration and has fostered a new perspective on human culture and society. It has forced scientific methods and concepts into the working process and aesthetic framework of an artist due to its very nature. The beginnings of the digital computer are inseparably connected to research into the biology of self-replication and the possibility of artificial life. The work of Alan Turing, John von Neumann, Stanislaw Ulam and many others was essential to the way the modern world operates and the fundamental concepts based on the spectacle of biological evolution and natural selection have been integrated into every piece of digital technology with which we have surrounded ourselves. However, we generally have little understanding of the nature of this technology

and the long-term impact it has on every aspect of our existence. *f*(x) is a performance environment created to reveal some of the aspects and principles of digital technology. It is based on a concept that has only been made possible with the advent of computers - cellular automata - and was born out of research into artificial self-reproduction. It is designed as a live audiovisual experiment in which the performer interacts with a world of 3-dimensional spatial functions defined - analogously with the principles of cellular automata - in terms of each other. The performance is seeking to reveal complex patterns of behavior, generated by relatively simple instructions and rules that would uncover some of the elusive characteristics of digital media surrounding us now in almost every situation.

3.2 Cellular automata

Cellular automata constitute an area of research belonging to the interdisciplinary field of complex systems science. The beginnings of the research in complexity can be considered as coinciding with the advent of the field of biology in the works of Jean Baptiste Lamarck and Gottfried Reinhold Treviranus in the beginning of 19th century. The idea that living organisms are more complex - or in other words having a higher degree of organization - than inorganic systems is not a modern concept. In the middle of the twentieth century two independent lines of research were started that have greatly shaped our understanding of complexity. In molecular biology, it became evident from the research that led to the discovery of the chemical structure of DNA, that every living system is highly organized and this organization is coupled to a complex molecular apparatus, which functions as a stored information code for regeneration. There are two complementary modes of existence embedded in each complex living system: the internal physical-chemical workings of a cell and the informational mode where information is selected, stored, and interpreted by the physical actions. In early computer science research an investigation was launched into the nature of complexity in general in a computational or mathematical sense. John von Neumann initiated the study of self-replicating automata in the 1940s. He recognized the dual functioning of information that self-replication requires. The two modes are present in any cellular automata system as well: the dynamics of an automata system and

the rules that evolve it [67]. In 1962, Herbert Simon [80] proposed four important aspects of complexity: hierarchy, evolvability, near-decomposability, and descriptive simplicity. Complex systems often take a form of a hierarchy, which is composed of subsystems, which in turn are composed of their own subsystems. Such hierarchic systems evolve faster than systems of comparable size with no internal hierarchy. Near-decomposability refers to the fact that interactions between subsystems are not substantial, but not negligible. The system components do not have a great effect on each other's short-term behavior, but depend on all other components in the long run. Descriptive simplicity derives from the generally high degree of redundancy in hierarchical systems that allow "chunking" up the description of its parts and describe it in simple terms. In addition to the four characteristics proposed by Simon, complex systems are thought to exhibit these additional characteristics:

- Simple laws can generate complex behavior.
- Time-asymmetric self-organization is a real phenomenon of the physical universe.
- Spontaneous pattern formation as an emergent property, not trivially predictable.
- Complex emergent phenomena can be simulated by a computer.
- Complexity reflects genotype-phenotype duality.
- Complexity is located between high order and high randomness - "the edge of chaos".
- Complexity needs to be explained in different terms from traditional reductionist methodology.

Cellular automata are deterministic dynamical systems, which are discrete in space and time, operate on a uniform lattice and are characterized by local interactions. An n-dimensional array of cells on a grid that evolves a number of discrete time steps according to a set of rules based on the states of neighboring cells. Every cellular automata system consists of four basic components:

1. The grid, which in computer terminology can be defined as an array. The simplest grid is equivalent to a 1-dimensional array. In two dimensions

square grid is the most common one, although triangular and hexagonal grids may be considered. Each slot in the grid represents a cell.

2. The number of distinct states - k - must be specified. The simplest automata are defined in terms binary states - 0 or 1. The states are usually designated by integers, although in case of continuous automata float values can be used.

3. Neighborhood over which cells affect each other. The simplest case is the nearest neighbors of a cell on a one-dimensional grid. In two-dimensional cellular automata the most common neighborhoods are the Moore neighborhood, which considers eight adjacent cells including the diagonal ones and the Von Neumann neighborhood, which excludes the diagonal neighbors (Figure 3.1)

4. Rule space. Number of possible rules N depends on neighborhood size r and number of allowed states for each cell k:

$$N = k^{k^{r+1}}$$

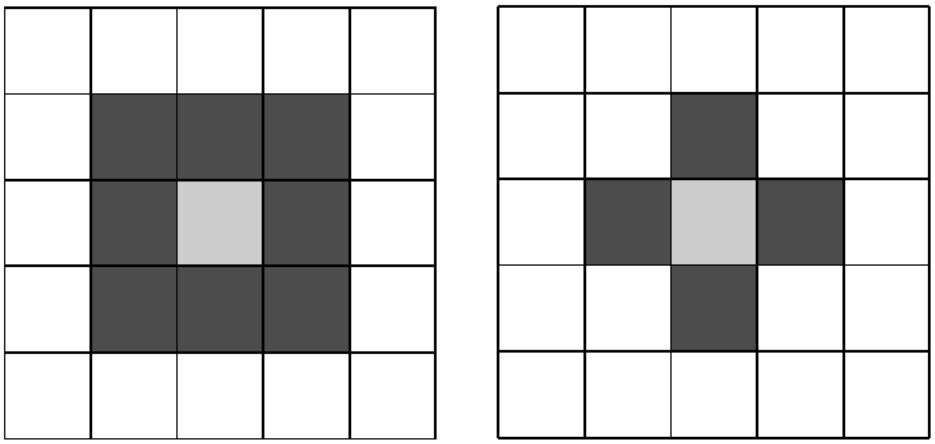


Figure 3.1: 2-dimensional automata neighborhoods.

In case of one-dimensional grid with nearest neighbor rules and binary states (i.e. r = 2, k = 2), there are a total of

$$2^{2^3} = 256$$

rules possible. The rules are encoded according to the decimal representation of the binary string of the rule. For example in case of rule 90 the transition

occurs according to the scheme illustrated in Figure 3.2:

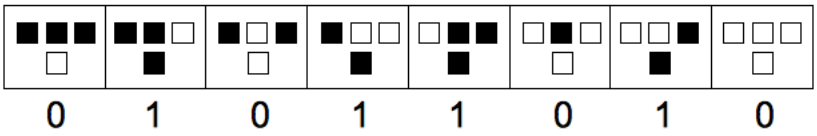


Figure 3.2: Rule 90 transition scheme.

Stephen Wolfram [94] has determined four general classes of behavior in which the evolution leads to: Class 1: homogenous state - the behavior is very simple and almost all initial conditions lead to the same uniform state after just a few generations. Class 2: a set of stable periodic structures, which are separated and simple. Class 3: chaotic pattern - the behavior is more complicated and seems random in many ways, although certain recurring structures are always present. Class 4: complex structures - a mixture of order and randomness, in which relatively simple structures move and interact in complicated ways.

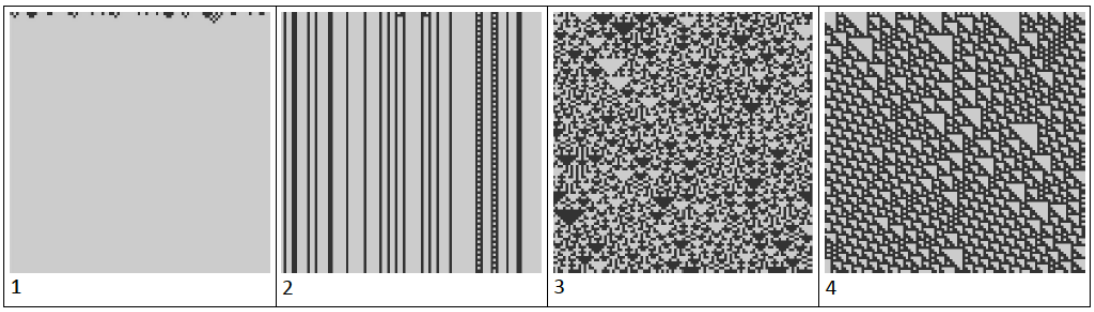


Figure 3.3: Cellular automata classes.

In complexity science in general the fourth class exhibits behavior that has been described as on "the edge of chaos". Although the term has been a center of a heated debate and conflicting interpretation, it is commonly believed to refer to behavior that is characteristic of living systems. It should be noted here that the above classification is fairly subjective, based on empirical observation rather than mathematical analysis, and, therefore, can be considered more of a working guideline than a definition.

Expanding the cellular automata concept to two-dimensional lattices indicates a significant increase in the number of possible rules and neighborhood configurations, which makes any systematic analysis increasingly difficult. For example, if possible cell states remained limited to binary [0 1] and neigh-

borhood was defined as what is known as standard Moore configuration or a 9-neighbor square, the number of possible rules would be 2^{512}, which is approximately 10^{154}.

3.3 Continuous automata

So far each location on the lattice has been considered assuming one of predefined discrete values represented normally by an integer. It is also possible to contemplate systems where the values are not discrete, but continuous within a specified range. In such systems, the value of each cell can be defined by calculating the average state of the defined neighborhood [94] [52] . In this scenario, it becomes necessary to either multiply the average of the neighborhood of each cell by a fraction greater than 1 - e.g. 3/2 - or add a value typically between 0 and 1 and then only use the fractional part of the result for the system to exhibit complex behavior similar to the elementary automata discussed in the previous section. Otherwise the averaging process will quickly settle the lattice into a uniform value all across. The multiplier or the added value becomes the determining factor in the behavior of the automata on the global level, with smaller values producing an overall smoother transition effect. In the image below, a close-up of a continuous-valued one-dimensional automata world is shown. The initial state is defined as all 19 cells being in state 0, except for the one in the center, which has a state value of 1. The constant value being added, after 2 nearest neighbor and the cell states have been averaged, is 0.1. Please note that the displayed values have been rounded to 1 decimal point for legibility.

The simple mechanism is evident when considering the calculation of center cell state from the 1st time step to the next and the role of the constant addition rule from 9th time step to the next. It is easy to see how the cell state progresses from initial value of 1 to 0.4: (0 + 1 + 0 / 3) + 0.1. Then increasing gradually to 1 because of the addition. Once the upper boundary is reached, the calculation yields a value of 1.1, of which only the fractional part is used in the next time step, resulting in the cell state of 0.1.

The images in Figure 3.5 show worlds of 60 cells each, with the same constant addition value of 0.1, but with different initial conditions: single cell initial condition on the left and random initial condition on the right.

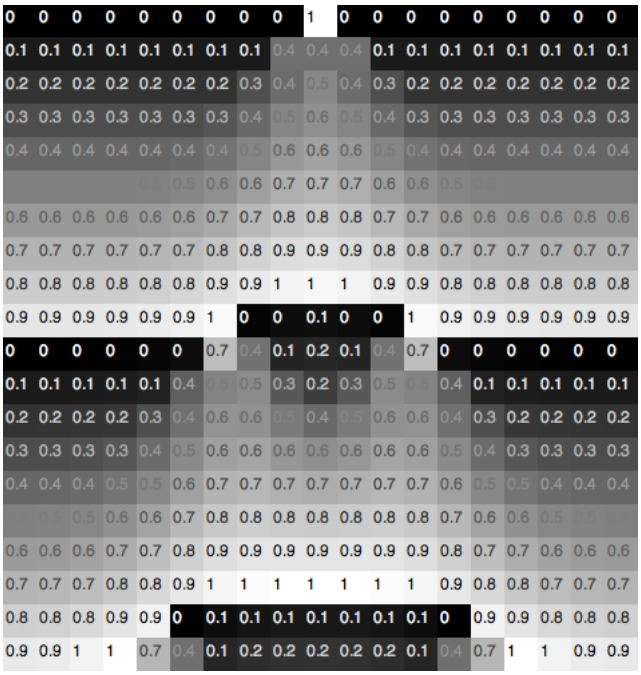


Figure 3.4: Illustration of continuous automata calculation procedure.

The behavior of the continuous-valued world can be further affected by assigning weights to each of the cell states in the neighbors, making variable the amount of influence each cell has in the neighborhood. In this case, weighted mean value is calculated for each cell when updating the states. The state s of a cell at position i in the one-dimensional lattice is calculated by

$$s_i = \frac{\sum^n w_j h_j}{\sum^n w_j}$$

where n is the size of the lattice, h denotes the values of cells in the neighborhood and w represents the weights assigned to the respective cell state values. Weighted automata rules increase in significance as the automata world is extended into higher dimensions as the geometrical properties of the system become more pronounced. The images in Figure 3.6 demonstrate the effect in one-dimensional lattices with different initial conditions. In the image on the left, the weights assigned to the 3-cell neighborhood configuration are specified as [1, 3, 2], while the add parameter is 0.1. In the second image, the weights are [1, 2, 3] and the add parameter is 0.95.

The same basic principles can be applied when expanding the automata lattice to two dimensions. Due to the inherent representational difficulties in the case of higher dimensional time-evolving systems, it is hoped that the above demonstrations using one-dimensional lattices will suffice to reveal the basic principles and dynamics of evolvement of continuous-valued cellular au-

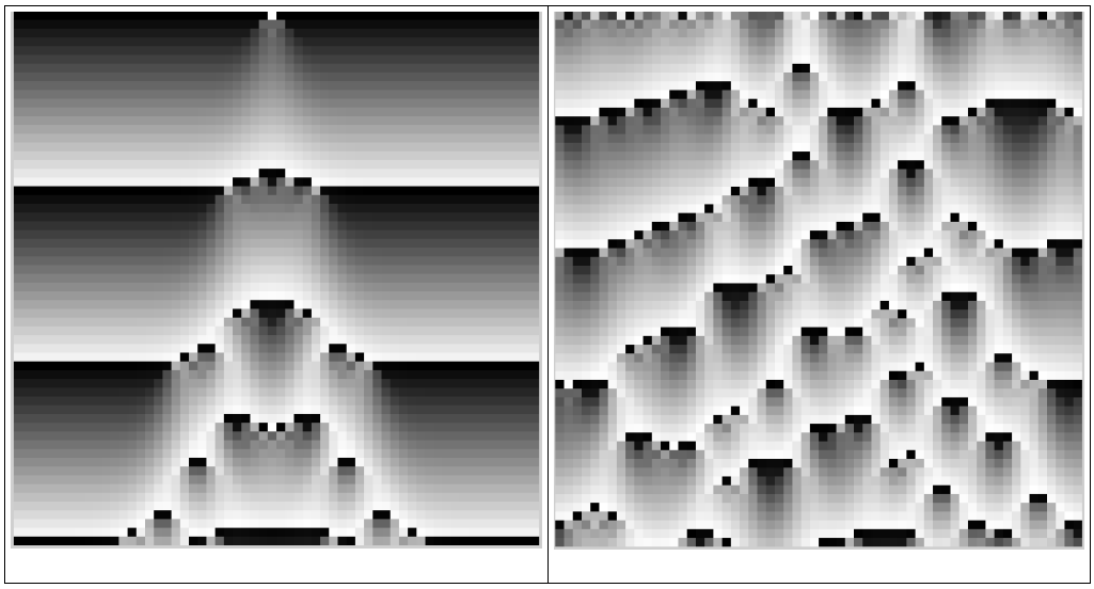


Figure 3.5: Continuous automata examples employing the same constant addition parameter (0.1), but with different initial conditions.

tomata worlds. There are, naturally, significantly more complex geometrical phenomena that emerge in the patterns created in higher dimensions and the behavior is even harder to quantify. In all the previously discussed worlds, the border conditions are handled by wrapping the edges over to the other side of the world. By this convention, in one-dimensional lattices, the nearest neighbor to the left of the first cell is the last i.e. the rightmost cell on the lattice. A two-dimensional world can in this context be viewed as having a shape of a torus as each of the edges wraps around to the opposite side.

Cellular automata have been frequently used in sound synthesis and composition processes previously. The different strategies employed have ranged from driving sample level synthesis [65] or parameters for granular synthesis [4] [60] to mapping states and coordinates to compose musical phrases [11]. There have been attempts to engage 3-dimensional automata in the process of composition, parameter mapping and performance previously, for example, *calories3D* cellular automata simulator and visualizer [89]. *f(x)* seeks to bring cellular automata into the realm of audiovisual performance and control the rules, the emergent behavior and audiovisual mappings in real time.

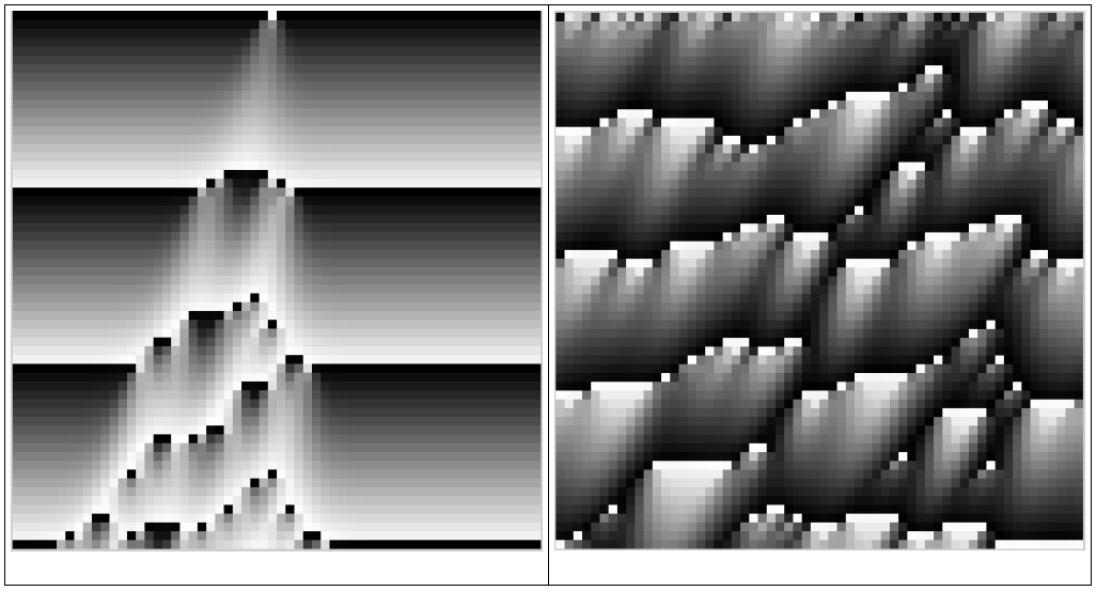


Figure 3.6: Weighted continuous automata examples employing different constant addition parameters and different initial conditions.

3.4 Structure of the performance environment

The concept of continuous-valued cellular automata is further expanded to the third dimension in the audiovisual performance environment *f*(x). One can imagine a cube-shaped region of three-dimensional space sectioned into smaller cube-shaped sub-regions, each of which has an associated state value and is defined in terms of surrounding space. The shape of the surrounding space depends on the neighborhood definition and is further molded by the weights assigned to each neighboring region. Thus these spatial fluctuations can be viewed as continuous functions permeating a region of imaginary space while being defined in terms of each other. This three-dimensional world forms the foundation of the performance environment. The environment facilitates a real-time exploration of the evolving spatial patterns by modifying the global addition value and the shape of the neighborhood and enables the user to manage predefined computer graphics algorithms and audio synthesis definitions in response to the automata behavior.

The system has been implemented in two physically separate parts. The basic architecture has been described in Chapter 2. The automata world consists of a three-dimensional array of cells, the actual number of cells in each dimension can be modified at any time during the performance. This enables finding a compromise between limits on computational power and

complexity of automata behavior, which increases as a function of world size. For example, a world that contains 20 cells in each dimension, a workable configuration under most conditions, is comprised of 8000 interacting cells in total. The definition of neighborhood shape is typically static and defaults to the standard Moore neighborhood, which in 3D includes 26 neighboring cells. The effect each neighbor has on the global behavior is managed by weight values specified for each neighbor cell. The 3-dimensional world is demonstrated in Figure 3.7. The image on the left represents all cells in the world as wireframe cubes, while the image on the right shows the 26 neighbors in the 3-dimensional Moore neighborhood.

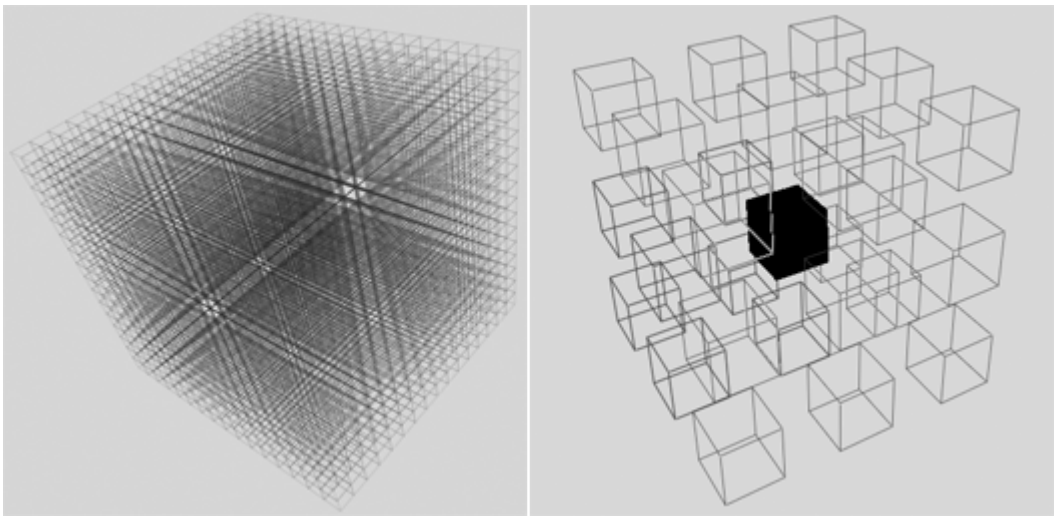


Figure 3.7: Expanding the automata world to three dimensions.

OGL drawing functions are implemented as encapsulated patches, each with a specific characteristic. These patches can be easily added and modified during development and constitute a basic unit of the graphics application. Each patch has a number of parameters for transparency, color mapping and other values that can be controlled remotely from SuperCollider (or any other OSC-enabled application). Besides the individual patch controls, the following global OpenGL parameters are remotely controllable from an external application:

- Global transparency
- Screen background color
- Zoom along each of the 3 axes
- Speed and direction of rotation

- Relative rotation angle around each of the 3 axes

Most of the functionality of the performance environment is controlled so as to make the most efficient use of the language-server architecture. It is designed to provide a flexible platform for handling OSC communications, dynamic user interface management (both graphical and external devices), and audio synthesis procedures. There are 3 types of trigger functions that can be added for evaluation and removed in real time, receiving different data from the automata world:

- global mean and standard deviation values and mean values of any polled sub-regions
- a selection of individual cell values in a specified configuration
- the x-y-z-coordinates of a cell within a specified state range

The added functions are expected to contain mapping logic from state values to audio synthesis parameters and can also be used to control certain parameters of the graphics application, creating parameter feedback loops during the performance. Audio synthesis definitions and time-structuring routines can be added dynamically during the performance either from predefined functions or specified in a live-coding just-in-time programming manner.

3.5 Audiovisual organization and mapping

The audiovisual composition relies on two elemental concepts: audiovisual macro compositional entities called zones and the separation of audio and visual content within these entities both physically and conceptually. The concept of a zone emerged in the process of developing this project as a means to have a meaningful segmentation of considerably different audiovisual material and allows introducing new approaches to audiovisual mapping without restructuring the existing organization. Each of these zones can be activated in any sequence during a performance and, for better or worse, have provided a general form for the entire composition. Each zone is characterized by unique and specific mapping procedures resulting in a distinct audiovisual entity. Normally each new approach requires a different configuration of cell state values to be polled from the automata world, depending on which part of the system is being explored. There are currently 4 configurations of 64 cell

state values polled at a rate of 12 cycles per second that are used in different zones:

- uniform sampling of every 4 cells throughout the world, starting with the third cell in each dimension a sampling of 16 cell values from each of the 4 vertical faces of the world
- wireframe cube-shaped sampling of cells from the middle of the world
- sampling of 8 values from 8 layers of odd numbered slices along the z-axis of the world

The 3D world is represented in OGL by mapping selected cell state values to low-level vertex drawing functions. Most commonly the cell state or inverse of the state is mapped to the drawn object's grayscale color, transparency and size. The variation of these three parameters already yields a significant palette of mapping options, considering that what really makes an impact in terms of the visual content are the intricate patterns that emerge even from the simplest of representations. In the 2 images below, an example of visual mapping is provided in a 2-dimensional automata world. In the image on the left there is a world of 30 by 30 cells, evolved from a symmetrical initial condition of uniform states 0 for all of the cells, except for a rectangular wireframe configuration of 20 by 20 cells in state 1 centered around the middle point. The state after 30 time-steps of evolution is shown. The image on the right in Figure 3.8 shows the same world at the same phase of evolution, but here the cell states are mapped to drawings of stroked rectangles with the rectangle size and the rectangle grayscale color mapped to cell state and the transparency mapped to the inverse of cell state.

Audio synthesis parameter mapping is driven by the messages received from the automata world and handled through different types of OSC responder functions depending on the configuration of cell states requested from the automata application. A commonly utilized technique is to activate a number of parallel synthesis processes, identifying a certain number of modifiable parameters and mapping the incoming cell state values to appropriate mapping ranges for each parameter. In SuperCollider, this mapping is achieved through ControlSpec class and Warp subclasses that enable the specification of range and curve of a mapping from between 0 to 1 to any desired range, which perfectly fits this particular purpose. Each synthesis process can have its cus-

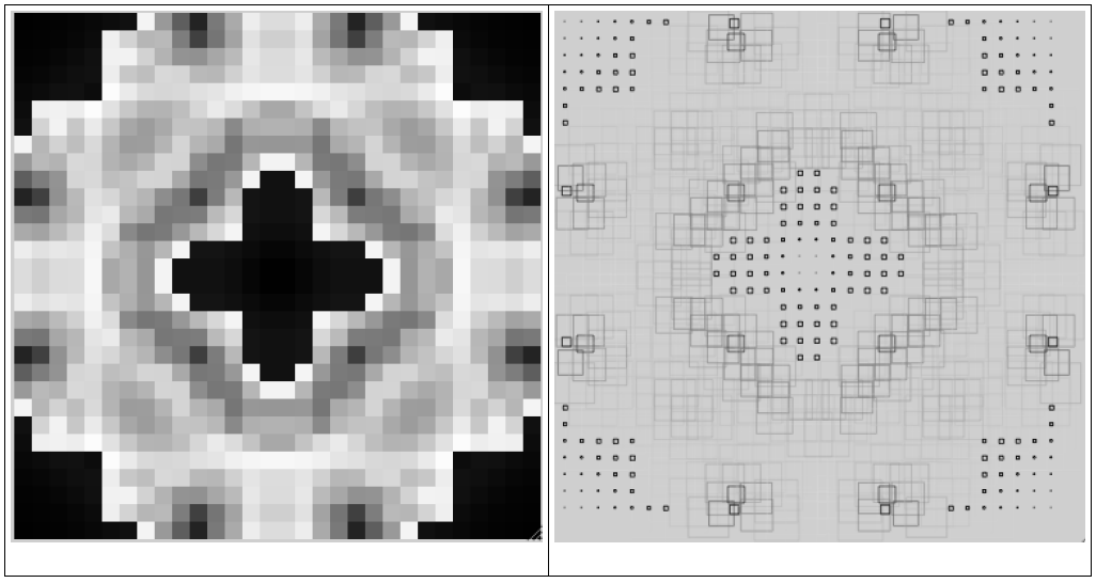


Figure 3.8: Simple mapping of 2-dimensional continuous automata values to drawing parameters.

tom mapping range for every selected parameter or synthesis processes can be grouped and the same mapping ranges used for equivalent parameters. The mapping ranges and curves themselves can also be modified during the performance, which provides even more powerful expressive tools for the performer.

The audio sources for a majority of zones developed derive from previous electroacoustic compositions and live electronic performances of the author. Some of these sounds have already been recycled in the similar manner in previous instances. The inspiration for this approach originates from a composition by Richard Karpen, *Camera Cantorum*, which in the words of the composer himself "presents a sonic analog of an array of old mirrors, each facet reflecting a different fragment of one of my previous pieces."[1] The different zones of *f(x)* each derive from a different performance. For example, the sounds in *zone 01* originate from a composition realized in SuperCollider2 called *detached* which explored the idea of a remix of a rock song in the contexts of electroacoustic composition and minimalist electronica. Thus there are already many levels of reuse and generally each new reiteration renders the source unrecognizable from its previous form.

The crucial aspect of any composition or performance system relying on emergent properties is the mapping procedures used to translate the global

[1] http://faculty.washington.edu/karpen/CameraCantorum.html

behavior of the automata into audiovisual composition entities. Peter Beyls [12] suggested a number of such interpretation procedures for CA as a non-exhaustive list of principles which are left open to different interpretations:

- static mapping - a lookup table
- dynamic mapping
- human performer mapping
- cultural concept mapping
- psychoacoustic mapping
- mapping according to natural laws

These principles can overlap - cultural concept mapping can be static or dynamic for example. So they do not really constitute a comprehensive ontology of mapping procedures, but rather provide a useful guide to the available possibilities. The *f(x)* zones employ all these strategies in some shape or form. There are a number of mapping strategies that attempt to find a balance between the emergent patterns of the automata world and top down compositional concepts of the performer, similar to the approach reported in the article by Beyls. Static mapping, for example, is widely employed in graphics functions where the cell state values are directly linked to a drawing parameter such as line length or size of a more complex shape. Similarly in audio mapping the cell states are directly interpreted as various parameters, including amplitude, pitch-shift or angle of rotation as discussed above. These represent simple static (cell state to amplitude), static cultural concept (pitch-shift frequency intervals are defined in terms of 24-step equal temperament musical scale), and dynamic psychoacoustic (human spatial hearing psychoacoustics) mappings respectively.

3.6 Discussion

The system was developed in the beginning stages of the project and was rather intended as a prototype for a more complex and interactive approach subsequently developed in the *mikro* environment. However, largely due to its primacy combined with the fact that it only requires the author to act as a perfomer, which makes the logistics of the performance significantly less

complex, it has been the system most frequently used in public performances during this project. The performances and presentations include the Super-Collider Symposium in Berlin, the International Symposium on Electronic Art in Istanbul, Piksel Festival in Bergen and the International Computer Music Conferene in Ljubljana among others. Figure 3.9 shows a capture from a live performance at the Shiny Toys festival in Germany in November 2011.

Figure 3.9: Performing *f(x)* at the ShinyToys festival in Germany, November 2011.

The automata world constitutes a novel artistic interface which enables exploration of complex patterns of computational behavior. The importance of the concept of cellular automata is at the very center of the theory of computation and is a product of research into origins of life and artificial intelligence by some of the most renowned protagonists in the recent scientific history. *f(x)* brings these ideas into the realm of live audiovisual performance and thereby fuses together artistic and scientific research methodology in an improvisatory setting. At the same time, this approach hopefully helps alleviate some of the skepticism surrounding the use of compuational systems like cellular automata in a live performance context and highlights the necessary shift in perspective of the performer towards real time applications from having control of a deterministic linear system to becoming a humbled explorer of complex digital universes.

The *f(x)* environment is designed to be controlled by a generic MIDI fader

interface for audiovisual mapping, synthesis and parameter control as well as gestural control of selected parameters by a mobile device like an iPhone or an Android phone equipped with the feature of sending accelerometer readings and other control information via OSC messages. The different settings for the automata world that change the rules of virtual physics and thereby the emergent behavior that can be mapped in real time to sound synthesis and graphics functions make it possible to interactively compose audiovisuals in a performance. The flexible structure of the control center enables reconfiguration, removal of existing and addition of new features depensing on the performance duration and context. The concept of a zone corresponds to predefined global settings that remain constant during a preformance, each zone represening a collection of mapping presets. It serves as a structuring element, loosely equivalent to a composition as an artefact. However, since the actual structure of each performance depends on the interaction between the performer and the automata driven interface, a distinction from the traditional compositional artefacts was necessary as the zones only come into existance during a live performance. Each zone produces a multimodal gestalt, an organized audiovisual whole that exhibits a definite connection between the modalities which is too complex to follow in terms of individual cells. The improvisatory element of *f(x)* arises from exploration by the performer of emergent audiovisual entities and the spontaneous decision-making processes involved when responding to the behavior patterns of the automata world.

Even though the automata are completely deterministic and follow simple local update rules, the resulting behavior is largely unpredictable. As with any other type of instrument or new musical interface, the behavior of the *f(x)* system can learned, thus behavior patterns and audiovisual mappings can be more or less recreated. The crucial difference between more traditional interfaces and the *f(x)* system is the non-linear response of the latter to performer's actions. This is where the audiovisual behavior of the system provides an immersive informative interface with constantly changing audiovisual feedback, in which the audio spatialization and 3D graphics play an important role.

4 | Towards autonomous systems: *mikro*

The wealth of compositional techniques and tools that have emerged as a result of digital technologies provides an enticing challenge to the artist. The sonic material is no longer bound to a physical instrument, nor is the musical structure necessarily articulated by human performers. The prospect of automating parts of the creative process by introducing autonomous agent enriched algorithms provides new avenues of exploration as well as computational challenges. This chapter introduces an approach to real time audio analysis as a source for interactive audiovisual performance. The catalyst of the audiovisual environment is presumed to be a live musical performance that emphasizes the attention to the spectral development of sound structures in the tradition of electroacoustic and computer music. This expression is further augmented by autonomous machine listening agents that interact with the incoming audio stream and generate an audiovisual response.

4.1 Introduction

The freedom to create and shape the spectral characteristics of sounds was recognized as the one of the most essential concerns in computer music composition since the very beginning. The notion of static timbres produced by physically immutable instruments is becoming fossilized amidst the explosive multiplication of algorithmic adaptability of spectra facilitated by computer programming. Algorithmic instruments not only enable exploration and real time composition of audio spectra, but encourage compositional considerations beyond narrow traditional cultural confines and foster artistic practices informed by scientific research and a more global understanding. These developments are to be considered within a more general framework, namely the position and function of art as a way to make sense of human experience which is a stage that has been reached after over roughly four billion years of

biological evolution. Biological evolution is an algorithmic process - blind, purposeless, myopic, lacking a long-term goal. Among the greatest challenges is comprehending and explaining how the apparent byproducts of such processes have given rise to such elusive and vastly complex phenomena as consciousness, communication and human culture. Technology facilitates closer comprehension and better explanation for such algorithmic processes in nature often by building simplified models and computational simulations. Significant advances in computation technology and artificial intelligence research are reflected in the development of artistic practices as interactive multimodal performance systems are becoming more commonplace. Artists working from the realms of sound are progressively embracing technologically facilitated choices. Largely due to increased processing power of accessible portable computers there is a shift from fixed electroacoustic composition (tape music, fixed media composition, computer music rendering) to real time interactive improvisation (supercollider, chuck). More advanced software and computer programming based analysis and synthesis tools have facilitated focus to be realigned from concerns with traditional harmony, melody within static timbres to spectral composition. The academic tradition of acousmatic aesthetics is slowly being enriched by multimedia environments as advanced and accessible (open source) programming languages and communication technologies proliferate. The progress in development of more powerful digital signal processing algorithms has encouraged the replacement of the MIDI paradigm with audio stream analysis involving machine listening and statistical methods. Yet another factor in this long list of interdependent developments is the accessibility to more time-scales of musical composition to use the musical time structure described by Curtis Roads [72]. In his exhaustive account of granular synthesis techniques, he emphasizes the importance of time-scales below that of the traditional note level, the micro time scale. Dennis Gabor's [36] theory of sonic quanta realized in granular synthesis algorithms has provided a new world to be explored. A point to be made observing these trends in the field of music technology is that more often technology directs the creative process, rather than the other way around.

4.2 The conceptual framework

The *mikro* system for interactive performance with live musicians is implemented as a collection of individual modules, in which each component performs a specialized task and interacts with other components to achieve an audiovisual output in response to the incoming audio stream. The features of such real-time interactive system include a capacity to extract information from the environment, analyze the gathered data, and respond within the intended context. Real time audio stream analysis forms the basis for the ensuing multimodal environment. Prior to sufficiently fast algorithms for real time feature extraction, a number of interactive systems - most notably Voyager [50] and Cypher [74] [75] [76] - utilized the MIDI protocol for most of the analysis tasks, including traditional harmonic analysis, key induction and metric segmentation. While these projects provide an excellent introduction to the problems that a programmer faces when designing intelligent artificial systems and the techniques and methods that are developed as solutions, MIDI is not a satisfying solution not only because of obvious inherent limitations as a music representation tool, but also because the ambition to work directly with incoming audio signals provides a more direct and detailed access to the sonic material on much smaller timescales and allows for significantly higher precision and intricacy of analysis. That in itself amounts to a more sensitive and fine-grained analysis algorithm and respectively poses a more difficult challenge for implementation, especially considering the real time nature of the system. More importantly, however, MIDI systems cannot track the most important aspect of the audio signal - the spectral dynamics. William Hsu's [46] "timbre-aware" performance systems focus more throughly on the timbral domain and consider it as an integral element in a performance.

The components of the implementation can be generally classified into 4 main categories: input, analysis, composition, and communication modules. Due to the disconnected structure of the SuperCollider environment, which runs the programming language and the audio synthesis engine as two separate applications simultaneously, each of these modules is further divided into algorithmic control structures and object structure in the SuperCollider language and audio synthesis definitions compiled as graphs of audio unit generators on the SuperCollider audio server. A unit generator in the Super-

Collider environment is a synthesis server object in which signal processing concepts are incapsulated [91]. The third component - the graphics application implemented in C++ and OpenGL - is another independent entity. The major advantage of this architecture allows different modules and their respective components to be deployed on physically separate computers running different operating systems. The input modules manage audio input and allow effects to be applied to the signal and apply Ambisonics spatialization algorithms in order to incorporate typically monophonic input signals into the Ambisonic framework. It also enables different types of audio streams to be patched through the system. Considering modularity, the input can be from three different sources:

- mono or multi-channel audio from the audio interface of the host machine,
- a local audio bus inside the SuperCollider application to enable the *mikro* system to be utilized in combination with other systems
- streamed audio from another application using the Jack audio framework[1] (particularly for network-streamed audio performances)

The input modules handle routing to analysis modules and Ambisonics spatialization of the audio source output which is always mixed to the global output bus. The audio analyzer module consists of machine listening algorithms that extract auditory features from incoming audio streams. Each audio stream typically originates from a single physical instrument or a microphone and is processed separately. The incoming audio stream is segmented into events based on a combination of onset detection and amplitude tracking. The audiovisual response is generated with a mixture of top-down symbolic and bottom-up connectionist artificial intelligence based concepts integrated into the system. The visual response of the system is currently based on an interactive combination of self-organizing neural networks with two- and three-dimensional continuous cellular automata. Each node in the self-organizing map of spectral data also behaves as a cell in continuous automata world. The neural network organizes incoming multidimensional data of spectral coefficients into a multi-dimensional map and the best matching unit is assigned a contrasting value in the cellular automata neighborhood to create moving patterns in response to the spectral content of the audio input. The neural

[1] http://jackaudio.org/

network data in combination with the automata behavior is then mapped to OpenGL drawing parameters of different predefined graphics functions that are activated according to different mapping strategies.

4.3 Real time audio analysis

Content-based audio feature extraction has been a well-established practice in the domain of music information retrieval (MIR) research [62]. In order to represent the musical knowledge, like harmonics, rhythm and timbre, it is necessary to extract meaningful information from audio signals and also reduce the amount of data for analysis. The *mikro* machine listening module implements a number of algorithms in order to extract musical information from the live audio stream and stores the information as a database of discrete event objects. The event segmentation algorithm tries to guess the beginning and end points of an event and stores each new event as an independent entity in the events database. For each event, the absolute start time, duration, continuous sampling of amplitude, overall amplitude envelope, frequency and spectral information are stored. In addition, each event also contains an audio buffer with the recording of the input audio, which can be used for further analysis or as a source for processed audio response. The event extraction algorithm combines onset detection and amplitude following. For onset detection, a threshold-based onset detection unit generator [84] is used which is triggered when the input signal transitions from below to above a given threshold, which can be determined according to the performance situation. The onsets in combination with amplitude following determine the duration of an event. The amplitude follower either falls below the threshold to signify the end of an event or a new onset trigger is detected. The amplitude follower also provides values for the overall amplitude envelope of the event. Pitch unit generator is used to capture the fundamental frequency information. Since the *mikro* system emphasizes the importance of spectral dynamics in a performance, there are three spectral feature detectors that are used: MFCC, SpecFlatness and SpecCentroid. Spectral classification is achieved by integrating data from these unit generators.

The MFCC unit generator calculates a set of Mel-frequency Cepstral Coefficients. MFCCs, which were originally employed in automatic speech pro-

cessing, have been used extensively in music information retrieval as a compressed representation of spectral dynamics, or, to use a more traditional terminology, they can be used to describe the timbre of the input [29] [51]. The number of coefficients extracted is parametrized in the *mikro* system and can be specified for each instance of the analyzer. MFCCs also play an important role in the automatic composition modules. The coefficients constitute a vital part of the fitness function in evolutionary computation of synthesis definitions, described in the following chapter, as well as providing the guiding principle in the improviser module to pick evolved instruments based on spectral characteristics. The MFCC data is simultaneously sent to the graphics application and used in training a self-organizing neural net which drives the real time graphics specification. These graphics processes are described later in this chapter. SpecFlatness outputs a value between 0 and 1 as a reflection of the noisiness of the input. A high spectral flatness - values closer to 1 - means the spectrum of the incoming audio has similar amount of power in all spectral bins and sound quality of white noise, while lower values indicate that the spectral power is concentrated in a small number of bins and the input is closer to a sine-like sound. The SpecCentroid unit generator calculates the 'centre of mass' of the spectrum, returning a frequency value which represents the weighted mean frequency of the spectrum. The analyzer extracts the values at a constant rate specified when the process is initialized. The standard rate used in the performances is 30 sets of values per second, which means that for an event with a duration of one second, the algorithm stores 600 MFCC values, and a total of 720 values for all the descriptors. This enables relatively fine-grained musical analysis of the input in real time without compromising system performance. The event objects calculate the mean and standard deviation values for all the descriptors. In the case of MFCCs, these values are calculated for each coefficient separately. These averaged values are useful in training musical composition modules on the event level. The values from the amplitude tracking algorithm are used to obtain an overall amplitude shape of each event, which can be converted into a SuperCollider Env object and used as an amplitude envelope in the synthesis process. Similar functionality is implemented for the spectral flatness measures to use dynamic timbral information in the synthesis process. The amplitude tracking, MFCC and spectral flatness values in the event database can also be converted into

a SuperCollider Buffer object and used as multi-dimensional dynamic control signals for different synthesis parameters. The analyzer automatically records the audio input into a predefined buffer, from where the audio is copied into individual events each time an event termination signal is received. Since the performance duration is not predetermined, the system keeps reallocating new buffers to record running input until indicated to stop.

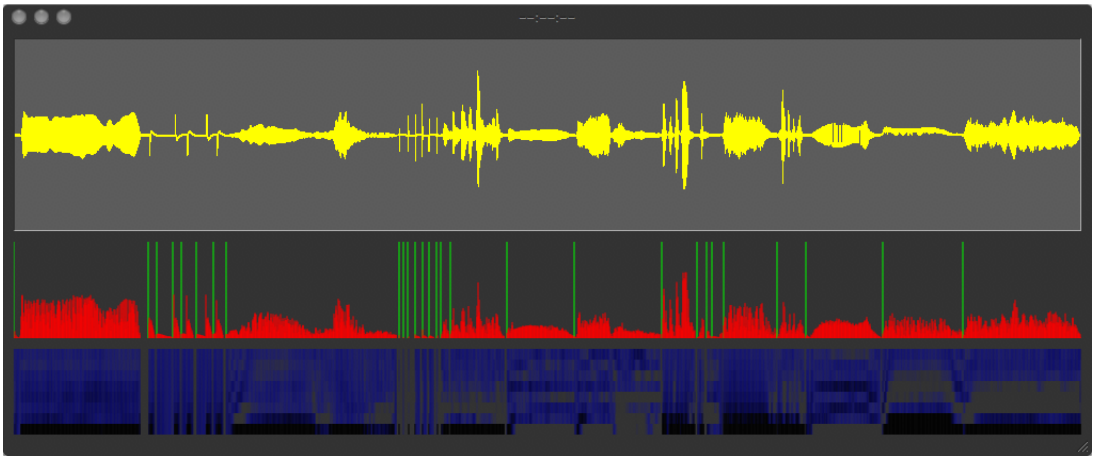


Figure 4.1: Analysis visualizer in the *mikro* system, showing the recorded audio input waveform (top), amplitude tracking data with onset detection (middle), and MFCC data (bottom).

The event analyzer also includes a visualization tool shown in Figure 4.1. The figure visualizes data for an excerpt of eclambone input of about a minute in duration. The top view in yellow represents the audio waveform. The middle view shows the amplitude tracking data along the corresponding timeline plotted as red lines, each line corresponding to a discrete value in the event's amplitudes collection. The green divider lines indicate onset detection results and thus the start time of each event. The onsets and therefore the event detection depends on the threshold parameter. The system was initially tested with Satoshi Shiraishi's custom timbre instrument e-Clambone, which produces sound in a very specific and idiosyncratic way. Some of the design decisions were influenced by these unique characteristics and had to be readjusted later for other instruments, like percussion or human voice. For example, the onset threshold for the e-Clambone instrument has to be quite low, because of the wide range in amplitude the instrument is capable of producing. Setting the threshold value too high would make the system ignore some of the more delicate output and the representation would become less accurate. When using the system with percussion or voice and electronics, the

threshold can be less sensitive as the output from these setups has an overall louder presence while the volume range is much smaller. The lower view shows the time progression of 8 mel-frequency cepstral coefficients. It reveals the evolution of spectral dynamics over the course of the audio timeline that the e-Clambone is capable of producing.

4.4 Automating the composition process

The objective for the composition module is to afford flexibility to different musical contexts while keeping the implementation relatively light-weight, modular, extensible and easy to deploy in rehearsals and performances. The audio analysis events database from the analyzer is accessed from this module and used in the composition process in response to the live performance. The live input processing functions are complex unit generator graphs which have been developed over years of performances, rehearsals and coding trial end error. These processes require time to develop the sonic output and, therefore, are designed to be activated for relatively longer durations, in a range starting from 5 to 30 seconds or more at a time. Some of these functions are internally reactive to the input by means of machine listening to changes in input audio. The activation or fade in and termination or fade out of these processes is determined by statistical analysis of the audio input over meso time scales equivalent to the compositional level of musical phrases or passages. There are a number of different scenarios that can be specified for activation and termination of these functions during a performance, depending on the nature of the input. The strategies employed with e-CLAMBONE involved training the system using a simple supervised learning method to recognize different patterns of spectral dynamics corresponding to the performer activating different modes of the instrument by pressing one of its five sensors. Since there are 5 sensors that turn processes on and off, there can be a total of 32 possible combinations, enabling this many different modes on the instrument. During the training phase, the mode recognizer algorithm created a dictionary of associations in which each of the 32 sensor combinations was connected to running mean and standard deviation values of MFCC analysis of e-CLAMBONE. The audio input was presented in the form of recorded audio samples, which were labelled with the corresponding combination of sensor

activity. These samples were specifically recorded and labelled for the purpose of training the algorithm. A total of 92 samples representative of the wide range of spectral dynamics of the instrument were analyzed in a random order until each combination had accumulated at least 1000 hits. Hits were counted as onset triggers. The dictionary enables mapping the spectral output of the instrument to rules which determine activation and termination of synthesis processes. The live input analysis during a performance can be used as a lookup vector in the dictionary. The algorithm is very simple, but achieves relatively accurate detection rates, comprehensively outperforming more complex solutions such as a neural network approach to the same detection problem. The detection rate of the simple dictionary recognizer algorithm turned out to remain constant between 75 - 85% (different modes possessing varying degrees of recognizability), while the neural net implementation, for example, achieved a maximum of 70% accuracy. The error of accuracy may perhaps be undesirable in scientific or commercial applications, however, in the sportive spirit of glitch art, which in its very essence promotes the exploration and exploitation of errors in the machinery, the system training errors can be employed in the performance as objects of artistic endeavor and curiosity. An illustrative example of such an approach underlies the development of *m*ikro:skism, a performance environment in which the performer is not totally in control due to system training errors and a schism is intentionally introduced into the human-machine relationship for exploration in a live concert situation. Further discussion of this uneasy relationship is presented in the section dedicated to this particular implementation later in this chapter.

Sample-based synthesis functions are used for short musical events, which derive their parameters like playback rate or time-stretch factor, amplitude, duration and amplitude envelope shape from the event database of the analyzer module as well as preloaded training data. The composer module uses Markov sets to store pre-analyzed musical data in order to generate short musical phrases as Markov chains in response to the live input. A Markov set in this context is a lookup dictionary of transition probability weights, which are calculated according to observations extracted during the training phase. Markov sets can be prepared with analysis data from varying sources. The *mikro* system uses an audio analysis library functionality, which provides different data sets depending on the type of instrument and other musical con-

texts. It involves analyzing desired musical works as training examples for the system and storing the results in the equivalent format to the live input analysis. There are Markov sets containing transition probabilities for 4 musically significant parameters in the analysis:

- event duration
- mean event amplitude
- mean event fundamental frequency
- inter-event interval duration

Due to the potential complexity and variability of amplitude envelope shapes, the envelopes are stored in a Fuzzy set and associated with the duration of the corresponding event. A Fuzzy set in set theory is known as a set whose elements have degrees of membership in order to be able to introduce more flexible assessment of membership [97]. In SuperCollider, a Fuzzy set is essentially a dictionary which associates objects with identical keys, in this case envelopes to event durations, keeps these in a set and, when queried, returns one of them by random selection.

The composer module provides an interface for dynamic addition and removal of musical phrase generator processes. The rules for process activation are defined in the composer. There is a limit on the maximum number of parallel processes allowed in order to keep the generation of new synthesis processes under control. An example of a process that is defined for a performance is shown in the listing below. Each process is passed the composer and the analyzer objects as parameters. As discussed above, the processes are activated in response to onset detection in the live audio input. Once activated, this particular process first determines that the synthesis definition called “bufplay” from the definitions library is to be used. Then it accesses the last two events from the live audio input events database and uses these to initialize a Markov chain for the parameters of the sample player. The envelope is picked randomly from the Fuzzy set of envelopes associated with the duration of the last event, the duration is the duration of the same event, the playback rate is the ratio of the fundamental frequencies of the last two events, the amplitude is the peak amplitude or maximum value in the amplitude tracking data of the event, and the interval until the next event is the

time between the two events. Then a musical phrase consisting of 4, 8, 12 or 16 events is generated, as the number determined by a random selection between 4 and 16, which is then rounded to the nearest multiple of 4. Each time the process generates a new synthesis event, a new set of values is queried from the Markov sets to be used in the next event, based on the respective probability weights.

```
{arg composer, analyzer;
	var srcA, srcB, current, def;
	def = composer.descLib['bufplay'];
	#srcB, srcA = analyzer.events.reverse.keep(2);
	current = (
		xenv: composer.envSet[srcB.duration.round(composer.timeQuant)],
		xdur: srcB.duration.round(composer.timeQuant).clip(0.0625, 4.0),
		xfrq: composer.roundFreq(srcA.meanFreq) /
		    composer.roundFreq(srcB.meanFreq),
		xamp: srcB.peakAmp.round(composer.roundAmp),
		xtime: abs(srcB.start -
		    srcA.start).round(composer.timeQuant).clip(0.0625, 4.0),
	);
	Tdef('proc00', {
		rrand(4, 16).round(4).do({
			var params, stream;
			stream = Pbrown(0.0, 1.0, 1/11, inf).asStream;
			params = ['out', ~mikro.decoder.bus, 'in', ~mikro.input.bus,
			    'dur', current.xdur ? 0.5, 'buf', srcB.buffer, 'rate',
			    current.xfrq ? 1.0, 'amp', current.xamp ? 1.0 ] ++
			    def.metadata.specs.collect(_.map(stream.next)).asKeyValuePairs;
			Synth.tail(~mikro.group, def.name, params).setn('env',
			    current.xenv);
			(current.xtime ? 0.25).wait;
			current = (
				xenv: composer.envSet[current.xdur],
				xdur: (composer.durChain.next(current.xdur) ?
				    0.5).clip(0.0625, 4.0),
				xfrq: composer.intChain.next(current.xfrq),
				xamp: composer.ampChain.next(current.xamp),
				xtime: (composer.timeChain.next(current.xtime) ?
				    0.5).clip(0.0625, 4.0)
```

```
                    );
                });
                Tdef('proc00').clear;
        });
        Tdef('proc00').play;
}
```

All parameter values must be quantized in some way in order to limit the number of eventual weight values. The time quanitzation value is applied in the case of amplitude envelopes to the associated duration as well as the duration of the event. The modifiable value is by default equal to $1/(2^6)$. The fundamental frequency values are by default rounded to 24-step equal temperament scale values, but any division by equal steps is possible. The rounded frequency can be calculated as

$$f_1 = 2^{\frac{[log_2(\frac{f_0}{f_r})d]}{d}} f_r$$

where f_1 is the rounded frequency value, f_0 is the input frequency from the analysis, f_r is the reference frequency, and d is number of octave division steps.

A similar approach is utilized for the synthesis functions resulting from evolutionary algorithms. The functions are stored in the database concurrently with the analysis data, including mean and standard deviation values for MFCCs, spectral flatness, fundamental frequency, and amplitude. This enables the database to be queried based on the statistics and used in the composition process. The way these functions are used in the real time environment involves classifying spectral analysis data with a k-means clustering algorithm in order to optimize matching the input analysis statistics to the closest match in the instrument database. The composition module handles the event creation in much the same way as the sample-based synthesizers by using parallel Markov sets of parameters. The major difference is the way instruments are selected for each event in a phrase. The instruments are loaded into a predefined number of clusters - by default there are 64 - based on the MFCC values from the analysis. The clustering algorithm assigns each instrument a cluster index based on the nearest mean of a prototype cluster. Once each instrument has been associated with a cluster, the selection process can be initiated by selecting an event from the live input event database, for example, the most recent addition. This event's MFCC data is then queried

and the result used to match the closest cluster prototype, defining the cluster from which the instrument is selected. Then it is either picked randomly or another matching process determines the instrument to be used based on the MFCC data. This approach significantly reduces the computational overhead that would otherwise be required to match the input MFCC vector of 20 values against a database of over 2000 each time a new event is generated. The musical phrases are composed with parallel Markov sets. The source of each phrase is analysis data from a selected live event and the parameters for each synthesizer are determined by querying the sets that have been trained with sample musical event data. There is an additional Markov set for MFCC data clusters. The system classifies the MFCC analysis vectors of the musical example database based on the MFCC clusters. So for each new event that is generated, the system can pick which cluster to use for the selection of the instrument. There are a number of operations that are used to vary the order in which the generated phrases are played. The prepared sequences of events can be played once or be repeated in the same order. They can also be repeated in reverse or in randomly scrambled order as well as rotated in either direction.

4.5 Interactive computer graphics

The graphics are based on three autonomous computational systems - cellular automata, self-organizing maps and swarm intelligence - which are reactive to the audio analysis data communicated to the graphics application. The three systems form an interactive environment, influencing and interacting with each other. The cellular automata world, similar to the f(x) system discussed in chapter 3, follows continuous automata rules of averaging neighbor states, which are floating point values between 0 and 1. A self-organizing map (SOM) is a type of artificial neural network proposed by Teuvo Kohonen [49] that is trained with sample data in form of input vectors using unsupervised learning. Once the map is trained, it can be used to classify input vectors. A SOM can consists of neurons or nodes arranged most commonly in 2 or 3-dimensional matrices. Each node represents a vector of weight values. The map is typically initiated by vectors containing random values. SOMs are first trained with example data until the specified training duration is reached. Training duration is

defined as number of iterations of sample data that the map is presented. For each iteration of new input vector D, the system proceeds through 2 phases:

- Find the similarity of D for each node in the map using Euclidean distance formula, where d is the distance and N the size of the vectors, W is the node weight vector and v is the input vector:

$$d = \sqrt{\sum_{i=1}^{N}(W_i - D_i)^2}$$

  retaining the smallest distance value which is called the Best Matching Unit (BMU)

- For all nodes within a defined radius of the BMU, update the node's weight vector scaled by the distance from the BMU:

$$W_{(s+1)} = W_s + ((D_s - W_s)\alpha I)$$

  where $W_{(s+1)}$ is the weight vector at next iteration, W_s is current weight vector, D_s is input of the current iteration, α is the learning rate and I is influence factor calculated by

$$I = \exp(\frac{d}{-2r^2})$$

  where d is the Pythagorean distance from the BMU calculated as

$$d = \sqrt{(BMU_x - N_x)^2 + (BMU_y - N_y)^2}$$

  and r is the radius of influence from the BMU defined as

$$r = \frac{\sqrt{WS}}{2}\exp(\frac{\lambda}{-t_c})$$

  where WS is the total number of nodes in the map, λ is the number of iterations the system has executed, and t_c is a time constant defined as

$$t_c = \frac{\lambda_{max}}{\log\frac{\sqrt{WS}}{2}}$$

Once the iteration is complete, the SOM can be used for classification of input vectors, which in the *mikro* system originate from live signal processing analysis. The map defines a location of each input vector using the BMU identification step in the multi-dimensional space (either 2D or 3D). In the mikro system the SOM is trained in real time during the performance with MFCC data

vectors received from the audio analysis of the live input. Figure 4.2 visualizes the spectral analysis data organized in a two-dimensional SOM that contains 2000 nodes. The first four MFCC coefficients of each node are scaled to a range of values between 0 and 1 and mapped to color specification of the squares as RGB and alpha values, the size of the squares is determined by averaging the next four MFCC values and the background color is determined by spectral flatness i.e. the lighter the background the noisier the signal.

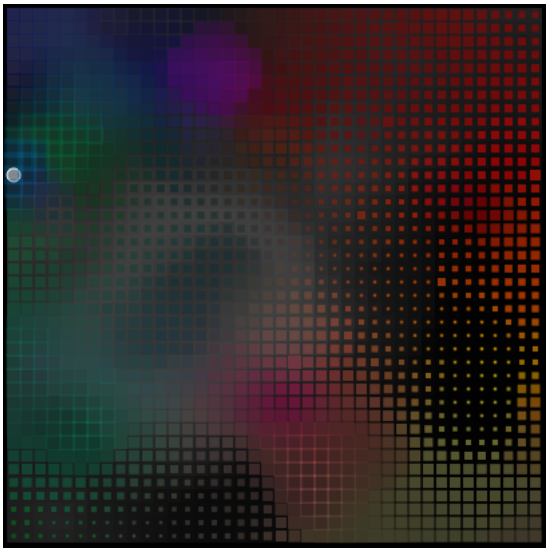


Figure 4.2: A graphical representation of a two-dimensional SOM of MFCC data from a performance with e-CLAMBONE.

The SOM and the automata world are linked so that each cell in the automata world is connected to a node in the SOM. The BMU identification sets the corresponding cell state in the automata world either to 1.0 or to contrast the mean state value in the cell neighborhood. This functions as a disturbance in the system, affecting the patterns of automata behavior. Since the automata world is defined in terms of periodic boundary conditions, different rules of pattern symmetry can be explored in this way. The affected cell can be bound to other cells in the matrix by mirroring its location along any of the axes. The affected cells are determined depending on the symmetry setting. In two-dimensional worlds there are 5 possible symmetry settings:

- No symmetry, only the BMU_{xy} cell is affected
- Symmetry along the X axis, the BMU_{xy} and $C_{n_x - BMU_x}$ cell affected
- Symmetry along the Y axis, the BMU_{xy} and $C_{n_y - BMU_y}$ cell affected

- Diagonal symmetry along both X and Y axes, the BMU_{xy} and $C_{n_{xy}-BMU_{xy}}$ cell affected
- All of the above combined

In three-dimensional worlds the number of symmetries grows to 12. All possible combinations of these are left at this point to the reader as a thought exercise. The real time graphics are implemented in the Open Graphics Library (OpenGL) as pattern functions which are activated and deactivated according to predefined rules. The pattern functions make use of the automata cell state and SOM weight vector and BMU values. There are many ways to map the computational behavior of these systems to graphics drawing functions and new ways are being continuously researched and implemented. For example, one method to harness the connectedness of SOM and CA is to draw only cells that are within a certain radius from the map's current BMU, which reflects the location of the node that most closely corresponds to the analysis data of live input. This kind of a pattern from a recording of *mikro:skism* is shown in Figure 4.3: there are 2 overlapping patterns activated simultaneously. The spherical structure composed of 3D box-shaped objects represent the automata cells, the sizes of which are mapped to respective cell states. The entire struc-

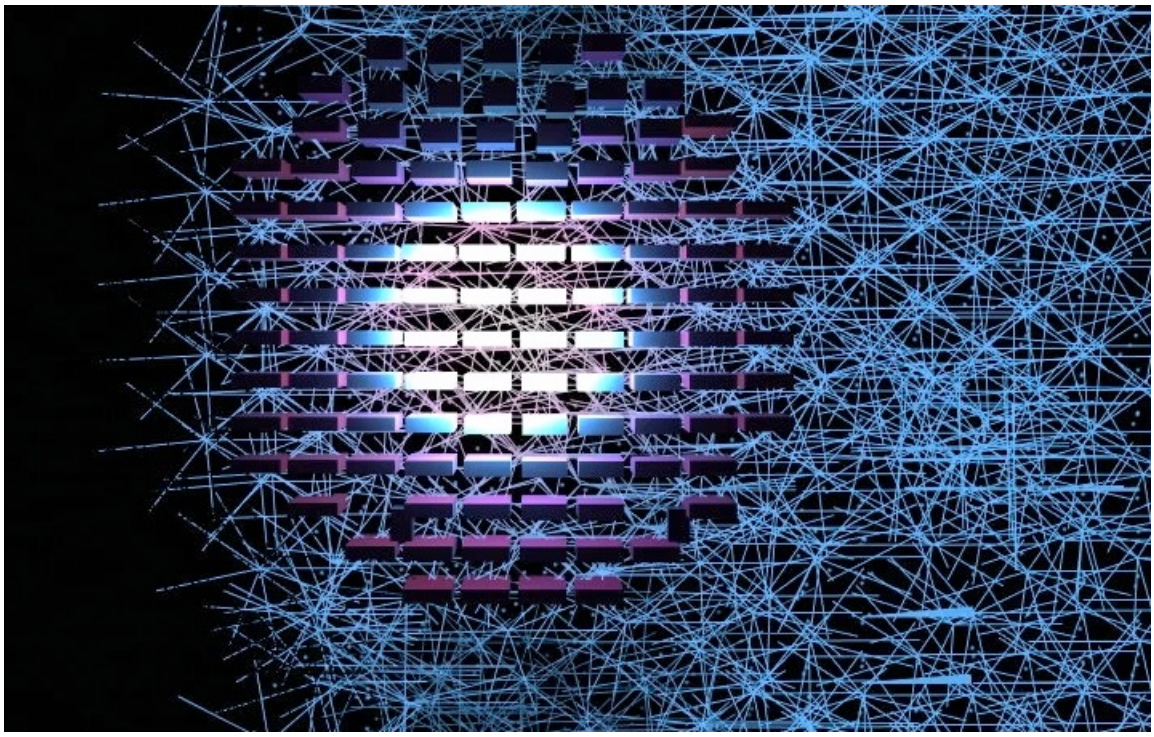


Figure 4.3: Overlapping*mikro* graphics pattern functions, demonstrating the interaction between automata and SOM worlds.

ture follows the location of the BMU and therefore moves around depending on the input analysis. The pattern also uses a OpenGL shader with lighting which is responsive to the input amplitude in a binary manner, i.e. rounded to the nearest integer, 0 or 1, which creates a strobe-like effect. The other pattern maps automata cell state values to trigonometric drawing functions that change the location of each cell within its local space in a circular motion and also draw extended lines from that moving center reaching out at different angles so they overlap with adjacent cells, which creates a dynamic and complex mesh.

The third computational system in the graphics application is derived from mathematical models of swarm behavior, particularly from the boids algorithm by Craig Reynolds [71]. Similarly to cellular automata, the overall system behavior exhibits emergent complex behavior, which arises from individual interacting agents following simple local rules. In the boids world, however, the agents are not fixed to a location, but move around at a velocity determined by three rules:

- Cohesion forces the agents to move towards the mean position of the swarm
- Alignment steers the agents in the direction of the mean direction of the swarm
- Separation repels the agents from each other when their proximity falls below a certain threshold

In the *mikro* system, the boid algorithm can react to the live audio input by mapping live audio analysis data to the different parameters that control the behavior of the boids algorithm. It is possible to force the boids to follow the location of the BMU of the SOM world. Another option is to map the amplitude following statistics to the alignment parameter, which controls how close together the agents tend to be. In this case, louder amplitudes increase the alignment parameter and allow boids to move away from each other and occupy a larger space, which can be utilized in the drawing functions. Figure 4.4 shows a snapshot of the 3D boids algorithm visualized with the help of OpenGL shaders and cube-mapping methods. Each boid can be a vertex of a complex, dynamic, three-dimensional object and simultaneously rendered as a cube-mapped sphere.

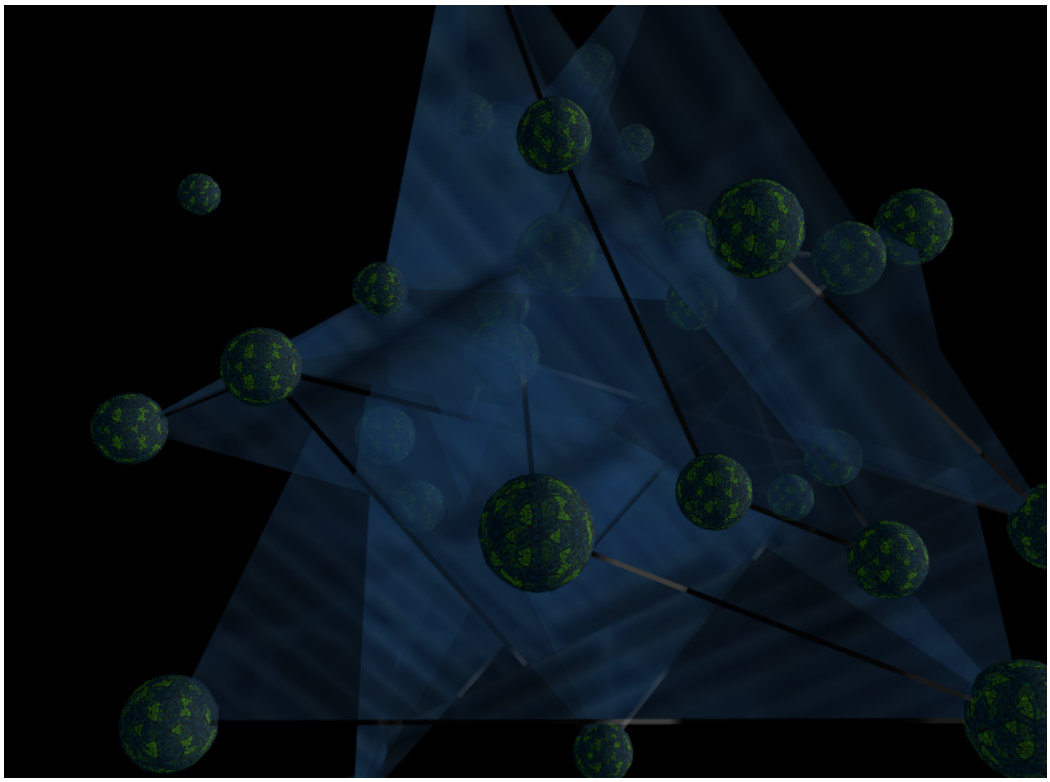


Figure 4.4: 3D graphics based on the boids algorithm.

4.6 Discussion

mikro system has naturally evolved during a number of collaborations with different musicians. The initial research goal was to investigate the degree to which control of the compositional process can be delegated to machine agents. There are four implementations that are all based on similar principles: the systems are driven by an instrumental performance which is analyzed in real time and an audiovisual response is generated allowing progressively larger degrees of autonomous agency.

mikro:strukt is a collaboration with Japanese instrument designer and performer Satoshi Shiraishi. Satoshi's performance on the eCLAMBONE provides an acoustic source for the piece. eCLAMBONE is a custom-built, fully integrated electronic instrument that consists of an aerophone supplied with haptic sensors and digital signal processing algorithms.

The eCLAMBONE was built with the specific interest in timbre manipulation by an acoustic instrumentalist to drive a sound synthesis engine [79]. The *mikro:strukt* environment incorporates a number fo machine listening processes that trigger changes in synthesizers and control synthesis parameters, but the overall structure relies on the intervention of a programmer to activate

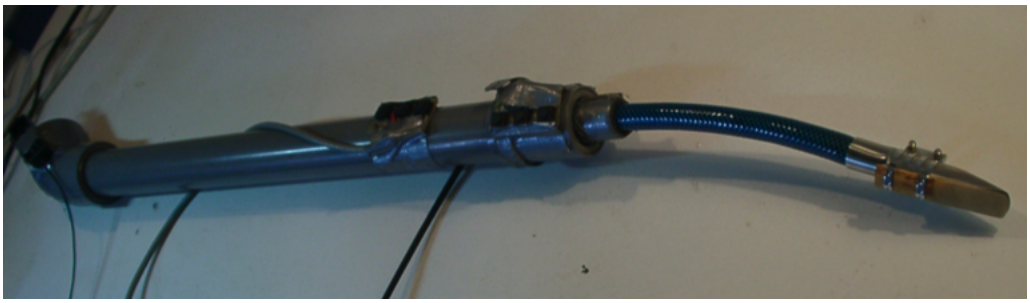


Figure 4.5: The custom-built electronic instrument eCLAMBONE by Satoshi Shiraishi.

and deactivate larger sections of the composition during performances. The machine learning methods are in this instance used to aid the programmer in the decision making process as well as to implement complex parameter control on micro time level that would otherwise be inaccessible in the real time setting of improvisatory performances. The overall structure of each composition is normally reached through discussions between the peformer and the programmer prior to a performance. This approach continues and extends an established practice of collaboration in which an immersive audiovisual environment is the result of improvisation within premeditated boundaries.

mikro:skism, the second in series of collaborative audiovisual performances involving Satoshi Shiraishi's improvisation on eCLAMBONE, introduces a higher level of automation to the compositional process. It attempts to concentrate on and, in effect, magnify the conflicts and disparities arising from human-machine interactions in parallel with tensions within a distributed audiovisual composition system. There are 5 buttons on the eCLAMBONE each associated with a mode of timbral analysis and resynthesis that can be activated in parallel. The *mikro:skism* system guesses the configuration of buttons pressed at any point in the piece according to a spectral analysis statistical map that has been created during the training phase before the performance. The statistical map links a set of mean MFCC values to each of the 32 possible configurations. Each button is mapped to an audiovisual entity that is activated when the system guesses the configuration, while the mappings change over the duration of the piece from one section to another. The performer is not otherwise notified of the changes until they happen and the change can be detected from the behavior of the system. There is always an error in the training procedure, so that the system is not always correct, 10-15% of the time it guesses wrong, which creates a tension between the expectations of the performer and the

response of the system. The role of the programmer is minimal during the performances. Decisions about overall compositional structure have been implemented into the system previously without informing the performer. The musical expression of eCLAMBONE is transformed into an immersive environment that can be controlled by the performer to a certain extent, limited by the unpredictability of the system.

Figure 4.6: Performing *mikro:strukt* with Satoshi Shiraishi at NIME 2011 in Oslo, Norway.

mikro:geen has been created to act as an autonomous improviser in a musical performance with human musicians or other artificially intelligent agents. The system uses the database of gene expression synthesis functions (described in Chapter 6) in combination with a library of musical analysis examples from a wide stylistic range of artists, performers an composers including Andrea Young, Satoshi Shiraishi, Yota Morimoto, Keiichiro Shibuya, Joel Ryan and Evan Parker, Pan Sonic, Yasunao Tone, Evala, Frances-Marie Uitti, Giancinto Scelsi, Konono No 1, Agostino Di Scipio, Pomassl, The Terminal Sound System, Gustafsson & Stackenäs and many others. The gene expression sythesis functions are selected based on a clustering algorithm that classifies hundreds of candidate functions into a predetermined number of categories according to 20 MFCC values. The parameters for event duration and amplitude envelope shape for each function are determined by stochastic transition rules derived from statistical analysis of musical features.

bocca/mikro is an audiovisual improvisatory collaboration with Canadian singer and composer Andrea Young. In this performance, the worlds of vocal experimentation, interactive machines and audiovisual environments collide as the catalyzing vocal input of bocca provides a source for the *mikro* system and the ensuing audiovisual environment, exploring the cultural and technological connotations of the schism between the natural and the artificial. *bocca* is an improvisation for voice and voice-controlled electronics which uses vocal feature extraction to control variables in algorithmic synthesizers. The combination of amplified voice morphing into voice-controlled machine noise extends the human voice, infuses electronics with the vocal envelopes and blurs the boundaries between voice and electronic sound. The implementation of *mikro* for *bocca/mikro* combines features from the previous three approaches. The system “learns” about musical structure from the incoming vocal improvisation as well as from other musical examples with the aid of various machine listening and learning algorithms like *mikro:geen*, while simulatenously processing and expanding the incoming audio stream directly using more reflexive synthesis processes from mikro:strukt.

5 | Live coding: *sparsematrix*

Writing computer code to create audiovisuals in real time is becoming an established form of expression in the realm of electronic arts. The practice introduces the spontaneity of improvisation into the realm of algorithmic composition and enables the performers to make computational decisions live on stage. This chapter proposes an approach to audiovisual live coding based on conversions of traditional rhythm patterns to sparse representations which are used as the raw material for time structuring the performance. The visual component of the performance employs reactive multi-dimensional outer totalistic cellular automata combined with a self-organizing map of spectral descriptors of real time audio analysis as the framework upon which computer graphics are built.

5.1 Introduction

sparsematrix is a computational experiment in the form of an audiovisual performance that explores computation as an aesthetic substrate for human culture. The dominant principle here is the assumption that all processes in nature are in essence algorithmic, in fact, all general physical processes are dominated by simple algorithmic rules. The algorithm becomes the aesthetic in the process of the human performer interacting with the machine by instructing it in code. Transcriptions of traditional rhythmic patterns, which have been subjected to conditions of sparsity, provide the raw structuring material. Sparse matrix is a mathematical idea applied here to dense rhythmic patterns in order to discover the skeletal structures of rhythm loops form various african traditions arguably one of the most enduring of cultural practices - filling the event cycles primarily with silences and turning the attention to pure perception of time stripped of the baggage of cultural connotations. Perception is inescapably multimodal and audiovisual cognition is a hard-wired reality we may at times wish to inhibit through specialization, nonetheless a reality we invariably inhabit. The visual component of the performance employs another

type of a simple artificial life algorithm from the theory of computation - that of multi-dimensional outer totalistic cellular automata which is essentially a model of complex behavior. The automata receive data flow from audio analysis of the rhythm loops in the form of onset triggers and spectral analysis and respond in spontaneous patterns of computer graphics which the live coding can only influence to a point, establishing an heterogeneous relationship between performer and this multimodal instrument, rather than the more traditional idea of one-directional flow of control.

5.2 Sparse matrices of musical structure

The central compositional concept in the system is the sparse representation of traditional rhythms in order to capture their basic time structure. This approach was partly inspired by Carsten Nicolai's - also known by the artist name Alva Noto - sonic experiments in which he "tried to strip bare pop music standards to a minimum level of rhythmic structure" [23], that produced remarkable results in terms of musical form, that could be described as rich in their sparseness. While Nicolai's experiments may imply an intuitive and trial-and-error type of approach, the concepts developed in this system rely on computational analysis and simple combinatorics. The algorithm implemented is not optimal and often does not find a solution that satisfies the constraints, however, it does preserve the most essential structure of the patterns, which can be used in the performance as a structuring element. African rhythm patterns were chosen as the raw material for their rhythmic complexity, as well as the rich and ancient cultural tradition, which have been historically often ignored in musicological research. The rhythms have been transcribed from the Djembe Font Notation [42] to a SuperCollider dictionary object.

Figure 5.1 shows the rhythm patterns in box notation before (on the left) and after (on the right) the sparse algorithm is applied. The box notation was originally developed by Philip Harland and used to notate polyrhythms in African music [87]. The example is based on a rhythm pattern called Yole from a library of African rhythms which have been transcribed from the original notation for 5 instruments: 3 djembe drums, 1 bass drum and a bell. Each of the djembe drums is represented in the notation as producing 3 distinct sounds: bass, tone and slap, each of which in turn is considered a different

instrument in the algorithm. This is how we end up with 11 parallel patterns: the Yole pattern specifies 3 distinct sounds for the first 2 djembes and just 2 distinct sounds for the third djembe and the bass, while the bell sound does not vary and is represented as the 11th instrument. The sparse representation is achieved by applying simple constraints to the matrix of events, not allowing an event to share the same row or column with another event. In order to preserve the most salient features of the pattern, the importance of each time unit is determined by the number of concurrent events in the original pattern. The algorithm then needs to find the best compromise between preserving the most important time units and keeping all instruments represented in the sparse matrix. This is a problem of combinatorial mathematics, somewhat similar to the nine queens problem, and presently the most optimal solutions for any of the patterns in the library have not been established. Furthermore, there is no clear single solution, as the constraints can be interpreted in a number of different ways.

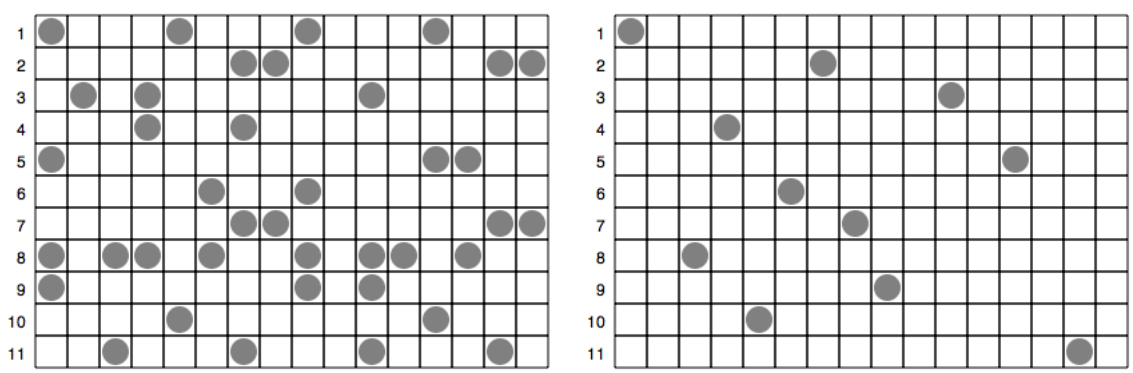
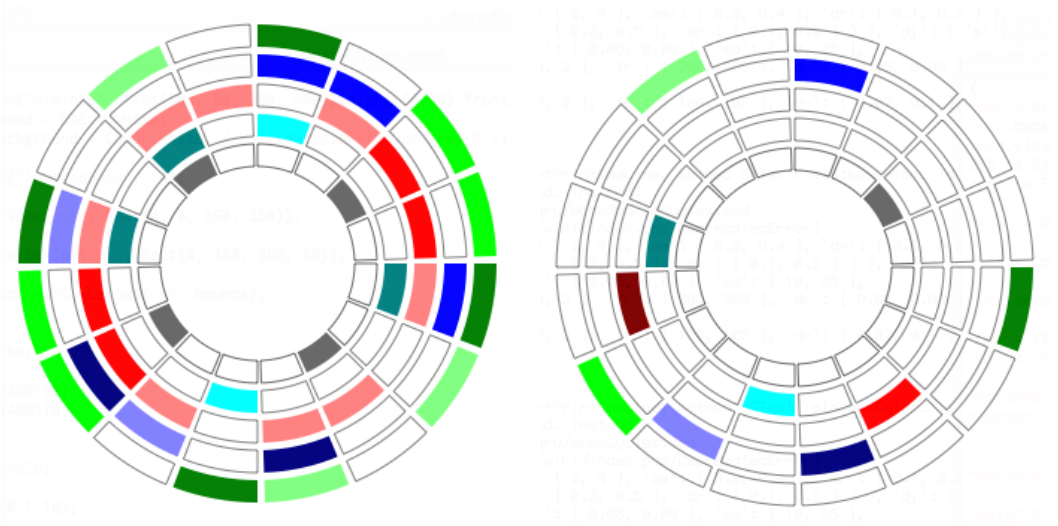


Figure 5.1: Application of sparsity rules to a rhythm pattern shown in box notation.

Figure 5.2 offers an alternative representation has been developed based on geometric representations of cyclic rhythms previously used in ethnomusicological research [87] [86]. The same Yole pattern is represented in Figure 5.2 in a cyclic wheel-like arrangement. The circular wedge patterns represent the 5 different instruments, while the different shades of colour fills designate the distinct sounds as discussed above. For example, the innermost circle represents the bell instrument pattern and is the same as row 11 in the box notation in Figure 5.1. The next circle represents the bass instrument pattern from rows 9 and 10, and and the 3 outer circles designate the 3 djembe drum patterns. The main benefit of a circular representation is that it allows for an alternative conceptualization of rhythm patterns to the traditional linear left-to-right



(a) Original Yole pattern

(b) Yole pattern after application of sparsity rules

Figure 5.2: Cyclic representation of sparsity rules

grid layout derived from Western music notation. The rhythm cycles do not necessarily have a definite beginning or end and this can be varied in the algorithm, inspired by the circularity of rhythm perception in traditional African drumming [5].

The algorithm in the current implementation solves the problem in the most simple terms by counting events in both dimensions for each instrument and time step and then fills the new matrix in one iteration without trying to solve the problem recursively. In this form, there are four preference combinations given the two dimensions of the matrix, which do not necessarily produce an optimal solution. It is not known whether the given constraints allow for optimal solutions for all the patterns included in the library. The solution would be somewhat similar to the eight rook polynomial in combinatorial mathematics, which investigates the number of ways in which to place rooks on a chessboard of a given size so that no two rooks are in the same row or column. However, there are some crucial differences: the rhythm patterns have variable sizes in both dimensions and even within each pattern individually, there are always more time slots than instruments, so the symmetry of the chessboard is lost. Moreover, there are constraints on specific locations of the rhythm events and the importance requirements of time slots. This probably means that each rhythm pattern has a set of unique solutions and the comparison can at best serve as a general guiding principle to finding the appropriate

algorithmic solution to the problem.

Incorporating the described library of sparse rhythmic patterns into a live coding performance exposes a fundamental dilemma within the relatively new and loosely defined practice. When considering a live coding environment as a musical performance tool, a decision has to be made about the granularity of the compositional building blocks to be used to find a right balance between prepared and real time coding. Since there are many levels to composing music with computers - including encapsulating common signal processing algorithms, combining these into synthesis algorithms, using the synthesis algorithms as building blocks for longer musical phrases and so on - the level of granularity largely depends on the compositional goals and the performance context. The *sparsematrix* system already presumes certain musical structures and concepts and therefore does not fit into the category of idealistic "pure" live coding environments. However it has been developed with consideration for flexibility on different levels of compositional structure, making it possible to select the balance between predefined and spontaneous on as many levels as possible, depending on the context. But before we can proceed further with the discussion of the musical concerns, let us for a moment delve briefly into the visual domain.

5.3 Outer totalistic automata for computer graphics

The graphics in the *sparsematrix* environment are based on another complex computational system of cellular automata known as outer totalistic cellular automata. In Chapter 3, the fundamental principles of cellular automata were introduced and continuous automata rules were discussed in more detail. Outer totalistic rules belong to the category of discrete-state automata and are a subset of what are known as totalistic rules. Totalistic rules are defined in terms of only the total or average of the values of the cells in a neighborhood, including the cell itself, and not the exact configuration as is the case with 1-dimensional automata frequently used as examples. Outer totalistic rules consider the center cell state separately from the outer total, i.e. the total of the neighborhood. The seemingly most popular of all cellular automata rules, John Conway's Game of Life, is a good example of an outer-totalistic rule. In

the context of calculating rule space for the life-like totalistic rules using

$$N = k^{1+(k-1)(r+1)}$$

which is a derivative of the familiar equation from Chapter 3, taking into account a significant reduction in the rule space. N is the number of possible rules, k is number of discrete states, and r is neighborhood size, and considering, that, in three dimensions, there are 26 neighbors in a Moore neighborhood, which means $r = 26$ and a number of discrete states $k = 2$, we get

$$N = 2^{54} = 1.8 * 10^{16}$$

or, in other words, an unimaginably vast rule space, which means we are far from any immediate danger of running out of behaviors to explore. In fact, the space is far too vast for any computational method to have a chance to find anything workable in a live performance situation, therefore, a library of life-like rules from the internet and literature has been compiled, which has been extended during development, experimentation and rehearsing. Another feature built into the rule definitions is the option to add decay when defining a rule, inspired by the *Generations* rule family from the MCell automata lexicon [93]. Decay means when a cell “dies”, it does not immediately switch to state 0, but goes through a number of discrete intermediate steps before reaching state 0. The rules are initialized by defining 3 parameters:

- An array of totals of neighborhood cells in state 1 that cause the center cell to transition from 0 to 1 or “birth” rules. These are only applicable if the center cell is in state 0.
- An array of totals of neighborhood cells in state 1 that maintain the center cell’s state 1 or “survival” rules. These are only applicable if center cell is in state 1. If the array does not contain the total, then the center cell’s state transitions from 1 to 0.
- the number of intermediate steps when the cell transitions from 1 to 0 or “decays”.

As an example we can define the Game of Life rule with these parameters. The MathWorld online resource defines the rules for the Game of Life [90] as:

1. Death: if the count is less than 2 or greater than 3, the current cell is switched off.

2. Survival: if (a) the count is exactly 2, or (b) the count is exactly 3 and the current cell is on, the current cell is left unchanged.

3. Birth: if the current cell is off and the count is exactly 3, the current cell is switched on.

The following SuperCollider code sends a rule initialization command to the graphics application to invoke the Game of Life rule:

```
~graphics.sendGenRule([3],[2,3],2);
```

However, as already suggested, the graphics application enables exploration of a much vaster rule space that allows roaming far beyond the tiny familiar confines of the Game of Life. The following code snippet, for example, initializes a more complex rule with a 16 step decay that produces very different dynamics in either 2D or 3D depending on the definition of dimensionality when the application is set up:

```
~graphics.sendGenRule([4,6,8],[3,5,7,9],16);
```

In order to generate patterns of behavior, an initial configuration of cells has to be specified and this can be done in many different ways. Similarly to the *mikro* graphics system, the cells are associated with a Self-Organizing Map (SOM) of live input analysis data and each time the Best Matching Unit (BMU) is determined, the corresponding cell in the automata world is activated, i.e. its state is set to 1. The same symmetry settings are also applicable here, however, due to the radically different nature of the rules, the resulting behavior is also very different. Figure 5.3 shows a screenshot of the graphics application computing the above defined complex rule. There are 3 OpenGL graphics patterns activated mapped to visualize system dynamics with different parameters. The floating text visible in the figure is the code that has been evaluated during the session and mapped to the edges of the cube-shaped world in an attempt to incorporate it into the visual aspect of the performance while finding alternatives to the wide-spread live coding practice of displaying the coding environment itself.

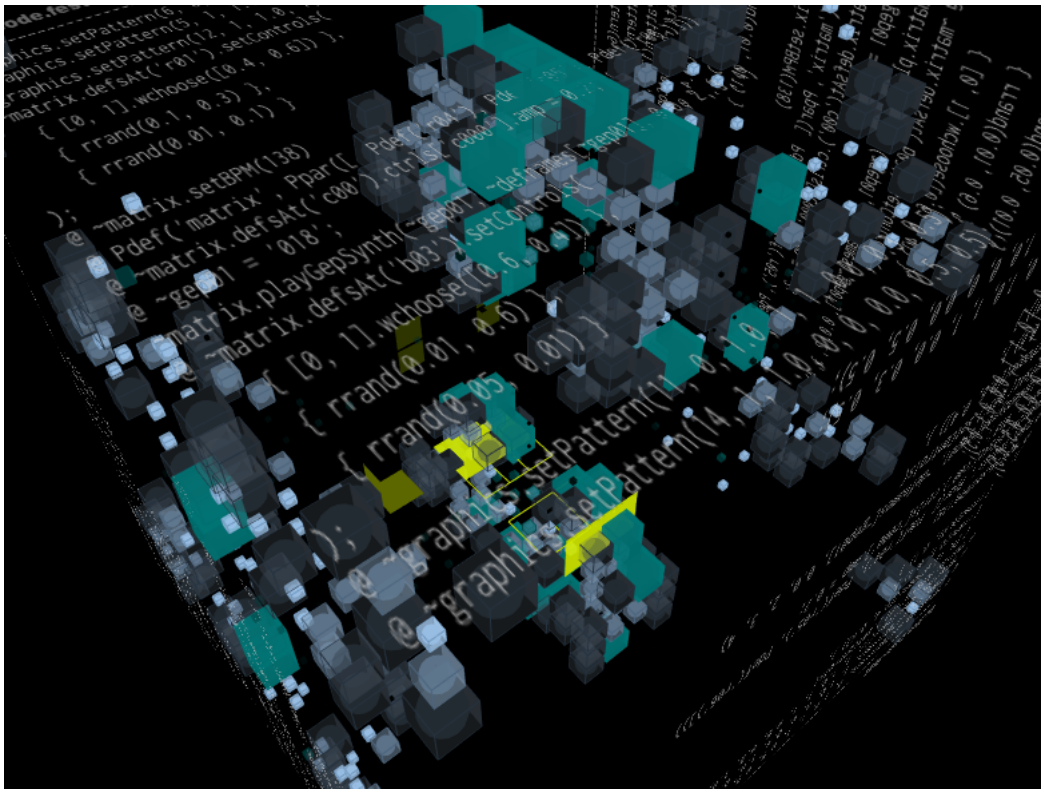


Figure 5.3: *sparsematrix* graphics with the code mapped onto the edges of the world.

5.4 Exploring the audiovisual space through live coding

Figure 5.4 shows the top section of a file uses in a live coding session and serves as a demonstration of different levels of control in *sparsematrix*. The system does not produce any sound or graphics until line 15 in the figure (The line beginning with Pdef('mx', Ppar....).play). This is a good illustration of one of the main obstacles in the beginning of a set that keeps the audience waiting while the performer types away to get the necessary system components running. The first five lines initialize the Ambisonic decoder, the graphical application interface for SuperCollider, the composition module of *sparsematrix*, and enables sending any code that is evaluated to the graphics application from the window that is used for coding. Then the graphics application is remotely launched and system audio is initialized, including activating the decoder, setting up necessary routing infrastructure and spectral analyzer that sends data to the graphics application for training the SOM. The musical structures explored are defined by the sparse rhythm patterns that were described

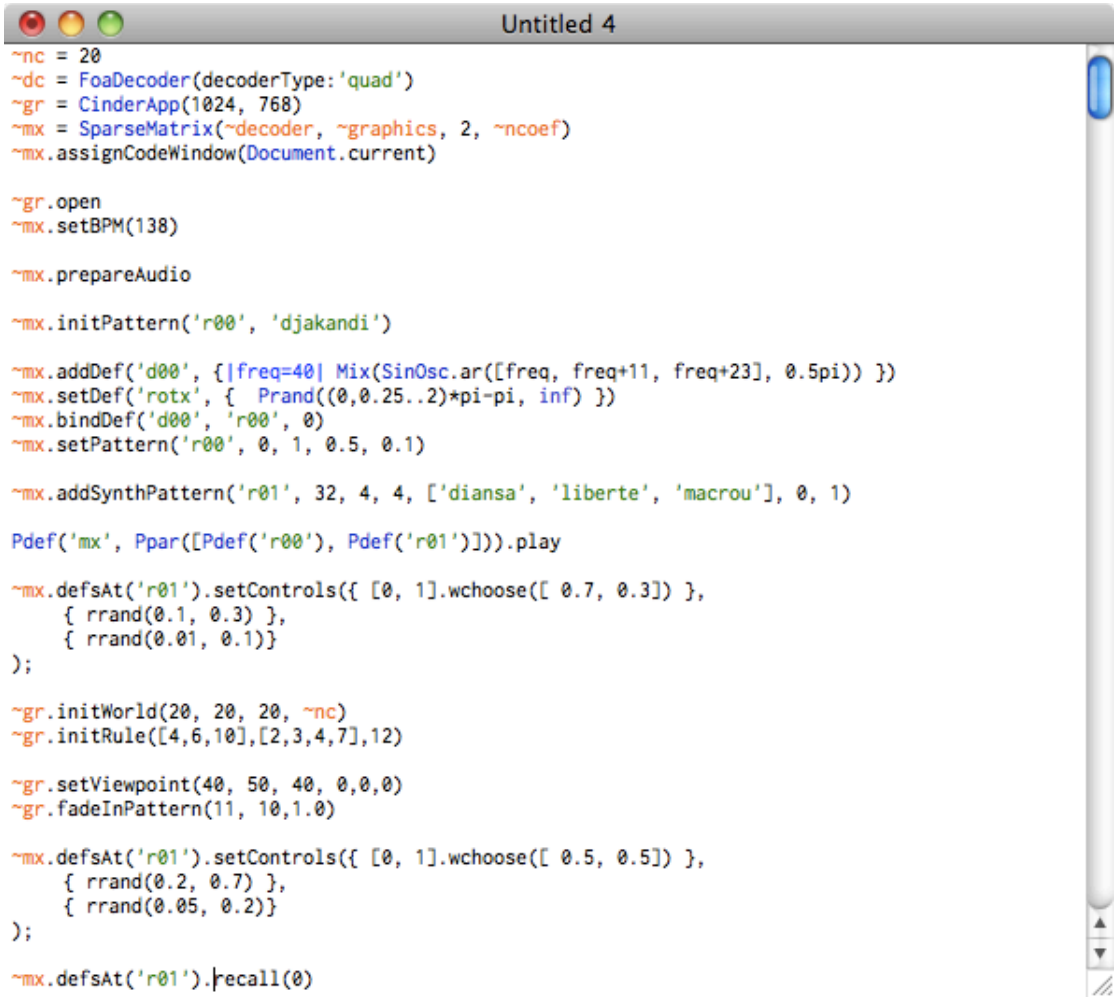


Figure 5.4: Screenshot of the top section of a live coding session file.

earlier in the chapter. Associating sounds - both synthetic and sampled - with the patterns can be accomplished on two levels of granularity: they can be added on-the-fly during the coding session or predefined and loaded when initializing the system. The 1s in each matrix represent sonic events, which can be assigned any synthesis definition together with amplitude, duration, envelope shape, effects bus, effects amplitude, and rotation parameters along 3 axes for the Ambisonics spatialization. This allows the painstaking build-up of live coded structures more akin to the idealist approach of “coding everything from scratch”. Alternatively, entire patterns can be populated with just a single line of code encapsulated in the addSynthPattern method call. On line 18 of the example, a pattern is defined to be identified as “r01” with 32 individual instruments. The instruments are grouped into layers of 4, parameters of instruments in each layer can be controlled in this way by layer as well as individually. The next 4 designates the beat division of the pattern, allowing

different patterns to be played at varying intervals. The next three symbols in square brackets select 3 patterns from the rhythm library to provide the 32 instruments with rhythmic content. This approach accomplishes more complex rhythms to be employed much faster and makes available more layers to be instantly available for control. The individual patterns are not immediately audible once added to the main player pattern Pdef('mx'), but have to be activated through a special parameter 'active'. This can be achieved by setting controls of all individual instrument patterns of a global pattern. In the example, the parameters for all 32 patterns of the "r01" super-pattern are set stochastically by providing functions to be evaluated for each individual. Activation is determined by a probability weight choice, where in each case it is 0.3 or 30% chance that the individual layer is activated. Amplitude for each is determined within a range of random values between 0.1 and 0.3, while durations for each event are similarly random values between 0.01 and 0.1. At this point the graphics application is not displaying anything, so the combined CA/SOM world is initialized with to contain 20 cells/nodes in each dimension and 8000 in total. The number of MFCCs for the training of the SOM is specified at initialization. The next step defines a custom outer totalistic rule for the automata world with "births" occurring if center cell has 4, 6, or 10 "alive" neighbors and it survives only if there are 2, 3, 4, or 7 neighbors in state 1. The transition from 1 to 0 lasts for 12 time steps. Next, the viewpoint of the camera is set to point at the center (0, 0, 0) of the world from coordinates (40, 50, 40) and finally, a visual pattern number 11 is faded in over 10 seconds until the transparency of the patterns is at maximum 1.0. Due to the large space of possibilities for musical exploration which is the product of combinations of all available patterns, synthesis processes, and parameters, stochastic functions for changing parameters are normally used quite liberally in order to find interesting or satisfying combinations, the system has a number of built-in memory functions that enable the performer to store and recall certain configurations that may be encountered. Each use defined super-pattern has an internal memory buffer of parameter settings which can be recalled by an index into the past. The last line in the example shows the recall function which returns the super-pattern "r01" to its previous state.

5.5 Discussion

The *sparsematrix* system grew into its current form through a combination of different factors, including the use of live coding as fast prototyping during the development combined with a certain disillusionment with graphical user interfaces for live performances. However, live coding is not an aesthetic to be explored in its own right, but rather affords the necessary means for exploration of a constrained musical concept-space defined as a collection of rhythm patterns. The objective here is to investigate the chasm between long-lasting cultural traditions and the exponential evolution of computational technology. The user of *sparsematrix* acts like a mediator between these two contrasting domains in a way, associating signal processing algorithms to skeletal structure matrices. The conditions of sparseness construct models of musical structure, only preserving the most essential components. Computational model building is an effective way to explain and understand natural phenomena and is widely used in many academic disciplines. The *sparsematrix* algorithm could potentially be used for structural analysis of any kind of music. At the same time, there is a definite allusion to DJ culture and electronic dance music as different rhythmic structures are layered and mixed during a performance, creating a sound world which is characterized by repetitive electronic beats and textures.

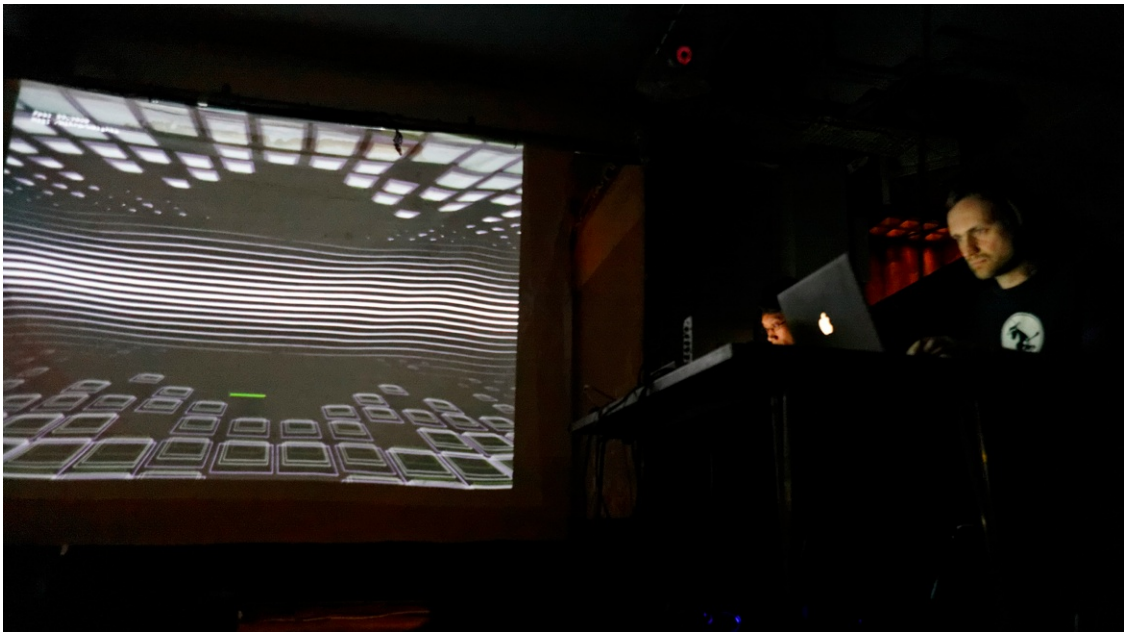


Figure 5.5: Live coding with Yota Morimoto at the SuperCollider symposium in London, April 2012.

The environment has been adapted for a collaborative project with Yota Morimoto. in this collaboration between two audiovisual artists who explore the nature of digital environments from different, mutually augmenting perspectives, the direct revelation of binary dynamics and rhythms of electronic circuits through custom-built sonic and visual models is expanded and contrasted with complex behavioral patterns of 3-dimensional artificial life and genetic programming algorithms. The tension and contrast between dense pulsating textures of binary audiovisual patterns on the one hand and rhythmic patterns originally derived from traditional ethnic music, but then digitized, filtered, and dehumanized in the network of computational agents and mechanical automata afford the audience a challenging perceptual experience, simultaneously challenging, disorienting and yet familiar. Figure 5.5 captures a moment from a performance with Yota at the City Arts and Music Project during a live coding evening of the SuperCollider symposium in London in April 2012. Technically the system consists of three computers, two of which have stereo audio outputs and the third operates as the combined graphics server with a VGA video output. The graphics application has been implemented in OpenFrameworks[1] C++ libraries and involved porting all the necessary *sparsematrix* objects and OpenGL patterns into Yota's *s0* performance system.

[1] http://www.openframeworks.cc

6 | Gene expression synthesis

Gene expression programming presents an alternative approach in the evolutionary computation paradigm evolving populations of candidate solutions as valid computer programs that can be used for a potentially wide range of problem solving tasks, including sound synthesis. This paper proposes Gene Expression Synthesis (GES) as a method to evolve sound synthesis functions as nested graphs of unit generators. The functions are encoded into linear chromosomes according to the principles of gene expression programming and then evolved by subjecting the functions to genetic operations and evaluating fitness. The design of the fitness functions involves statistical methods and machine listening algorithms in an attempt to automate the supervision of the synthesis process. The specification of synthesis parameters explores the idea of artificial co-evolution. A parallel population of functions share their fitness values with their respective synthesis functions for which they compute parameter values, while being subjected to genetic operations including recombination separately.

6.1 Introduction

Since the first artificial life experiments by Nils A. Barricelli in the 1950s, evolutionary computing has inspired numerous problem solving and model building techniques including ways to evolve sound synthesis algorithms inspired by processes of evolution by adaptation and natural selection. In our attempts to understand these natural algorithmic processes, which are purposeless and devoid of any intention, but nonetheless directly responsible for all the complexity and intelligent behavior in the natural world, we keep developing increasingly more powerful technology that enables us to model and simulate, albeit on a vastly simplified scale, the power of cumulative selection. Genetic algorithms and genetic programming have been firmly established as efficient and productive stochastic search and optimization methods within the artificial intelligence field and have been widely used in various disciplines for years.

Gene expression programming was introduced as an improvement to the existing paradigms, proposed by Candida Ferreira in 2001, by combining the best features of genetic algorithms and genetic programming [34]. The fundamental differences between gene expression programming and its predecessors stem from the separation of genotype-phenotype representations and the modular multigenic structure of the chromosomes. These improvements account for significant increases to the efficiency of the algorithm for a number of benchmark problems. The following account describes an experimental approach to evolving sound generating programs with the proposed principles and explores creative applications of evolutionary computation which do not necessarily presume a definite solution to a problem, but rather an open-ended solution space to be explored for aesthetic experimentation.

6.2 Sound synthesis with evolutionary algorithms

The evolutionary paradigm has been harnessed in a broad spectrum of applications in the realm of computer music, applying the processes of gene expression, selection, reproduction and variation on many different levels of compositional hierarchy. Examples can be drawn throughout all musical time levels, including producing waveforms directly by expressing binary genotypes as sample level time functions, evolving synthesis graphs and optimizing parameters, generating longer time structures and patterns of motives and phrases, all the way to composing comprehensive artificial environments inhabited by listening and sound-generating agents. Magnus [53] developed a modified genetic algorithm that works directly on time-domain waveforms to produce genetically evolved electroacoustic music. Garcia [37] [38] proposed using evolutionary methods for selecting topological arrangements of sound synthesis algorithms and for optimizing internal parameters of the functional elements. On the phrase and motive level, there are two classic studies that paved the way for countless later explorations: John Biles [13] hierarchical GenJam system that generates on-the-fly jazz chord progressions and the “sonomorphs" proposed by Gary Lee Nelson [66]. Jon McCormack [56] developed an interactive installation of evolving agents influenced by the presence and movement of audience as an example of a comprehensive digital sonic ecosystem. These are but a few examples of the wide range of applications

for evolutionary algorithms and by no means meant as a review, rather a random sampling of applications on different levels of the compositional process.

The abundance of different possibilities explored demonstrates the potential inherent in evolutionary processes which can exhibit unparalleled efficiency and problem-solving resourcefulness even in a vastly simplified form as compared to the forces operating in the natural world. The idea of automating the design process of sound synthesis algorithms using evolutionary methods has to be considered in the context of computer music specification. Generating waveforms by the direct principle of sample-by-sample calculation, for example, does not necessarily require any higher level infrastructure or a specialized programming environment, however, such an approach may complicate the design of an efficient fitness function, especially considering unsupervised learning methods. Since the Music N programming languages (most prominently Csound), the encapsulation of sound generating and processing functions into unit generators has cultivated a modular graph based concept of synthesis with interconnectable functions as building blocks. Most contemporary synthesis software, regardless of whether the interface is graphical or text-based, operates based on this model. The method presented here has been implemented in the SuperCollider environment, but is applicable in any audio programming environment that has adopted the graph based paradigm, where sound synthesis programs are defined as interconnected unit generator graphs. These graphs can be evolved by evolutionary programming principles just like any other computer programs that serve as the solution space for a particular problem. The question then becomes how to define or, in other words, encode these graphs in terms of evolutionary programming.

SuperCollider synthesis topologies have previously been studied in the context of evolutionary programming. Dan Stowell [83] presented a genetic algorithm for live audio evolution at the first SuperCollider symposium in Birmingham 2006. The system demonstrates how genetic methods can be used in a live setting, with modifications to the synthesis process occurring in real time. Fredrik Olofsson [70] released a similar algorithm for sound synthesis through his personal website. The goal of his project was to create genomes that would translate into realtime synthesis processes and allow the user to evaluate the results in a framework of a realtime sequencer. The algorithm is, similarly to the one described above, based on arrays of floating point values serving as

genomes, which were translated into SuperCollider synthesis definitions.

The SuperCollider implementation of the gene expression programming proposed here expands on the foundations of the methods described above. The problem addressed is how to encode SuperCollider unit generator graphs as populations of chromosomes and evolve these graphs using genetic operators. In a similar way, there is a constrained selection of unit generators that are included in the graphs and the translation process produces valid sound generating functions that are evaluated for fitness. However, the following description introduces a number of modifications and distinct features in accordance with the techniques of the gene expression algorithm to introduce an alternative strategy for evolutionary sound synthesis.

6.3 Components of GES

Gene expression programming (GEP) is a method of evolutionary computation providing an alternative to the established paradigms of classic genetic algorithms (GA) and genetic programming (GP) [34] [35]. The basic premises that these methods share in common have been inspired by biological evolution and attempt to model the natural selection process algorithmically in computers. All these methods use populations of individuals as potential solutions to a defined problem, select the individuals from generation to generation according to fitness, and propagate genetic variation within the population by random initiation and applying genetic operators. The differences between these algorithms are defined by the nature of individuals. In GAs the individuals are fixed length strings of numbers (traditionally binary); in GP the individuals are non-linear tree structures of different sizes and levels of complexity. GEP combines these approaches by encoding complex expression trees as simple strings of fixed length to overcome the inherent limitations of the previous methods. In GEP the genotype and phenotype are expressed as separate entities, the structure of the chromosome allowing to represent any expression tree which always produces a valid computer program. Another feature to set GEP apart from its predecessors is the structural design of GEP individuals that allows encoding multiple genes in a single chromosome. This facilitates encoding programs of higher complexity and expands the range of problems that can be solved with evolutionary computing.

GEP consists of two principal components: the genes (genotype) and the expression trees (the phenotype). The information decoding from chromosomes to expression trees is called translation. The genome or chromosome consists of a linear, symbolic string of fixed length composed of one or more genes. Each gene is structurally divided into two sections: a head and a tail. There are two types of smallest units called codons that make up a gene: functions and terminals. Terminals operate as placeholders for static variables or arguments to the functions. The head of a gene contains symbols representing both functions and terminals with the start codon always holding a function while the tail is entirely made up of terminals. This structure and the particular rules of translation in GEP ensure that each gene encodes a valid computer program. Despite the fixed length of the genome, each gene has the potential to encode for expression trees of different levels of complexity and nesting. The translation from genotype to phenotype follows a simple, breadth-first recursive principle: as the codons of a gene are traversed, for each function encountered, the algorithm reserves a number of following unreserved codons as arguments to that function regardless whether they are functions or terminals. The number of codons reserved depends on the number of arguments the function encountered requires. In order to illustrate this process, encoding of a simple phase modulation graph is shown in Figure 6.1. Such a gene would have to consists of a head section with at least 3 codons and tail with at least 6. The first 3 positions in the head of this gene contain the two sine oscillator functions and a terminal in between (the head part of the gene is indicated by a shaded grey background). The tail is entirely made up of terminals.

0	1	2	3	4	5	6	7	8	9	0	1	2
O	a	**O**	b	c	d	e	f	g	a	c	d	b

Figure 6.1: Encoding a phase modulation instrument as a single-gene sequence

In the *Karva* notation [33] this chromosome is represented as a string of upper and lower case letters with position reference numbers above:

```
0123456789012
OaObdefghcdbc
```

The expression tree that emerges form this gene after the translation pro-

cess is shown in Figure 6.2.

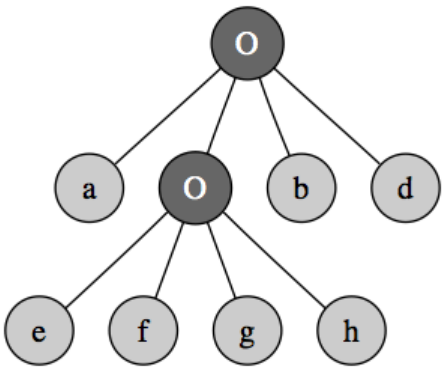


Figure 6.2: Decoded expression tree of phase modulation as a graph of sine oscillators. Sine oscillators are designated by captial O and terminals by lower case letters

The first codon designating a sinewave oscillator audio rate method (represented here by symbol O) - which in SuperCollider language specification expects four arguments: frequency, phase, mul and add - is translated as the root node in the expression tree with four branches deriving their values from codons in positions 1 to 4 in the chromosome string as they get reserved according to number of arguments into the function. When the algorithm encounters a terminal, there is no need to reserve anything and the terminal is assigned its position in the tree with no further branching, however, when it comes across another function at position 2 in the head of the gene, it looks ahead to reserve the next sequence of codons, in this case four arguments are expected again, therefore terminals at positions 5 to 8 fill these nodes. Once the algorithm has filled all the function arguments, the process stops and the rest of the terminals in the tail section of the gene are ignored. This mechanism allows to define the potential complexity and nesting in the resulting computer programs as a function of overall gene length. The expression tree above translates into a corresponding SuperCollider unit generator graph function:

```
{arg a, b, c, d, e, f, g;
        SinOsc.ar(a, SinOsc.ar(d, e, f, g), b, c)
}
```

The size of the gene tail t is calculated based on the size of the head h and the number of terminals n required by the function with the largest number of

arguments.

$$t = h(n-1)+1$$

Another feature that sets GEP apart from other evolutionary algorithms is the use of multigenic chromosomes. Multigenic chromosomes can be combined together by a function that serves as a linker. In order to provide an example of a multigenic chromosome, let us consider a slightly more complex example than the phase modulation graph above. This time there are four unit generators involved: sine oscillator SinOsc (O), sawtooth wave oscillator LFSaw (S), random values oscillator with quadratic interpolation LFNoise2 (N) and band-limited pulse wave generator Pulse (P)[1]. Since the generator with largest number of arguments is the sine tone oscillator and the head size remains the same for the time being, the gene size is also the same as above, but this time the chromosome consists of two genes which are linked together by mathematical multiplication function in the translation process.

The gene expression tree of this chromosome consists of two independent sub expression trees corresponding to the multigenic structure: the first one has a noise generator as the root codon and the second one a sawtooth oscillator. There is an additional linker function, in this case multiplication, which combines the genes together into a single composite function, as shown in Figure 6.3

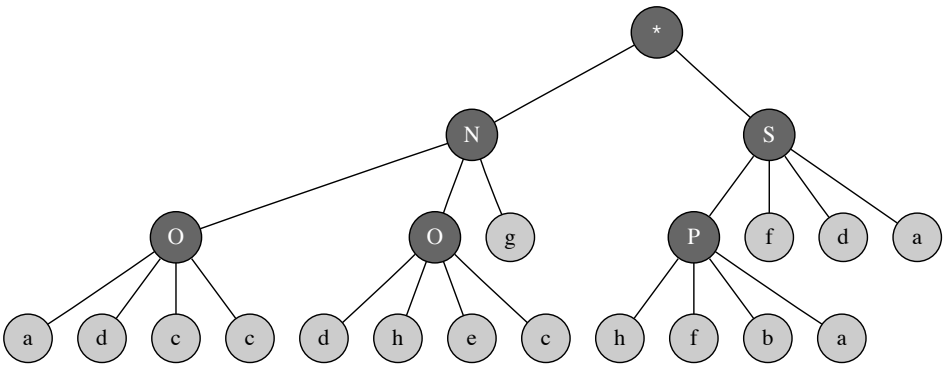


Figure 6.3: Expression tree of a multigenic chromosome with multiplication function serving as a linker.

[1]All the examples presented in this paper use these four unit generators with their corresponding abbreviations

This expression tree translates to a unit generator graph function in SuperCollider:

```
{arg a,b,c,d,e,f,g,h;
        LFNoise2.ar(
                SinOsc.ar(a,d,c,c),
                SinOsc.ar(d,h,e,c),
        g) *
        LFSaw.ar(
                Pulse.ar(h, f, b, a ),
        f, d, a )
}
```

GEP chromosomes contain several genes each coding for structurally and functionally unique expression trees. Depending on the problem to be solved, these sub-trees may be selected individually according to their respective fitness or they may form a more complex multi-subunit expression tree and be selected according to the fitness of the whole chromosome. The linker between the individual expression trees can also be any function and depends on the context of the task at hand. For example, in the above structure, the multiplication could be substituted by addition to produce additive synthesis instead of modulation synthesis or any other function that requires two arguments.

6.4 The selection process

The gene expression process does not differ much from that of the classic genetic algorithms. It begins with the random generation of chromosomes of a certain number of individuals for the initial population. In the next step, these chromosomes are translated into computer functions to be executed and the fitness of each individual is assessed against a set of desired examples which act as the environment to which the individuals are to be adapted. The individuals are then selected according to their fitness (their performance in that particular environment) to reproduce with modification, leaving progeny with new traits. These new individuals are, in their turn, subjected to the same developmental process: expression of the genomes, confrontation of the selection environment, selection, and reproduction with modification. The

process is repeated for a certain number of generations or until a good solution has been found.

The initial population in gene expression programming is created in the same way as in other evolutionary computation algorithms either by randomly populating the gene codons with functions and terminals determined to be part of the solution space or using pre-existing individuals from a pool of previous successful runs. In case of random generation of the population, which is by far the most common method used, the genes are constructed, first, by randomly selecting a root node from the included function definitions, then the head codons are filled by randomly selecting a function or a terminal for each position and, finally, the tail only includes random selections of terminal values. Although, it is not absolutely necessary to define a root node as a function according to GEP principles, especially in multigenic chromosomes, it proves more crucial of a factor in the special case of sound synthesis. Sound synthesis is a special case for more than one reason and the many constraints that it imposes on the GEP paradigm will be discussed in detail in the following sections.

As in any other evolutionary programming model, the most important and challenging component in GEP is the design of the fitness cases as this is what drives the fitness of the population and ultimately decides the success of the problem solving algorithm. In most cases which are trying to find the single best solution to a particular problem, the goal must be defined clearly and precisely in order for the system to evolve in the intended direction. Although it may not always be the case, particularly while evolving candidate solutions for complex, open-ended situations including sound synthesis or musical phrase composition, a poorly designed fitness function tends to produce random meaningless results and either converges on an inappropriate solution or will not converge at all producing consistently large error values in individuals with the highest fitness.

The selection process commences once each individual in the population has been assigned a fitness value. The purpose of this phase of the algorithm is to propagate the fittest solutions to the following generation. Again, there are a number of different methods by which to select the individuals, stochastic and deterministic, however in the long run it makes little difference which one is used as long as the best traits of the current population are preserved in

the new population. The preferred method in GEP is stochastic, which entails assigning each chromosome in the population a probability weight value proportional to its relative fitness. This may mean that the fittest individual may not always survive the selection process while mediocre individuals might be selected.

6.5 Genetic operators

The selection process has a tendency to converge towards a single high scoring solution and, without genetic operators, would rapidly get stuck in a local optimum. Therefore it is essential to maintain genetic diversity, which is mainly achieved by several modifications introduced during the replication process of the genomes. There are a variety of genetic operators in GEP divided into three main categories: **mutation**, **transposition**, **recombination**.

Mutation entails modifying a single value in a randomly chosen position and can occur anywhere in the chromosome. However, the structural organization of the chromosome must be preserved to ensure that when expressed the individual still produces a valid program. This means that the root can only be replaced by another function, any codon in the head section of the chromosome can be substituted by a function or a terminal and only terminals are allowed as replacements in the tail section. Mutations of a single codon can have a dramatic effect on the phenotype a chromosome is encoding, especially if it occurs in the head section. The following Karva notation strings display a mutated chromosome before and after the mutation, in which a terminal that occurs in position 1 in the original gene has mutated into a sine oscillator in the next generation:

```
0123456789012
NcOgadccdhecc

0123456789012
NOOgadccdhecc
```

Figure 6.4 shows the effect on corresponding expression trees of this single-point mutation.

Mutation rate is defined as a global constant in the GEP algorithm and can be specified as a probability percentage which each chromosome is subjected to. If the mutation rate is defined as 0.1, it means each chromosome has a 10% chance of being subject to a random one-point mutation.

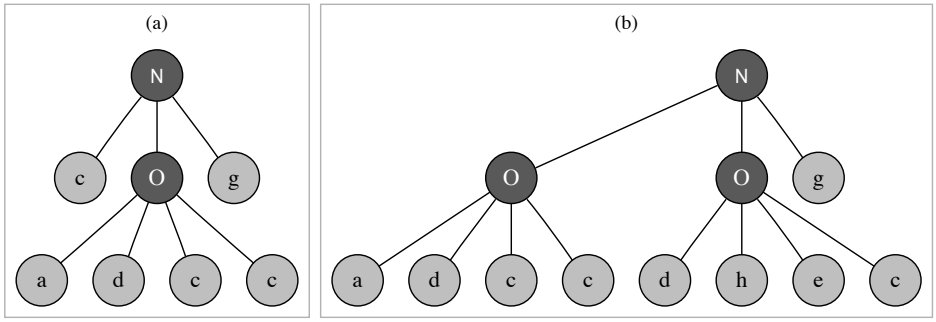


Figure 6.4: Single-point mutation. A terminal in (a) changes into a sine oscillator function in (b)

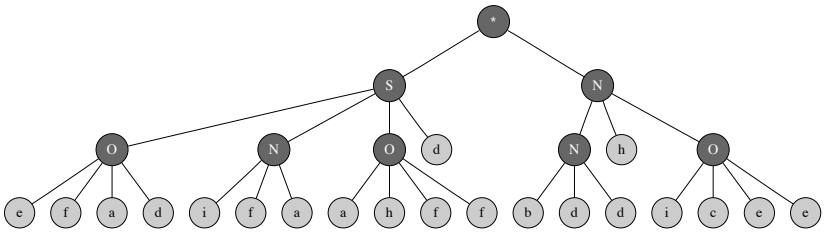
The transposition operations in GES copy short fragments of the genome from their original locations to another location in the chromosome. For example the already familiar gene from two previous examples is subjected to transposition of a short codon sequence shown in Karva notation and Figure 6.5. The terminals at locations 5 and 6 are copied into the head section of the gene, which results in the first two parameters - frequency and phase in this case - of the root codon sawtooth oscillator of the first gene to be replaced by a noise generator and a terminal instead of a sine oscillator and a noise generator.

```
012345678901234567890123456789012345
SONOdefadifaahffbN NhO bddiceedaebcd

012345678901234567890123456789012345
S NhO defadifahffbaNNhObddiceedaebcd
```

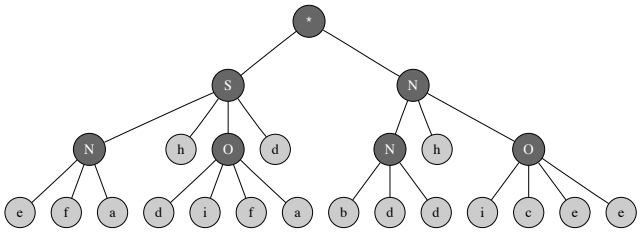
The SuperCollider synthesis function that is derived from the transposed tree is shown in the code listing below:

```
SynthDef('r00_g02_s001', {arg a,b,c,d,e,f,g,h,i;
        Out.ar(0,
                (LFSaw.ar(LFNoise2.ar(e,f,a), h, SinOsc.ar(d,i,f,a),d))*
                (LFNoise2.ar( LFNoise2.ar(b,d,d),h, SinOsc.ar(i,c,e,e)))
        )
})
```

Recombination involves choosing chromosomes from the pool of individuals that have successfully passed the selection process and exchanging their genetic material. This process results in creation of two new individuals. A defined number of points are randomly chosen along the two parents and their



(a) original



(b) transposed

Figure 6.5: Transposition of a codon sequence

codons are copied to the child chromosomes as mixed set containing codons from each of the parents. In order to illustrate the basic principles and effects of recombination let us consider two chromosomes derived from the same four unit generators presented previously. The listings below display two parent chromosomes in Karva notation (head sections in bold):

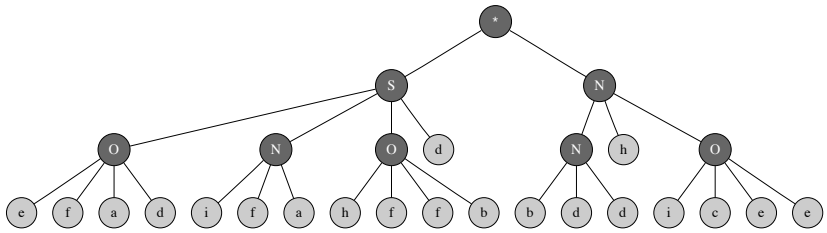
```
0123456789012345678901234567890123
SONOdefadifahffbaNNhObddiceedaebcd

0123456789012345678901234567890123
PPNSahihgifbbcdafOePNhbddhgbhhgdee
```

After subjecting these chromosomes to recombination, the result is two new individuals that have characteristics of each of the parents. In the symbol strings below, the components that made up the original chromosome 1 are indicated in bold to illustrate the effect of recombination. The first of the two randomly selected recombination points is located at position 3 of the chromosome and the second occurred at position 27 located in the head section of the second gene.

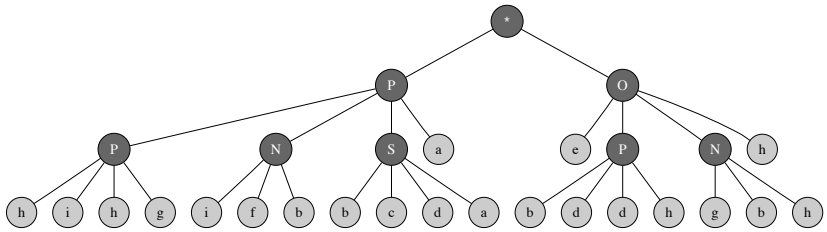
```
0123456789012345678901234567890123
SONSahihgifbbcdafOehObddiceedaebcd

0123456789012345678901234567890123
PPNOdefadifahffbaNNPNhaadhgbhhgdee
```

The corresponding expression trees of the two parents and their progeny is

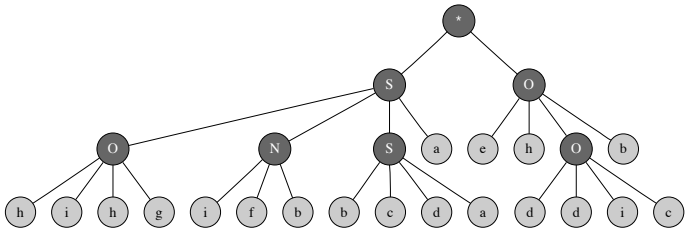
shown in Figure 6.6



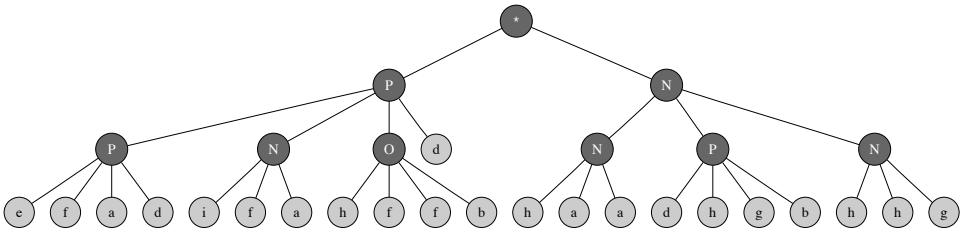
(a) parent A



(b) parent B



(c) child A



(d) child B

Figure 6.6: Recombination

These are relatively simple examples in order to demonstrate the principles of genetic operations in gene expression synthesis. The synthesis functions that have been evolved so far using this technique typically originate from chromosomes consisting of at least 4 up to 8 genes and head sizes ranging between 8 to 16, resulting in much more complex graphs with more levels of nesting. While the genetic operations ensure variability within the population, evolution towards a goal is largely determined by a fitness function.

6.6 Evolving unit generator graphs

The algorithm works in a cyclical pattern as illustrated in Figure 6.7, first an initial population of n individuals is generated, then each individual is expressed as a recursive expression tree beginning with the root node which can then be translated into a sound synthesis function string. The function string is evaluated and a synthesis process is started on the server. An analyzer agent then assigns a fitness value to each individual. The selection process is stochastic and associates a probability weight to each individual based on their relative fitness. Replicated individuals are then subjected to a series of genetic operations depending on the settings of the algorithm. Once every new individual has been exposed to the genetic operator phase, the cycle is completed by replacing the original population with the new individuals which then are ready for the subsequent repeat of these steps.

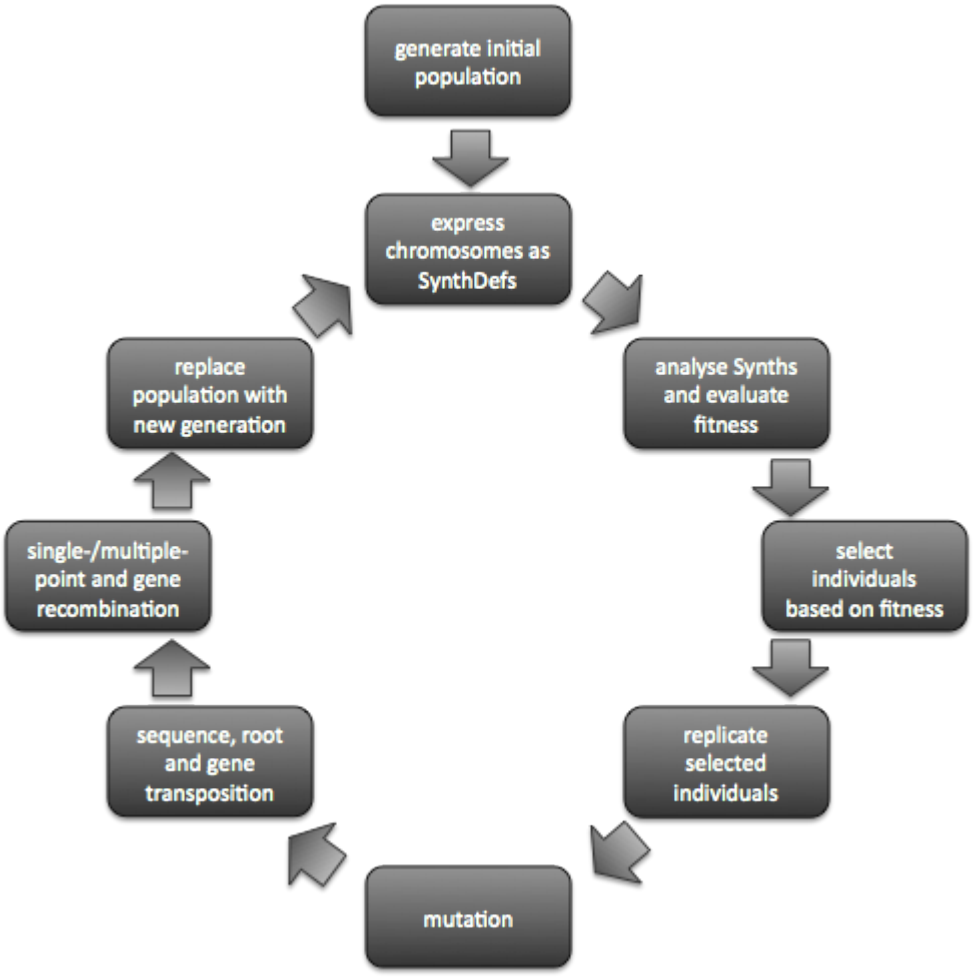


Figure 6.7: The cyclic gene expression programming algorithm.

In order to begin the process of evolving unit generator graph functions

for sound synthesis, there are two crucial components to be defined: (1) the specification of terminals as synthesis function arguments and (2) the design of the fitness function so that the evolution proceeds towards desired goals with minimal human supervision. The specification of terminals was solved by introducing a parallel population of calculation functions in which each individual becomes expressed as a list of floating point values. This parallel population is evaluated simultaneously with the sound generating functions and each individual receives the same score as its counterpart in the sound generating population. However, the selection process and the genetic operators are applied separately so the population retains a certain degree of independence. The number of genes in this parallel population corresponds to the number of terminals necessary to fill all the parameters. The functions used in this population are not sound generating functions, but binary arithmetic operators of addition, subtraction, multiplication, and division and the terminals are static floating point values. This solution imitates the phenomenon of co-evolution in the natural world where two interdependent species indirectly cause mutual evolutionary changes across the confines of their genotypes.

6.7 Defining the fitness function

The fitness function uses machine listening algorithms to analyze the candidate solutions once they have passed an initial basic compilation test on the SuperCollider server. Before the machine analysis can commence, any individual that fails the basic fitness check and the expressed function fails to compile, is automatically assigned a weight value of 0 and consequently excluded from the selection process. Compilation may fail for any number of reasons, the most common being invalid input type and since initialization is completely random, unsuitable function arguments become quite frequent in case unit generators that have arguments of specific type. A good example of an invalid unit generator argument would be in case of a filter algorithm which expects the first argument to be a signal of the same rate (typically audio rate in this case) as it is running itself, therefore a floating point number is not accepted and compilation fails. There is an option to start the process by filling the initial population exclusively with candidate solutions that pass this check.

The machine listening process analyses a set of 20 mel frequency cep-

stral coefficients (MFCC), spectral flatness, spectral centroid, and amplitude features into running mean and standard deviation values over a desired duration, 3 to 8 seconds in the runs reported in this account. Invalid output from any of the analysis processes (mostly NaN or unrepresentable value as a result of a calculation, dividing 0 by 0 for example) is assigned an error value greater than one which gets treated the same way as uncompilable functions and is thereby excluded from the selection process. The fitness function that was used in all the variants of the gene expression experiments under investigation in this case used example analysis sets extracted from sound examples towards which the algorithm was expected to converge. A number of different reference sounds were used including sounds synthesized with GES, other types of synthesized sounds as well as sounds of traditional musical instruments. The score of each individual was determined as the difference between maximum possible score and the total actual error in each of the analysis categories. The mean and standard deviation statistics of each of the MFC coefficients were given double weighting relative to other statistical values and the maximum error in each of the statistical categories was set to 1.0. Spectral centroid values, which are expressed in frequency values, were mapped to range between 0.0 and 1.0. This meant a maximum individual score of 10.0 as the sum of scores from MFC coefficients adding up to 2.0 for both mean and standard deviation statistics, and to 1.0 for spectral flatness, spectral centroid, and amplitude.

Table 6.1 represents the assignment of initial fitness scores which were calculated as difference measures from the corresponding features of the reference sound. Each of the 20 mean MFCC coefficients were each assigned a weight value of 0.1, which means that the maximum score possible from the sum of these features is 2.0. The same weight is assigned to the standard deviation values of MFCC. The remaining 6 features - mean and standard deviation values for Spectral Flatness, Spectral Centroid, and Amplitude - were each assigned a weight of 1.0. Therefore the maximum similarity score possible is 10.0 in case of identical features.

In order to imitate the condition of limited resources of natural selection, each candidate solution is assigned a CPU usage value measured during the execution of the synthesizer. At the end of each evaluation cycle, the CPU usage percentage is normalized relative to the minimum and maximum values

FEATURE	NUMBER	WEIGHT	MAXIMUM
MFCC (mean)	20	0.10	2.00
MFCC (std dev)	20	0.10	2.00
Flatness (mean)	1	1.00	1.00
Flatness (std dev)	1	1.00	1.00
Centroid (mean)	1	1.00	1.00
Centroid (std dev)	1	1.00	1.00
Amplitude (mean)	1	1.00	1.00
Amplitude (std dev)	1	1.00	1.00
		Total maximum	10.00

Table 6.1: The weights of features used in the fitness function. The maximum score of each feature is calculated by multiplying the number of features by the weight

of the population and the scores recalculated adding in the CPU percentage as 10 percent of the total score. This pressure introduces a tendency in the population of favoring simpler synthesizer graphs over more complex ones. To counteract this tendency a conflicting fitness pressure is introduced to encourage structural complexity. Maximum depth of unit generator nesting is a straightforward indicator of complexity in graphs, so the scores are adjusted according to the maximum depth of a chromosome relative to the maximum of the population. This way, the complexity can be maintained in populations, while still encouraging resource usage effectiveness. These parameters can be adjusted depending on the purpose of the experiment.

6.8 Discussion

The most striking feature of the implemented synthesis system that emerged during the experiments is perhaps the phenomenon of high fitness scores being present starting from the initial randomly generated population. The maximum score remained fluctuating within a limited range at the top of the fitness landscape and did not seem to improve. Figure 6.8 shows a graph of mean and maximum fitness scores plotted against each generation in an experiment, in which a synthetic bass sound was used as reference.

This reveals the crucial characteristics of the algorithm and informs of inherent properties and constraints of sound generating functions going for-

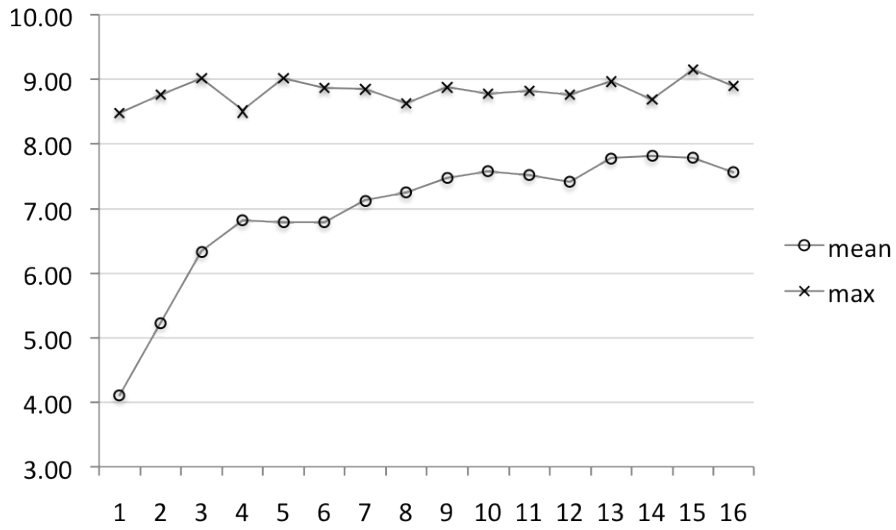


Figure 6.8: Maximum and mean scores of a gene expression synthesis experiment with the fitness scores on the x-axis plotted against the generation number

ward. One of the factors for this outcome is the additional server compilation check applied prior to exposing the functions to the statistical fitness evaluation procedures. This is an additional layer, which in standard framework of GEP problem solving would seem redundant, but has been devised to accommodate for the specific language-server architecture of the SuperCollider environment, which is essentially comprised of 2 different computer languages. Another characteristic that emerged from these experiments was the explosion of variety combined with rapidly improving mean fitness of the entire population after 2-5 generations, which produced the most interesting (as subjectively judged by the author) synthesis functions in large variety compared to later generations. The variety tends to diminish after a number of generations similar to classic GA-s as strands of highest fitness individuals take over the population. Furthermore, evolving sound-generating functions imposes rather strict constraints on the algorithm, which in these experiments were largely ignored as much as possible. For example, the co-evolution of parameters to the functions from a set of random calculations exposed the audio system to unexpected output values which were weeded out by the statistical analysis of amplitude tracking and bad value checks. Further normalizers and limiters were employed to try and keep the synthesis output within a perceivable (and tolerable) range. The persistence of relatively high but static maximum scores also underlines the limitations of the statistical fitness functions used in the experiments. The mean and standard deviation statistics of

the example sounds do not provide sufficient time-domain information, therefore, there was always a significant variety of functions with differing spectral and temporal characteristics attaining high scores. The main limitation to the sonic output is the selection of unit generators, which were naturally not expected to conjure up complex spectra of the human voice or traditional musical instruments. The fitness statistics were intended rather as rough guides to acceptable ranges of spectral characteristics of the candidate solutions and near exact matches were understood to be virtually impossible from the outset even in the case of GEP evolved synthesis experiment. In total, over 3000 new synthesis definitions have been selected as additions to the GES database as of this publication. These synthesis definitions are stored together with their statistical analysis data and linearly encoded chromosomes to be utilized in the interactive autonomous mikro improvisation system, live coding improvisation performances, and as genetic source material for further experiments in GES.

Garcia [38] presents a somewhat similar approach to evolving sound synthesis graphs. The sound synthesis algorithms are similarly represented as topologies of interconnected nodes in a graph while the fitness function also uses an audio feature based distance metric to compare the candidate solutions to a target sound. The evolution of graphs is driven by genetic programming, in which the population consists of individuals each of which represents a non-linear tree of varying complexity. This variation makes it difficult to apply genetic operators and thereby new solutions are not introduced into the populations. This means that in order for the algorithm to converge, the components for the optimal solution would already have to be present in the initial populations. [34] Since GES uses the gene expression programming algorithm to guide the search, this problem is solved by encoding the graph represenation (phenotype) into a linear representation (genotype) which enables more complex and efficient genetic operators to be applied. In Garcia's experiments, the algorithm is designed to converge to the target and in the example presented shows the error diminish significantly lower over 200 generations than in experiments reported here. This is mainly because (1) the implementation of Garcia's algorithm is on a lower representational level of synthesis using C++ and matlab while the experiments presented here utilize the large variety of unit generators available in SuperCollider, and (2) Garcia's

experiments are designed to converge to an optimal solution whereas GES experiments are designed for open-ended exploration.

The claims that the gene expression programming algorithm is superior to the traditional evolutionary algorithms [34] appear to be corroborated in the GES experiments based on the speed of population mean fitness increase and high maximum scores starting with initial populations. There are no direct comparisons with previous sound synthesis algorithms due to the fundamentally different nature of the studies considered. Existing genetic algorithms that have made use of the SuperCollider programming environment used human listeners as the fitness function, therefore it is not possible to compare the evolution of the fitness landscapes. These initial experiments have provided a rich insight into the myriad of sound synthesis possibilities latent in the GES algorithm. Based on these experiments and taking aboard the methodology explored in previous evolutionary programming attempts with SuperCollider, the algorithm can be expanded to include range limitations for unit generator parameters to safeguard against unreasonable output values. Special classes of unit generators - such as filters, buffer players, reverberation and spatialization functions ,Äì can only be incorporated by providing structural constraints in the design of the chromosome, but would significantly increase the complexity of the potential sonic output. Further, the parameter definition could be optimized more efficiently.

The gene expression synthesis method was designed as a tool to be used in the creative process, particularly to build up a database of synthesizers that could be employed in different live performance situations. This objective has guided the decisions made while designing fitness functions and experiments with the recognition that there are many alternative ways to implement the basic concept presented here.

There are many practical applications for gene expression synthesizers, that have been implemented in different projects. In *f(x)* the gene expression synthesizers have been added as additional textural sound sources controlled alongside the original layers created from recycled sonic material. The *mikro* system implements a set of these synthesizers as part of the sonic vocabulary of machine improvisers who make selections based on the audio analysis data stored in the repository. Recent live coding performances of *sparsematrix* have made use of synthesizers evolved for purpose to be computationally

less expensive and have sonic characteristics more suitable for percussive envelopes.

7 | Conclusions

> "Every last scrap of our external experience is of virtual reality. And every last scrap of our knowledge - including our knowledge of the non-physical worlds logic, mathematics, and philosophy, and of imagination, fiction, art and fantasy - is encoded in the form of programs for the rendering of those worlds on our brain's own virtual-reality generator"
>
> from *The Fabric of Reality* by David Deutsch

The three performance environments presented in this thesis explore different strategies for real time improvisation in audiovisual performances, at the same time representing different ways to interact with algorithmic systems on stage. Research and development of performance strategies is necessary due to the volume and complexity of real time composition and performance tools available to artist-programmers. The function that music and arts perform in society is constantly shifting and it is only natural that in the current context of an increasingly technological society this function is undergoing drastic changes. A number of trends previously difficult or impossible to realize in music and visual arts are becoming prominent as methods previously unthinkable have become commonplace. The strategies that have been introduced and improved over the last few decades, coinciding with the advances in computational power and capacity of personal computers, represent an impressive range of new opportunities. Whether it is interpreting data flow on the Internet as a musical instrument [68], using biological signals of a human body [32] or brain activity by EEG [69] to control musical parameters, all these represent attempts to improve on ways in which we explain the environment we inhabit and the technology we have created to help us explain. At the same time the realization that works of art are not imbued with some intrinsic value of authoritarian or mystical origin, but subject to similar conditions present in the biological evolution of selection and adaptation, provides a more objective backdrop to the audiovisual artifacts produced by performance systems

like *lambda*. The most important realization when describing the approach to composition in these audiovisual works is that musical compositions as well as any other artworks are not imbued with an intrinsic value originating from some higher authority as we often would have believed, but rather are subject to a selection process that resembles natural evolution by selection and adaptation. This cultural evolution is described in terms of memes, a concept introduced by biologist Richard Dawkins [27]. Audiovisual composition must be viewed in the context of current social, cultural and scientific developments, in which computation theory and artificial intelligence very much occupy a prominent role and thus constitute the main theme of exploration into generative computer music, audiovisual performance and immersive digital environments.

Virtual reality is not just an area of research within computer science typified by large headgear and tactile interfaces connected by bundles of cables. The tradition of electroacoustic music, particularly the acousmatic trend of engulfing the audience in darkness in order to be better conditioned for immersive imaginary sound worlds, is evidently a practice of virtual reality generation, a specific case of purely acoustic kind. The emerging practice of generative audiovisual performance, the central concern of this thesis, represents another class of virtual reality generators. These are examples of immersive environments created with technology, both digital (computers, audio interfaces) and analog (sound amplifiers, speakers). However, as suggested in the quote by David Deutsch at the top of this chapter, it is important to realize that virtual reality generators are not necessarily only digital computers executing instructions specified by humans, but every information processing device or organism is in possession of such a generator in order to create a representation of its environment through perception and cognition. Human experience can be viewed as a type of highly complex virtual reality generated by the nervous system which has evolved into the current state over millions of years of natural cumulative selection. The technology with which we have surrounded ourselves serves largely the purpose of virtualizing the human experience in order to model, explore, explain and possibly modify the environments we inhabit. Virtual reality can be put into the context computation as a representation of one information processing entity by another and was perhaps first formulated in Alan Turing’s concept of computational univer-

sality [88], which originally meant that any computation engine that is capable of computing anything that the universal Turing machine can compute is considered a universal computer. His thesis has been used to gauge the limits of computation and, in the course of this exploration, it has been extended to virtual reality generators. A virtual reality generator is deemed universal if it is capable of imitating any other digital machine, implying that, given resources and time, it is capable of rendering behavior of any computable phenomenon. Determining the limits of what is computable has been the object of rigorous research in the field of computation theory for years and is still very much an open question.

The implications of the concept of computability for creative processes that are the primary objects of interest in this project - such as musical composition, improvisation, and visual art - have been reshaping the way we consider artistic practices from the time first computers came about. One thought provoking aspect of our interaction with technology is the extent to which it is often overlooked how much technology guides the creative process. From conversations with other composers and many lectures and presentations I have attended, it seems that very often significant nuances of the interface to a piece of creative technology remain imperceptible. The users often claim intention where their choices have been mostly forced by the limitations of the interface they use. A great example of awareness of such limitations was presented by David Rokeby [73], a long-term practitioner of creative interactive technologies, at the 11th International Conference on New Interfaces for Musical Expression (NIME) in Oslo. In his keynote address, the artist described what must be a common occurrence in contemporary arts practice, particularly in cases where technology plays a significant role. His experience of inadvertently becoming influenced by the nature of the technology he was working with during development of interactive installation pieces in the 1980s provides ample insight into this common phenomenon and perhaps a cautionary note for anyone working with technology. It also suggests a method for artistic exploration of the technology itself: to purposefully design systems and performances based on the idiosyncrasies of the hardware and/or software involved. The common frustration stemming from what is sometimes called a "semantic gap" between top-down design of a project and technological facilities to realize it seems to be often caused by lack of understanding how the

chosen technology operates.

The different techniques for the artist-programmer to interface with algorithmic systems explored in these performance systems are an attempt to move away from conventional point-and-click graphical user interfaces and find ways to interact with algorithmic composition environments through external controllers, autonomous algorithmic agents or writing and modifying computer code directly. The acoustic nature of the performances is defined by several synthesis methods, which all have in common the focus on the spectral dynamics and development of timbre as the primary musical parameter. In case of *f(x)*, the sonic material is recycled from recordings of past performances and used as the source material to be manipulated and processed until the original material is unrecognizable. The *mikro* system relies almost entirely on the audio input of the human performer to process and transform the recorded samples into new material. However, a novel method of computer sound synthesis is also introduced in the shape of Gene Expression Synthesis (GES), which entails evolving SuperCollider unit generator graphs functions by the principles of gene expression programming.

Chapter 1 of this thesis establishes the dualistic interdisciplinary methodological framework utilized in this project combining practice-based methods with the principles of design science. The former methodology applies to the artefacts produced by the systems, i.e. the actual performances and the multimodal environments created as a result, whereas the design and development principles behind the software systems that are used in these performances belong in the domain of design science. This also informs the evaluation method most pertinent to the context to determine the utility, quality and efficacy of the performance systems. The following evaluation combines analytical with observational methods in an attempt to assess the research and development outcomes.

From static system architecture prespective, the performance environments can be evaluated according to modularity and extensibility principles, which then can be then put into the wider context of efficacy. Modularity in performance systems facilitates deployment of different components according to the performance context. The systems developed during this project share the same fundamental archtecture of hierarchical modularity. On the highest level of this hierarchy, the physical separation of the audio and graphical ap-

plications and the facility to deploy the applications in separate locations and performance contexts has not only enabled resource distribution at a performance venue between multiple computers, but also distributing applications to different geographical locations. In practice this has involved deploying the graphics application at a remote venue while streaming control parameter information and audio content over the Internet. Furthermore, the physical separation enables system components to be deployed individually and repurposed for applications originally not intended (for example the graphical component has been adapted as a VJ tool in club settings on a few occasions). The modular design is most evident in the audio modules of the *mikro* system where the different task-based components can be utilized without the rest of the system. For example, the analysis module is used to build the musical feature databases without the need to launch the entire system, but becomes an essential part of the compositional process during actual performances when real time input analysis informs the musical structure. The *mikro* system has benefited the most in terms of functional modularity consisting of modules desgned to perform a specific task such as audio input routing, audio analysis, machine learning, musical composition, data storage, and information retrieval. In contrast, the degree of modularity in *f(x)* and *sparsematrix* is considerably lower due to the idiosyncratic nature of the embedded compositional and computational concepts and, to a certain extent, the lack of premeditated modular design. In this respect these environments constitute independent modules in themselves and can be included and excluded from performances depending on context, but they have not been designed to be easily broken down into smaller independent components. The *f(x)* system is designed to be expandable by adding new distinct compositional components in the form of audiovisual mapping presets that can be utilized in shaping the overall audiovisual structure of the performances. The modularity in *sparsematrix* serves a similar structuring purpose enabling on-the-fly changes to the mapping of different patterns to synthesis functions, flexible signal routing to audio effects processes, and incorporation of gene expression synthesis functions into the sonic composition of the rhythms.

While the modularity and extensibility of the systems have achieved compositional and functional requirements, the storage and retrieval systems for musical analysis and gene expression synthesis data need to be redesigned to

optimize data access in live performance situations. In the current state, the systems need to be prepared in advance, using an initialization process that can take up to several minutes to complete. This situation is particularly undesirable in the middle of a performance when fast recovery is necessary due to unexpected system failure (certainly a familiar circumstance for any experimental electronic musician). There may be no immediate solution to speed up the musical training phase of the composition module due to large volume of data to be processed, but manipulation of gene expression synthesis data structures would potentially benefit from a Semantic Web solution involving JSON or RDF storage mechanisms in terms of speed as well as networked public accessibility.

The contextualization of the performance environments from the practice-based perspective relies on the concept of computability discussed in the beginning of the chapter. Anything computable can be expressed in algorithms, in other words in sets of explicit instructions. Algorithms, at a certain level of complexity and possessing certain characteristics, can be considered inherently emergent, creating larger and more complex entities through interactions among simpler building blocks. A common theme throughout the three performance systems under discussion is how computational emergence can be used as a general principle of interactive composition in an improvisatory context. It could be argued that emergent systems in this kind of performance context cultivate improvisation, since it is impossible to predict the behavior beforehand to be able to establish a set of premeditated instructions akin to a musical score. In this computational environment, the artist-programmer acts as explorer of algorithmic behavior interacting with a virtual multi-dimensional data space. There is no singular, linear path through the vast parameter space of compositional possibilities and each instantiation of one of these performance systems always establishes an unfamiliar initial condition distinct from any previous instance. The behavioral feedback loops established between computational systems and the actions of the performers reinforce the improvisatory nature of the resulting performance. No performance can ever be repeated even approximately, a characteristic already ingrained in the non-linear nature of many of the low level signal processing and synthesis methods that create the spectrally diverse and dynamic sonic makeup of the performances.

The musical forms that emerge from the performances are very different from the linear left to right matrix structure familiar from audio and video editing and sequencing software. In a similar manner they deviate from the frontally focussed sound space of the prevailing cinematic sound systems like 5.1 and its many variants. No direction is emphasized or marginalized, the diffuse sound field surrounds the listener from all sides, the dynamic shape of which does not tend in any one direction, but rather through constant readjustment and reforming creates an immersive and unpredictable sonic environment. The general structure of acoustic forms emerging from *f(x)* tend to be cyclic, not in the sense of a cyclic musical form from the history of Western classical tradition, but has to do with the cyclic configuration of the borders of the automata world which causes the cell neighborhoods to wrap around each dimension and thereby propagate the global behavior in an infinite loop over the edges. This forces the turbulence and complex waves propagating through the cell states and which are being affected by the performer into cyclic patterns of acoustic streams. These streams create complex dynamics of behavior which the performer can respond to and try to modify. In this sense the performance entails an activity more akin to surfing (albeit on a very small, but rather turbulent pond) than Xenakis' metaphor of a composer as a pilot. In contrast, the emergent multimodal forms in the variants of *mikro* are shaped largely by the spectral dynamics of the audio input and built-in probabilistic training functions. The autonomous improvising agents can be considered still rather crude in their complexity and significant improvements can be achieved by more detailed statistical analysis of the input stream, more efficiently implemented training modules and more careful consideration of the different response options by implementing interacting critic algorithms on multiple hierarchical levels, starting from micro level synthesis to large scale compositional structures. And, finally, *sparsematrix* explores rhythm loops as the building blocks of musical structure. The paradigm of software sequencers and audio trackers used in the production of electronic dance music is replaced by the SuperCollider pattern libraries, algorithmic structural filtering functions and live coding is preferred to graphical user interface based interaction. This encourages exploration of polyrhythmic structures, rapid changes in tempo and downbeat, introduction of syncopation, and probabilistic parameter control of large number of parallel rhythm cycles, again emphasizing the

exploratory nature of the performance rather than routine repetition of predetermined and precomposed phrases, sections and movements.

In the context of the main objective of this project - to develop a multimodal autonomous improvisation system that interacts in real time with human musicians - the research and development has produced a fully functional performance environment in which the three discussed performance systems have been combined. This enables the performer to combine different compositional strategies in a single performance and adapt the system to the performance context. For example, it is possible to perform *sparsematrix* and use the audio output as the input to *mikro* analysis and response system. In this kind of a setup, an autonomous agent based artificial improviser can accompany the live coder and create new avenues of interactive exploration of the audiovisual parameter space. A number of solutions to the problem of multilevel audiovisual parameter control on multiple time scales and levels of compositional hierarchy have been designed, implemented and deployed for 30 public performances and presentations throughout the duration of the project. A comprehesive list of all performances and presentations can be found in Appendix A.

This does not mean that the work is done. Complex software performance environments require significant amount of time and effort to implement and maintain, however cannot ever in any way be considered finished. This sets them decisively apart from the traditional model of compositions as finished immutable artifacts. Analogously to the general endeavor of knowledge creation, these performance environments constantly give rise to new problems to be solved, as novel concepts to be explored and implemented are created unremittingly by fellow artists as well as curious minds from other fields of research, all the while components that do not stand up to the process of constant re-evaluation by conjecture and refutation, trial and error are regularly phased out. At the same time, these complicated instruments are very much like any other musical instruments, for they too require a vigorous routine of practice for the performer to acquire a level of virtuosity.

A | Public performances and presentations

During the project, all the described performance systems were put to practice at a number of concerts and presented to audiences at festivals, conferences and symposia, both within the UK and internationally:

- 23 Jun 2010. Performance of f(x): zone 00, zone 01, zone 02. Access-space (Sheffield, UK, http://www.access-space.org/)
- 9 Sep 2010. Presentation and screening of f(x): zone 00, zone 01, zone 02 at SoundingOut 5 Conference (http://soundingout.bournemouth.ac.uk/), Bournemouth University (Bournemouth, UK, http://home.bournemouth.ac.uk/)
- 24 Sep 2010. Performance of f(x): zone 00, zone 01, zone 02 at Sounding Code: Supercollider Symposium 2010 (http://supercollider2010.de/), Ausland (Berlin, Germany, http://ausland-berlin.de/)
- 10 Nov 2010. On the edge: Imaginary worlds, PS2 University of Hull (Scarborough, UK, http://www2.hull.ac.uk/scarborough/)
  - ◇ Performance of mikro:strukt with Satoshi Shiraishi
  - ◇ Performance of f(x): zone 00, zone 01, zone 02
- 18 Nov 2010. Performance of f(x): zone 00, zone 01, zone 02 at Piksel Festival (http://www.piksel.no), Studio USF (Bergen, Norway, http://www.usf.no/)
- 3 Dec 2010. Screening of f(x): zone 01 at Pixxelpoint Festival (http://pixxelpoint.org), Galeria Metropolitana (Gorizia, Italy, http://www.mediainmotion.de/pixxelpoint2.html)
- 11 Feb 2011. Performance of f(x): zone 00, zone 01, zone 05 at Soundings Festival (http://www.facebook.com/event.php?eid=183135291711888), University of Edinburgh (Edinburgh, UK, http://www.ed.ac.uk/home)

- 16 Mar 2011. Performance of f(x): zone 00, zone 01, zone 03, zone 05 at Risk of Shock - Sonic Science (http://www.riskofshock.org/), Prifysgol Bangor University (Bangor, UK, http://www.bangor.ac.uk/)
- 25 Mar 2011. Performance of f(x): zone 00, zone 01, zone 03, zone 05 at Hear This Space (http://www.hearthisspace.com/), Fabrika Arts Centre (Leicester, UK, http://www.taoleicester.co.uk/)
- 28 May 2011. Performance of f(x): zone 00, zone 01, zone 02, zone 03, zone 05 at International Videofestival (http://www.videofestival.org/), Ruhr-Universität (Bochum, Germany, http://www.ruhr-uni-bochum.de/mz/index.htm)
- 31 May 2011. Performance of mikro:strukt with Satoshi Shiraishi at the 11th International Conference on New Interfaces for Musical Expression (NIME) 2011 (http://www.nime2011.org/), Chateau Neuf (Oslo, Norway, http://studentersamfundet.no/)
- 4 Jun 2011. Todaysart Sessions #2 (http://todaysart.nl), Korzo Theater (The Hague, The Netherlands, http://www.korzo.nl):
  - ◇ Performance of mikro:strukt with Satoshi Shiraishi
  - ◇ Performance of f(x): zone 01
- 8 Jun 2011. Performance of f(x): zone 01 at Sound, Sight Space and Play 2011 (http://www.sssp.org.uk/), De Montfort University (Leicester, UK, http://www.mti.dmu.ac.uk/)
- 23 Jun 2011. Performance of f(x): zone 04 at IPC Cluster Symposium, PS2 University of Hull (Scarborough, UK, http://www2.hull.ac.uk/scarborough/)
- 1 Aug 2011. 2 performances of mikro:strukt with Satoshi Shiraishi at the International Computer Music Conference (ICMC) 2011 (http://icmc2011.org.uk/), CeReNeM, University of Huddersfield (Huddersfield, UK, http://mhm.hud.ac.uk/cerenem/)
- 15 Sep 2011. Presentation of f(x) at the International Symposium on Electronic Art (ISEA) 2011 (http://isea2011.sabanciuniv.edu/), Sabanci Center (Istanbul, Turkey, http://isea2011.sabanciuniv.edu/location/sabanci-center-levent)
- 11 Nov 2011. Performance of f(x): zone 00, zone 01, zone 02, zone 04, zone 05 at ShinyToys Festival (http://shinytoys.eu/festival/), Bollwerk 107 (Moers, Germany, http://www.bollwerk107.de)

- 24 Nov 2011. Performance of sparsematrix at Sonic Odyssey: Imaginary Worlds 2, PS2 University of Hull (Scarborough, UK, http://www2.hull.ac.uk/scarborough/)
- 14 Jan 2012. Performance of f(x): zone 01, zone 04 at noise==noise (http://nnnnn.org.uk/doku.php?id=noise_noise_-_supercollider_electronics), nnnnn Studios (London, UK, http://nnnnn.org.uk/)
- 19 Mar 2012. Performance of sparsematrix at Phipps Hall (University of Huddersfield, UK, http://www.hud.ac.uk/)
- 15 Apr 2012. Live coding performance of with Yota Morimoto at the SuperCollider Symposium (http://www.sc2012.org.uk/live/code/), City Arts and Music Project (London, UK, http://thecamplondon.com/)
- 17 Apr 2012. Performance of mikro:skism with Satoshi Shiraishi at The SuperCollider Symposium (http://www.sc2012.org.uk/live/concert/), City University London (London, UK, http://www.city.ac.uk/)
- 27 Apr 2012. Performance of sparsematrix at the Leeds International Festival For Innovations in Music Production and Composition (http://postgraduate.lcm.ac.uk/), The Wardrobe Club (Leeds, UK, http://www.thewardrobe.co.uk/)
- 7 Sep 2012. Performance of sparsematrix at Live Interfaces: Performance, Art, Music; The HiFi Club (Leeds, UK, http://www.thehificlub.co.uk/)
- 12 Sep 2012. Performance of f(x) at the International Computer Music Conference (http://www.icmc2012.si/), Kino Šiška (Ljubljana, Slovenia, http://www.kinosiska.si/)
- 26 Oct 2012. Performance of f(x) and sparsematrix at Pluto Festival (http://www.pluto-festival.be/), Nijdrop (Opwijk, Belgium, http://www.nijdrop.be/)
- 5 Dec 2012. Performance of lambda with Satoshi Shiraishi and Andrea Young, PhD Project Final Concert, PS2 University of Hull (Scarborough, UK, http://www2.hull.ac.uk/scarborough/)

B | Resources included on digital media

B.1 Audiovisual recordings

f(x)

- fx/f(x)_zone_00.m4v (05:49)
- fx/f(x)_zone_01.mov (07:36)
- fx/f(x)_zone_02.m4v (03:42)
- fx/f(x)_zone_03.m4v (05:07)
- fx/f(x)_zone_04.m4v (04:35)
- fx/f(x)_zone_05.m4v (05:04)

mikro

- mikro/bocca-mikro.mov (08:25)
- mikro/mikro-geen.mov (28:25)
- mikro/mikro-skism.mov (12:46)
- mikro/bocca-mikro.mov (12:02)

sparsematrix

- sparsematrix/matrix3d.mov (09:49)
- sparsematrix/sparsematrix.mov (09:25)

B.2 Audio recording

lambda - final performance

- lambda/lambda.wav (34:42)

C | Online resources

C.1 Project websites

f(x): http://fx.tehis.net

lambda: http://lambda.tehis.net

mikro: http://mikro.tehis.net

sparsematrix: http://matrix.tehis.net

Gene Expression Synthesis: http://geen.tehis.net

C.2 Live recordings of performances

Performance of f(x) at Hear This Space, Fabrika Arts Centre, Leicester, UK on 25 March 2011 (excerpt):
http://www.vimeo.com/22597217

Performance of mikro:strukt with Satoshi Shiraishi at On The Edge, PS2 University of Hull, Scarborough, UK on 10 Nov 2010:
http://vimeo.com/18549321

Performance of mikro:strukt with Satoshi Shiraishi at the 11th International Conference on New Interfaces for Musical Expression, Oslo, Norway on 31 May 2011:
http://vimeo.com/27694202

Performance of mikro:strukt with Satoshi Shiraishi at the International Com-

puter Music Conference, Huddersfield, UK, August 4 2011:

http://www.youtube.com/watch?v=m3UNrYHZE7Q

Performance of f(x) at the ShinyToys Festival, Moers, Germany, November 11, 2011 (excerpts):

http://www.youtube.com/watch?v=KoFBtj5JBqc

http://www.youtube.com/watch?v=cFURvv0wxSk

Performance of mikro:skism at the SuperCollider Symposium, London, UK, April 17 2012:

https://vimeo.com/42186998

C.3 Online recordings

- f(x)
  - zone 00: https://vimeo.com/18763431
  - zone 01: https://vimeo.com/18767837
  - zone 02: https://vimeo.com/19193886
  - zone 03: https://vimeo.com/19414076
  - zone 04: https://vimeo.com/28905481
  - zone 05: https://vimeo.com/19476095
- mikro
  - mikro:strukt: https://vimeo.com/69230003
  - mikro:skism: https://vimeo.com/47340107
  - bocca/mikro: https://vimeo.com/75621701
- sparsematrix
  - sparsematrix 2D: https://vimeo.com/42571333
  - sparsematrix 3D: https://vimeo.com/76418183

C.4 Source code repositories

f(x) (SuperCollider, C, Objective-C, OpenGL):

SuperCollider code files: https://github.com/darkjazz/Fx3D/tree/master/Audio

SuperCollider class files: https://github.com/darkjazz/sc-lib

Objective-C project: https://github.com/darkjazz/Fx3D/tree/master/Visual

lambda (SuperCollider, C++, OpenGL):

SuperCollider code files:

https://github.com/darkjazz/lambda/tree/master/supercollider

SuperCollider class files: https://github.com/darkjazz/sc-lib

Cinder C++ project: https://github.com/darkjazz/lambda/tree/master/src

mikro (SuperCollider, Objective-C, C, C++, OpenGL):

SuperCollider code files: https://github.com/darkjazz/mikro/tree/master/audio

SuperCollider class files: https://github.com/darkjazz/sc-lib

C++ project: https://github.com/darkjazz/mikro/tree/master/visual

sparsematrix (SuperCollider, C++, OpenGL):

SuperCollider code files: https://github.com/darkjazz/sparsematrix/tree/master/supercollider

SuperCollider class files: https://github.com/darkjazz/sc-lib

Cinder C++ project: https://github.com/darkjazz/sparsematrix/tree/master/cinder

Gene expression synthesis library (SuperCollider):

https://github.com/darkjazz/gepdefs

This thesis (Latex, Bibtex):

https://bitbucket.org/darkjazz/phdthesis

D | Practice-based Research Degree Guidelines

D.1 Practice based Research Degree - PhD/MPhil by Composition

A student may register for a programme of research in which musical composition forms the basis of the work.
Creative output shall be accompanied by a written commentary, which supports the work in its relevant theoretical, critical and aesthetic context.
The research degree in musical composition, shall consist of:

1. A substantial folio of compositions demonstrating originality, experimentation, coherence and technical acumen when addressing issues of form, language and forces. The student will also show a clear and detailed knowledge of contemporary issues in the field.

2. The compositions shall normally be a total of 90 minutes duration for the degree of PhD and 45 minutes duration for the degree of MPhil. In the case of the PhD submission the folio must contain one piece of substantial proportions.

3. The written commentary shall address the compositional approaches used in each piece and shall demonstrate an understanding of the techniques used, the context of the piece, and appropriate aesthetic issues. In the case of PhD, the document should be a maximum of 25, 000 words and a maximum of 12,000 words for the MPhil.

4. A permanent record in the form of recordings shall accompany at least a third of the compositions.

Students will be encouraged to have their works reviewed through public performance at conferences and festivals.

Bibliography

[1] Theodore W. Adorno. *Philosophy of Modern Music*. Oxford University Press, 1947.

[2] Amy Alexander and Nick Collins. Live audiovisuals. In Collins and d'Escrivan [23], pages 126–139.

[3] Alo Allik. Trigger Feedback Networks: creating musical patterns with models of neural firing networks. Unpublished manuscript and code library, 2003.

[4] Alo Allik. Tehis: a cellular automata programming environment for computer music composition. Master's thesis, Institute of Sonology, Koninklijk Conservatorium, the Hague, the Netherlands, 2005.

[5] Willie Anku. Circles and time: A theory of structural organization of rhythm in African music. *Music Theory Online*, 6(1), 2000. http://www.mtosmt.org/issues/mto.00.6.1/mto.00.6.1.anku_essay.html.

[6] Jacques Attali. *Noise. The Political Economy of Music (Theory and history of Literature, Vol. 16)*. University of Minnesota Press, 1985.

[7] Bret Battey. On the presence of water. http://www.mti.dmu.ac.uk/~bbattey/Gallery/otpw.html, 1997.

[8] Bret Battey. Writing on the surface. http://www.mti.dmu.ac.uk/~bbattey/Gallery/wots.html, 2000.

[9] Bret Battey, Peter Bill, Neil Chowdhury, Cris Ewing, Chad Kirby, Michael O'Malley, Juan Pampin, and Richard Karpen. Terraform 1. Multimedia installation curated by Richard Karpen, Henry Art Gallery in Seattle (Washington, USA), http://www.henryart.org/exhibitions/exhibition/82, 2000.

[10] Paul Bertelson. Ventriloquism: A case of crossmodal perceptual grouping. *Advances in psychology*, 129, 1999.

[11] Peter Beyls. The musical universe of cellular automata. In *Proceedings of the 1989 International Computer Music Conference, (ICMC 1989)*. The International Computer Music Association, 1989.

[12] Peter Beyls. Cellular automata mapping procedures. In *Proceedings of the 2004 International Computer Music Conference, (ICMC 2004)*. The International Computer Music Association, 2004.

[13] John Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference*, 1994.

[14] John Biles. Interactive genjam: Integrating real-time performance with a genetic algorithm. In *Proceedings of the 1998 International Computer Music Conference*, 1998.

[15] Antonio Camurri and Pasqualino Ferrentino. Interactive environments for music and multimedia. *Multimedia Systems*, 7(1), 1999.

[16] Linda Candy. Practice based research: a guide. http://www.creativityandcognition.com/research/practice-based-research/. Accessed: 2013-06-30.

[17] Joel Chadabe. *Electric Sound*. Prentice Hall, 1997.

[18] Michel Chion. *Audio-Vision. Sound On Screen*. Columbia University Press, New York, 1994.

[19] Thomas Ciufo. *Computer-mediated improvisation*. PhD thesis, Brown University, 2004.

[20] Nick Collins. Generative music and laptop performance. *Contemporary Music Review*, 22(4), 2003.

[21] Nick Collins. *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. PhD thesis, Centre for Science and Music, Faculty of Music, University of Cambridge, 2006.

[22] Nick Collins. LL: Listening and learning in an interactive improvisation system. *Unpublished research report*, 2007.

[23] Nick Collins and Julio d'Escrivan, editors. *The Cambridge Companion to Electronic Music*. Cambridge University Press, 2007.

[24] Christoph Cox and Daniel Warner, editors. *Audio Culture: Readings in Modern Music*. Continuum, 2004.

[25] Francis Crick. *The Astonishing Hypothesis*. Touchstone, 1994.

[26] Palle Dahlstedt. *Evolutionary algorithms as creative tools for the contemporary composer*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2004.

[27] Richard Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, UK, 1976.

[28] Richard Dawkins. *The Blind Watchmaker*. Penguin Books, 1986.

[29] Giovanni De Poli and Paolo Prandoni. Sonological models for timbre characterization. *Journal of New Music Research*, 26, 1997.

[30] Daniel C. Dennett. *Darwin's Dangerous Idea*. Penguin Books, 1995.

[31] David Deutsch. *The Fabric of Reality*. Penguin Books, 1997.

[32] Marco Donnarumma. Xth sense: A study of muscle sounds for an experimental paradigm of musical performance. In *Proceedings of the International Computer Music Conference (ICMC), Huddersfield, 2011.*, 2011.

[33] C. Ferreira. Karva notation and k-expressions. from gep tutorials: A gepsoft web resource. http://www.gene-expression-programming.com/tutorial002.htm.

[34] Cândida Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13, 2001.

[35] Cândida Ferreira. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, volume 21 of *Studies in Computational Intelligence*. Springer, 2006.

[36] Dennis Gabor. Acoustical quanta and the theory of hearing. *Nature*, 159(4044):591–94, 1947.

[37] Ricardo A. Garcia. Automating the design of sound synthesis techniques using evolutionary methods, 2001.

[38] Ricardo A. Garcia. Growing sound synthesizers using evolutionary methods. In *In Proceedings ALMMA 2001: Artificial Life Models for Musical Applications Workshop*, 2001.

[39] Murray Gell-Mann. *The Quark and the Jaguar : Adventures in the Simple and Complex*. W. H. Freeman, September 1995.

[40] Michael A. Gerzon. Practical periphony: The reproduction of full-sphere sound. In *Audio Engineering Society Convention 65*, 1980.

[41] N. Griffith and P.M. Todd, editors. *Musical networks: Parallel distributed perception and performance.* MIT Press, 1998.

[42] Lennart Hallstrom. African drum rhythms - educational tools for djembes, bass drums, and bells. http://www.djembe.net/. Accessed: 2012-10-02.

[43] Paul Hegarty. *Noise/Music*. The Continuum International Publishing Group Inc, 2007.

[44] Alan Hevner, Salvatore March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28, 2004.

[45] Douglas Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid*. Basic Books, Inc, 1979.

[46] William Hsu. Two approaches for interaction management in timbre-aware improvisation systems. In *Proceedings of the 2008 International Computer Music Conference*, 2008.

[47] Takashi Ikegami. Evolvability of machines and tapes. *Artificial Life and Robotics*, 3, 1999.

[48] James Joyce. *Finnegan's Wake*. Faber and Faber, 1939.

[49] Teuvo Kohonen. *Self-Organising Maps*. Springer-Verlag, 1994.

[50] George E. Lewis. Too many notes: Computers, complexity and culture in Voyager. *Leonardo Music Journal*, 10, 2000.

[51] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, 2000.

[52] B. J. Maclennan. Continuous spatial automata. Technical report, Department of Computer Science, University of Tennessee, 1990.

[53] Cristyn Magnus. Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms. In *Proceedings of the 2004 International Computer Music Conference*, 2004.

[54] Max V. Mathews and John R. Pierce, editors. *Current Directions in Computer Music Research*. MIT Press, 1989.

[55] James McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4), 2002.

[56] Jon McCormack. Eden: An evolutionary sonic ecosystem. In *Advances in Artificial Life, 6th European Conference, ECAL 2001, Prague, Czech Republic, September 10-14, 2001, Proceedings*, pages 133–142, 2001.

[57] Harry McGurk and John MacDonald. Hearing lips and seeing voices. *Nature*, 264, 1976.

[58] Alex McLean. *Artist-Programmers and Programming Languages for the Arts*. PhD thesis, Goldsmiths, University of London, 2011.

[59] Dennis H. Miller. Seven animations. Microcinema International, DVD, 2006.

[60] Eduardo R. Miranda. Granular synthesis of sounds by means of a cellular automaton. *Leonardo*, 28(4), 1995.

[61] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[62] Dalibor Mitrovic, Matthias Zeppelzauer, and Christian Breiteneder. Features for content-based audio retrieval. *Advances in Computers*, 78, 2010.

[63] Manfred Mohr. Artist's statement. Bulletin of the Computer Arts Society, London, January 1973, 1973.

[64] Manfred Mohr. Dimensions. Generative Drawings. Galerie D+C Mueller-Roth, Stuttgart, Germany, March 4 - April 29, 1979, 1979. Exibition Catalog.

[65] Yota Morimoto. Hacking cellular automata: an approach to sound synthesis. In *The SuperCollider Symposium, Berlin, Germany, 2010*, 2010.

[66] Gary Lee Nelson. Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In *Proceedings of the Fourth Biennial Art & Technology Symposium*, volume 155, 1993.

[67] John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

[68] Netochka Nezvanova. The internet, a musical instrument in perpetual flux. *Computer Music Journal*, 24(3), 2000.

[69] Jestern Alberto Novello. Invisible to visible: the eeg as a tool for music control and creation. Master's thesis, Institute of Sonology, Koninklijk Conservatorium, the Hague, the Netherlands, 2012.

[70] Fredrik Olofsson. Work with Mark: Genetics. (a blog post.). http://www.fredrikolofsson.com/f0blog/?q=node/144. Accessed: 2012-08-27.

[71] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics*, 21(4), July 1987.

[72] Curtis Roads. *Microsound*. The MIT Press, 2004.

[73] David Rokeby. Adventures in phy-gital space. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2011.

[74] Robert Rowe. Machine listening and composing with Cypher. *Computer Music Journal*, 16, 1992.

[75] Robert Rowe. *Interactive Music Systems: Machine Listening and Composing*. MIT Press, Cambridge, MA, USA, 1993.

[76] Robert Rowe. *Machine Musicianship*. MIT Press, Cambridge, MA, USA, 2001.

[77] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2003.

[78] Keiichiro Shibuya and Takeshi Ikegami. Personal communication. Club Transmediale 2008, Berlin, Germany. 2008-01-28.

[79] Satoshi Shiraishi. A real-time timbre tracking model based on similarity. Master's thesis, Institute of Sonology, Koninklijk Conservatorium, the Hague, the Netherlands, 2006.

[80] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6), 12 December 1962.

[81] Denis Smalley. The listening imagination: Listening in the electroacoustic era. *Contemporary Music Review: Live Electronics*, 13, 1997.

[82] Denis Smalley. Space-form and the acousmatic image. *Organised Sound*, 12, 2007.

[83] Dan Stowell. Supercollider code written by Dan Stowell. http://www.mcld.co.uk/supercollider/. Accessed: 2012-08-27.

[84] Dan Stowell and Mark Plumbley. Adaptive whitening for improved real-time audio onset detection. In *Proceedings of the 2007 International Computer Music Conference*, 2007.

[85] Peter M. Todd and Gregory M. Werner. Frankensteinian methods for evolutionary music composition. In Griffith and Todd [41].

[86] Godfried Toussaint. Classification and phylogenetic analysis of african ternary rhythm timelines, 2003.

[87] Godfried T. Toussaint. The geometry of musical rhythm. In Jin Akiyama, Mikio Kano, and Xuehou Tan, editors, *Lecture Notes in Computer Science*, volume 3742, pages 198–212. Springer, 2004.

[88] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59, 1950.

[89] Ge Wang. Calories: 3d cellular automata simulation + visualization for music. https://ccrma.stanford.edu/~ge/software/calories/. Accessed: 2012-09 -10.

[90] Eric W. Weisstein. Game of life. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GameofLife.html. Accessed: 2013-08-29.

[91] Scott Wilson, David Cottle, and Nick Collins, editors. *The SuperCollider Book*. MIT Press, Cambridge, MA, USA, 2011.

[92] Trevor Wishart. *On Sonic Art*. Harwood Academic Publishers, 1998.

[93] Mirek Wojtowicz. Mirek's cellebration - 1-d and 2-d cellular automata viewer, explorer and editor. http://www.mirekw.com/ca/index.html. Accessed: 2012-10-15.

[94] Stephen Wolfram. *A new kind of science*. Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.

[95] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1999.

[96] Iannis Xenakis. *Formalized Music*. Indiana University Press, 1972.

[97] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3), 1965.