Synthesised Sound & Synthesised Composition

COMPOSING ELECTRONIC MUSIC WITH COMPUTER ASSISTED COMPOSITION

Master's thesis in music theory: music technology

Magnus Bugge

Norwegian Academy of Music — Spring 2014

Abstract

This is a master's thesis concentrated around the construction of a suite of digital signal processing routines programmed in Max, with the purpose of composing, or aiding composition of electronic music. The routines are synthesis models operated by arbitrary generated numbers. Sounds produced are used in composition of acousmatic (fixed media) electronic pieces. Further, the processes is evaluated to see if, how and why the composition of electronic music can benefit from being crossed with computer assisted composition. The project is set in an academic context using artistic research as a method to overview, document, and reflect over the project.

Keywords: *Algorithm*, *artistic research*, *computer assisted composition*, *digital audio*, *DSP*, *generative music*, *Max*, *random numbers*, *visual programming*.

Acknowledgements

For contributions in various ways, I would like to express my gratitude towards my supervisor prof. emer. Tore Simonsen, the librarians at the Norwegian Academy of Music, my friends in SkRR, Andrew Hill, Lars Erik Sparby, and everyone at Notam—Norwegian center for technology in music and the arts.

Contents

Part I
Introduction

Can machines compose music? This is the classic opening line in texts about computer assisted composition. The answer to this should be obvious by now. Yes, they can. It is far more interesting to ask *how* we can use machines to compose music. In this thesis I document and reflect upon my own approach to CAC (computer assisted composition), to use visual programming as a tool for composing electronic music. The thesis is one out of three parts that together constitutes the master's. The two other parts is a collection of programming examples and a set of sound files, which can be found on the websites github.com and bandcamp.com.[1] Both the programming examples and the sound files are also presented on the DVD accompanying the paper version of the thesis.

Programming examples downloads here: `http://github.com/magnusbugge/SSSC/archive/master.zip`[2]
Sound files streams or downloads here: `http://magnusbugge.bandcamp.com/album/sssc`

[1]The source code is protected under GNU GENERAL PUBLIC LICENSE, meaning that it can be downloaded, modified, and shared, as long as derived works uses the same license. The sound files are considered the author's compositions and are licensed under a standard TONO/NCB-contract.
[2]To visit the Github repository, use this link: `http://github.com/magnusbugge/SSSC`

1 About this thesis

1.1 Research questions

As a point of departure for the thesis, there are specified two, a main and a secondary, research questions. I use the term point of 'departure due' to the explorative nature of the project. From beginning to end I was always looking for unexpected and possibly interesting observations. The main research question is also of an open character, not limiting itself to any specific ways of answering it. I believe that while working with an attitude of this kind is a potentially weak practice in many scientific inquires, it is a natural way to explore a topic as an artist using practice-based, artistic research. The research questions are:

- Main question: *How may computer assisted composition assist or stimulate the composition of electronic music?*
- Secondary question: *'Where' in computer assisted composition is the artwork? Is it in the code (computer program), in the music composed, or 'somewhere else'?*

The way the main question is asked, a natural approach to finding an answer is to try it—to compose electronic music using computer assisted composition. As for the secondary question, the idea is that the answer (although this question's answer is a matter of definition) to this should be enlightened during the work with the main question.

My approach in researching this subject is identifying—or establishing—a definition and understanding of the subject based on observation and reflection of selected parts of the available discourse,[3] followed by my own artistic take on the subject—including a thoroughly and practice based documentation in the thesis, and, at last a reflective part considering my own and others findings.

1.2 Author's background

It should be mention that I am not a composer in the typical definition of the word. I do not have a degree in composition, although I have taken composition courses

[3]There are, however, no scientific discourse theory/analysis applied. The discourse term is here used as a unifying word for all the literature, music, computer code, experiments, and general culture built up around CAC from the 1950s up until now, but also the history of algorithmic and procedural composition that goes hundreds of years back in time.

as part of my education. My interest for computer assisted composition stems from a desire to work with digital audio, music theory, and computers, more than it stems from a wish to 'compose' music (whatever 'compose' means).

1.3 Purpose of thesis

For the master's student there are several sides of writing and working with a master's thesis. It is the master's itself—the product, and it is the process—the experience, the learning. The thesis is also useful to me on a personal and professional level, giving me valuable training in working with music, sound, programming, text, and research. A master's is of course education, with the purpose of reaching a high level of expertise within a field.

Also, with this master's thesis I hope to be able to convey an understanding of what CAC is, and what it may be: to demystify the use of CAC. CAC often brings up the debate about 'heart' and 'soul' in music, which are seemingly used as metaphors for interpretable human quality. David Cope's colleagues nicknamed him 'The Tin Man' after the *Wizard of Oz* character, since they both supposedly had no heart [Bli10]. This paper tries to show that CAC simply represents another approach to making music. Creativity, hard manual labor, and genuine creation is still required to be able to make high quality music, no matter how computers are involved.

1.4 Theory

There is a relatively large amount of articles and books written on CAC. Much of the available material is practice based research written by composers that have experimented with CAC. There are also a number of musicologists who have written reflections and reviews on the subject, both in general and for specific works. I have used a selection of references to my aid in building up a useful background section for the thesis. I believe that by presenting and discussing the works that I do, most readers should be able to achieve a sufficient base of knowledge of CAC, and be able to set the rest of the thesis in context, including potential readers with little or no knowledge of the subject.

1.5 Research tools

The project is a practice based research project, meaning there is little or no inquiry or data collecting. The main tool for deriving information is working with the Max environment, the generation of audio based upon this, and my own observations and reflections during and afterwards programming.

1.6 Methodology

The methodological approach in the project is in the field of *artistic research*. A procedural course of action that I have called *systematic experimental programming* is used to construct composition programs, which are used for composing and evaluated.

1.7 Thesis structure

The thesis outline is of course printed in the table of contents in the preamble, but I'll include a short paragraph here on why it is organised as it is. There are, excluding the appendix, six chapters in the text. These chapters are paired two by two (illustrated in the table below), based on three major processes in the project: identification, where the subject is defined, explored, and evaluated; execution, which features my take on the subject, and, as an artist trying to create something of my own, while still keeping a professional distance to my work; and contemplation, the reflection and observation done around both my own work and the concept itself.

Identification		Execution		Contemplation	
Introduction	**History**	**Method**	**Programming**	**Reflection**	**Conclusion**

When describing Max programming, Max-objects are written in a monospaced font, like this: `object`, and MSP-objects the same way, but with a tilde sign after (like they are written in Max), like this: `object~`. Dates are written in the ISO 8601 date format YYYY-MM-DD. For the pdf using reader, urls, all referring to figures; chapters; and references, table of contents, footnotes, etc are clickable.

2 Presentation of subject

I define CAC as programming (instructing) a computer to execute composition related tasks, primarily in order to generate arbitrary[4] musical material. There are not necessarily an obvious definition of what is to be considered a composition related task, but in general, the use of conventional notation software and DAWs falls outside the field of CAC—there are in most cases some sort of programming language or programming environment involved, of which may or may not be specialised for working with music in terms of MIDI, digital audio, or other forms of inner/inter-application music communication.

CAC commonly relies on receiving some sort of data and processing this data into use for musical purposes. The data used may be generated internally—with an algorithm, or imported, converted, and scaled from some other digital or analogue source (like musical notes or sound, but also various sensor data, weather and space, stocks, etc). The mapping concept is a crucially important feature of CAC. The human decision of which data shall be mapped to which parameter, how the data is to be used, has a potentially much higher influence on the musical output than the data used. Though the data may appear to be music, sensor readings, or whatever, it is of course always numbers, or something else disguised as numbers. Seen from the computer, our interpretation of the data is just a metaphor for numerical information. I found keeping this thought in my head while working with the project useful—it helped me understand the computer, and, in a strange way helped me understand why the computer sometimes did not understand me.

2.1 Relevant terms

It is useful to make some general reflections around a few central terms before starting on the main body of the text. Four important words are 'computer', 'assistance', 'algorithm' and 'composition'. These words are interesting separately and in context of each other.

[4] Arbitrary as in random, but within a predictable range.

2.1.1 Computer

What is actually a *computer*, and what was a computer in the 50s? Though the difference in technology and power in todays computers and the ones of the 50s are of huge contrast, the definition of the device itself remains reasonable. According to the Oxford online dictionaries, a computer is:

> an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals. [Oxf13]

The ability to receive and interpret data is a crucial feature, but it should be added that the computer is programmable, which I define as that the interpretation of data is modifiable by the user.

2.1.2 Assistance

The word *assistance* is well defined by the same source: 'the action of helping someone by sharing work' [Oxf13]. In the context of CAC, this definition helps to understand how the process is actually conducted. The computer will not be able to take the place of the composer, it is an assistant who will share the most difficult and ponderous tasks.

2.1.3 Algorithm

Much of the works within the category of CAC is also CAAC—computer assisted algorithmic composition.[5] This introduces the word *algorithm*:

> The term algorithm is associated with the Greek arithms (number) and the Arabic term algorism (number series). Algorithm is usually defined as a set of rules for solving a particular problem in a finite number of steps [...]. Typically, algorithmic composers employ computers to realize these finite steps. [Cop97, 192]

[5] Whenever I use the term CAC, I do not exclude CAAC.

This gives us a reasonable, but not satisfying definition. For example, the phenomenon of chaos-algorithms are well known. Such algorithms predicts the movement of an object constantly dividing itself (like split atoms). The algorithm does not necessarily 'solve any problems', but it is considered an algorithm nevertheless (note that chaos-algorithms are useful in CAC, to rapidly generate masses of musical material). In the book *Electronic Music and Sound Design* an algorithm is defined as

> a sequence of instructions, written in a programming language, that enables a computer to carry out a defined task. [CG09, 3]

This definition is closer to what I previously have associated with the term, and is also a good general description of the programming examples following this thesis, but it is problematic, since it seemingly proposes that algorithms only exists within computers. The mentioned book is however written as sort of a workbook for Max users, and the definition should be understood within the context of computers science.

2.1.4 Composition

The word *composition* stems from the Latin term *compenere* meaning 'put together' [Oxf13]. In musical communities the term is more problematic. Composing, being something that composers do, creating music, is, in my experience something that many musicians have a very specific definition of. For many people, very little music making is actual composing (like song writing). In this text I will use the term simply as a word for writing and preparing some sort of music that is possible to perform later, but prepared with such a high level of detail that it can not be viewed as improvisation.

2.1.5 Combined: computer assisted algorithmic composition

If we combine these terms with each other, we get other terms like computer assisted composition, algorithmic composition and computer assisted algorithmic composition. These terms will often be understood as the same, but may also be viewed as separate entities (like non-computer algorithmic composition and

non-algorithmic computer assisted composition). One interesting example of non-computer algorithmic composition (or procedural composition) is the musical dice games of W. A. Mozart, were the user rolls dices to select precomposed bars of notes that are combined into a piece of music. One of the first CAC-experiments was an implantation of this procedure, which is clearly an algorithmic approach to composition. The mentioned computer implantation is further described in section 3.2.4.

2.1.6 Music technology

The term *music technology*, which is the name of my curricular specialisation, should also be examined, as it helps defining what one should expect from this paper. Especially the last word *technology* is often used in unclear contexts. Music technology, I find, is typically used by faculties and individuals alike as an umbrella term including everything in amplified stage sound, recording studios and other types of studios, live electronics, synthesis and signal processing, and musical activities using computers with sound or music software.[6] Based on this it would seem as music technology is something that is musical and needs to be connected to a power outlet, which seems strange—is not also the delicate mechanics of the hammers in a piano an example of amazing technology? Turning again to the Oxford online dictionaries, the definition of technology is:

> The application of scientific knowledge for practical purposes, especially in industry, (...)[etymology:] early 17th century: from Greek tekhnologia 'systematic treatment', from tekhn 'art, craft' + -logia [Oxf13]

This definition provides little help, illustrating the problem I associate with the term—it covers too much. The functionality of the mentioned piano hammers is also derived from experiments, therefore scientific knowledge for practical purposes. Etymology and definitions aside, the word 'technology' has an unspoken quality to us. It brings association of complex engineering, applied mathematics, and computer science. Also in terms of music technology the term has a certain

[6]This, plus film music, is basically everything covered by the 'music technology' optional subjects currently available at the Norwegian Academy of Music (`http://nmh.no/studenter/studiene/studiehandboker/startkull_2014/emner/valgemner/musikkteknologi`) [2014-04-15].

ring to it, that makes us think of the likes of Iannis Xenakis, Max Mathews, Karlheinz Stockhausen, and Don Buchla.

2.2 Computer music and software

The term *computer music*, which presumably could be used to describe the music derived from this project, does not really say anything about the music. Peter Hoffman exemplifies this with:

> Are the novels and poetry of contemporary authors 'computer novels' and 'computer poetry' only because virtually every author today prepares them with the help of a word processor? (...) Score-writing software coupled with electronic MIDI-instruments has greatly increased the output of many composers—just like word-processing software has enormously enhanced the human capacity for creating bureaucratic nonsense and boring novels. [Hof09, 23]

This quote also illustrates the difference from 'conventional' use of computers for making music, and using them for CAC. The computer should take part in the act of composing itself, but the computer will usually not replace the composer, it is foremost an assistant, useful for increasing productivity and organisation, and can possibly speed up certain parts of the process of composing some types music for some types of composers.

If the process are to fall within the category of computer music, the computer have to play a significant role in the creation of the music itself, not simply acting like a replacement for a pencil, tape recorder, mixing console, synthesiser or anything else that used to be separate objects, but now is commonly found as computer software. There are two very normal types of software that by this definition is not included within the subject of CAC. The first one is score-writing software, of which there are mainly two major commercial programs; Finale and Sibelius, the second is DAWs (Digital Audio Workstation—which is software combining recorder, sequencer, mixer, virtual instrument and effects, etc), of which there are a lot of popular programs, including ProTools, Cubase, Logic, Ableton Live, and many others. Although it is possible to use all these programs for simple CAC (for example with MIDI-plugins and functions for shuffling notes in various ways),

they are not considered very useful for actual 'musical computing', which is something I define in two main sections: DSP and CAC.

2.2.1 DSP, CAC, and environments

DSP (Digital Signal Processing) refers to the use of digital processing (e.g. something with a microprocessor, like a personal computer or a specialised digital audio unit) for executing synthesis or processing of recorded or incoming audio signals. Of course, audio as 'sound' does not really exist in a digital environment, the term refers to either digital recordings at its respective sample and bit rate, or digital realisation (synthesis) of specific waveforms and their features. Reaktor, Max, Csound, and SuperCollider are typical environments for this purpose.

CAC differs from this in that it is often associated with the process of generating the formal structures in music. Computer assisted generation of tone rows, chord progressions, forms, and such falls into this category. For this purpose, OpenMusic, PatchWork, JMSL, and MUSICOMP are typical environments.

The programs mention above are so-called 'programming environments' (which some would say is not really programming) that do not require any special programming knowledge to work with (perhaps with the exception of Csound). Of course, there is also the possibility to work musically with CAC or DSP using general purpose programming languages like C, Python, C++, Basic, Lisp, etc, which many notable CAC composers have done.

2.2.2 Synthesised sound and synthesised composition

There are of course also possible to use several of the DSP-environments for CAC and visa vi. OpenMusic for instance, has many excellent tools for processing audio[7] (especially those described by Jean Bresson [Bre06]), and Max can be used as a powerful system for generating and controlling MIDI.

CAC and DSP may also be combined into systems where both the composition itself and the sounds are generated by a computer. Yannis Xenakis's system GENDY (section 4.1) is an example of this, using stochastic principles for calculating the synthesis parameters of each sound. Also Tristan Murail have used re-

[7] Much more than only spectral analysis and partial tracking which seems to be a common association to OpenMusic.

lated approaches, mainly since the additive synthesis sounds in compositions like *Désintégrations* (1983) had several hundred parameters that were simply too laborious to adjust manually [Mur84]. Some of the first systems for CAC included sound generative subsystems (like the systems of Olson and Belar [OB61] and Caplin and Prinz [Ari11, 43]), foremost simple synthesis systems where generated melodies were sent for instant playback and review.

2.3 The problem

In texts written about CAC, there is a reoccurring statement that basically says it is difficult to have computers compose music of as good quality as a skilled human composer. One of these texts is the paper *Six Techniques for Algorithmic Music Composition* by Peter Langston, in which he in the introduction writes that

> (...) software engineers find formidable challenges in areas such as music composition; simulation of this complex human activity requires expertise in algorithm design, expert systems, optimization, and other related software engineering disciplines. Designing an algorithm to compose music, unlike designing an algorithm to invert a matrix or solve the traveling salesman problem, has no simple, mechanical test for success; if such a test existed, the computer analogy to the infinite number of monkeys and infinite number of typewriters trying to write Shakespeare could be tried (...). [Lan89, 1]

This and similar statements ([Mur05], [Cop00], among others) are well known challenges in CAC. Seemingly there is too much to ask of a computer to be able to see the process of composing complete works as a wholeness, and be able to see connections between all elements that together is a composed work. Norwegian composer Lasse Thoresen once answered the following when asked to define what composition is:

> To make good music is like solving a seven dimensional crossword where all elements has to match in every direction: harmonic, melodic, rhythmic, form wise, timbre wise, expression wise, spiritual: all parts must enlighten the others without becoming obsolete; everything must be justified in countless ways (authour's translation).[8] [VG04]

[8]Untranslated: Å lage god musikk blir som å løse et syvdimensjonalt kryssord der de samme

Is composition to much to ask of the computer? Are we waisting our time trying to make them do things we could do better ourselves? It really depends on what we want to see composition as. Adjusting synthesis parameters can be composition, just as much as composing melodies and harmonies. During this project I found myself viewing programming and composition as not separate tasks, but the same thing—programming as the composition itself. By viewing programming as composition, letting the algorithm not only make the music, but *be* the music, the perspective of what one can expect from it, and what it is possible to create, is more in contact with reality than if we want to 'hit the button' to compose the next great symphony.

elementene skal stemme på kryss og tvers alle vegne: harmonisk, melodisk, rytmisk, formmessig, klanglig, uttrykksmessig, åndelig: Alt skal belyse hverandre uten å være overflødig; alt skal være berettiget på utallige måter.

Part II
History of computer assisted composition

3 Early history of computer assisted composition

The field of computer assisted composition have, from the mid 50s to the present, presented a number of compositions of significant difference in musical style, technological approach, and artistic quality. These differences should make it clear that CAC is not really a musical style or composing technique, but rather an attitude towards the use of technology. CAC composers share little with each other but the wish and will to integrate computers in their writing. Even the motivation for doing so often varies greatly, from the programmer's approach of having the computer make music, to the pragmatic composer's approach of having the computer do tasks (s)he can not—or do not want to—do manually, wether being related to labour, practical reasons, or simply a way to execute tasks impossible to do without this kind of technological approach.

In this section I will briefly present some of the first attempts at CAC, which were realised almost as soon as computers became available, and also mention a few approaches done later, when computer technology had developed into a more practical work environment. A tendency that separates the older approaches

from some of the newer ones (especially the examples mention in this text) is that while the early attempts focus on reproducing musical writing of existing styles, the newer approaches is integrated in a contemporary musical discourse, utilising the technology to generate new forms of expression. This can possibly be seen in the context of most of the early experiments were executed in the field of computer science, by scientists, while the later approaches were executed by composers wanting to composing music.

3.1 General historical context

The immediate post war western music history is dominated by serial composers of the Darmstadt school, and the two technological approaches electronic music (which also soon appeared as serial music), and musique concrète. The nominal interpretation of these tendencies are that serial music is established as a reaction to fascistic politicising and romanticising of romantic music during the 30s and WW2, while electronic music and musique concrète is a natural development due to new available technology and research. All three directions should of course also been seen in context of the post romantic era liberalisation from tonality.

The post war years is also the start of the cold war, (often dated 1947-91), with the first successful detonation of a Soviet nuclear device in 1949, starting the cold war arms race. Generally, an arms race is, a race between two powers to have the best armed forces. In this case nuclear technology, the cold war symbol itself, was the most important type of weapons, but effectiveness and innovation in other kinds of military technology were also of great importance. This arms race had two interesting effects on non-military life:[9] First, technology developed/sold for military use, have later become of great importance for civilian use (like micro-processors, Internet, GPS). Second, while the two superpowers competed in having the most destructive arsenal of weapons, there were also a competition in having the more advanced technology in non-military science, leading to a lift in the overall development in research and technology.

[9]Of course the Cold War and the arms race generated much more complex synergies than these two, but these are the ones relevant in the context.

3.2 Pioneering projects

The history of computer assisted composition starts in the 1950s, which also is the decade of the emerging age of computer science, with the outfitting of universities and research facilities with computer systems to perform calculations, aid research, and other tasks. Within the field of CAC an interesting notion is that during a short period of the middle 50s, several experimental approaches to generating music with computers were initiated, seemingly unaware and uninspired of each other. It was only when Lejaren Hiller published a paper on the *Illiac Suite* (a string quartet named after the computer used to compose it, an *ILLIAC 1*) in 1961, four years after the research was completed, that he started to receive letters from other who also had conducted similar experiments [Ari11, 41] .

Although Hiller and Isaacson is considered responsible for the first true composition composed by a computer, Hiller is the first to mention that there were other successful experiments executed before the realisation of the *Illiac Suite*. There were also experiments done at the time that were conducted without the use of computers, but using very similar approaches. I would like to mention particularly two non-computer interesting innovations from the 50s that is of relevance to the subject.

3.2.1 Machine to Compose Music

The first one is Cohen and Sowa's *Machine to Compose Music* [Ass98], which was based on the educational toy 'GENIAC' (an electronic construction kit that may be used to aid calculation, but has no internal computing abilities) [Sow13]. The *Machine to Compose Music* was built specifically for the purpose of composing pentatonic melodies based on simple coin flipping like decision making (coin flipping being a random result from two potential outcomes; easily representable with 0 and 1 in a binary system). In figure 1 [Sow13] we see a simple table describing how random melodies are created from the coin flipping trial. Solid lines represents heads, dotted lines tails. Starting at the left top C, flipping a tails, the next note is another C. A flip leading to a 0 will result in a prolongation of the previous note (a quarter note becomes a half note). Note that the bottom number is a new 1., bringing the table up to the first level, ergo the processes may theoretically continue forever.

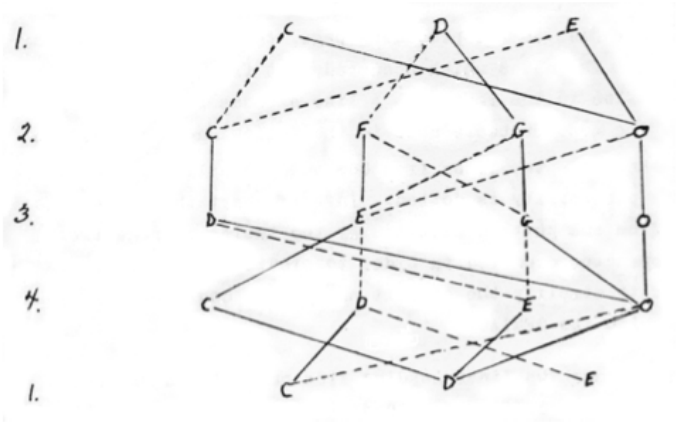

Figure 1: Diagram showing a coin flipping table from Sowa's *Machine to Compose Music*. Data flows downwards. [Sow13]

3.2.2 Combination Random-Probability System

The other non-computer innovation is the work Harry F. Olson and Herbert Belar. Olson will for some readers be known as a major inventor in the fields of music technology, sound, and recording, throughout the 20th century, responsible for several now industry standard microphone designs and the 1955 *RCA Electronic Music Synthesizer*, known as the first modern synthesiser. The *Combination Random-Probability System* was a machine built in the early 50s and patented in 1961 [OB61]. It basically consisted of two random number generators with weighted probabilities controlling pitch and rhythm, along with a sound generating system producing sawtooth waves [Cop91, 6]. Assayag argues that this device was not actually a computer, but 'a set of electronic circuits compromising a subsystem for sound generation and another for stochastic composition' [Ass98]. Previously we defined a computer as a device with the possibility of receiving and modifying information, with the user having the possibility of altering (through the act of programming) the way the information is processed. The *Combination Random-Probability System* has the ability to generate information and send it to a predetermined set of instructions. Since the instructions are not variable (they are defined in soldered, unmodifiable circuits), the device is not programmable, hence not a computer. It is still a device which has qualities interesting within this field of research, as the musical output is closely related to the CAC projects of the time, which I assume is Assayag's point of view as well, as he has included it in his text about computer

assisted composition. The device resembles a different approach for executing a similar task. Olson was an inventor and an electronics engineer, not a computer scientist, which makes an electronic machine a more natural platform than a computer.

3.2.3 Banal Tunemaker

Of the more truly computer based experiments, an early entry is a program by Richard C. Pinkerton who worked with the appliance of information theory in melodies. Based on this he designed, in 1956, a stochastic procedure called a *Banal Tunemaker*, which were fed with 39 existing folk melodies and nursery songs, that were separated into segments and combined into new pieces, using random numbers [Ass98] [You58, 24]. The available technical information on this experiment is sparse, which makes it unclear what the computer actually did, and what was done manually. This approach is related to the newer approach known as 'style emulation' described in section 4.2

3.2.4 Musikalisches Wúrfelspiel

Another early attempt is the one that involves Mozart's *Musikalisches Wúrfelspiel* (musical dice game). This is of course completely possible to execute by hand, but it is time consuming and repetitive, which is exactly the task computers are so suited for. In 1955, David Caplin and Dietrich Prinz did exactly this—they successfully created a computer-driven version of Mozart's dice game,[10] which is probebly the first use of a computer to compose music, as it was done before Pinkerton's experiment, and Olson's and Sowa's machines were not computers [Ass98] [Ari11, 42].

There are at least 20 different dice games attributed to Mozart, and this specific one was published in 1793 [Ari11, 44]. It is a composition tool, containing 176 pre-composed bars of music, that are systematically selected and combined by rolling two dices several times and placing bars corresponding to the dice's numbers after each other, resulting in 16 bar long compositions consisting of two homophonic lines to be played on a keyboard instrument [Ari11, 45]. In Chaplin

[10]Should the reader be interested in this phenomenon I recommend downloading Gary Lee Nelson's Max addaption of the concept. `http://cycling74.com/project/mozarts-dice-game/` [2014-05-05].

and Prinz's implementation the present hardware's computing power (a Ferranti Mark I) only allowed them to compose one line (they used the right hand voice), at an octave lower than specified, since the correct octave frequencies was to high to compute (reducing the notes by one octave reduces the frequencies by half, therefore reducing the computing power needed to calculate each note). There were also only possible to assign pitch and duration to each note, as the speaker system (integrated in the computer) could not interpret varying dynamics [Ari11, 42]. The dice's function was executed using the Ferranti's random number generator.

This experiment may seem banal from a modern view. It was based on a pre-computer algorithmic system, that even for being a dice-game wasn't especially advanced or even musically interesting. To actually fulfil such an experiment in 1955 though, requires dedication and insight. Ariza writes:

> While Caplin and Prinz could have generated endless random melodies, the implementation of Mozart's Musikalisches Wúrfelspiel offered a connection to historical practice and a suggestion of musical legitimacy.' [Ari11, 46]

The experiment had shown that actual music, not mere musical gibberish, was not only possible to compose with a computer, but *by* a computer. The extra effort of generating a procedure for sound realisation is of particular interest. Mozart's 'assignmet' is to produce a music sheet that can be played by a pianist—not making a computer automatically play it. This task could easily be done by printing the numbers as notes. It is a visionary idea of the time, to further develop the concept into a complete computer music procedure, with the random selection of bars, the conversion to frequencies, and the audible final realisation of the music.

It is interesting that already in this period, computer time was given to scientists that wanted to work with an 'unnecessary' discipline like music. Computing time was a rare commodity at the time, as the systems were slow, ponderous, extremely expensive, and calculations took hours or days (although this was fast compared the time that human labour would have used to execute similar calculations.) Ariza [Ari11, 44] writes that David Chaplin and Dietrich Prinz in their initial experiments at KSLA[11] used, in addition to their own programs, some of

[11]KSLA is located in Amsterdam and now called Shell Technology Centre Amsterdam [Ari11, 41].

the so-called 'visitor programs' that were used to demonstrate what computers could do for various people touring the facility. As Chaplins and Prinz's procedures actually would produce sound (melodies in the style of Mozart even) from the systems integrated speaker, these were also used to impress visitors by showing the vast possibilities within computer technology—while still maintaining a human connection through music, which most people can relate stronger to than logarithmic calculations.

3.2.5 Lejaren Hiller and the Illiac Suite

Despite the achievements of Chaplin and Prinz it is common to coin the first composition realised with a computer to Lejaren Hiller in collaboration with Leonard Isaacson, with the piece called *Illiac Suite for String Quartet*, composed during 1955-57 [Ari11, 40] named after the *ILLIAC 1* computer that were used at the University of Illinois [Hil63, 100]. This project was, as illustrated above, not the first entry in the history of CAC, but it was the most rigorous experiment with the highest degree of computer executed composing.

The suite consists of four movements based on separate programming routines, the three first focusing on different musical phenomenons, while the fourth uses statistical principles more than musical principles, more specifically Markov chains. A Markov chains is

> a probability system in which the likelihood of future event is determined by the state of one or more events in the immediate past. [Roa96, 878]

Obviously, this is an advantage in the composition of music, where one rarely would want completely randomised musical structures, but rather a sensible form and wholeness to the music. As a source for the fundamental musical material, Hiller applied a Monte-Carlo algorithm,[12] that allowed for creating large quantities of material with a probability of 'errors' (it is not clear how the errors were applied). This material, which were of course numbers, was mapped to basic musical

[12] 'Any method which solves a problem by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties.' (From: Wolfram Alpha 2014 [iOS application], Wolfram Group LLC `http://itunes.apple.com/en/app/wolframalpha/id334989259?mt=8`)

parameters as pitch and dynamics, but also to instrumental playing technique (like arco and pizzicato, this being a string quartet). The material was then run through a set of compositional rules (Fuxian similar rules as counter point, voice leading, etc) before being evaluated as 'valid' or not in Markov chain tables [Ass98]. Trimming down the massive amount of material by selection seems to be the main task for the Markov chains here.

Hiller's research project is of great importance in the history of CAC and computer music in general for several reason. Mainly, it is the first thorough inquiry in computer music which is also academically documented. It is also an original musical project that uses new ideas for generating material, unlike computer implementations and simulations of 18th century composition games. Assayag calls the project a

> (...) major breakthrough as it opened a new perspective for musical engineering, even if the interest of the artistic result itself may be discussed. It initiated the practice of algorithmic composition, which is still alive, especially in the United States. [Ass98]

It is also interesting due to the fact that the Monte-Carlo algorithm and Markov chains were such important components in the realisation of the project. In the time of serialism and computer technology as a new phenomenon, viewing music as information or data was a very contemporary observation by Hiller and Isaacson.

4 Four examples of newer approaches

Before moving on to the programming of my project I will present four different approaches to CAC, all from a newer time than the early projects mentioned above. There are hundreds of different projects, compositions, programs, or approaches that could have been mentioned as well, these four are chosen because they have a certain relevance to my research.

4.1 Stochastic sound & stochastic composition

Iannis Xenakis, which is probably the most known composer mentioned in this text (excluding Mozart of course), is the programmer and composer behind the very

interesting composition *GENDY3* from 1991. The idea behind *GENDY3* is, as in the adaption of the *Musikalisches Würfelspiel* by Chaplin and Prinz, an idea that stems from before the use of computer composition. Xenakis's *Metastasis*, from 1955, composed right in the very infantile years of CAC, also one of his most known compositions, is a stochastic work that uses the same principles that are utilised in the computer program used to compose *GENDY3*. Part of what makes this piece relevant is the complexity of the computer program and the fact that not

> (...) only is the musical structure of *GENDY3* stochastic, but the sound synthesis is also based on a stochastic algorithm that Xenakis invented and called 'dynamic stochastic synthesis.' [Ser93, 236]

GENDY3 is composed with program written by Xenkais in Basic, called GENDYN (GENeration, DYnamic), both the works *GENDY3* and *GENDY301* is created with this program [Ser93, 239] [Hof09, 9].

What the *dynamic stochastic synthesis model* may be viewed as is sort of an alternative oscillator. Instead of producing periodic, repeated waveforms, as those used in classic analogue synthesis (sine, triangle, sawtooth, and square), Xenakis defined an algorithm that calculates the amplitude of each separate sample of the waveform. When a waveform is 'completed' (finished one period), it is subjected to a stochastically calculated variation of itself. Figure 2 shows an example of how

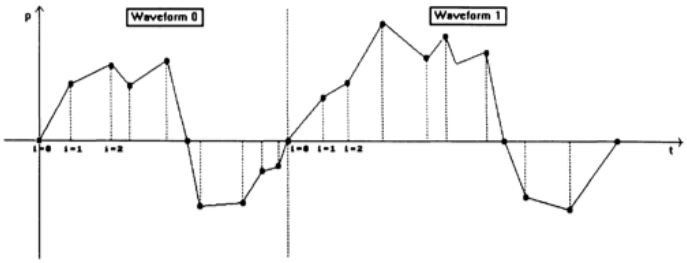

Figure 2: Examples of waveforms realised with dynamic stochastic synthesis. [Ser93, 241]

two waveforms may appear. Note that signals between the ten points per waveform are completely linear. This will result in some aliased waveforms:

> The sounds are usually very rich in harmonics and present a lively and dynamic quality that is noticeable. The polygonization of the wave-

> form introduces discontinuities into the numerical signal that produce high partials, some of which will be aliased by the digital-to-analogue conversion. Digital filtering can be applied in order to attenuate the aliasing, but then the signal may lose some variability that is valuable for the dynamic quality. [Ser93, 250]

GENDY3 also uses randomised calculation to construct the macrofrom. There is a total of 16 voices that play at different times. This decision is done with a 'Bernoulli trial', which is a random process with two possible outcomes, typically called success or failure [Ser93, 253] (like Sowa's coin flipping procedure). If a voice rolls a success it will start playing, and, using a formula, calculate how long it will be active.

Let us keep in mind that this piece and computer program was completed in 1991, after a long period in musical computing with reduced research around the field of CAC, which were of little interest and at the time had little practical use, and larger focus on DSP, which has a more understandable application and the possibility to enhance the quality and sound of much more music:

> After this pioneer period [the 50's and the 60's], CAC suffered from the considerable development of digital audio technologies. Massively attracting means and people, tempting by its immediate rendering of a new sound world, researches in digital sound synthesis and processing also gave a more scientific status to computer music and perhaps rang the bell for the likened 'composer-engineer' character who had been so often associated to former works. [Ass98]

It is then of no surprise, that around 1990, when also computer assisted spectral composition, as mention below, were at a peak, a piece which truly combines technology, artistry, and musicality of both CAC and DSP is realised. Xenakis,

> unlike many computer music composers, had no ambition whatsoever to emulate traditional musical thinking with the computer. Instead he realized his sonic vision in an abstract physical model of sound pressure dynamics yielding higher-order musical structures as emergent epiphenomena. [Hof09, 9]

4.2 Style emulation

Some composers, of there is one of that have become especially known, David Cope, have used the approach of *style emulation*. This idea demands the reduction of a musical style down to a set of rules that can be repeated over and over again for creating massive amounts of musical material.

> Cope produced interesting simulations in the style of Bach, Mozart, or Beethoven; but Cope is also a composer. In this case, the problem for him is to define the atoms of his own musical style. There is here the implicit assumption that musical creation consists of the recombination and working out of preexisting cognitive elements representing the non formalizable part, the absolute originality of a creator, in other words, the style. If this database-oriented approach makes sense for musicology, it seems marked by a too naive idealism as far as contemporary creation is concerned. It denies the idea of invention for which it substitutes that of combinative discovery of the musical 'self', and, by there, seems not very likely to reach real innovation. [Ass98]

The process of isolating the musical atoms is the crucial phase in this approach. Cope's emulations of Bach sounded significantly less dull after he implemented the chance to break rules of voice leading [Bli10]. Of course one can never manually program a computer to have a fantasy as rich as a humans, while still maintaining the possibility of creating an artwork with a sense of wholeness and continuity. One may however adapt CAC into the field of artificial intelligence, evolutionary algorithms, and machine learning. Emily Howell is a program by David Cope, where Cope has the ability to 'like' or 'dislike' the program's output to his preferences. This program has produced some interesting and very listenable musical pieces, like those released on the CD *Emily Howell: From Darkness, Light*.[13]

4.3 The pragmatic approach in spectral music

I would also like to, in this small survey of CAC, include a part on the mainly French style of music known as *spectral music*. Spectral music is a kind of music, or perhaps more an attitude towards the compositional process,[14] emerging pri-

[13] Cope, David. Centaur Records, 2010. Compact Disc.

[14] Are not most of the 20th century musical 'styles' just as much attitudes towards music as they are musical styles?

marily in France (hence the term *The French spectral school of music*) during the late 70s and 80s. A key feature is the use of Fourier transform analysis (or fast Fourier transform—FFT) of sound waves done with computers: a technique (among other things) used for revealing the harmonic spectra of a sound, which gives the composer a detailed understanding of the sound's partials: their frequency, amplitude, and phase, and the movements and interactions of these components. This information becomes the raw material of which the composition may be realised from on several levels, either directly, transposed in some way, or purely metaphorical. There are four composers that are usually mentioned in most discussions of spectral music, including Tristan Murail, Gérard Grisey, Hugues Dufourt, and Jean-Claude Risset.

The role of the computer in this type of music varies greatly from each composer and composition. One of the most known pieces from the movement, Murail's *Désintégrations* is a composition for a small orchestra and tape (synchronised with a click track for the conductor). The tape part of the piece is realised using additive synthesis.[15] The sound of the tape has, in addition to playing non-orchestra sounds, the role of amplifying, modulating or distorting the sound of the orchestra. What is interesting in this context is how the synthesis of the tape is realised. Murail is here discussing the use of a single sine wave generator to compose complete pieces of music in the initial phase of electronic music, recording different frequencies over and over again.

> This all became much easier with computers. Nevertheless, creating sounds with additive synthesis remains complex and difficult. For example, in *Désintégrations* to create an interesting sound it was often necessary to keep track of 10-30 components per sound, with 10-15 separate parameters for each component: pitch, dynamic, duration, time of attack, dynamic envelope, spatialization envelope, vibrato—with its different parameters (envelope, frequency, amplitude), spatialization, etc. There were often several hundred parameters for a single sound. [Mur05, 249]

[15] Additive synthesis is a synthesis technique that consists of combining (adding; additive) a large number of sine wave oscillators with individually set frequency, amplitude, and phase to achieve a desired sound. Since all sounds are combinations of sine waves, every imaginable sound is theoretically possible to achieve with additive synthesis. Frequencies are often based on partials withdrawn from spectrum analysis, or mathematically calculated to achieve a specific harmonic or inharmonic spectrum.

As we see here the primary agenda of using a computer is pragmatic. The precise settings of each synthesiser module is not what is important, but in order to 'excavate' sounds from it, adjusting individual parameters manually is simply to laborious, and randomisations by the computer is a more sensible approach. When the synthesis modules already are defined within the programming language, there is a small task to equip them with some sort of random number generators or any kind of control via mapping of other data. Murail continues:

> I needed to write a program that could calculate all of the necessary parameters as a function of global musical data. (For example, I needed to be able to specify to the computer that an oboe spectrum would be used, that the global duration would be x seconds, that the attacks would not be simultaneous, but rather staggered with acceleration effect), that the vibrato would have a certain frequency (speed) for the lowest component and another for the highest component, etc. The program then performed all of the necessary intermediate calculations, carried out any interpolations need, and supplied the list of parameters required for synthesis. [Mur05, 249-50]

This points at some interesting aspects of computer assisted composition. It's is clear to us here that the composer of the music is the human. However, the composer of the *sound* seems to be the computer. The synthesis programs built by Murail could probably create a magnitude of different sounds. As we know, for Murail, the sound itself is an extremely important component of the music, channeled through the idea of 'Posing sonic material, simply offering it to the listener's hearing' [Mur05, 174].

4.4 Autonomous systems

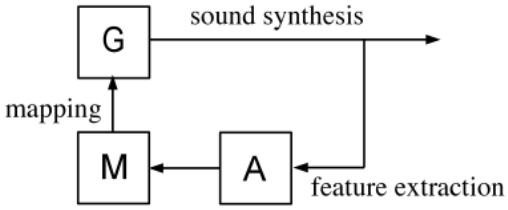

Figure 3: Basic feature-feedback system. [Hol12]

As a final mention in this brief overview, I will include a short mention of the concept of autonomous systems. Figure 3, from the research of Risto Holopainen, illustrates a basic system that executes realtime analysis on it's own output and, based on this analysis, changes constantly changes itself, resulting in a sort of *adaptive synthesis* [Hol12]. The ability for the software to unaided be able to interpret it's own output makes the computer active in composition with a new level of depth. This is related to, but still very different from, the previously mention work on evolutionary algorithms by Cope.

Part III
Method

The primary goal of this project is exploration of possibilities in composing music with systems based on visual programming for generating and controlling digital audio. This imposes that the ultimate goal is creation of music, but it is actually the system, the process of evolving the system, and the use of it, that is of highest importance—the music composed is considered a secondary goal and a desirable side effect. There is of course also the other primary goal of documenting the work thoroughly in this thesis, which contributes in making the project a research project. This section deals with the methods used in the project, primarily the *research method*, but also some mention of practical solutions in various parts of the project.

To research something means to systematically investigate it, in which the word *systematically* implies applying some sort of method to the investigation of the subject. The choice of research method for a master's thesis, or any research project of any size, is as crucial and as important as the choice of research questions. While the research questions deals with what one is researching, the method obviously deals with how one approaches the questions and subject.

5 Artistic research

The method used for this project is based on what is known as *artistic research* (AR). The use of AR allows us students and researchers within the fields of arts to explore, develop, and execute our artistic works in an academic presence. AR is not necessarily applied as science, and projects using AR is not necessarily a scientific inquiry. By this I mean that we (the artistic researchers) are not (re)searching with the goal of obtaining scientific proofs of something. The scientifically obtainable facts about arts are in many, but not all cases, not interesting, at least not as interesting as other aspects of art may be. There is not an obvious reason for turning an artistic project into an artistic research project. The project should be fitted for academia, and the artist should also want to be a researcher as much as an artist.

> I don't think an artist should (intentionally at least) become part of an institutional body purely for the reason of creating art. [Bro12]

The artist should, instead, 'become part of an institutional body' for working with artistic research—if (s)he has a desire to do so. Research and arts become interesting when the research turns into developing new forms of arts, of that which were not possible without the research. In music, such scientific achievements would be the all kinds of things like the microphone, amplifier, and speaker, the possibility to record and store sound, the hammerklaver, all kinds of synthesis, signal processing, i.e hundreds of innovations that changed how musicians works, but also musical works like *GENDY3* and *Désintégrations*.

An interesting conjunction of science and arts, or rather science applied with artistic material, is the research of José Antonio Bowen presented in the paper *Tempo, Duration and Flexibility: Technique and Analysis in Performance* [Bow96]. The work displays a rigorous work on measuring tempo and tempo changes in a large number of recordings of the same symphonies, with different conductors and orchestras. This is a good example of a scientific (statistical) method applied on artistic material, revealing interesting cultural and historical tendencies, since the plots displays changing of tempos throughout the 20th century, differences between American and European conductors, and difference in different recordings by the same conductors (like Bruno Walter going slower and slower for every event). This

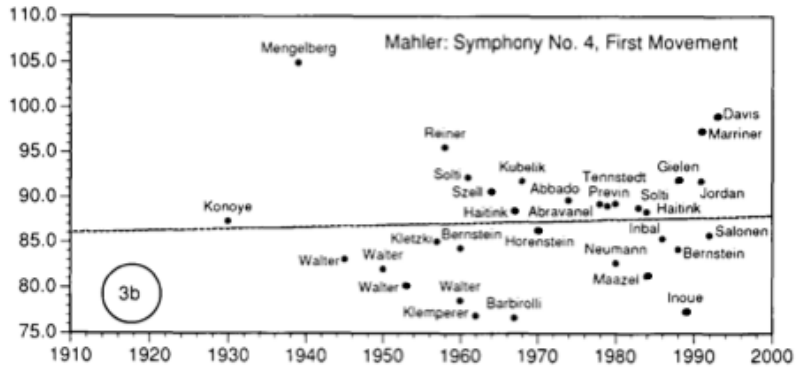

Figure 4: Scatter plot showing the initial tempo from different recordings of Mahler's 4th symphony, 1st movement. The trend line shows an average raise from 86 to 88 bpm throughout the 20th century. [Bow96]

project is, as Bowen writes introductory, an attempt to bring the work of musicology away from 'what remains the same' (the score), and over to 'the event' (the concert and/or recording), which is always different [Bow96]. I find this attitude towards musicology sensible, as it raises the ultimate musicological question of what music is. While traditional musicology has for years focused on the score as the actual 'work', and the performance as an interpretation of the score, the average listener will view a performance as the actual music, and if the listener does not know how to read music the score is useless. Both options are of course both right and wrong. A score is a stack of paper, which is not music, but without the score the work doesn't exist other than in recordings and/or memories. Today a piece of music (in this context a 'piece of music' refers to composed, printed music meant for acoustic performance; a classical composition) is several things: scores in different editions, various recordings, memories in peoples minds, plagiaries, etc, and all aspects of music should be considered legit, musicological, researchable subjects, from note structures to timbre, performance, reception, recording principles, development, history, industry, etc.

5.1 How artistic research was used

Much of the thesis is based on readings of existing literature, including both CAC and DSP specific writings and discussions of AR in academics. AR is, opposed to research in the terms of social and scientific research, based on a rigorous explo-

ration of a subject, and not the practice of testing of a hypothesis, though there is nothing wrong with using testings of hypothesises as part of the exploration. AR is also a method where the researcher enters both the role of a researcher and the artist, therefore becomes the research subject itself, or rather, the art of the artist becomes the subject. A crucially important feature is that AR also allows the subjectivity of the researcher to account for a valid result. This illustrates perhaps the difference between universities and other higher education institutions (especially art academies). In the words of Henk Borgdorrf:

> Research in higher professional education differs from that in university education in the degree which it is oriented to application, design and development. As a rule, 'pure' or fundamental scholarly or scientific research (if indeed that exists) is and remains the province of the universities. Research at theatre and dance schools, conservatories, art academics and other professional schools of art is therefore of a different nature to what generally takes place in the academic world of universities and research institutions. [Bor06]

This is probably different in different countries. This project in done at the Norwegian Academy of Music, which by name is an 'art academy', but still has the same duties and requirements within education and research as universities.

The authors of the book *Artistic Research* isolates several features and goals in common in most projects using artistic research [HSV05, 20-21]. One of the features is that the artwork is the 'focal point' and is the most important priority in the work. In this project this is different, the actual research (the development of the programs) is in this case the 'focal point', while the artwork serves the role of supporting the research—it's process, development, and final form are the research material.

The research part of this project can be broken down to defining a procedure (an algorithm), executing it, observing the results, and evaluate the result. Based on the evaluation the process starts over again and the procedure is changed accordingly, which leads to a new result and a new evaluation. This process theoretically never stops, but at some point it will be necessary to withdraw some musical information and do something with it.

6 Systematic experimental programming

I have called the approach used when programming the patches 'systematic experimental programming'. 'Systematic' as in a procedural approach, a planned line of steps, and 'experimental' as in that though I have an idea of what is going to happen, the course of action may change due to unforeseen events, making me change the idea. The initial idea may just as well be a desire to cultivate a certain musical event—and, based on this, try to execute it with programming, or the opposite, trying to use some certain objects in a certain way—and see what musical results is possible to achieve. The methodological process in this project has the following stepwise course:

1. Form an idea for a procedure
2. Define procedure in programming environment
3. Test
4. Evaluate
5. Redefine
6. Record material to DAW
7. Arrange material into musical composition
8. Evaluate process

Step 7 is obviously very open of character, providing no limitations or guidelines on how to use the DAW as a compositional tool. At this point in the process, the musical material is anyway taken out of the process associated with CAC, so there is, in the context of this text, not particularly interesting to discuss what happens in the DAW. It would be against the projects nature though, to either process the audio samples so much that it becomes sonically disconnected and unrecognisable to the patches that generates them, or to add extra audio to the compositions using samples, virtual instruments, or any other external audio source.

7 Software

7.1 Max

Max, also known as MaxMSP or Max/MSP/Jitter, is a graphical/visual object-oriented programming environment, and is mainly used for making custom DSP applications, both for studio, installations, and live use. Max is object oriented, meaning that the programming consists of combining ready defined objects into interacting with each other. Figure 5 illustrates a simple procedure made in Max

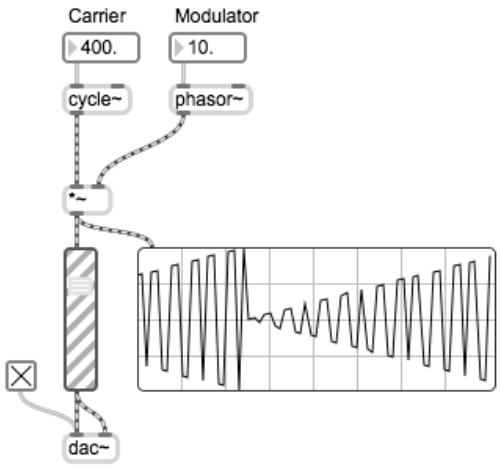

Figure 5: Simple amplitude modulation application made in Max 6. The `cycle~` (sine wave oscillator) operates as a carrier with a frequency of 400 Hz, while the `phasor~` (sawtooth wave oscillator) functions as a modulator at the frequency of 10 Hz. A `scope~` is used for visualising the modulated waveform.

for this introductory text. It features two floating number boxes, two oscillators, an audio signal multiplier, a gain slider, a scope, a toggle and a digital-analogue converter. All these a called objects, and are readily designed and included in Max (Max objects will from now on be written in an own font for easy distinguishing, like this: `object`). To make objects interact with each other, one connects their inlets and outlets to each other using virtual patch cords. Striped cords represent audio signals, monochrome cords represent data (numbers, lists, bangs,[16] etc).

[16]The bang message is the basic message in Max: 'it's the message that tells many objects to *do that thing you do*.' (Max Tutorial 2 `http://www.cycling74.com/docs/max5/tutorials/max-tut/basicchapter02.html` [2014-05-05])

What is happening in this patch is that the two number boxes at the top sets the frequency of the oscillators below. The `cycle~` (the tilde sign is used after the object's name if it is an audio object) is a sinusoidal oscillator, while the `phasor~` is a sawtooth oscillator. 'Audio' in this case is of course simply streams of numbers—when multiplied in the `*~` we will experience it as that the low frequency sawtooth will modulate the amplitude of the high frequency sine wave. This is visually illustrated in `scope~` object below to the right, which functions as a conventional oscilloscope. The striped fader is a `gain~` object, an exponential gain controller, and a quick solution for adjusting volume levels. Below is the `dac~`,[17] which is what actually sends sound out of the software and to the computer's audio interface. It needs to be switched on (audio objects will stay inactive until), for instance by using a `toggle` (the small box with an x—usually used as an on/off switch for some objects). Using a visual programming environment like this has the great advantage of creating a user interface automatically while programming. On will often clean up the look of the programming with hiding of objects, presentation mode or other methods (in this patch only the number boxes, toggle, fader and scope needs to be visible).

7.2 Other software tools

For recording the Max-generated audio, signals were routed out from Max with Soundflower (application-to-application routing software with up to 64 channels), and in to a DAW (digital audio workstation). The DAWs used were Logic Pro 9 and Reaper 4. Reaper had to be used to produce the 8-channel version of *Marise*, since Logic lacks the ability to do this.

The thesis written and typeset with LaTeX, using the editor Texpad 1.6 with the MacTex Latex distribution. References are kept and organised with a Bibtex-file built using Bookends 11. Pages 4 was used for designing some of the figures.

[17] Digital to analogue converter—of course it is not truly a dac, the dac is a hardware unit in the audio interface.

Part IV
Programming — composing

This section is written more as a report than the rest of the thesis, as it's purpose is to inform the reader of how the patches work and what kind of ideas they are based on. What kind of *ideals* they are based on is a subject for part V. Each section in this part focuses on one main patch and it's subpatches.

8 Matrise

Matrise (the Norwegian word for matrix) is a patch, and a piece, named after one of the objects used in the patch, the `matrix~` object. Basically *Matrise* uses six instances amplitude modulated oscillators, that are randomly sent to four unsynchronised pulse generators, which are connected to huge sounding reverbs. The patch, and the others in the project, is self-operating. Gradually as the patch runs, it will develop into new sonic structures that are more and more different from each time. The intitial idea was a very simple one: to define a simple multi-output synthesiser that could be routed to different processors using a `matrix~`.

This patch is the only one in the project that outputs on more than a stereo pair of channels. Though it uses eights channels, it makes more sense to view the output as four stereo pairs (as shown in figure 6). Among the recordings done in the patch there is included one 8-channel wav-file with a routing sheet for perfor-

mances, and a stereo mix for demonstration purposes.

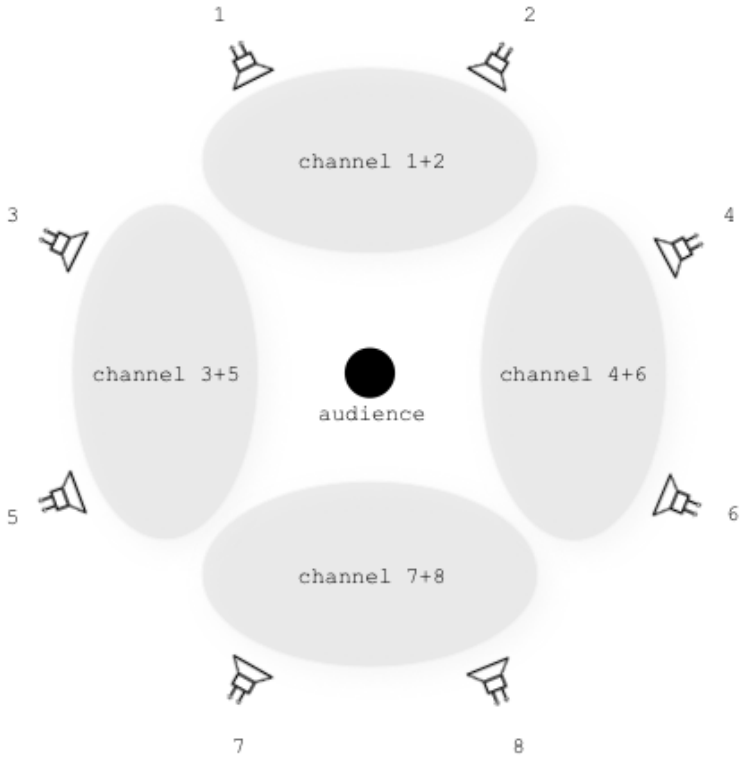

Figure 6: Speaker placement for performance of *Matrise*. Rather than typical and active surround sound (like using VBAP [Pul02]), the piece is installed as four static stereo pairs.

8.1 Programming Matrise

8.1.1 Synthesiser

The initial sound in *Matrise* is generated in the subpatch `matrisesynth`.[18] In this subpatch we find a set of oscillators and an algorithm for setting frequencies. There are 12 oscillators which is paired two by two. Each pair has a sine a wave oscillator, and an either square wave oscillator or a sawtooth wave oscillator.

The oscillators in each pair is connected to each other, using the left (saws and squares) and right (sines) inlets of a $\star\sim$, creating six instances of amplitude mod-

[18] *Matrise* opens in presentation mode, so the patch has to be switched over to patching mode to view subpatches, patchcords, and most of the objects.

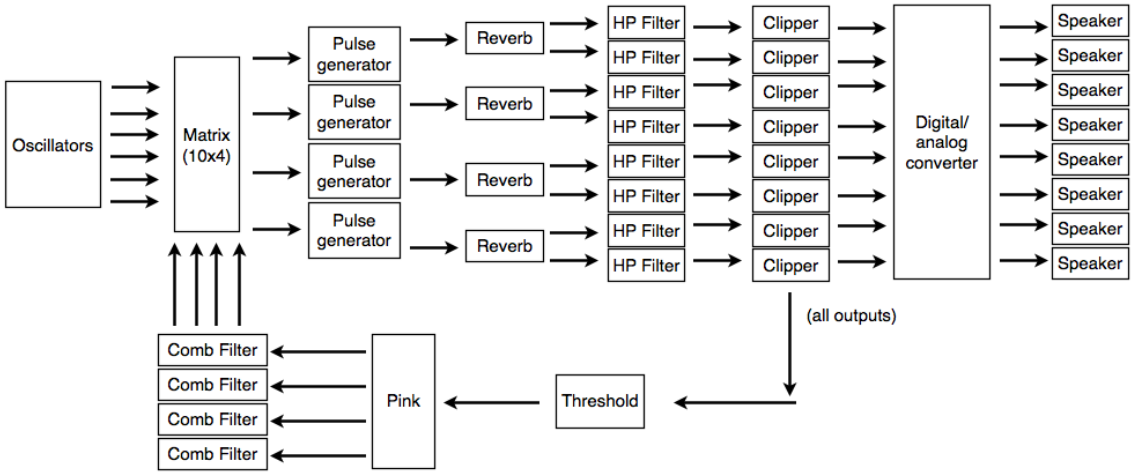

Figure 7: The signal path in the patch *Matrise*.

ulation. The oscillators will, when close to each other in frequency and sharing outlet, modulate each other and create a desirable beating effect.[19]

When the patch is banged from further down the signal path, there is a 1/20 chance that the patch will set a new main frequency. If not, the bang will produce either new LFO frequencies or new micro variations (detunes of the main frequency) and shuffle the oscillator's frequencies. There are at any time six oscillators using the main frequency (40-120 Hz), three oscillators using a micro variation (a few Hz below or above the main frequency), and three oscillators functioning as LFOs (up to 5 Hz). The `zl.scramble` shuffles the order of these numbers, randomising their destination oscillators. There is also trial with a 1/2 chance before each oscillator that decides if the oscillator is to change to the new frequency at all, adding the chance that at a new main frequency, several of the oscillators will not change, resulting in a duo-phonic output. The six amplitude modulation instances the leads to the six first inlets of the 10x4 `matrix~`.

This results in nice sounding, thick, detuned, and beating textures. A particularly nice effect is the difference of when a sine wave has a high frequency and the connected saw or square has a low frequency—resulting in the sine wave being amplitude modulated by a saw or square, contradictory to when the saw or square has a high frequency and the sine has a low frequency—resulting in a saw

[19]Beating occurs when two oscillators are close, but not identical in each other in frequency, creating a third frequency with low frequency amplitude modulating effect.

or square wave being amplitude modulated by a sine wave.

There is also a small noise generator (the patch `noisethresh`) using the remaining four inlets of the `matrix~`. This part is a very simple autonomous algorithm, using a signal measuring (after the high pass filter further down the signal path) to detect loud levels. Over a set threshold, the patch will output comb filtered pink noise into the audio stream, adding more character and colour during intense moments. The same pink noise generator (`pink~` object) is patched to four filters (`comb~` objects) with randomised parameters, and the four filters leads to the `matrix~`.

The 10x4 `matrix~` uses a simple algorithm based on `random` to create three elemented lists of numbers: 0-9 (horizontal), 0-3 (vertical) and 0-1 (on/off). Each time a list is received, the `matrixctrl` changes the active inlets and outlets of the `matrix~`.

8.1.2 Pulse generators and reverbs

After the routing of signals in the `matrix~`, ten audio signals have become four, and are at this point sent to the `trainjaff` subpatches.[20] The pulses (the tones, really) in each voice is made using four `train~`s with a very high pulse rate (15-25 seconds) and a variation in how much of the pulse is 'on' and how much is silent. To avoid clicks, the signals are routed through separate `rampsmooth~` objects using the system sample rate times two (typically 44100x2), meaning there will two seconds fade in and out for each pulse tone. This is an extreme way to use `rampsmooth~`, but it works well and provides a uniform envelope for the tones.

Below the pulse generator we find the reverbs, where I have simply used the `yafr`-patch.[21] Though the `vst~` object and an external plugin reverb probably could have worked better, I quickly became used to the sound of this reverb and decided on using it.

Since there were four outlets from the `matrix~` and the yafr-reverb has two each, we are now up in eight channels. After the `trainjaff`s, channels are individually filtered in the deepest frequencies to remove some unwanted rumbling.

[20] There are three `trainjaff`s and one `trainjafflogic`, of which the only difference is that the latter sends the 'trainbang' that controls change in most parameters.

[21] A patch included with Max. Plate emulation in 'the style Griesinger', a name some readers will associate with Lexicon.

This was done using onepole~ as a high pass filter with the cutoff set to 50 Hz, which is very gentle. onepole~ is not a resonant filter.

8.1.3 Sigmoid soft clipper

During development the patch would sometimes peak, especially at times where the train~ objects were using a high pulse width, or the matrix~ were outputting many of the inputs. Such effects are in one way desirable since *Matrise* is a piece focused around sound masses, process, and digital noise. However, they should be maintained under control, since they sometimes may ruin an otherwise good recording or performance. To avoid this I made a soft clipper for the patch, using gen~, with the purpose of smoothening out signals that would otherwise clip and distort. Since a soft clipper is purely a DSP-event, gen~ seemed like the proper environment for this. The soft clipper is based on a sigmoid function, which the WolframAlpha iOS application[22] presents like this:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

After some work this was how the gen~ adaption looked (figure 8) with the code output as follows (although gen~ uses visual programming, it automatically outputs the code of the generated object):

```
int_1 = int(1);
neg_2 = -in1;
pow_3 = pow(e, neg_2);
add_4 = int_1 + pow_3;
div_5 = int_1 / add_4;
sub_6 = div_5 - 0.5;
mul_7 = sub_6 * 2.;
out1 = mul_7;
```

Notice I had to withdraw 0.5 (sub_6) at the end to have the signal focus around 0, rather than 0.5, and then multiply by 2 (mul_7) to bring the signal up to a useful level. See the practical demonstration in the included patch sigmoidpresentation.maxpat to view the soft clipper in a predictable situation.

[22] Wolfram Alpha 2014, Wolfram Group LLC http://itunes.apple.com/en/app/wolframalpha/id334989259?mt=8

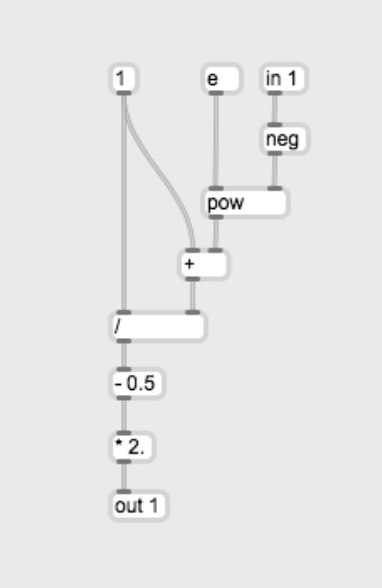

Figure 8: Sigmoid function in `gen~`.

8.2 Sound results

In total I recorded about four hours of 7-10 minute long 8-channel audio segments made with the patch, while tweaking settings of the patch in between. I consider this part of CAC, the listening and selecting of material, as important as programming.

In *Matrise* the patch is the artwork, with much resemblance to a score that, as I quoted Bowen on earlier [Bow96], remains the same, but the performance is always different (of course I may change the patch in the future, but so can a composer working with scores do). A recording derived from the patch, as the one the were performed and is attached to the thesis, will naturally stay the same. However, performing this recording again will result in a different performance, since in such minimal music, every detail becomes transparent. At first I pictured the piece to be performed at non-typical venues with long, and/or characteristic natural reverberations, that in traditional musical performance culture would be considered a very bad place to perform music, like parking garages, caves, subway stations, etc. A natural reverb of this kind would greatly colour the music, and change the music into something else.

Matrise was premiered as the 8-channel version, along with five other electronic and/or electroacoustic compositions, at the concert *STILLE LYD - "Quiet Sounds"*

2014-04-28 in PACE Studio One in Leicester, UK, curated by Andrew Hill (programme notes in section 22).

9 Matrise redux

I was pleased with the technical and musical results achieved with *Matrise*, but somehow it seemed as the potential of the basic principle—feeding different inputs to a matrix with different outputs—should be explored further, especially in terms of making something less massive sounding. This was the point of departure for making the patch *Matrise redux*, which is used for quickly making polyphonic variable amplitude drone textures.

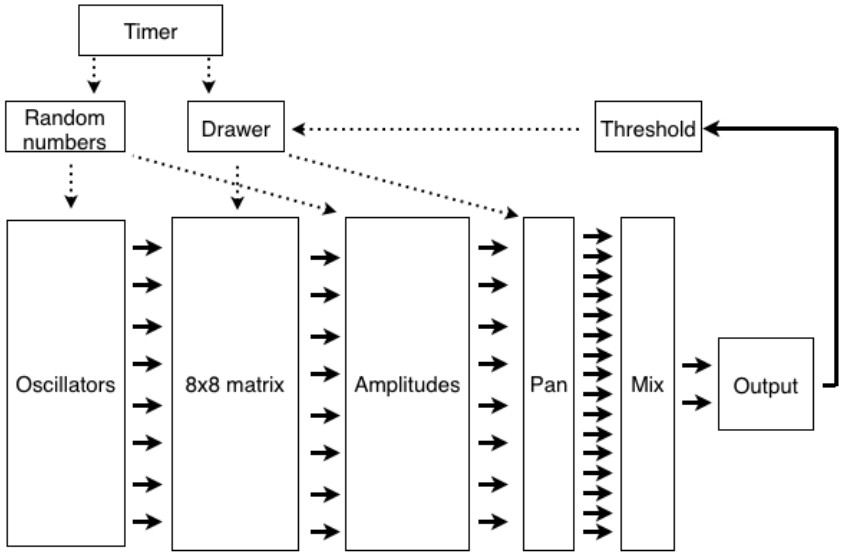

Figure 9: Signal path in *Matrise redux*.

9.1 Programming Matrise redux

9.1.1 Oscillator frequencies

Upon launch, eight oscillators (I've used the patch both with `tri~` [*Texture 1* and *2*] and `cycle~` [*Texture 3*], and simply changed the objects manually) are set to random frequencies: 19-319 Hz, with exception of the eighth oscillator, which will always have it's random number multiplied by five, meaning it's frequency will be

95-1595 Hz. This has off course no particular effect unless the multiplied number results in something notably higher than 319 Hz. If, however, the frequency is significantly higher, the eight oscillator will be experienced as something else—more special, since the tone will stand out with a significantly higher pitch—than the other seven. In other words, the eight oscillator will sometimes be special, sometimes not.

9.1.2 Variable amplitudes

The amplitude variables are set in a similar way as the oscillators. The `rand~` object is used for making the amplitudes variable in a randomised way. Of course oscillators could have been used for this task as well, but I wanted a more random sounding texture than the pulsating feel of an LFO. To use an analogy from analogue synthesisers, I wanted to use something sounding like sample hold for the VCAs. `rand~` generates a signal varying from -1. to 1., interpolating linearly from value to value, which occurs at a certain frequency set with a floating number. In other words, setting 1. as the frequency will make the `rand~` output a new number at every second. It will, however, slide linearly from number to number. Setting the frequency to 1000. will make it output a new number every millisecond, which will not allow us to actually hear the amplitude changing, only a buzz, really the same as pushing a low frequency oscillator up in frequency from 'signal range' to 'audio range', having it produce a tone, rather than a pulse.

9.1.3 The matrix

The patch uses an 8x8 matrix, in which the oscillators described above is connected to the inputs, and the outputs leads to eight `*~` objects which are connected to the outputs of the `rand~` objects described above. The matrix has a `matrixctrl` connected to it, with an algorithm assigned to switch on and of connection points.

In the encapsulated part of the patch (the `p` above the `matrixctrl`) is the 'drawing algorithm' located. It's purpose is to generate three elemented lists of number for the `matrixctrl`. This `matrixctrl`, takes list consisting of a numbers from 0-7 (horizontal), 0-7 (vertical), and 0-1 (on/off). For the first two elements, `random` objects generating 0-7 is used, but instead of using the numbers directly they 'draw' lines on the grid, using the `line` object: if the random rolls a

2 followed by a 5, the `line` generates the series 2-3-4-5 over a period of time set randomly (maximum five seconds). The same process is used for drawing both vertically and horizontally in the `matrixctrl`. The third number is always either 0 or 1 which decides if the drawer should activate or deactive the points. If the total output becomes too loud a message is sent that will set the drawer to deactivating points (0), before, after a randomly timed interval, goes back to activating (1) them again. There is also a gate at the end of the drawing procedure that irregularly closes and stopping the algorithm from changing connections. Finally, there is a global timer (outside the encapsulation), set to 90 seconds by default (but user changeable) that will at this specified interval roll, with a 1/2 chance, to clear all connection points in the matrix and set new frequencies for the oscillators and for the amplitude generators (the frequency changes are delayed by a second so the don't change in the middle their fade out [all connection changes in the matrix use a fade in/out timed to a second using the '@ramp 1000' argument]).

9.1.4 Output

The whole idea with this patch was to make something simple, so I did not want to output all eight channels separately like in *Matrise*. Feeding out two identical mono channels as stereo felt to simple though. To get a more interesting signal output I included a simple panning section using one of the Max included panning algorithms from *MSP Tutorial 23: MIDI Panning*; the 'constant distance xfade'.[23] This algorithm takes numbers in the range of 0-127, whereas 0 represents left and 127 represents right, to control panning of a signal split into two $\star\sim$ objects. To control them, eight 0-127 `drunk` objects with a step size of 16 (which is rather high) sets the panning number, making sure the panning is prominent and properly audible. The `drunk`s are controlled by irregular bangs based on the procedural drawing algorithm: every time a connection point is changed, the panning amplitudes also changes.

[23] MSP Tutorial 22 `http://www.cycling74.com/docs/max5/tutorials/msp-tut/mspchapter22.html` [2014-05-02]

9.2 Sound results

The 'shaking' sound character caused by the `rand~` objects makes the sound character in this patch interesting. They were originally sine oscillators, but the sound became to predictable and transparent. The patch works well with several kinds of waveforms for sound generation. *Matrise redux* were used in the all three *Texture* pieces.

10 Dronetool

Dronetool is another patch built to make drone textures. *Matrise redux* and *Dronetool* were used simultaneously in the composition of *Texture 1* and *2*. The idea was to compose thick, randomised structures of superimposed synth tones.

10.1 Programming Dronetool

Dronetool is the simplest of the main patches in this project. Sound wise (and CPU wise), it makes up for it's simplicity using the `poly~` object to load 50 instances of itself. The patch does not, however, usually use all instances at the same time, so the number could have been lower, but I wanted to ability to create an intense, overloading feel. The `poly~` is set up so every time a new tone is created, it is allocated to the next free voice (up to 50).

The sound source in *Dronetool* is a combination of four oscillators (the classic sine, triangle, saw, and square) and a pink noise generator. Every time *Dronetool* is instructed to produce a new tone it sets a value for all five sources that are combined into a single, more complex signal (but still a simple, typical synthesiser tone). A modification of the `polyise` patch, `interpolyise` integrates the possibility to morph between different combinations of waveform levels, using the line `object`, providing a richer texture by adjustment of synthesis parameter during the sound, not only in between each sound.

For envelopes, *Dronetool* uses a simplified envelope generator that is commonly known as an AR-envelope (contradictory to the ADSR-envelope), made with a `curve~` that ramps up for attack, waits, and then ramps down for release. These three values, the length of the attack, sustain, and release, are randomised for each

tone, using a combination of `random` and `delay` (`delay` is used for delaying a bang for a specified amount of time).

A `lores~` (resonant lowpassfilter), which can be omitted, filters the signal. The cutoff frequency and resonance gain constantly changes, and the gain sometimes peaks which is what creates the 'screams' that can be heard in *Texture 1* and *2*. This sounds noisy in a good and desirable way, but required the use of the `sigmoid.gendsp` here as well.

10.2 Sound results

The sound generated with this patch is quite uniform, which is what makes the patch useful. The morphing of levels in the waveform mixer, and the filter, is completely required to make it interesting.

Part V
Reflection

This part contains reflections upon what was achieved with the work described in the previous part, and general observations around working with CAC in this way.

11 Audio as music — Max for composition

This kind of music represents a special kind of utilising audio: the audio is no longer a representation of music, the audio *is* the music.

Max is generally a DSP programming tool, widely and commonly used for both daily and more specialised solutions for synthesis, effects processing, installations, video arts, performance software, and controllerism. For most users it is probably not associated with CAC, at least not since the MSP (Max Signal Processing) addition from 1997. Adding the possibility of audio processing and generation has, for a majority of the users, made the software mainly an open audio platform.

This is what makes Max a natural composition platform for me, since as a composer, my 'media' is sound represented by audio,[24] rather than sound represented

[24]By 'audio' in this context I mean sound that is either converted from sound waves to digital or analogue audio, or the other way around. Oxford Dictionary defititon of audio: 'Sound, especially when recorded, transmitted, or reproduced' [Oxf13].

by notes (or a combination of audio and notes, which seems to be a preferred media for many in my generation of composers). Some propose a difference between these two approaches:

> One might say that while the traditional composer working with traditional instruments composes using sounds, the electronic composer composes the sounds themselves. [CG09, 3]

I can understand how such statements are tempting to use, especially in the opening of a book about computer music and sound design. We, the composers of electronic music, should be careful about claiming property on composing sounds, as composers writing for acoustic instruments also composes sounds. The action of combining a bassoon tone with a viola tone is just as much composing a sound as modulating a sine wave with another sine wave. Second, if electronic composers composes the sound itself, what about all the electronic works based on recorded sound, the tradition of musique concrète, field recordings, the vast remix culture, etc? In the early history of tape music, one used a genre separation based on the sound source—recorded sound; musique concrète, and electronically generated sound; electronic music—but this separation of tape music soon became obsolete and unnecessary, since composers started to use both techniques in the same pieces. Stockhausen's *Gesang der Jünglinge* composed 1955-56 is a notable example where the composer composes using sound, composes sounds themselves, and composes using composed sound based on used sound.

In this project the sounds were generated using oscillators (and some noise generators). Periodic waveforms have since the beginning of electronic music been a cornerstone and are no less relevant today then in the 50s. A lot of software now allows for the use of custom waveforms, either by drawing them yourself,[25] or, with sample based synthesis, using any sound as a waveform for synthesis. As much as I find such ways of sound generation interesting, in this project I found the classis synthesiser waveforms—sine, triangle, saw and square—sufficient as these waveforms, especially when using the anti-aliased msp-objects (`tri~`, `saw~`, and `rect~`), sounds good and are well suited as 'raw material' for further manipulation and signal processing.

[25] Especially interesting is the software *DIN is Noise* by Jagannathan Sampath, which uses fully real time modifiable Bezier curves for oscillators and literarily everything else. (`http://dinisnoise.org` [2014-05-08])

11.1 The concept of a Max patch

A way to view some (my) Max patches is as a musical statement: something that says something musical, or communicates a musical idea. To bring this idea further, I think of the Max patch as a combination of a score and an instrument. All though the imagined 'score' in my patches in some ways are created in real time, the patch still functions as a score with extremely precise details on how every piece of the patch is to act.

One find is that the idea of a sound algorithm feels almost identical to the idea of a musical piece. The sound algorithm will produce a specific kind of sound. The patch becomes the score, since the programmer is constantly working on adjustments in the patch, and the patch is what communicates the musical idea. Ideas arise as actual musical results occurs, and the original idea becomes either more transparent or more defined based on how the idea was first conceived. Like in Bowen's tempo measuring experiments, the infrastructural framework is the same every time, but each performance features a unique presentation of the same idea.

12 Random numbers as data source

The data source used in this project was random numbers, generated in real time as the compositions came together, as sort of a synthesised data stream, hence the second part of the title *synthesised composition*.

For random numbers and decisions, four Max objects are of extra relevance: `random`, `drunk`, `decide`, and `urn`. `urn` (which outputs *all* possible numbers in a random order without duplicates) were not used in this project, but `random` and `drunk` were used to feed random numbers into very many audio and audio related objects, therefore having a major impact on the final result. `random` is a simple object, it has an integer number as an argument, say 100, and will then, every time banged, output a number within the range of 0-99. The `drunk` is more complex, it outputs a number one step next (up or down) to the previous number, or the same number again. Step size may be set using a second argument. One important thing to know about `drunk` is that it almost always starts from the middle of it's range, meaning a range of 100 (0-99) with default step size will output either 49 or 50 as the first number. `random`, `drunk`, and `decide` may be further specified

with a 'seed' message, controlling the frequency of reproduced numbers. This possibility has not been explored in this project, the objects have been used with their default, 'flat' seed. Reproduced numbers are though very useful in musical composition, since they potentially can be used for easily creating musical relations using repetitions and contrasts.

The MSP objects `noise~` and `rand~` are also useful random generators, outputting random *signals* rather than data. `noise~` is a white noise generator outputting random values between -1. and 1. `rand~` is basically the same object, but with a frequency inlet where the frequency of new values output may be set (1 Hz of course being every second, turning it up to the sample rate of the system [typically 44100 Hz] will make the `rand~` object output white noise in the same way as `noise~`). Note that `rand~` interpolates linearly to the next value.

The decision of using such objects to generate the numerical material for the compositions is important. In my systems, the numbers only in a small degree decide what the composition will sound like, but they do control how the sound producing parts of the system acts. Like written in section 2, the type of numerical material used does not necessarily decide much of the outcome, since the system do not care wether it uses numbers from say a gyroscope or stock numbers. In the case of the gyroscope, which I have experimented with in live electronic setups,[26] the gyroscope data itself is not what is interesting (one parameter produced a signal at -0.001 to 0.001, not outputting much between other than 0.), it is how the data is used. To make such a signal useful it requires scaling and mapping. Some times, like in the case of the extremely low signal of -0.001 to 0.001, scaling can make the original and scaled signal so different that they do not seem relevant to each other anymore. I believe that my use of randomly generated numbers rather than data streams and/or controller data is a good illustration of how irrelevant relevant data can be. The random numbers has no meaning at all, they are arbitrary, and do not represent anything before they are used for synthesising the sound and the composition. When they do this, they do exactly what they are intended for, since the algorithms are carefully calibrated to produce numbers within a range that is artistically interesting for their destination object.

[26] Using an iPad with the Cycling '74 application *Mira*, which provides access in Max to the iPad's gyroscope, accelerometer, touch, etc, via wifi.

13 The actual music

13.1 Lack of controllerism

Sound wice, the music in this project share much with a lot of other examples of contemporary computer realised music. The important difference is located at the controllerism, or actually in the lack of controllerism. Controllerism is of course important in live performance, especially improvisation, but is also relevant in studio and composition situations.

Controllerism is a term referring to hardware and interfaces for controlling sound producing or sound manipulating hardware or software. At one point there were electronic controllers based on virtually every musical instrument (like in the fusion group 'Steps Ahead' where Micheal Brecker played a MIDI saxophone, Mike Mainieri played a MIDI vibraphone, both connected to synthesisers, and Steve Smith played drums fitted with triggers connected to a drum synth.[27]), before controllerism entered it's current phase, which is sort of postmodern controllerism age—*anything* can be a controller.[28]

Making a system where there is no controllers brings up two interesting conditions: First, it makes the music acousmatic, since there is no one performing it, and nothing to look at for the audience. Second, it requires much more preparation (much performed 'laptop music' is often prepared improvisations, using a software setup with a defined framework), since the piece have to be completely finished before performing it.

13.2 Musicscapes — sound art

When starting this project, I had no special desire to compose music in a specific style, other than that I knew it was going to be realised with CAC.[29] The style of the music I ended up with falls within the categories of 'ambient' and/or 'drone', which I probably should have expected, since this is one of my fields of interest in music. The pieces composed have sort of a 'musicscape' quality to them, since

[27] Album: *Live in Tokyo 1986*, Steps Ahead. NYC Records 1995. Compact disk.

[28] I once attended a concert were a musician was shaking an artificial sensor equipped palm tree.

[29] For instance I took a course in OpenMusic to see if the software was relevant to the project, which it could have been, just as much as Max.

they share many features with soundscapes. They were however always intended as music.

Matrise (the patch) has the ability to be used as a generative, exhibited sound installation. It can also be started and stopped manually in a concert situation, but I chose to record the generated audio and bounce it as fixed audio files. There are three important aspects of this:

- One has the possibility of editing the recordings, removing certain parts, making adjustment, etc.
- One maintains maximum control over what the audience will hear.
- Related to the above bullet, one decides how long the piece is going to be, and therefore how long the audience will listen to it.

Generative music shares some aspects with exhibited sound installations. A common practice in sound installations is that there is a room, where some sort of audio process is going on. The audience will enter the room, experience the audio, and then leave the room to continue their visit at the gallery. This means that the artist has no control over how much time the audience spends on (or in) the installation, and they may because of this not experience the full artwork. This may be solved by providing listening instructions, but this may also cause restrictions on how the audience listens, and therefore also potentially removes a full experience.

A term I encountered during literature searches was what is known as *sound art*, which as I read about seemed relevant to my project. Alan Licht defines sound art as something divided into three categories:

1. An installed sound environment that is defined by the space (and/or acoustic space) rather than time and be exhibited as a visual artwork would be.
2. A visual artwork that also has a sound-producing function, such as a sound sculpture.
3. Sound by visual artists that serves as an extension of the artist's particular aesthetic, generally expressed in other media. [Lic07, 16-17]

Based on this, the fixed media presentation of my works are not sound art. Licht's three categories all feature an 'unfixed' character, something that has a semi-self

consciousness it uses for development of itself. If I rigged *Matrise* as an installation in a room and left it running it would be in category 1. Since I have recorded and edited it, it is not a part of Licht's definition of sound art—it is music.

13.3 Time and form

13.3.1 Real time

Building audio generative Max patches and recording audio from them has the (dis)advantages of being forced to generate the audio in real time (e.g. creating 60 seconds of audio takes exactly 60 seconds). Being able to activate a patch and instantly hear the results is, compared to working with DSP some decades ago, a luxury. Compared to many other workflows though, which can be 'faster than real time', it is slow. Having to manually record audio routed from Max to a DAW, means it takes a lot of time to generate not exactly much material (remembering Cope that had 5000 pieces generated during his lunch break, though this was not DSP). There is of course the possibility of having several ongoing processes on either one computer, which is not practical when executing complex DSP with reverbs on a six years old laptop, or on several computers, which is not practical since in the initial test phase the patch changes all the time and it is much easier to have just one copy of the patch that all of the modifications are executed on. Working on several different patches on different computers simultaneously is not an option either, since my mind simply do not have sufficient multitasking abilities to do this. On the plus side, working in real time with one process at the time has the advantage of instantly having the material presented, allowing to also evaluate the quality of the material in real time, which is positive since this is something that has to be done anyway. The story do not tell what David Cope did with the 5000 pieces, but I have a hard time imagining he has ever listened to all of them.

13.3.2 Working with form

Intentionally I wanted to produce a system that would compose the form—the macro level of the piece—using randomised parameters similar to those that were used for the synthesis, this way creating a 'synthesised' form. This became the approach in *Matrise*. In the other patches, this approach was abandon and replaced

with a more traditional way of working: manually combining pieces of audio to form a composition. To be able to compose this way, the use of the DAW became more important than originally planned. It required me to work the same way as a composer working with recorded sound does, one have to find the sounds, organise them, and utilise them in an interesting way.

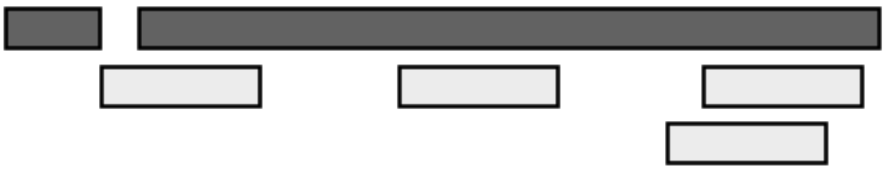

Figure 10: Graphic score (roughly!) showing organisation of audio segments in *Texture 1*. The dark bars represent sound made with *Dronetool*, while the lighter bars represent sound made with *Matrise redux*.

Figure 10 shows the layout of audio segments combined to compose *Texture 1*. *Texture 2* would present a similar score with a few, long bars, while *Texture 3* is made by combining a lot of short, superimposing segments.

14 Finding sounds — objet trouvé?

Working with CAC is related to working with objet trouvé, or found object. The composer searches for interesting bits of information within the algorithm and, based on the complexity and restrictions built in the algorithm, the composer 'never' know what (s)he will find. Still, in classic objet trouvé art, like the one by Marcel Duchamp, the objects are usually completely unmodified ordinary or 'daily' objects. An algorithm programmed to produce compositional output is not related to anything daily at all. Translating the term objet trouvé to the fields of composition and DSP, the practice of field recording is a more natural analogy. Field recorders tend though to visit locations knowing that the specific location is going to produce a desired soundscape, like rivers, highways, urban areas, etc. They look, or listens, for the sound, just like I listen to the sound of the algorithm to find something interesting, something useful that can become part of a composition.

15 The future of CAC

Allowing myself to make an assumption, I do not believe the future of CAC lies within mapping of random numbers, as have been my approach. As previously stated, a problem with CAC is that the computer does not 'know' that it is working with music, which should make us ask the question, if it is composing without knowing it is composing, is it composing? Further, if the computer 'knew' it was composing, would it compose better music? This thought, combined with future progression in technology makes it likely that artificial intelligence, machine learning, and evolutionary and adaptive algorithms will be utilised for composing music. I already mentioned David Cope's 'Emily Howell'-software, and the feature-feedback systems of Risto Holopainen. This is possibly only the beginning, and if so, we are going to have a very exciting future of CAC.

Part VI
Conclusion

16 Answering research questions

16.1 Main research question

- *How may computer assisted composition assist or stimulate the composition of electronic music?*

I wrote introductory that there was no obvious way of answering this question. In retrospective, I feel that Part IV and Part V together is one out of extremely many possible answers to this question. My approach is only one approach, which is stylistically greatly coloured by my preferences in music, my 'way' of using Max (there are countless ways to use Max, especially with external third party objects, GEN, and communication with other softwares), the actual use of Max rather than other software (I truly believe the produced music, and possibly also the content of the thesis, would have been something completely different if my workflow was based around SuperCollider or another environment), and countless other variables that have influenced the project.

I believe, though, that the project is a good example of what is possible to achieve with CAC, based on little more than curiosity for an unfamiliar field and the will and interest to work with a master's project. Ironically, the amount of

hours invested in the project could probably, if used for 'normal' composing resulted in a lot more music than what I have produced during the project. This music would though have also been something completely different than the music composed, since CAC, as this project has shown, is a tool that can allow the composer to compose music s(he) would not compose otherwise, since it conveys a different set of thinking, another way of viewing music.

16.2 Secondary research question

- *'Where' in computer assisted composition is the artwork?*

This was perhaps a loaded question from me to me. I had a thought about making musical 'statements' in Max, and letting them make sound, based on their own infrastructural character. This was also what I did, I made situations where programming became composition or composition became programming. I think this attitude towards CAC was what let me make music of the level that I did. By viewing programming as composition or visa vi, and by this letting the algorithm not only make the music, but, in fact *be* the music, the perspective of what one can expect from it, and what it is possible to create, is more in contact with reality than if we want to compose masterpieces without effort. The computer's lack of the ability to see arts the way we do is what makes them useful exactly for making arts. David Cope has a wonderful view on this:

> He's [Cope] now convinced that, in many ways, machines can be more creative than people. They're able to introduce random notions and reassemble old elements in new ways, without any of the hang-ups or preconceptions of humanity. [Bli10]

17 Application of method

Working with a research method was the most difficult part of the project, possibly since it was the aspect I had the least experience with in advance. Let us recap the programming steps I sat up in section 6:

1. Form an idea for a procedure

2. Define procedure in programming environment
3. Test
4. Evaluate
5. Redefine
6. Record material to DAW
7. Arrange material into musical composition
8. Evaluate process

During work with the programming I often found that step 1. and 2. grew into each other and became a conjoint step. Even though I had an idea while I sat down to program, the idea would soon fail, either because I had not planned it in deep enough detail, or that the programming environment and/or my knowledge of it had shortcomings with regards to fulfilling it. I would at this point find a new way of dealing with the function I wanted for the specific patch, which of course resulted in a different, but nonetheless usually useful output.

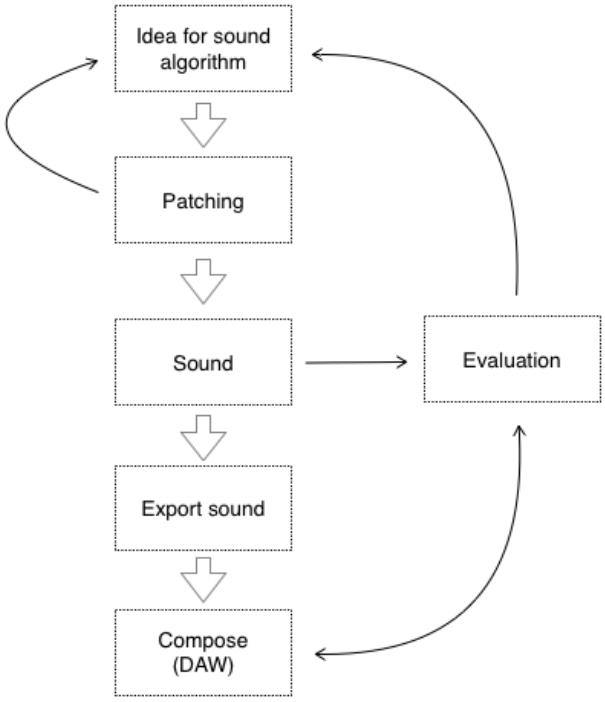

Figure 11: Actual workflow.

One could also see step 1-5 as a superimposed action, as I often would program, create, test, evaluate and redefine/redesign as in one action. Figure 11 shows a better representation of the actual workflow. Looking back, the testing and evaluation steps should have been more formalised, ideally organised in a systematic way, and all the work kept in logs. This could possibly have resulted in finalisation of more ideas, since very many patches were abandoned at step 2 or 4. The ones abandoned at step 2 was either impossible to program—due to my programming skills and/or limitations of Max, or simply lost and forgotten about among the 602 files associated with the project. The ones abandoned at step 4 didn't turn out as good as I had imagined, some of them possible also due to too low skills. I am now, at the end of the project, a much better Max programmer than at the beginning, and some of the abandoned patches (ideas) are probably possible to realise now. Although I ended up presenting only three patches in the project, the approximately 20-30 abandoned patches played a significant role in the development of the three presented ones.

The initial idea is often unclear, and do not intuitively tells how much work it will involve. For example, the main structure in *Matrise*, with the exception of the pink noise generating feature and the soft clipper, was done in one day's work, followed by about two weeks of adjustments, calibrations, and listening. Like written in the section describing the patch, the initial idea was not much more than using the `matrix~` object. This shows that it is not necessarily the idea that comes first, or the first idea that ends up being used, but that the musical potential in a work becomes more transparent during time and work. I have no reason to believe that this, the changing of ideas as they are explored, is experienced differently for composers working with notes, or any other musicians working in any other way.

18 Synthesised Sound & Synthesised Composition

My title for this work is *Synthesised sound & synthesised composition*, which obviously refers to the use of synthesis as sound source, and the artificial generation, the synthesis, of the composition. But as I quoted Cipriano and Giri on in section 11, in electronic music the composer also composes the sound. Based on this the title should perhaps be *Synthesised composition of sound & synthesised composition*, which is a little too long. The point is, that this way of composing can be viewed

as superimposing of composition of synthesis, and composition of composition. They are not separate entities, they are one element in a way that makes it meaningless to talk about one without mentioning the other. Remove the synthesis and there is only a stream of numbers. Remove the composition and there is only a sound generator not creation sound. There is a synthesis of sound, which is the composition, which is the sound.

Part VII
Appendix

19 Bibliography

The bibliography is realised with BibTex-file, formatted in the Alpha-style.

To reduce the amount of visual noise in the bibliography I have avoided unnecessary extra words like 'retrieved from', 'access date', 'online', etc, for the online sources. If the source is retrieved online it will inform the reader with this, simply by having an included url in a monospaced font, with the access date attached at the end in brackets.

The access dates are written in the ISO 8601 date format YYYY-MM-DD. Urls are clickable, however, some sources (like articles retrieved from JSTOR) requires subscription and/or log ins for further viewing.

References

[Ari11] Christopher Ariza. Two pioneering projects from the early history of computer-aided algorithmic compositions. *Computer Music Journal*, 35.3:40–56, 2011. `http://hdl.handle.net/1721.1/68626` [2013-05-29].

[Ass98] Gèrard Assayag. Computer assisted composition today. *1st symposium on music and computers*, 1998. http://www.ircam.fr/equipes/repmus [2013-09-03].

[Bli10] Ryan Blistein. Triumph of the cyborg composer. 2010. http://www.psmag.com/culture/triumph-of-the-cyborg-composer-8507/ [2012-11-14].

[Bor06] Henk Borgdorff. The debate on research in the arts. *Sensuos Knowledge: Focus on Artistis Research and Development*, 02, 2006.

[Bow96] José Antonio Bowen. Tempo, duration and flexibilety: Technique in the analysis of performance. *J. Musicological Research*, 16:111–156, 1996.

[Bre06] Jean Bresson. Sound processing in openmusic. 2006. http://www.dafx.ca/proceedings/papers/p_325.pdf [2014-04-11].

[Bro12] Scott Brown. Response: Who's afraid of artistic research? *kodamapixel.com*, 2012. http://kodamapixel.com/2012/03/response-whos-afraid-of-artistic-research/ [2013-05-10].

[CG09] Alessandro Cipriano and Maurizio Giri. *Electronic Music and Sound Design: Theory and Practice with Max/MSP*, volume 1. ConTempoNet s.a.s., Rome, Italy, 2009.

[Cop91] David Cope. *Computers and Musical Style*, volume 6. A-R Editions, Inc, Madison, Wisconsin, 1991.

[Cop97] David Cope. *Techniques of the Contemporary Composer*. Shirmer Thomson Learning, USA, 1997.

[Cop00] David Cope. *The Algorithmic Composer*, volume 16. A-R Editionns, Inc, Madison, Wisconsin, 2000.

[Hil63] Lejaren Hiller. Electronic music at the university of illinois. *Journal of Music Theory*, 7:99–126, 1963. http://www.jstor.org/stable/843024 [2013-01-16].

[Hof09] Peter Hoffman. *Music Out of Nothing? A Rigorous Approach to Algorithmic Composition by Iannis Xenakis*. PhD thesis, Technischen Universität Berlin, Berlin, 2009. http://hdl.handle.net/1721.1/68626 [2013-01-11].

[Hol12] Risto Holopainen. *Self-organised Sound with Autonomous Instruments: Aesthetics and experiments*. PhD thesis, Univeristy of Oslo, Oslo, 2012. https://www.duo.uio.no/handle/10852/37664 [2014-04-08].

[HSV05] Mika Hannula, Juha Suoranta, and Tere Vadén. *Artistic Research*. Academy of Fine Arts, Helsinki, Finland & University of Gothenburg / ArtMonitor Sweden, 2005.

[Lan89] Peter Langston. Six techniques for algorithmic music composition. 1989. `http://www.langston.com/Papers/amc.pdf` [2014-04-11].

[Lic07] Alan Licht. *Sound Art: Beyond Music, Between Categories*. Rizzoli International Publication Inc, New York, 2007.

[Mur84] Tristan Murail. Spectra and pixies. *Contemporary Music Review*, Musical Thought at Ircam:157–170, 1984.

[Mur05] Tristan Murail. *Models & Artifice The Collected Writings of Tristan Murail*, volume 24, Parts 2+3, April/June. Routledge. Taylor & Francis Group, 2005.

[OB61] Harry F. Olson and Herbert Belar. Combination random-probability system. *Radio Corporation of America*, USA, 1961. `http://www.google.com/patents/US3007362` [2013-09-3].

[Oxf13] Oxford. Oxford dictionaries. 2013. `http://www.oxforddictionaries.com` [2013-10-18].

[Pul02] Ville Pulkki. Compensating displacement of amplitude-panned virtual sources. *AES 22 International Conference on Virtual, Synthetic and Entertainment Audio*, pages 186–195, 2002. `http://www.acoustics.hut.fi/research/cat/vbap/papers/pulkkiaes22.pdf` [2014-01-28].

[Roa96] Curtis Roads. *The Computer Music Tutorial*. The MIT Press, Cambridge, Massachusetts; London, England, 1996.

[Ser93] Marie-Hélène Serra. Stochastic composition and stochastic timbre: Gendy3 by iannis xenakis. *Perspectives of New Music*, Vol. 31, no. 1:236–257, 1993. `http://www.jstor.org/stable/833052` [2012-04-02].

[Sow13] John Sowa. A machine to compose music. 2013. `http://www.jfsowa.com/misc/compose.htm` [2013-09-02].

[VG04] Øivind Varkøy and Erling Guldbrandsen. *Musikk og mysterium*. Cappelen akademisk forlag, Oslo, Norway, 2004.

[You58] Joseph E. Youngblood. Style as information. *Journal of Music Theory*, 2:24–35, 1958. `http://www.jstor.org/stable/842928` [2013-10-22].

20 Sound examples

Overview of the sound files generated for the project. All tracks mastered by Lars Erik Sparby.

URL: `http://magnusbugge.bandcamp.com/album/sssc`

- `Matrise (stereo mix).wav [09:08]`
- `Matrise (8 channel).wav [09:12] [DVD only, accompanied by Matrisechannelrouting.png]`
- `Texture 1.wav [05:58]`
- `Texture 2.wav [03:42]`
- `Texture 3.wav [04:47]`

21 File list

Overview of the files uploaded to the github repository. Second level items are files associated with their imposed first level items. File extensions: `maxpat` requires Max to edit, but may be viewed and run with Max Runtime (free), `gendsp` requires GEN to edit, but may be viewed with Max or Max Runtime, and `gcx` requires Mac OS X Grapher to open.[30]

Download all: `http://github.com/magnusbugge/SSSC/archive/master.zip`
Visit the Github repository: `http://github.com/magnusbugge/SSSC`

- `Dronetool.maxpat`
  - `polyise.maxpat`
  - `interpolyise.maxpat`
- `LICENSE`
- `Matrise.maxpat`
  - `gridctrl.maxpat`
  - `hipass.maxpat`
  - `matrisesynth.maxpat`
  - `noisethresh.maxpat`
  - `trainjaff.maxpat`
  - `trainjafflogic.maxpat`
- `Matrise redux.maxpat`
- `README.md`
- `sigmoid.gendsp`
  - `sigmoidraw.gendsp`
  - `sigmoidpresentation.maxpat`
  - `sigmoid.gcx`

[30] Download Max, Max Runtime, and/or GEN from Cycling '74, `http://cycling74.com/downloads/`.

22 Attachments

Attachment 1/1: Concert programme from the performance of *Matrise*.

STILLE LYD – "Quiet Sounds"

A concert of electroacoustic works from Norway

7pm, 28th March 2014 – PACE Studio One

In the winter of 2013/14 Andrew Hill travelled to the Norwegian Centre for Technology in Music and the Arts (NOTAM) to work with their composers and technicians in the development of electroacoustic works around the theme of "Quiet".

This concert is the culmination of this project, featuring the newly composed works 'Stille Lyd', pieces by contemporary composers based at NOTAM, and a performance of the original four channel version of the piece 'Solitaire' by the Norwegian composer Arne Nordheim, which provided inspiration for the project.

This project was supported by the British Council and Arts Council England, NOTAM and Production Network for Electronic Art (PNEK).

CONCERT PROGRAMME

UhnKunkh by Anders Tveit
(2003) 2 Channel.

Stille Lyd: Part I – NOTAM by Andrew Hill
(2014) 2 Channel.

Matrise by Magnus Brugge
(2014) 8 Channel.

Losing Control: Part I and Part II by Gyrid Nordal Kaldestad
(2011) 2 Channel.

Stille Lyd: Part II – Høvringen by Andrew Hill
(2014) 2 Channel.

Solitaire by Arne Nordheim
(1968) 4 Channel.

Supported By:

notam. Supported using public funding by ARTS COUNCIL ENGLAND | BRITISH COUNCIL pnek Production Network for Electronic Art, Norway