

**Music Out of Nothing?**  
**A Rigorous Approach to Algorithmic Composition by Iannis**  
**Xenakis**

vorgelegt von  
Peter Hoffmann

Von der Fakultät I - Geisteswissenschaften  
der Technischen Universität Berlin  
zur Erlangung des akademischen Grades

Dr. phil.

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Stefan Weinzierl

Berichter: Prof. Dr. Christian Martin Schmidt

Berichter: Prof. Dr. Helga de la Motte-Haber

Tag der wissenschaftlichen Aussprache: 29.04.2009

Berlin 2009

D 83



**Music Out of Nothing?**  
A Rigorous Approach to Algorithmic  
Composition by Iannis Xenakis

Peter Hoffmann

October 11, 2009



“Perhaps we have grown too accustomed to the idea that it is necessary to exist. I think there is another way but I cannot see clearly, I cannot yet say what it is. There may be another reason for our life, our action than to strive for immortality, power or the justification of existence [...] There must be other reasons. [...] [One] may die and disappear but that applies only to the individual. But not ...”

(Iannis Xenakis, 29.05.1922–04.02.2001, in his 1980 conversation with Bálint András Varga.)

Next page: The sensational discovery by Rudolf Pfenninger and Oskar Fischinger (photo) in the 1930s that sound can be arbitrarily invented by “painting” it on the optical sound track of a film strip (instead of recording and processing it) is reflected in a headline reading “Music out of Nothing” presented as the title of a short introductory prelude to the film “Barcarole” by Rudolf Pfenninger, a puppet animation film with a synthetic optical sound track ([Goe98], [Mor93]). Iannis Xenakis, sixty years later, goes one step further. He does not define the sound signal by painting but lets it come into being by probability fluctuations of stochastic processes, creating sound from silence, “out of nothing”. But in doing so, does he really create a “Music out of Nothing”?

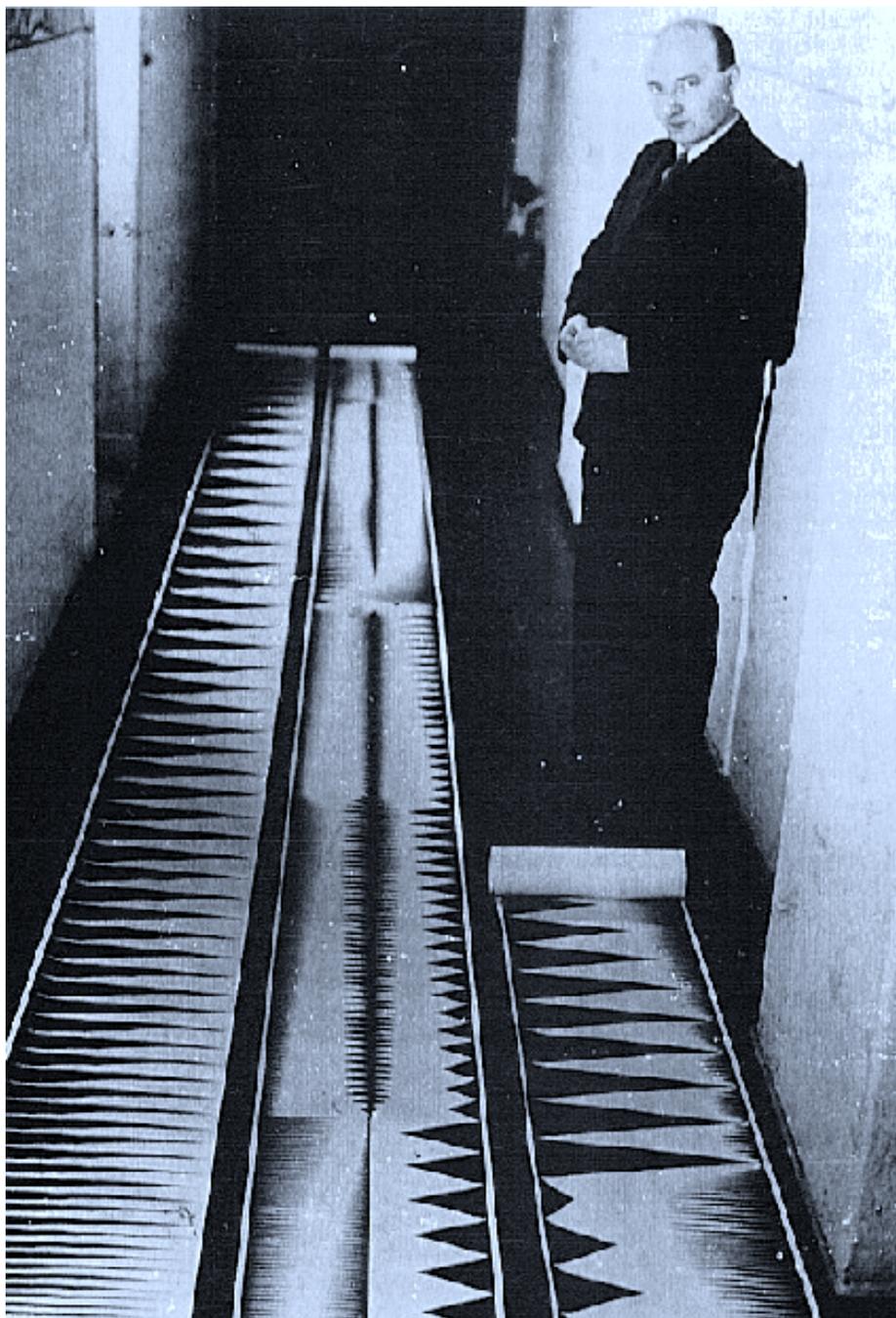


Figure 1: Oskar Fischinger's: "Music Out of Nothing"



# Abstract

GENDY3 (1991) by Iannis Xenakis (1922-2001) is a piece of computer generated music. But it is more than just “computer music”. GENDY3 is the culmination of Xenakis’ lifelong quest for an “Automated Art”: a music entirely generated by a computer algorithm.

Being a radical instance of a pure algorithmic composition, GENDY3 is, in a precise mathematical sense, a computable music: every aspect of its sonic shape is defined by an algorithmic procedure called “Dynamic Stochastic Synthesis” (“Génération Dynamique Stochastique”, or GENDYN for short).

The GENDYN Project, started by the author in 1995/96 with a research at CEMAMu, then the composer’s research center near Paris, exploits this computability for developing and documenting the GENDYN concept, in order to understand its various ramifications and to make it accessible for further research and production. To this end, the author implemented Dynamic Stochastic Synthesis in a new program called the “New GENDYN Program” which, in addition to “recomposing” GENDY3 in real time, makes it possible to inspect and control the algorithmic composition process, thereby opening up new perspectives both in Computational Musicology and in computer music creation.

For music analysis purposes, GENDY3 has been completely resynthesized. The simulation of the genesis of GENDY3 “in vitro” made possible by the New GENDYN Program permits the systematic exploration of the “decision space” of the composition model, contributing to a deeper understanding of both its potentials and limitations, and the complex interaction between “material requirements” and compositional freedom within which the composer navigated.

The study of the GENDYN compositions provokes many fundamental questions about computing, listening and understanding, of creation, interaction and computer music aesthetics. It is shown that 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. This unusual approach addresses the medium of electroacoustic algorithmic music, i.e. the physics of sound, as well as the computability of sound as subjects of artistic creation. This approach seems to the author to be of a higher value for the foundation of a “true” computer art than the widespread ambition to emulate human creativity by computers and to build up an artificial brave new world of music.



# Preface

This is the documentation and discussion of a research project in algorithmic computer music, started in August 1995 with a research in Paris, in the framework of a dissertation at the Technische Universität Berlin. Its scope is the latest achievement by Iannis Xenakis (1922-2001) in composing music with the computer at his institute CEMAMu (“Centre d’études de Mathématique et Automatique Musicales”<sup>1</sup>) in Paris. The composer called his new computer composition method “Dynamic Stochastic Synthesis”. It is a rigorous algorithmic procedure for not only composing the macro-structure of a composition (i.e. on the level of duration and pitch) but also the micro-structure of sound (i.e. on the level of the digital sample). GENDYN is the name Xenakis gave to his computer program(s) realizing the stochastic synthesis, short for the French term “GÉNération DY-Namique”. The original program was written in BASIC by the composer himself at CEMAMu, Paris, with the assistance of Marie-Hélène Serra. In 1991, a single run of this program, called “GENDY301”, generated *GENDY3*, a piece of about 20 minutes. It was later released on CD (NEUMA 450-86). In the aftermath, Xenakis tried to extend the program to include additional time-variant effects by making some parameter settings of GENDYN’s stochastic processes time-dependent. This resulted in a second piece *S709*, premiered in 1994, which is beyond the scope of this study.

The “GENDYN Project”, the subject of this study as I would like to call it, is a project that takes Xenakis’ legacy in algorithmic music creation and tries to complete it in a way Xenakis either did not care or has not been able to do during his lifetime. First, the idea is to keep GENDYN on top of current computer technology in order to ensure its usability for the present and future. The result of this was what I baptized the “New GENDYN Program”, a complete new implementation of Xenakis’ program using C++ for synthesis and Visual Basic for the graphic user interface, on the Windows platform. Second, using the tools constructed in the first step, the GENDYN project aims at understanding the richness of Xenakis’ unique approach to computer creation, its various implications and ramifications in the fields of music aesthetics, analysis and production. Third, the idea is to feed back the results of steps one and two to the computer music community for artistic use. Xenakis of course was the first to be offered the New GENDYN Program but he declined, either on the basis of personal conviction or due to misunderstandings on his or my side. But other composers have used it or are interested in using it, similar to what happened to Xenakis’ early ST-program (1958-62) which Xenakis was proud of

---

<sup>1</sup>The English equivalent is *Centre for Mathematical and Automated Music*, a name actually given to a sister institution at Bloomington University, Indiana, led by Xenakis in 1967-72.

handing out to universities and research centers.

One aspect of the GENDYN Project, therefore, is to reproduce and study Xenakis' original composition *GENDY3*. Another aspect is to go further and to systematically explore the sonic space of GENDYN by controlling the parameters of synthesis while listening to the aural result. This makes the New GENDYN Program, in addition to being a customized tool for the analysis of a single musical piece, a generic "stochastic composition instrument", much in the sense of interactive computer composition.

It follows from the above considerations that the nature and the goals of the project are actually double-sided: one is "scientific": the musicological analysis of *GENDY3*, a specific piece of computer generated music. The other is "artistic": to preserve Stochastic Synthesis as a "composition instrument" good for creating new computer music. These two aspects, retrospective analysis and prospective development are two sides of the same coin. If we are interested in a compositional procedure and the music made by it, both from a scientific and an artistic point of view, its analytical as well as its productive aspects prove to be mutually enlightening.

The (pre-) history of this project dates back to late 1991, when I heard the sounds of the Dynamic Stochastic Synthesis for the first time. Xenakis' composition *GENDY3*, premiered at the *Rencontres Internationales de Musique Contemporaine de Metz* in November 1991, was broadcast by *Sender Freies Berlin* in the framework of a program covering the festival's most prominent events. This was a startling experience. Never had I heard sounds like this before. Not only was it the richness and the strangeness of the sounds, but also the fact that these sounds were "generated by the computer". They sounded totally different from what I was used to associate with "computer generated sound". Never had I experienced a computer music so brute, fresh and immediate. As I later learned, GENDYN sound is immediate indeed: all standard means of transmission (score, instruments, microphones, tape) are bypassed as the music is digitally composed by a computer "direct-to-disk". Of course, the music came from a DAT tape over the air, but it felt as if my loudspeakers were driven by a strange force governed by rules that were at the same time alien and familiar — an experience which maybe is rather typical for listening to Xenakis' music in general.

During the time between my first encounter with Stochastic Sound described above and the end of my studies in musicology and computer science at Technische Universität Berlin in 1995, I tried several times to contact CEMAMu, without any result. It was Radu Stan from Editions Salabert, responsible for the promotion of Xenakis' music, who finally got involved and mediated a visit to the institute in the presence of the composer. It was decided that I should try and add a graphical user interface to Xenakis' GENDYN program, which the composer had used for a last time in late 1994. Marie-Hélène Serra, who had helped the composer preparing the version of 1991, had later left the institute, and Xenakis had been trying alone to develop a somewhat extended version of GENDYN until the end of 1994, when he finally created a second piece with it, called S709, shorter and somewhat less ambitious than *GENDY3*. When I came to Paris in August 1995, he had stopped working on GENDYN altogether, concentrating on the composition of instrumental music.

I was quite afraid of starting this task. It was to be my first major software project, but I had no guidance either by the composer nor by the people working

at CEMAMu. Yet this open situation proved extremely fruitful, as I realized later. I was free to decide what to do and how to do it; I could apply what I had learned during my studies and what I had to learn by doing. My motivation was to provide the composer with a new user interface to his own program and to possibly assist him in composing new pieces with GENDYN. After I had learned that he was not going to use it, I was happy that I could use the program myself to find out everything I wanted to know about the composition of *GENDY3*.

I had bought a portable computer before going to Paris so I could work day or night in my small student's room at "Maison Heinrich Heine", the German pavillon at the Cité Internationale Universitaire de Paris. By the time the project started, Windows 95 was not yet available. Only very expensive laptops had sound cards built in. 8 Mbytes of RAM were a luxury. I bought one with 8 MB and no sound card. Programming was still 16 bit unless one wanted to mess with Microsoft's unreliable Win32s library. The choice of Visual Basic as a development tool was both suggested by the fact that Xenakis' original program was written in BASIC as it was imposed by the people at CEMAMu. It was only later that I realized I had to also (re-)implement the sound synthesis in order to get a usable tool. For reasons of execution efficiency and sound card access, I did it in C++ and I linked it to the Visual Basic user interface by remote object invocation using Microsoft's OLE 2.0 library (this was called "OLE Automation" at the time). This way I was able to provide an implementation generating GENDYN sound in "real time" (using the DA converter of an external PCM-CIA sound card that I had to acquire for that purpose), and this was quite an achievement given the fact that Xenakis' own program ran for days and nights before one could get any sound out of it. In analogy to CEMAMu's version 3 of UPIC which they called the "New UPIC", I called my implementation "The New GENDYN Program".

After my return to Berlin, and during the second year of my NAFöG scholarship, I fine-tuned the program to exactly simulate the algorithmic behavior of the original Xenakis program, which I had to analyze in much more detail for this purpose. The idea was to be able to do an analysis of all the layers and details of *GENDY3* by resynthesizing each of them with the new program. In addition, I felt compelled to prove that the new program was not just another way of doing Stochastic Synthesis, but that it was capable of producing exactly the same results as Xenakis' original program. Indeed, I demonstrated the equivalence of the old and the new program by combining one channel of the output of Xenakis' original program and one channel of my resynthesis into a stereo sound file. (This is possible because GENDYN synthesis is mono. Xenakis combined two mono soundfiles produced by two different program runs with a 80 ms delay to obtain the published version of *GENDY3*.) The combination of GENDYN and New GENDYN, with the same 80ms delay is impossible to tell from Xenakis' original piece.

Well, almost. In spite of all efforts, there remained a few seconds in the piece where the original and the resynthesis slightly diverge. There is an undulating glissando movement (track #11) in the before-last section of the piece starting at 17'01", which begins on the same pitch but after a few seconds starts to evolve differently, first almost unnoticeable, then more and more obvious, between the two channels of the combined recording.

First I was frustrated that after so much hard work, I was still not able to produce a perfect clone of Xenakis' piece. But then I started to like the

idea that there was something in Xenakis' creative work that finally escaped my profanatory actions. Fortunately, as I later realized, because in the end, creating a perfect clone is much less interesting than creating a perfect twin! No-one would have believed me that my resynthesis was not a simple copy if it had turned out to be exactly the same music! In the same spirit, I kept using a different formula for linear interpolation than Xenakis, and therefore the exact sample values of the original music are not reproduced. Even if I had, they would be different, because unlike the faithful reconstruction of pitch movement, I did not bother to emulate exactly the same random number sequences that are responsible for the actual amplitude sample values of *GENDY3*. Sound is the differential of sound pressure, and so it is not the absolute sample values but their dynamics of change that had to be reproduced in order to get the music of *GENDY3*. (This does also, by the way, avoid any problems with copyright, since the sound files are not identical.)

In order to construct the perfect twin, I had to modify my implementation of the New GENDYN program in order to emulate some of Xenakis' idiosyncratic programming style which proved crucial for the actual musical output of the program. Therefore, I had to add some (undocumented) features of Xenakis' original implementation to my (documented) program which were not part of the algorithm's realization but had to be there to obtain his music. These added features act much like a "personal card" (in the sense of a chip card to be inserted into e.g. a cellphone providing customization) with the help of which I "customized" my implementation of the algorithm to produce "original Xenakis". In other words, Xenakis' idiosyncrasies in coding Dynamic Stochastic Synthesis are distilled to "customizing code" which is exterior to the implementation of Dynamic Stochastic Synthesis as such. Once the exact resynthesis of the original Xenakis music is no longer required, this "customizing code" can be removed, since it slows down computation considerably.

The difference between the "personalized" version and the underlying straightforward implementation is similar to that between an *identical* and a *typical* simulation of a process: only the former reproduces the same musical result, whereas the latter composes a musical result of the same quality. For the purposes of analysis, the "personal card" contains important information about the specific way Xenakis actually implemented his compositional ideas into a working composition tool.

The idea of a computer directly driving a loudspeaker was my vision on the first encounter with GENDYN. At the end of the project, this vision was to become true. A composition like *GENDY3* can in principle be composed "on the fly" and played back "in real-time" during a concert. Imagine yourself sitting in the middle of a circle of 16 loudspeakers each being connected to a small laptop computer which would be normally used at home for typing letters (or dissertations). Each of these computer/loudspeaker combinations distributed in space would generate the sound of one out of 16 channels of *GENDY3*. This is not as far-fetched as it seems to be at a first glance, as the Structured Audio approach, proposed for the MPEG-4 standard, aimed exactly at this idea of a synthesis "in situ" where the data to be stored and transmitted are only control data. The sound itself would be produced at the consumer terminal in real time (cf. [Sch98]).

The motivation for this study comes from the listening experience and is quite simple: why does *GENDY3* sound as it does? From where does the quality

and richness of that music come, given that it is entirely computed by a (relatively simple) algorithm? How did the composer arrive at this result? What part of this result was the composer really looking for, what was accidental and had to be accepted together with the other part? I hope that the reader will find the answers to these questions in what follows, or at least some stimulating ideas for further discussion and research.

All people I spoke with during the whole project spontaneously seized on the question if and how a computer can be used for activities in the creative domain of Arts. In the case of Dynamic Stochastic Synthesis, we are confronted with this question in a conceivably most radical and compelling way. First, because the computer is not simply used as a powerful working tool in the hand of a composer but is entrusted itself with the task of defining a whole musical work in its entirety and in all of its aspects including the actual aural rendering. Second, because Stochastic Synthesis is a relatively simple algorithm which allows to scrutinize the very coming-into-being of a musical artwork in every and all of its details. It is relatively easy to understand the working of Xenakis' program. There is nothing like "artificial intelligence", fancy self-modification, or obscure hardware effects involved. I do not know of any computer program comparable in its simplicity which a composer could have dared to use for the composition of a masterpiece of almost 20 minutes duration. This music is able to aesthetically compete with systems of an incomparably greater degree of sophistication.

While writing the present text, I realized that, almost despite myself, it sometimes became polemic and tended to take on features of a manifesto. Originally, I only wanted to present the dry "facts", very much in the style of a technical report. But slowly I discovered that the techniques of electroacoustic music cannot be dissolved from very basic attitudes toward composing in general, much in the same way as it seems that composing cannot even be dissolved from very basic attitudes toward life! I never felt particularly inclined to philosophy, and I generally prefer to leave it to people who understand more about it than I do. The same holds for aesthetic discussions — sometimes they seemed to me to obscure musical phenomena more than elucidating them.

But when it comes to discussing algorithmic composition, questions of aesthetic or even "philosophical" nature can hardly be avoided. Rigorous algorithmic composition stirs questions like the following: if the computer makes the music, is there still any place left for human compositional craftsmanship? Where is the act of creation to be located? Inside the machine or inside the human creature programming it, or somewhere in between? What is the meaning of composition anyway when it comes to machine generated music? Instead of hiding these questions in asides and footnotes, I decided to confront them right at the beginning of this text. With this critique formed, I realized that many of the questions that had occurred to me during the course of my research were actually the subject of current discussions within the computer music community itself. I discovered a few authors in the vast computer music literature that I will be happy to quote in the following. They describe actual composing experience where I could only theorize.

This present document about a very specific instance of algorithmic composition carried out by a very specific composer could therefore function as a very specific contribution to a broader discussion about the question of what might be an idea of machine music. In order to do so, I sometimes prefer to put an edge to my arguments in order to make my point. I would like to ask the indulgent

reader not to take them as pure polemics but rather as the thought-provoking statements they are intended to be.

The text is grouped into three parts. Part I “Introduction” is the description of Xenakis’ artistic project GENDYN, of its aesthetic and philosophical implications. It serves to put Xenakis’ work into the context of computer music in general and his own music in particular. Part II “Theory” is the analysis of the theoretical background and possible ramifications induced by some of the questions raised in part I. It is also the place to describe the theoretical foundations (or what I think these theoretical foundations to be) of the New GENDYN Program. Part III “Analysis” is aimed at musical analysis proper: based on the clarifications presented in parts I and II, the music of *GENDY3* will be the main subject of this part.

For the purposes of science and pedagogy, computer technology somehow seems to provide us with all means to enter a kind of “Golden Age”. The advent of inexpensive computing power and internetworking standards permit us to explore, in an interactive manner, all kinds of complex phenomena. They give us the possibility to acquire an intuitive understanding of even the most intriguing creations in the realm of art and music. The internet links artists, scholars and the public and offers a world wide information repository, drastically reducing publication latency and granting immediate access to information about related work and activities. In this text, there are quite a few information sources and documents indexed by their **Uniform Resource Locator** (URL), the address of information on the internet. A Web browser pointed to such an address displays the information in its appropriate graphical rendering and invites to follow more pointers to other information sites somewhere else in the world. It would be just adequate to feed the results of the present study back to the Web and tie it up with related information. However, there are also reasons to provide for a stand-alone, printable documentation (for being able, e.g. to scribble on the margins, underline phrases, or read it in the subway). So I decided to produce this nice old-fashioned book using the  $\LaTeX$  typesetting compiler, originally invented by famous Donald Knuth for the making of beautiful mathematical text books. There is not much maths in this text but even the prose still looks good in  $\LaTeX$ .

This project would not have been accomplished without the generous help of a 2 year’s scholarship from the land of Berlin (NAFöG). Survival in Paris in 1995/96 was possible by a supplementary grant of the German Academic Exchange Service (DAAD). While the technical part of the dissertation project was a more or less solitary undertaking, the discussion of its ideas was to a great extent promoted by many people I had contact with in Paris, Berlin and elsewhere. They helped me to clarify ideas, forcing me to formulate them in a way understandable to myself, making many suggestions and opening my mind toward questions more fundamental than the pure technical ones. I would like to thank all who engaged in these private or public discussions.

Special thanks go to Christian Martin Schmidt who supported this project from the very beginning with enthusiasm. Iannis Xenakis granted me full access to his workplace at CEMAMu. The CEMAMu staff (Gérard Marino, Jean-Michel Raczinski, and in particular Vincent Fontalirant and Stéphane Sladek) provided me with software and were helpful on all occasions we met. I am also indebted for their particular support to Stéphane Paumier, Anas Rassem, Angelo Bello, Makis Solomos, Ronald J. Squibbs and Paul Doornbusch. Peter Löhr

spontaneously offered me a working place in his institute at the Freie Universität Berlin in 1996/97. The programming part of the project is the application (albeit an “exotic” one) of what I had learnt with him for my diploma in computer science back in 1995. Special thanks go also to Fridolin Klostermeier who shares my Xenakis frenziness since its very beginnings 20 years ago, and to his co-CEO, Christian Küttner at Intranetz GmbH, Berlin. Very special thanks go to my parents whose continuing support allowed me to indulge for a long while in such unprofitable activities like studying musicology and writing scientific papers.

A final word about the presentation of this text. It seems a bit strange for an author to write in a language which is not his own mother tongue. There is a danger of offending native speakers with a stylistic concoction of incompatible linguistic backgrounds, e.g. a mixture of German, English or American “cultures of writing” (not to mention the problem of English vs. American spelling). I hope that this will not obscure too much of what I would like to convey in this text. The aspect of communication is in fact the strongest argument for writing this text in English. Those interested in theorizing on Xenakis’ computer music are comparatively few and scattered all over the world. Writing in English also avoids much awkwardness in either translating or assimilating English technical terms, especially in computer science, into other languages. Another advantage of expressing thoughts in a foreign language might be the fact that it discourages the mannerisms of a technical (here: musicological) jargon in favor of a more common language which might prove beneficial for an interdisciplinary study like this one.

Quotations from French and German sources have been translated into English with best personal effort (i.e. by me, unless otherwise stated). It is implicitly understood that references to humans always apply to both sexes, i.e. “composer” always implies “female composer” etc.

This document has been typeset using the e- $\TeX$  version of Leslie Lamport’s  $\LaTeX$  macro language (based on Donald E. Knuth’s  $\TeX$ ) in its  $\text{MiK}\TeX$  implementation (version 2.4) by  $\text{MiK}\TeX$ .org using  $\text{TeXnicCenter}$  ( $\text{TeXnicCenter.org}$ ) as a frontend.  $\LaTeX$  figure code was created using the  $\LaTeX\text{cad}$  package, v1.9 by John Leis. Music examples have been created using the  $\text{Music}\TeX$  package, v5.30 by Daniel Taupin. Citations and bibliography have been prepared with  $\text{Bib}\TeX$ , following the standard of van Leunen, Mary-Claire: *A Handbook for Scholars*, Alfred A. Knopf, New York, 1979, as indicated in the documentation of [Lam86].

Peter Hoffmann



# Contents

<b>I</b>	<b>Introduction</b>	<b>19</b>
1	Computer Music	23
2	Art and Technology	35
3	Xenakis' electroacoustic Project	47
4	The Paris electroacoustic Scene	59
5	Algorithmic Composition	67
6	Rationalizing Culture?	93
7	Creativity, Communication, Originality	103
<b>II</b>	<b>Theory</b>	<b>113</b>
8	Models of Sound	117
9	Algorithms and Computer Programs	133
10	Computation and Computability	155
11	Randomness and Complexity	169
12	Aspects of Software Engineering	187
<b>III</b>	<b>Analysis</b>	<b>193</b>
13	Computer Music Analysis	197
14	Analysis of <i>GENDY3</i>	209
15	Description of <i>GENDY3</i>	261
16	Future Work	291
A	Stochastic Distributions Used in <i>GENDY3</i>	315

**B Code Listing and Manual**

**Part I**

**Introduction**



This part is called “Introduction”. In fact, it is more than just an introduction. This part is to set up the framework necessary to discuss Xenakis’ latest and last contribution to computer music. This work on computer composition, culminating in the generation of two pieces, *GENDY3* (1991) and *S709* (1994) is actually more than just the creation of these two pieces. This is because GENDYN, the computer program behind these two pieces, represents an artistic project in its own right. Xenakis’ approach to computer music composition, as expressed in this project, is a radical one. In order to do it justice, I feel compelled to frame its discussion by its historical, aesthetic and technological context. Therefore, the idea of this introduction is to attack the aesthetic problems, technological questions, and philosophical ramifications provoked by Xenakis’ work on GENDYN right at the beginning of this study. This will clear the way for the discussion of GENDYN in general and *GENDY3* in particular. In this sense, this introductory part serves as a foundation to either the theoretical and the analytical discussions to follow in parts II and III.



# Chapter 1

## Computer Music

“Most composers and musicians have of course encountered the computer by now. In electroacoustic music, for instance, the digital computer has replaced all previously existing sound synthesis machinery. [...] It is equally helpful for the conservative composer who writes pieces for orchestras. 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.” [pHH96], p. 152

What is “computer music”? Is it any music where the computer is involved to execute specific tasks? This could be the composition of musical scores, sound synthesis, sound transformation, interactive performance systems, but also sound engineering, sound design, the programming of computer games, computer typesetting of instrumental parts, digital restoration of historic recordings, and the generation of automatic accompaniment to melodies for the film industry. Since the computer is a multi-purpose machine, virtually anything in music can come under the heading of “computer music”.

In this sense, the term “computer music” would be applicable to any music in the creation of which the computer is involved in some way. We see at once that this definition is odd. Is the present text a “computer text” only because I use the computer to type it? 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? With the same reason, one could argue that all music today is electroacoustic music because it is recorded, broadcast, stored and played back with the help of audio engineering technology. By the same token, with the ongoing computerization of electroacoustic equipment, all music will have to be classed as “computer music” if we do not want to restrict ourselves to acoustic music-making within the four walls of our living-room.

In this perspective, sound computation by a computing device would be no more than part of a general tendency toward mechanization and automation of the music production and consumption processes, in the sense established above. Given this consideration, the use of the computer alone as a universal tool for sound recording, processing and musical performance in the modern studio, concert hall or even in the private home cannot be the sole criterion for music to be “computer music” in the rigorous sense to be established in the framework

of this study. What I would like to call “computer music” must entail some sort of a conceptional attitude toward sonic creation with the computer, some reflection of one kind or other about the conditions and the modes of artistic creation with the help of machines. Indeed, we are looking for an attitude that addresses some of the following questions concerning the use of computers in art.

- When the structure of a composition or even its sounds are generated by a computer, where is the compositional craftsmanship to be located? With the programmer or the machine? Shouldn’t the copyright, then, be transferred to the machine?
- What is the difference between the interpretation of a classical piece by a musician and its simulation by a computer?
- Does art in general and music in particular finally become the slave of machines?

If a computer program is used in the creation of music, it should be somehow noticeable to the listener. This can go very far. In its extreme form, the music will be equated with the audification of the program action ([Kra94], or the sonification of digital data ([SC91], [Nez00])). For a deeper discussion, see section 2.3 on page 41.

Therefore, we want to establish a definition of computer music for the scope of this study which stresses the algorithmic nature of computer composition. It is evident that this definition is in stark contrast to the widespread and often unreflected use of computers in making music.

**Definition 1.1: Computer Music.** Art in general and music in particular is to be called “computer art” resp. “computer music” if there is a specifically computational aspect in their creation.

## 1.1 Two “Cultures” of Computer Music

Should this introduction contain any general survey of music produced with the help of computers, given the fact that we have restricted the scope of this study to a very narrow definition of computer music, i.e. algorithmic composition, in its computational sense? Is there any use trying to establish a “research context” to Xenakis’ activities in view of its being completely irrelevant to his singular approach? One reason could be that Xenakis’ GENDYN project will distinguish itself before the background of other approaches to computer composition. The other reason is that there will certainly be some interesting information gained from contemporary activities to the ones that Xenakis undertook. Even if there does not seem to exist any mutual influence, a rough picture of computer music research will reveal the uniqueness and the strangeness of Xenakis’ approach to computer composition, as evidenced in his late GENDYN composition program.

When examining the practice and aesthetics of computer composition in a wider context, we find antagonisms and contradictions that I would like to describe as the effect of “two cultures” of computer music. The “first culture” would be defined by its endeavor to emulate the human by the machine, to hide

the ugly wires of computer technology behind a discreet curtain of humanizing interfaces. The second one, in contrast, would be characterized by its trying to find out what computers can genuinely contribute to art and what is beyond standard human conception. This second approach strives to demonstrate the specificity of computers and computation in the domain of the arts. Interestingly enough, the second culture, although it stresses the importance of computer technology much more than the first one, seems to be somewhat less present in the public awareness of computer music, and even within the computer music community itself.

The aesthetics of these two cultures pertain to their sonic ideal: within the “first” culture, the computer is used to produce “natural” and “beautiful” sound, i.e. sound that does not step too much off traditional musical habits. In contrast, composers whom I regard as representatives of the second culture deal with computer composition in a more fundamental, radical, conceptual way, without caring too much to create “musically pleasing” sound.<sup>1</sup>

Let us, in order to give the dog a name, call the first culture the “disguised” and the second the “explicit” one.

**Definition 1.2: Disguised Computer Music.** This majority trend in computer music strives at emulating human music making by computers, e.g. by using Artificial Intelligence, Expert Systems, Neural Networks, Psychoacoustics, and Cognitive Sciences. These people want the machines to do what humans do. Humans are supposed to appreciate the machine’s artifacts within their inherited cultural framework.

**Definition 1.3: Explicit Computer Music.** This attitude, put forward by some computer music composers aims at creating music which is specific to machines, stressing the computational aspect in its composition, by using rigorous formalisms, machine sounds which have no equivalent in Nature, and by conceptualizing and problematizing the use of computers in music.

Without wanting to impute underground links and influences, I would like to associate Xenakis with the small number of composers I regard as proponents of the “explicit culture” of computer music. Xenakis shares the conceptual strength of such outstanding composers like e.g. Herbert Brün (1918-2000) and Gottfried Michael Koenig (b. 1926), like him pioneers in computer music, who, with their groundbreaking aesthetic reflections, laid the foundation of a true computer art.

When I started the GENDYN project I often found myself “defending” it against the expectations of many people interested in computer music, because I had not yet well understood its explicit concepts. GENDYN is different from most electroacoustic music which sounds so much “nicer”, more integrated into the “natural” world of instrumental sounds, more “like music”, if I dare to say. I would tell people: “Well, Xenakis has this idea of a brute, raw, unprocessed sound, starting and ending abruptly, and if you are not used to it, you may find it a bit strange.”

Today, after some time and reading, I have stopped feeling the need to match Xenakis’ computer music against the standards of the disguised computer music

---

<sup>1</sup>This term has been used by [Smi91], p.11, to cite only one of the most distinguished proponents of the “disguised culture”.

culture. Certainly, the latter meets the expectations of a classically trained concert public better, but only at the cost of perpetuating the classical instrumental paradigm, embellishing it here and there with computer-enabled additives, and serving the product as a “human-like”, technologically enhanced art. Against this practice, I would like to put a challenge: Why should computers sound “natural” and “beautiful”? What sense is there in training computers to extrapolate musical styles and ideas that have been developed by humans without giving any thought to computation?

The argument that will be put forward in this study is that “true” computer music must be music which takes the technical means of its production into consideration, which uses the computational strength of computers (which is enormous but qualitatively limited) not in order to extrapolate existing concepts but to put forward ideas that only the computer can realize. It is clear that such assertions make the present text much more controversial, but as it is hoped, also much more interesting.

## 1.2 Using Computers for Music Composition

There is an ever increasing number of composers in the world who use the computer in order to calculate parts or sketches of their compositions based on computational models. Some composers use the computer for interactive “prototyping” of compositional procedures in view of the unimaginable possibilities of composition today. I will cite no names: almost every young composer today has experience in computing, and some of the music conservatories even oblige their students to work in a computer music studio.

There are many systems for computer assisted composition, either commercial, freeware, or self-made. Most of them provide the means to turn sketches and scores into sound. There are universal tool boxes for Computer Aided Composition (CAC) like IRCAMs OpenMusic [Ago98], and sophisticated programming environments with large libraries for music composition like Csound [Bou99], Common Music [Tau91], SuperCollider [McC96] and many, many others (for a bulky, but not exhaustive survey, see [Loy89]). Many systems which started as control languages for hardware synthesizing units are by now, with the increasing power of personal computers, pure software: PureData [Puc96] and MSP [Dob97] are now replacing MAX [PZ90] which was originally developed to control IRCAMs 4X hardware synthesizer. Even Xenakis’ UPIC system, once an expensive hardware setup with many components, is now available as a small software program.

Due to the universality of computing, there are many individual attitudes to computer music composition resulting in ever new approaches to computer generated music, much in contradiction to the general unifying tendencies that generally prevail in the technological domain. Listings of systems and software can be found e.g. in [Mir98].

Examples for more specialized music programming have been the exploration of complex dynamic systems for the generation of musical structures ([Lit93], [Bey91]), genetic programming exploited for musical purposes ([Bey97], [Jac95]), grammars for the generation of musical structures [Roa85a] or, more specifically, Lindenmayr systems [MS94], the tracing of stochastic processes in music [Jon81], and every year, new methods and procedures are presented at international con-

ferences, the best known of which is ICMC (the International Computer Music Conference), organized by the ICMA, the International Computer Music Association. Any larger town in Europe, the United States, Canada, Latin America, Australia, and some other countries have electroacoustic studios with a specific history and aesthetics of their own, and with the advent of computer technology can be said to produce some kind or other of “computer music”.

In addition, there is an increasing number of individuals who do not feel the need to associate with computer music studios or larger institutions since music can be computed on any home computer. Only the work of a very small fraction of those involved in computer music is covered by articles in scientific journals or presented at congresses. The World Wide Web is in some way helpful to find out what is going on in the music community. But even the Web is a filter.

### 1.3 Xenakis and the Computer Music Community

Iannis Xenakis is to be regarded, together with US-American Lejaren Hiller and French Pierre Barbaud, as one of the pioneers in computer music, by virtue of his stochastic composition program, conceived from 1958–1962 (see chapter 5 on page 67). However, it would have been difficult to class him as a member of the international computer music community. With his small research institute CEMAMu in Paris, he had little influence on generations of young composers as had the bigger institutions, like e.g. IRCAM with a large user group, regular competitions, prestigious commissions, pedagogical activities and music academies.

Conversely, Xenakis seemed to be virtually immune to influences from current trends in computer music. His ideas on the use of the computer in music were so fundamental that they even did not have to change much since he started to think about computers by the mid-1950s. Although his name figured sometimes in lists of editorial boards or members of academic institutions he did not seem to make much use of it, nor did he participate in interest groups or multinational research projects. Ignorant of the intense research taking place all over the world in computer music, he followed the lines of his personal interest in the use of computers as formed in his artistic mind. This is not to say that his latest achievement in computer generated music, dating from 1991, should be regarded as “outdated” in any sense. Rather, his work seems to be “timeless”, although this might seem a bold statement in the context of music computing. However, viewed against the turmoils of a juvenile computer music history, Xenakis’ contribution shows a surprising steadiness. This is partly due, as we shall see later in this study, to his awareness of the principles of computation, its strength and limitations, that have not changed since electronic computers came into being around the time of World War II.

The work of Iannis Xenakis in the field of computer music is based on theoretical foundations of mathematics and computer science. Xenakis was more conscious about these than most of his composer colleagues. Only a few composers shared his concern toward the philosophical and mathematical foundation of their artistic practice. One of them was the philosopher-composer John Myhill [Myh52] who had great influence in the Urbana “school” of computer

composition. Another was Herbert Brün who was also active in Urbana since 1963.

## 1.4 The Issue of Computation

The question seems to be how and to what purpose the computer is used in creating works of art. According to definition 1 on page 24 above, where “a specifically computational aspect” must be involved, we shall not deal in this study with simulations or emulations of techniques of different technological background. Rather, we are interested in the creative act designed for and aimed at the use of the computer and which could practically not be realized without it. “Computer Music”, in this sense, stands for something that could not be enacted without the specific computational nature of computing machinery. This is a rather narrow definition, and it implies nothing less than an elaborated “computer aesthetic” associated with the use of the computer for the creation of a work of art. Such an aesthetic, conformal to the “digital medium”, has indeed been formulated by outstanding artists in computer music, as we will see in the following.

Definition 1 on page 24 immediately raises a question: “What is computation?” This question will be answered in depth in part II, but here is a provisional definition.

**Definition 1.4: Computation.** Computation is the mechanical application of rules to data. “Data” are symbolic representation of information. “Mechanical” means that the rules are laid out beforehand, and the decision when to apply what rule is also rule-based.

According to this definition, it is irrelevant if the computation is carried out manually by a human, by a physical environment, or by an electronic computer. The information can be represented by physical or immaterial patterns of data: text, planet constellations, bit patterns, or ideas in a composer’s head.

The definition stated above sounds quite abstract and wide-ranging, but is in fact a very strict definition. It means no less than: the entire procedure must be completely formalized, the rules exactly specified and the data meticulously described in order for a mechanical process to take place. Everything that calls for intuition, understanding or insight into the procedure being executed must be excluded in order not to violate its purely mechanical nature. We call this strict formalization of a computational procedure an algorithm. There is no place for intelligence within an algorithm. However, a lot of intelligence is needed to construct such an algorithm.

A definition of computer music exclusively in terms of its computational aspect is indeed a strict one. Yet, it is the only one appropriate to serve as a foundation for the discussion of Xenakis’ latest work in computer composition. Only if the computational aspect of algorithmic composition is singled out in this clear and distinct sense can the specific contribution of Xenakis’ latest activities to the issue of machine-generated music be properly assessed.

The consequence is that any aesthetic assessment of this kind of art must consider the computational aspect. With computers involved, one simply cannot make the same music with the same “pre-industrial” mind, and the same “autonomous” aesthetics as before. The idea of autonomy of art from the technique

of its making is a fiction. Even more so, if a work of art is entirely generated by a computational procedure, as is the case with GENDYN by Iannis Xenakis, the result is entirely determined by computation. GENDYN music is a computable music, in the strong mathematical sense. The compositional procedure is fully represented by the algorithm. This algorithmic nature of a GENDYN composition becomes an integral part of the musical experience. GENDYN music is explicit computer music.

## 1.5 The Relevance of Technology

Kristine Burns [Bur94a] holds that Algorithmic Composition does not necessarily have to be realized with a computer. This is certainly true: any automaton can be made to realize rule-based composition, even physical systems can be considered as computing their own evolution (as evidenced by cellular automata, see part II of this study). A nice example even of a stochastic composition without computer is Marcel Duchamp's mechanical composition system (around 1913) with which he created a piece *The bride stripped bare by her bachelors, even. Erratum musical*, where little balls are randomly caught by a toy train passing under a bridge (cf. e.g. [Hoc96]). However, for our purposes, I would like to restrict the notion of Algorithmic Composition to the computer for two reasons. First, the computer is capable of universal computation. This supports a general programming approach which is compatible with any computer (in particular, the computer on which Xenakis made the music and the computer on which I studied it). Second, a computer algorithm is self-contained: it does not depend on the environment as does an installation, nor does it depend on human interaction as does e.g. collective composing. I do not agree with [Bur94a] who describes these examples as algorithmic composition.

When it comes to algorithmic art, the term "technology" must be understood in a much broader sense than just programming languages or computing hardware. It is true that in practice the execution of composition algorithms, especially when they extend to the composition of sound as does GENDYN would not be possible without computer technology. However, it is the specific algorithm devised by the composer which is of major importance for the artistic result. It is the algorithm that makes the machine behave in a way so as to produce a work of art. Algorithms exist independently of computing machinery. They describe the manipulation of symbols. Only if the symbolic data are to materialize as a physical phenomenon, e.g. as electric pulses or sound waves, do we need the help of a machine. The machine, then, executes the algorithm within physical time and creates sound data as a result which are converted to acoustic sound with the help of a digital-to-analogue converter.

Algorithms have to be consciously and laboriously designed by humans. They cannot in general be machine-generated, unless the generation process is algorithmic in itself.<sup>2</sup> In a second step, the algorithm is implemented as a computer program targeted to a specific computing environment (programming environment and/or hardware).

At the same time, not only do humans implement their thinking in algorithms, but the specific condition of working with the computer also affects

---

<sup>2</sup>This is the case with e.g. Genetic Programming: a human-designed algorithm controls the evolution process of a "population" of computer programs.

the way humans think when working with the computer. Therefore, composers working with the computer must be aware that their attitude toward composition is conditioned by the specific nature of machine computation.

The term “technology” in the domain of computer music production is therefore a shorthand for a complex web of aspects and interdependencies resulting from the interaction of humans and machines and encompassing a whole field reaching from the “technical” to the “artistic”. Composer and researcher Agostino Di Scipio defines music technology in this sense as

“[...] a complex of design activities that crystallize around techniques, tools, practices, shared conceptual conventions and representations, constituting the musician’s working environment”. [DiS97]

To that end, the constructive processes have to be studied in order to be matched against the perceptive processes. Algorithmic composition is therefore a subject that includes the question about the process of production and its technical and conceptual conditions.

“electroacoustic music emphasizes [...] that the work of art is always created by creating the technique of its making [...] Ultimately, models and representations in one’s working environment reflect the way in which a composer explores him/herself and affirms a personal view of the actual world (while raising visions of a different, possible world). Clearly, then, for an aesthetic perspective of an technologically based music, these models and representations are at least as relevant as the receptively intelligible events which eventually emerge from the composed structure. This process questions previously established music theory (personal or shared by many) and requires (from the composer as well as from the analyst and the listener) a continual revision of musical concepts, the leaving of all presumed ontological foundations of music.” [DiS95a], pp. 375–377

Applied to the case of GENDYN, the fact that a musical work like *GENDY3* is algorithmically created becomes an integral part of its aesthetic substance. This piece, a purely algorithmic composition, does indeed present itself as an ideal subject of a study on machine generated music. It represents an extreme case where compositional thought is entirely formalized and represented in the form of a computer program. In this case, we can replace the word “technique” by “algorithm” and reformulate the above quotation as follows.

**Definition 1.5: Rigorous Algorithmic Composition (RAC)** The work of art is created by creating the *algorithm* of its making.

There is much debate about why and how Xenakis applies mathematical theories and algorithmic models in his creations. Some regard his collected writings in “Formalized Music” as a sort of cookbook for composition. I think this is an over-simplification. While in composition, Xenakis sometimes exhibits a crude pragmatism, there is more to his project of “Art/Science” than just inventing new composition techniques, as I have tried to show elsewhere ([Hof94a]; see also [Eic94]). His projects of “Symbolic Music”, “Metamusic”, etc. have a *raison d’être* of their own, quite apart from their acoustic results, which by the way

are, as for themselves, more than just exemplifications of composition theories. I think that the scientific and intellectual background of Xenakis' music must be seen as an integral part of his work, in the sense of a "concept-art".

## 1.6 Three Technologies of Music: Writing, Editing, Computing

We have seen that artistic production in general and algorithmic composition in particular are closely interrelated with the technology used. With this in mind, let us now focus on three generic modes of musical production each of which can be associated to a specific kind of technology. The first one is the "classical" production of writing music scores. A second, and more recent one, is working with a sound recording medium. The third, even more recent, is music programming with the computer (cf. [Eck98]). These three approaches are quite different, and, according to the mutual interdependency of technology and art, yield different qualities of artistic output.

### 1.6.1 The Technology of Writing

The "technology of writing" has its roots in the ninth century of occidental Europe when Gregorian Chant was codified into written form. As a side effect of this codification, the notion of a musical text developed slowly over the centuries and found its hypostasis in Listenius' dictum of an "opus perfectum et absolutum". This was the idea that the written document of a musical oeuvre has a permanent value beyond its sonic realization, which may be deficient, and even beyond the life of its author. We shall not recapitulate here the legends and facts about the occidental culture of writing but merely point out some consequences here. (For a historical account, see any good book on music history, e.g. [Egg91]).

The symbolic notation of a timely phenomenon like music makes it indeed amenable to manipulations that are simply not conceivable within the flow of time. It is evident that counterpointal techniques like retrogradation, inversion etc. would probably never have been developed without musical literacy. It has also been argued that musical literacy is responsible for the concept of progress within occidental music history. In any case, it is obvious that the fixation of a musical text provokes transgression and variance. This is very different from oral tradition, where a core model is faithfully preserved over generations who at the same time have much freedom in making it come to life.

One immediate consequence of the "technology of writing" is the separation of composition and interpretation. The musical text is an "incomplete symbolic representation of the music" in view of an interpretation (which may also be reading). The interpretation, then, is the "completion [of the text] by relating it to the current cultural conventions of interpretation [and] translate it into an in-time acoustical representation, i.e. into sound" ([Eck98], p. 45).

A very good and informed account on the technology of writing gives [Lév03]. He stresses the importance of symbolic abstraction and grammatological approach to musical phenomena, as a prerequisite to make them accessible to musical thought.

Xenakis has created the overwhelming majority of his works by writing notes on staves, using the technology of writing. However, many of these scores are actually transcriptions of graphic and/or algebraic sketches into traditional notation in view of an execution by orchestra or instrumental ensemble. In cases where these graphs contain the whole of the music, as e.g. *Evryali* (1971) for piano solo, the essence of the work is captured by the graphics rather than by the score transcription. Therefore, Xenakis' musical thinking was not so much conditioned by symbolic musical scripture as it was by architectural and algebraic construction.

### 1.6.2 The Technology of Editing

“After the invention of machinery for direct reproduction of sound, why still pass through writing a score? [...] Why still pass through the ‘note’ as the foundation of a work while access to the sound itself and all its qualities has become possible?” [Gor96]

The “technology of editing” is not much older than half a century (see [Dav96]). It came into being after Second World War II, mainly with the activities of Pierre Schaeffer at Radio France, the Elektronisches Tonstudio at Westdeutscher Rundfunk (then Nordwestdeutscher Rundfunk) and the pioneers of Tape Music at universities in the US who started to use standard sound engineering machinery for creative purposes. (Note however, that John Cage in the US worked with turntables and radio mixers already from the late Thirties, see e.g. the cycle of *Imaginary Landscape* 1939-1952.) For the first time, the technique of storing sound on a recording medium (turntables, later tape machines) was creatively (ab)used to (re)assemble, i.e. “edit” sounds in time. The effect was that the sounds took on a completely new sense within a context radically different from the original context they had been taken from.

In its variant of Pierre Schaeffer's *musique concrète*, the editing of sounds on a recording medium meant a radical challenge to the technology of writing. Schaeffer wanted to put music from its head back to its feed, working with the concrete sound phenomenon, instead of manipulating abstract symbols on paper.

“As a matter of fact, two outstanding modes of musical production, known under the headings of *musique concrète* and *Elektronische Musik*, were born almost at the same time [...] They more than opposed traditional notation: they simply left it behind. The first one had to do without notation in view of a sonic material that was too varied and complex to be transcribed. The second made notation look anachronistic through its rigorosity so total that traditional music scores grew pale in view of its precision.” [Sch66], p. 17.

It has been commented that the working with recorded sound means manipulating a temporal phenomenon (sound and music) in space (e.g. the groove of a turntable disk or a piece of magnetic tape). Just to give a very simple, but speaking example: In some of his early rhythmical studies with a tape recorder, Xenakis designed spatial patterns onto a magnetic tape and pressed the record button on each of them, creating a pattern of acoustic clicks which gave him a feeling for durations and their proportion in music [Var96].

Indeed, a recording on tape is a spatial representation (a magnetization pattern along the tape strip analogous to the sound amplitude) of a temporal acoustic event. This allows the possibility of working on temporal aspects of sound and music “outside time”, much like a music score, but on the sound itself instead of its symbolic representation. It permits the inclusion of all kinds of sounds, especially noises which are impossible to grasp by symbolic scripture because they are too complex to be abstracted to one-dimensional parametric values like e.g. pitch and duration. Moreover, sound is stretched out on a tape in the sense that a small piece of a tape represents a microscopic part of recorded sound. Therefore, the tape (or a sound file) offers the possibility to deconstruct and recompose the micro-structure of sound by splicing, looping, changing record or playback speed or direction. Even seemingly “modern” sound manipulation effects were already achieved with tape machinery, e.g. pitch-shifting and time-stretching, by the so-called “Springer Machine” ([Spr57]) and Pierre Schaeffer’s *Phonogène* from 1953 ([Dav96], p. 8). All of these techniques of *musique concrète* are today done with the computer. But regardless if done on the computer or on tape, the technology of editing penetrates into the micro-structure of sound, opening radical new ways of “composing-the-sound” as opposed to just “composing with sounds”. For an insider account on the technology of editing, see e.g. [Del02].

Xenakis worked in Schaeffer’s studio from 1954 to 1963. He actively took part in the technology of editing, creating 5 electroacoustic pieces during that period (see chapter 3 on page 47). But at the same time, Xenakis was a dissident to Schaeffer’s ideology of “*musique concrète*”. He conceived differently of a music artwork than Schaeffer, in a more conceptual and planned way which transgressed Schaeffer’s phenomenology of hearing which was intimately coupled with his personal distaste for written music. When Xenakis promoted the use of computers within Schaeffer’s group, he found no response and left the studio (cf. [Sol96]).

### 1.6.3 The Technology of Computing

The “technology of computing” is radically different from the first two technologies of composition by the fact that it reaches far beyond the analogy of spatio-temporal dimensions. A computer program has no aspects that “show” the temporal extension of the music. It is neither thick like a score nor is it long like a tape. In order to tell how much and what kind of music a computer program may produce, one has to understand how it works. In some cases there is even no way of telling anything about the program’s behavior and result from a look at the program text. In that case, the only way to find out is running the program. (We will discuss these issues of computation in part II of this study.) On the other hand, quantity and quality of the music output can be much different on each program run, depending on input data.

These aspects of “programmed” sound and music, its extreme openness on the one hand, and its implicitness on the other, are the most difficult and at the same time the most fascinating features of algorithmic computer music. The generation of musical sound by a computer program is very much implicit because its (digital) representation may come about as a result of a computation of any degree of complexity. If the structural complexity of a program is such that the result of a computation cannot be predicted beforehand, then there

is no way of knowing in advance how the result will sound. We will see later how this fact relates to chaotic systems and the specific quality of the artistic work realized with such systems on the computer. Here I shall only drop some buzzwords: “Explorative and cooperative attitude to composition”, “interactive modes of composition”, “improvising with the machine”, the principle of “Sonological Emergence” (see 5.7.5 on page 76).

The computer is a universal and versatile tool, so it can just as well emulate techniques related to the two foregoing technologies of writing and editing. Editing is nowadays almost exclusively done on the computer. Computer assisted composition (or CAC for short), in most cases, is nothing else than an extrapolation of the “technology of writing” with the computer (see, however, [Ass98], [Ass93] for a computationally extended use of CAC systems). Yet our interest here will not be the simulation of foregoing cultural technologies by a new one, but the implications and ramifications that this new technology specifically brings about, and which are picked up and made explicit by artists who are conscious of the specificities of the technology they use.

“The artist’s work becomes a metaphor for a cultural view in which technology is less a powerful instrument to actualize dreamt-of visions than something to excite new, autonomous visions. It is less to answer than to challenge, in sharp contrast with the advertised technological industrial products presumed to augment the consumer’s comfort by empowering his/her well consolidated, habitual action, by satisfying his/her needs. [...] A revolutionary cultural paradigm shift [lies] behind the new sonic art. By fostering a reconsideration of the artist’s *téchne*, electroacoustic art shows that the realm of the *artificial* is the realm of the *humanly devised*. Ideally, the composer of electroacoustic music becomes fully responsible for the conventions, representations, structural possibilities and constraints in the working environment. [...] The whole structure of the musical work, at many scales of time, becomes thoroughly *artifact*. [...] — in fact, *τεχνημα* — fully and peculiarly human, *entirely cultural*, rid of traits as yet independent of an individual’s acts of intent” [DiS95a], p. 375/376, emphasis original.

Xenakis adhered to the technology of computing as one of the first composers in the world, shortly after Lejaren Hiller in the US (around 1955) and with Pierre Barbaud in France (around 1961), by virtue of his early Stochastic composition program (1958-62). As a preliminary, automated score composition is mentioned for the first time by Xenakis as early as 1957 [Xen57]. Around the same time, too, Xenakis put forward his idea of algorithmic sound synthesis (cf. [Var96]). Xenakis is also known to have preconceived Granular Sound Synthesis [LR81]. However, in spite of intense research over many years, Xenakis realized his dream of creating an entire music by the execution of a computer program only in 1991 with GENDYN.

## Chapter 2

# Art and Technology

“Just as machines took over physical work from people in the Industrial Revolution, now machines are starting to take over intellectual work. [... This] will free human beings to discover the essence of what it means to be human.” [Bee88], p. 214

To describe the work I have done during two years of research, and later during years of spare-time activity, I would have to class it under the heading of “computer music”. Computer music is a tiny, esoteric niche somewhere in the reservation of “contemporary music” which itself is a tiny niche somewhere in the reservation of “music”. And “music” is, in turn, a tiny niche in our daily life. I shall leave no doubt, however, that I consider music (as art in general) to be of major importance to our very existence, and I do not hereby mean its commercial or economic aspects. Art gives us the chance to come to terms with ourselves as being human beings. Art is a prominent discipline of human self-reflection and self-awareness. Algorithmic composition (not only the composition with the help of a computer, but composition *by* the computer) is an extreme, perhaps the most extreme form conceivable to practice art in this understanding. If computer music is a tiny niche in the reservation of New Music, algorithmic composition again is a tiny niche in the reservation of computer music. And within that niche, we are left with a unique concept of machine music thought of by a unique composer resulting in two unique works of art, *GENDY3* and *S709*.

If one conveys the creation of a work of art to a machine, in how far does this mean to reflect about the “human condition”? Well, it is because algorithmic composition is conceivably the most radical and most effective confrontation with the technological revolution of the second half of the 20th century, namely the development of computing machinery, in the realm of arts. Algorithmic music, as one of the most distinguished disciplines of an “Automated Art” is the stage where we are confronted with our condition as being subjected to the rule of machines in virtually all aspects of our life.

Using computers for the creation of art means to force them to devote part of their computing power to the reflection of humans about their human condition. Forcing machines to serve artistic goals much different from the purposes they have been built for amounts to gaining a temporary victory over uncompromising rationalization and automation in our environment. In using computers for art, we have a chance to actively reflect and possibly object to the fact that

technology determines our life.

We shall see that there is a strong contradiction between the use of computers for artistic purposes and the standard, engineering perspective. From an engineering standpoint, it does not seem very sensible for Xenakis to put in so much time realizing a composition program only to use it for two pieces (for the second piece, *S709*, Xenakis even considerably rewrote his program). What is the explanation for this seeming inefficiency? Well, the explanation is that art does not work that way. The idea of artistic originality which has prevailed for so long in occidental music history seems to be so strong as to affect even its antagonistic opposite, automation, once it is put to the service of art. This fact applies even more to a composer like Xenakis who promoted the very idea of originality throughout his artistic life (cf. [Xen94b]).

In engineering, and especially in computer science, a problem algorithmically solved is said to be solved once and for all. Quite on the contrary, solving a compositional problem in one piece raises a dozen more problems to be solved in the next compositions to come (cf. [Dah74]). To give an example in the context of GENDYN: GENDYN sound synthesis of time-varying spectra solved the problem of frozen spectra produced by the UPIC system. However, Xenakis still found it a problem that sounds in *GENDY3* tended to be stationary. Therefore, he tried to alter the parameters of GENDYN synthesis over time by additional programming. The result was the composition *S709* which is very different from *GENDY3*. If Xenakis had continued research on GENDYN, he certainly would have constructed yet another version of the program and created a different piece with it.

## 2.1 “Heretical” Use of Technology

The programming language which Xenakis used for GENDYN is BASIC, an interpreted low-level coding scheme used all over the world, from the programming of pocket calculators up to scientific computing. It has been widely spread with the IBM-PC computer and many pupils have been confronted with it at school. Later, Microsoft extended BASIC for scripting and application-level programming within their Windows operating system (and called it Visual BASIC resp. Visual Basic for Applications).

How could a highly individual piece of music as *GENDY3* ever emerge from a sequence of unspecific, common statements of an everyday programming language? Since in *GENDY3* there is no human input other than the program text itself, the artistic quality of the music must somehow emerge from the set of these anonymous statements.

What an armada of programmers all over the world use for bookkeeping, salary accounting, or the calculation of interest rates, Xenakis used to calculate music. Under his hands, seemingly neutral programming transforms into a highly individual specification of a music composition algorithm. The standard concept of algorithm, namely an automated procedure to solve a given routine problem, is creatively transformed to generate a unique, individual work of art. Therefore we can say that in the case of Xenakis, programming is used in a “heretical” way, as opposed to its “orthodox” use in the world of business, the cradle of Personal Computer technology. With the term “heretical”, I refer to Agostino DiScipio’s discussion of the relation between art and technology in electroacoustic music

([DiS97]) which has already been the scope of section 1.5 on page 29.

In this section I would like to show that heretical (ab)use of technology is not just an attitude of artists who are not capable of making any “skillful” use of standard engineering techniques. Quite on the contrary, I would like to demonstrate that the alternate or even “improper” use of technology is fundamental to the very role of arts. Let us start with a statement about the origins of electroacoustic music.

“In the 1940s and 1950s *Elektronische Musik* and *Musique concrète* were born by an unprecedented re-interpretation of technical instruments which were solely meant for scientific measurements and control. In that case, means of reproduction, control and storage were bent to a form of creative production - of *poiesis* - which was completely alien to their original technical code.” [DiS97]

We clearly see by this statement that heresy is the original sin of electroacoustic music. The impact of art upon technology, however, is not a one-sided process. The conquest of technology by art has also in turn affected artistic creation as “technological”. Technology questioned seemingly self-evident and well-established conventions in art and urged for finding novel categories. For example, well-established concepts like “part-writing”, “instrumentation”, and “interpretation”, centered around the technology of writing musical scores, become either obsolete or take on a completely different meaning when technologies of editing and computing are employed where sound is created by means other than playing instruments. Traditional concepts of harmony and counterpoint, which are based on the paradigm of a traditional score focussing on pitch and duration, have thus been called into question.

“Electronic music involves no performance [...], hence no need for common tonal or metrical systems. Melody, harmony, and counterpoint therefore lose their meaning, becoming historically and in a functional sense wrong.” [Koe83], p. 28

In view of the explicit computer music culture defined above, putting computer technology to the service of traditional academic music composition must be considered a misconception. We now know why: It would mean to confuse the technology of writing with the technology of computing.

“A computer program which allows for writing music in common music notation reflects and supports the hegemonic opinion that composing is writing; however, this can be [...] a questionable view.” [DiS97]

In the heretical perspective established above, computer art is not the ultimate triumph of technology over human self-awareness. Quite on the contrary, the accessibility of the atomic structure of vision and sound offered by technology is a challenge to implement human imagination in every aspect of an artwork, down to its atomic matter. The fact that art is invaded by technology in the field of computer music is a chance rather than a flaw, but only if art takes up this challenge creating tools and techniques of its own logic and not the logic of standard engineering.

An example from my work on GENDYN may serve to illustrate this statement. Xenakis calculated GENDYN sound samples on a numeric interval of integer numbers between -100 and +100 only multiplied later by a constant factor to yield sample values within the standard 16 bit CD sound quality resolution (integer numbers between -32768 and +32767). The effect was that the computed sound signals became step functions with an effective resolution of less than 7 bit<sup>1</sup>. In my first implementation of the New GENDYN Program, I wanted to do better than him. I decided to compute the sound within the full 16 bit quantization space right away. From an engineering point of view, my implementation was certainly superior to his. Yet, the resulting sound was dull and uninteresting. It was only when I changed my implementation to Xenakis', inferior from an engineering perspective, that I was able to produce the captivating GENDYN sound. I had to understand that the quantization noise of 25.15 dB<sup>2</sup> induced by the arbitrary restriction to the number interval between -100 and +100, was an integral part of GENDYN sound.

This example shows that Xenakis, consciously or unconsciously, had made the very fact of working in the domain of digital sound synthesis a distinctive feature of his late activities in computer music creation. GENDYN sounds are digital artifacts impossible to obtain with physical vibrating bodies. Consequently, GENDYN sounds cannot, by definition, be an imitation of "natural" sound. In this sense, Xenakis meets the categorical imperative of authenticity and explicitness in art which demands that the choice of means be appropriate to the intended artistic goal.

## 2.2 The Non-Standard Approach to Computer Composition

We have seen that the use of technology for artistic purposes involves and implies a certain potential of heresy. In early electronic music, tape music and *musique concrète* standard radio equipment was bent to serve creative musical purposes. The same applies for the creative use of computers by composers since computers were originally designed for scientific, military and business use (in that order). One can say that in all these cases, a given technology was used by composers in a non-standard way as opposed to standard engineering techniques. For example, playing back a tape recording with reduced or increased speed or in reverse direction is a non-standard technique considering the fact that tape machines have been designed to faithfully reproduce natural sound with high fidelity. As for computers, they were programmed by composers in wild ways, most often in blatant violation of the most common software engineering techniques. Contrary to business, scientific and military programming, the resulting programs were not even checked to work "correctly" as long as they produced interesting results for the composer.

However, science, business (and in some instances even the military) have before long reconquered the computer music domain. Research centers like IRCAM. (see chapter 4 on page 59) were set up where artists and researchers were

---

<sup>1</sup> $\log_2(100) = 6.64$ . Even worse, in *GENDY3*, Xenakis restricted the quantization interval of sound synthesis even further to about 4 bit quantization by setting the random walk barriers to values around  $\pm 20$  (see chapter 3.8 on page 56).

<sup>2</sup> $10 \log_{10}(\frac{32767}{100}) = 25.15$

supposed to collaborate. In times when funding gets difficult, business managers are hired in order to find industry partners and to commercialize research. As for the military use of computer music research, one can cite IRCAM's deal with the military-industrial complex in 1984, where the prestigious 4X synthesizer was sold to be used in an aircraft flight simulator ([Bor95], p. 110, 282). Another example is the interest of the Los Alamos Laboratories, famous for the development of the nuclear bomb, in the CARL music programming environment ([Bor95], p. 161). In view of these facts, I am happy to state that it would be very difficult to win a war with Xenakis' GENDYN program!

In modern computer music research and industry, standard engineering techniques are applied to music systems which are more or less commercialized, distributed and used by computer music studios and individuals for artistic creation. Yet the more general-purpose a music system, the more its ambition to serve a wider user group of different composers, the greater the danger that it impose itself on a whole generation of composers in a standardizing way. This danger is aggravated when the system is either expensive or difficult to handle, so that it will be only usable through the mediation of an assistant. The assistant will willingly or unwillingly imprint his or her personal preferences of using the system onto the compositions realized with her/his help. Any different, possibly "heretical" use of such a "standard" composition environment is thus prevented.

However, there have always been composers who use existing software in ways that were not taken into account by the designers of that software. In addition, the output of such systems can be used in a completely different way than originally conceived. In the same spirit, the input that is fed into such a system could be of a completely different nature than that "expected" by the system. In this way, computational data of standard music systems can be (re-) interpreted in various non-standard ways.

Another alternative to using standard software is forging one's own tools for artistic creation. This is, however, only possible when either programming is done by the composer, or when a system is flexible enough to be entirely (re-) configurable by the user. For this purpose, a viable conceptual approach is required, taking several artistic imperatives into account. In addition to the principle of explicitness defined above, one could name the principle of appropriateness of the chosen means for an artistic goal.

Therefore, a prerequisite for a creative use of technology is that artists invent new techniques of using it. Any application of established industrial and standard techniques would mean that the artist, instead of controlling technology, would him/herself be controlled by technology.

"Primarily I am very annoyed with composers using the most modern tools of music making, like electric music, voltage control, even computers, and making twelve-tone series for instance, or trying to imitate existing instruments. This has, of course, its scientific value, but not necessarily a creative value in new music making. [...] So just to be able to avoid that, to open up new fields of sounds you would not be able to produce or would not think of describing in classical terms, I have chosen [my] non-standard approach." [Roa85b] p. 573

The aesthetic consequence of this conceptual approach to computer music

is, indeed, that there are no “natural laws” of creation. Once the sound signal as such has become accessible to artistic design, the application of acoustical or instrumental models loses its privilege of being the dominant conceptual framework for composition. “Non-standard” use of computer technology means using the computer to rethink what music is about in the computer age and to accept artistic responsibility even for the tiniest details of sonic design with the machine.

“Models of standard synthesis are instances of known theories of sound; one utilizes them in his/her own model of musical design. In contrast, non-standard models instantiate a possible theory of sound; one explores them, and learns how they can mediate the sonic structure.” [DiS94], section 2.2.

The core idea of this approach is to bypass any known model of sound in terms of physics, acoustics or music theory and to generate and directly manipulate the physical sound. Since the representation of sound in the computer is in the digital form of a sequence of sound pressure (amplitude) sample values, i.e. numbers, the problem of creating a piece of music translates to producing the “right” number sequence. Any process generating series of numbers can be chosen as long as it yields musical results which justify their being made the subject of aesthetic concern and as long as they are able to function within the context of a musical artwork.

In the case of algorithmic composition, a composer is to invent the algorithm that is to produce the music. He or she has therefore to start from scratch, inventing the processes and the structure of their own computer art.

It is such a radical notion of artistship that lies at the bottom of Iannis Xenakis’ working with the computer.

However, Xenakis never discussed terms like “standard” and “non-standard” in his theoretical writings. Yet a “non-standard” approach to computer composition is evidenced by every part of his GENDYN project. In light of the above discussion, we better understand Xenakis’ notorious distaste for Fourier analysis: Fourier analysis is the central standard technique of (spectral) analysis and (re-)synthesis of musical sound. It is a technique mostly used to process existing sound instead of inventing new sounds.

A non-standard approach in the context of electroacoustic music is therefore a prerequisite for harnessing technology for an explicit computer art. One of the most innovative exponents of such a kind of “heretical” strategy, assimilating standard music technology for his own creative purposes, is Argentinian composer Horacio Vaggione who teaches at the University of Paris VIII. In his opinion, a consequence of the symbolic representation – the coding – of data in the computer is the fact that “all is text”: score data, the results of spectral analysis, numerical output of algorithmic processes, raw sound sample data. All these data can, in his view, be treated by the same transformational processes: folding and stretching, convolution between different textures, etc. in an iterative procedure, taking the results of one stage as the input of other stages.

A most striking example of his is the arbitrary interpretation of MIDI data as e.g. sound data and vice versa. This represents, indeed, a blatant challenge to fundamental conventions in computer music, for MIDI is one of the most successful product of industrial standardization in the history of electroacoustic

music ever. Vaggione's deliberate re-interpretation of MIDI data defies all those introductory textbook lessons on computer music, where students are taught that MIDI does not code the sound signal itself, but the "events" (the playing information) of a musical performance, in a sense restricted to the "electronic instrument" paradigm.

## 2.3 Idiomatic Digital Aesthetics

The above discussion tells us that in the domain of the arts, computers serve a purpose much different from that in everyday life and also different from that of standardization in music itself. Criteria such as efficiency and standardization that are vital for science and business have either no value in the artistic domain, or even prove counter-productive. This statement is all the more pertinent when applied to rigorous algorithmic composition because here the entire music is a result of computation.

Using computers for the production of artworks cannot seriously be done without carefully reflecting on the specificities of mechanical (i.e. algorithmic) computation. Today's computers comply to the specification and limitation of what it means to be "computable". Computable art therefore benefits from the incredible strength of rigorous formal reasoning which is incorporated into a computer program. Conversely, it has to meet the specificities of formal systems and computation. An authentic aesthetics of computer art can only be established by taking this double-sidedness into account.

"Each art-form exploits its special production methods in order to endow the phenomena with unmistakable characteristics. Artistic economy demands that the means be appropriate to the end and the exploitation of the means be an end in itself." Gottfried Michael Koenig, quoted by [BRT80], p. 25

Any adequate use of computers in music has to deal with the very nature of machine action. Any attempt to "humanize" the computer by ignoring or denying the specific nature of computation amounts to either abusing the machine for tasks a human could better do or abusing the human to succumb to the tyranny of a machine in one of his most sensitive aspects: the confrontation with art as one of the most noble activities of human self-consciousness and introspection. In this sense, using the computer to automate conventional compositional thinking would not only mean to waste machine resources but also to deceive the human on the essence of art and on the fundamental question of what it means to exist in a computerized world.

In view of the fact that computers take over more and more aspects of our daily life, there is a need, more urgent than ever before, to define a true computer art as a way to consciously reflect on the human condition in the era of "thinking machines". It is the merit of composers that I class as members of the explicit computer music culture to have consciously addressed these problems and prompted for reflections by means of their specific way to compose with the computer.

The most radical ideas of what could be a music aesthetic appropriate to computer composition seem to have originated within a group of composers/researchers working at the Institute of Sonology at the University of

Utrecht which later moved to the Royal Conservatory of Den Haag. The “school” of Utrecht developed a notion of what might be called an “idiomatic aesthetics of computer music”, where the specificity of machine computation was made the subject of composition. The urge to strive for an aesthetics that would be appropriate to the use of computers in music composition has best been expressed by one of its then-members, Stephen Holtzman:

“From a creative perspective, what is interesting is not how well computers can emulate traditional human models for performing their tasks and solving problems, but, rather, the new territory that computers will reveal. What are the new possibilities opened by computers? What ideas and means of expression will we discover that are only conceivable with computers? What new models will we develop for viewing the world in light of computers? *What means of expression are idiomatic for computers?*” [Hol94], p. 240. Emphasis original.

One other member of this group, Paul Berg, put it like this:

“It is basically an attempt “[...] to hear that which could not be heard without the computer, to think that which could not be thought without the computer, and to learn that which would not be learned without the computer.” [Ber79], p. 161

But what is the idiom of the computer? Holtzman holds that the idiom of the computer is its digital nature: the manipulation of discrete binary states 0 and 1. Computers represent, store and create music in terms of a mass of binary digital numbers, the samples. The Sampling Theorem (see part II: Theory) states that all sound can be represented as a stream of digital samples, each of which codes a momentary sound pressure value as a binary digital number.

Holtzman, like Koenig, Berg, Brün, Xenakis and others apply the idiom of the computer to music by using the Sampling Theorem in a creative way. Instead of processing existing or simulating conventional sound they create new sounds by changing the descriptive character of the Sampling Theorem to a productive one. This can by itself be considered a heretical use of sound technology. Holtzman, Koenig, Berg, Brün and Xenakis did not care if the sounds so produced correspond to pleasant, natural or even plausible sound. They were interested in creating coherent compositional processes which would manifest themselves in new sounds and sonic structures. In the case of Holtzman and Berg, it was sequences of arithmetic and logic operations on binary numbers (called “Instruction Synthesis” by [Roa96], pp. 327-29). In Koenig’s case, it was techniques taken from serial composition applied to the molecules of sound, i.e. sound waveform segments (his SSP sound synthesis program, cf. [Roa96], pp. 326-27). In Brün’s case, it was abstract operations on waveform segments (his SAWDUST program, cf. [Roa96], pp. 324-26, see also section 5.7.2 on page 74). In Xenakis’ case, it was stochastic processes working on the shape of a waveform period.

In Instruction Synthesis, the temporal structure of the music is not a composed aspect, but an emerging feature of computation. Timing of Instruction Synthesis depends upon the computational complexity of the instructions and the processing speed of the machine used. In the case of Instruction Synthesis,

we can neatly identify the program text with the music score, to be read only by the computer. Instruction Synthesis is described more deeply in section 5.7.5 on page 75.

It is possible to regard Berg's or Holtzman's activities at Utrecht as an early example of what later became *program audification* (cf. [Kra94]). Audification is the attempt to make computer programs sound e.g. in order to help a programmer or a system operator to better detect bugs or anomalies in a complex software system. Again, we can clearly see here the idea of heresy in bending a (now) standard engineering technique to artistic goals: instead of monitoring existing algorithmic processes with the aid of sound, artists like Holtzman and Berg, and later Xenakis, were interested in creating new algorithmic processes in order to obtain new sounds.<sup>3</sup>

The inestimable merit of such radical composers of a computer music idiomatic to the computer is that their concepts are a strong remedy against a false humanization of the machine. Such music never pretends to be nice, smart and beautiful, while at the same time hiding under its false human complexion the wires and circuits of an electronic brave new world.

As a result of this chapter, we can now rephrase our preliminary definition 3 on page 25 of *Explicit Computer Art* as follows:

**Definition 2.1: Explicit or “True” Computer Music.** An Explicit Computer Music is heretical, non-standard and idiomatic.

A mode of working appropriate to the chosen medium has also been demanded by [Sup97]. Supper cites the outstanding, almost classical example of Alvin Lucier's early tape piece “I am sitting in a room” (1969). This piece can be viewed as a radical example of concept-art applied to the technology of tape recording. A magnetic tape is repeatedly recorded with its own playback, in a self-referential manner comparable to an infinite visual regress of two mirrors being set up opposite to each other. The sound source is speech, not music, and the content of the speech is not poetic, but a sober verbal reflection on the recording setup (again in a self-referential manner). At every playback/recording step, the room in which the recording takes place adds its specific resonance to the recording. The result after so and so many recordings is that, on the one hand, all information content of the original speech is lost, but on the other, sound emerges as a product of the auto-amplified resonance of the room as a response to the acoustical impulses of the speech signal (together with the artifacts of tape technology like noise, hiss, etc.), a structure of pitched sounds and timbres of an unreal flavor.

The electroacoustic piece “I am sitting in a room” can be conceived of as an homage to a whole era of analogue recording technology, of the interdependence of room acoustics, microphone techniques, and tape recording machinery. In a way, the tape recorder is used in its own (reproduction) logic as an “instrument” to “create” sound. The specific conditions and artifacts of analogue recording become part of and are consciously reflected upon in an artistic context.

The Lucier piece is a wonderful example for a work of art idiomatic to the medium used. In provocative contrast to the standard endeavor to making the

---

<sup>3</sup>Both Paul Berg and Steven Holtzman remained faithful to their original concepts: Berg's Instruction Synthesis concept has been evolved into a real-time synthesis instrument [Roa92], p. 408, whereas Holtzman still propagates his idea of directly turning symbolic computation into sound [Hol94].

recording process unheard, and creating a perfect illusion of high fidelity sound (by synthesizing artificial acoustic images in the recording studio out of chunks of recorded material), Lucier’s artwork exploits the “dirt effects” of recording and raises them to the status of a sonic event.

Another example of an “idiomatic” use of technology is Herbert Brün’s reflexion on the conditions of artistic production in the electronic music studio in his short composition “auf und ab und zu” (1978). It takes as its basic material the clicks of analogue filter bank switches. This is another instance where a “dirt effect” typical for the medium used is exploited in an artistic manner (cf. [FGG96], p. 239).

## 2.4 Rigorous Algorithmic Composition (RAC)

GENDYN is more than just a custom sound synthesis method. The idea of GENDYN is that a whole musical structure, reaching from the tiniest detail of sonic micro-structure up to macroscopic configurations such as duration, pitch and timbre, representing some hundreds of megabytes of sound data, unfolds as the outcome of an algorithmic process which is described in a few lines of code. In other words, the GENDYN algorithm transforms a few kilobytes of control data into megabytes of sound data without any human interaction except for the making of the program itself. I call this phenomenon Rigorous Algorithmic Composition (RAC).

**Definition 2.2: Rigorous Algorithmic Music** Rigorous Algorithmic Music is a rigorous approach to algorithmic composition where the entirety of a musical artwork is computationally defined, up to and including every atom of the sound itself.

RAC is distinct from “mainstream” algorithmic composition in two ways: first, the algorithm computes the whole sonic gestalt of the piece, instead of “only” score data or “only” sound material. Second, the program is responsible for 100% of the final musical output. (The only post-processing Xenakis applied to *GENDY3* was combining two mono output files stemming from two separate program runs into a stereo file with a 80 ms delay between the channels.) In a way, the composer “composes” a computer program, which in turn “composes” the music. This is what Xenakis strived for during his whole career under the heading of “Automated Music”. In the next chapter 3 on page 47, we will see to what an extent Xenakis was committed to electroacoustic composition and in how far GENDYN can be regarded as the culmination of Xenakis’ electroacoustic Project.

However, “to compose” means a different activity in both cases. The composer’s activity is a creative one. It is reserved to an intelligent being. The computer’s work on the other hand is mechanical. It is bound by the limits of computation. GENDYN is a demonstration in how far a mechanical structure-generating process can bring about a result that can be experienced as a piece of art.

Composing a computer program is different from composing a piece of music. RAC means devising a computational procedure in a way that it may, when executed by a machine processor, result in a desirable musical output. The computer program “is” not the music but just a description of the generating

procedure. One can say that the algorithm contains the music “in potentia”. This is not very different from a musical score which is not the music either but only a code of symbols and conventions for a human interpreter to turn into music. (This human interpreter need not be a conductor, it can also be a musician silently reading the score.) The difference is that the human interpreter can be creative in re-imagining the intended piece while the computer processor follows an interpretation schema laid out beforehand at design time of the programming language used.

Let us, just as an aside, mention that such conceptual approaches also exist in the domain of instrumental music. They may be understood as a thorough reflection on the conditions of composing with the inherited instrumental apparatus. Helmut Lachenmann is probably one of the most famous exponents of such an approach, but many others have found their original contribution to using traditional instruments in ways different from those described in the instrumentation treatises of the 19th century. Interestingly enough, it seems that this sort of discovery of a “*musique concrète* for instruments” (Lachenmann) became only conceivable at a time where the classical orchestra was challenged by the advent of alternate sound sources, namely electroacoustic means of sound production. I once met a Polish composer who explained to me that he brings back into his compositions for instruments all those “dirt effects” that have been banned from concert music by a process of acculturation over the centuries, thereby creating wonderful “airy”, “scratchy”, “tweaky” sounds. I think that this attitude is a demonstration of the fact that he is writing for real instruments capable of many “side effects” that are so hard to mimic by electroacoustic means (although scientists still try hard to do so<sup>4</sup>). It’s like saying: Hey, I’m writing for real instruments, and I want to make it obvious to you in view of all those artificial computer instruments that you are already used to hearing every day without even knowing it!

---

<sup>4</sup>In 1998, I assisted a demonstration at IRCAM of a trumpet sound synthesis model capable of imitating the natural “canard” effect of pitch instability when changing between natural tones. The people present seemed to be very happy of this achievement, including myself!



## Chapter 3

# The electroacoustic Project of Iannis Xenakis

In Xenakis' œuvre, we count 24 pieces which depend, in one way or another, upon electroacoustic technology. Only 14 of these pieces are explicitly listed under the heading of "electroacoustic music" in his catalogue of works ([Xen97]). Let us briefly mention the others:

- Two "mixed" pieces for tape and instruments: *Kraanerg* for tape and orchestra (incidental ballet music) and *Pour la paix* for tape, speakers and choir (a radiophonic work). The latter also exists in alternate versions.
- Another eight works where soloists have to be electrically amplified: *Khoaï*, *Naama*, *Komboï*, *à l'île de Gorée*, *Oophaa* (where the solo harpsichord is amplified), *Aïs* and *Kassandra* (where the bariton is amplified as well as, in *Kassandra*, his psalterion à 20 strings), and *N'shima* (where the two sopranos, 2 horns and cello are amplified).
- Three more works have later been withdrawn by the composer: *Vasarely* (1960), *Formes rouges* (1961), and *Erod* (1997).

All of these compositions could not have been produced nor conceived of by Xenakis without the technology of electroacoustics.

electroacoustics is the science and the practice of electric sound processing: recording, synthesis, transformation and diffusion in space thanks to the help of electric devices and specific techniques of operating them. At the beginning of Xenakis' electroacoustic practice, there was nothing but microphone, tape recorder and loudspeaker. Over the course of time, specific sound processing machinery has been developed. With the advent of the computer, the number of "virtual" electroacoustic devices has become unlimited. All those who know how to program a computer (as Xenakis did) are able to forge their own tools as they need them.

Xenakis' œuvre as a whole is marked by the electroacoustic experience. electroacoustic practice promotes an abstract notion of sound. Sound gets detached from its environment. Thus, sound can be aurally explored in an immediate way, without the attenuations and modifications which a "natural" acoustic setting would involve. electroacoustics can function as an acoustic magnifying glass.

This experience has without doubt conditioned Xenakis' sonic aesthetics and overall approach to sound. Xenakis preferred an analytic approach to hearing as if he were listening to the tracks of a multi-track recording. He liked to “enter” the sound, to be immersed in it.

“The physical proximity of the instruments makes their sonority much more alive than when you hear them from a distance. In the latter case the energy they radiate dwindles and components of the matter of sound are lost. If we are as near the instruments as the conductor and the musicians then we find ourselves in the very middle of sound. [...] We can listen to the individual parts separately and then reconstruct the entirety of the composition in our head. We can also walk round a building, look at all its details and then build up in our minds its image without being there in all its corners and niches at the same time. [...] The composers [...] consider music as an object by itself with real sounds and by no means from a certain distance which would be silly.” [Var96], quoted from the manuscript, part 1, pp. 109-111.

This “engineering” approach to sound is responsible for the fact that Xenakis, when present at recordings of his music, asked that the microphones be put as close as possible to the instruments in order to obtain a sound more immediate and rich. This approach is also responsible for the plethora of unusual playing modes in Xenakis' instrumental music. These playing modes are not used as an effect in Xenakis' music. They are used as raw material for his sonic constructions. Some passages of recorded instrumental sounds, most probably of Xenakis' own music, have been used by the composer in his electroacoustic compositions as an electroacoustic building material for his most complex sonic structures.

### 3.1 electroacoustic Compositions

The 14 electroacoustic pieces are “pure” tape pieces. They can be divided into four groups:

- 5 early compositions made at GRM (see section 4 on page 59): *Diamorphoses* (1957), *Concret P.H.* (1958), *Analogique B* (1959), *Orient-Occident* (1960) and *Bohor* (1962)
- 4 “soundtracks” for his daunting “multimedia productions” of the *Polytopes*: *Hibiki Hana Ma* (1969-70), *Persépolis* (1971), *Polytope de Cluny* (1972) and *La légende d'Eer* (1977)
- 3 pieces made with the UPIC system at CEMAMu: *Mycènes Alpha* (1978), *Taurhiphanie* (1987) and *Voyage absolu des Unari vers Andromède* (1989)
- the subject of this study, the GENDYN pieces: *GENDY3* (1991) and *S709* (1994)

These four groups cover, chronologically and “technologically”, two major periods. In the compositions of the first period, from 1957 through 1977, Xenakis

used sounds recorded from natural sound sources. Most of these compositions were produced in the studio of Pierre Schaeffer at Radio France (his “Groupe de recherche de musique concrète” was re-baptized “Groupe de recherche musicale (GRM)” in 1958). In the compositions of the second period (1978-1994), Xenakis used sounds synthesized at CEMAMu, his research center near Paris (1972-2001).

Within this first period, the “concrete” one, as it were, two important exceptions are to note: in *Analogique B* (1959), the sound sources are not concrete ones but sinusoids taken from an oscillator, and therefore synthesized, and in *La légende d’Eer* (1977), some sounds are already generated by computer.

The second, “synthetic” period of Xenakis’ electroacoustic production is divided in two parts. The first one is the work with the UPIC system, developed by the CEMAMu team led by Xenakis. Xenakis realized four works with the UPIC: *Mycène alpha* (1978), *Pour la paix* (1981), *Taurhiphanie* (1987), and *Voyage absolue des Unari vers Andromède* (1989). The second part is the rigorous algorithmic composition with the help of the GENDYN program (1991), in spirit and continuation of his pioneering work of the years 1958-62 (his ST computer program) as well as his research of the 1970s in sound synthesis (cf. [Xen92a]).

An electroacoustic composition by Xenakis has a specific character, to whatever group or period it might belong. He conceived his sonic constructions as superpositions and simultaneities of sound events which are, in general, complex sounds. As a compensation, the global structure of Xenakis’ electroacoustic compositions often shows a slow-motion process. The richness of such processes which are at the same time monolithic and highly differentiated may account for their hypnotizing effect. A good example is *La légende d’Eer* for the *Diatope* (1977), a sort of *gesamtkunstwerk* uniting architecture, visual effects and music. The music is a huge crescendo-decrescendo stretched over 45 minutes with a climax that can be called apocalyptic.

The same concept has also left its footprints in the conception of Xenakis’ GENDYN composition algorithm. *GENDY3* is a superposition of up to 16 tracks of complex sound but which evolve rather slowly in time, being at the same time extremely minute in detail. *GENDY3* shows a rather static picture of a strange sound world which is just there, in a hypnotizing way, with a multiplicity of internal processes but without the ambition to present any further musical discourse.

## 3.2 The case for the computer

The passage from the first “concrete” period to the second “synthetic” period is motivated by Xenakis’ interest in exploring the inner structure of sound.

“At the same time [i.e. around 1960], I wanted to take possession of the sound in a much more conscious and thorough manner — the material of the sound — and also wanted to be able to create it. That is why I was interested in computers. I wanted to produce sound with the help of the theories which I had applied earlier in the field of instrumental music.” [Var96] manuscript, part 1, p. 46

This quote describes well Xenakis' concept of *micro-composition*. Micro-composition is the concept of composition at the microlevel of sound (i.e. the time level of microseconds, which in digital sound corresponds to the temporal order of the standard digital pulse code rate of 44,100 samples per second). Only with the computer could Xenakis control sound at the sample level. Moreover, he could do it with the help of his mathematical theories of composition, in particular stochastics.

“But if one could produce sounds with a computer, the circle would become complete not only in the field of macro-form but also in the smallest domain, the sound synthesis.” [Var96], manuscript, part I, p. 44-45

Curiously enough, it was not before the composition of *GENDY3* (1991) that Xenakis could realize his vision of an all-encompassing composition with computer, a grand unified theory of composition, as it were, in the form of a computer program.

### 3.3 Performance

Xenakis' concept for the diffusion of his electroacoustic compositions in space is unique. I call it, for want of a better term “mixing in space”: heterogeneous, even disparate sources, are assigned to different loudspeakers (4 in general, sometimes 8, or many more) which surround the public and which yield a result very different for each person in the hall, due to his/her geometric position relative to the sonic sources. A good example is the tape of *Persépolis* (for a *Polytope* in 1971), which is a dense mixture of disparate sources which extends over more than one hour. Xenakis applied this principle also in his spatial compositions for orchestra: the public is mingled with the orchestra (*Terretektorkh* (1965-6), *Nomos gamma* (1967-8)). Each instrumentalist is treated as if he/she were a loudspeaker, i.e. a sonic source which can be approached while being immersed in a cloud of sounds from all directions (cf. [Hof07])

*GENDYN* is different from this approach. First of all, it is not a multi-track composition. The algorithm composes a mono channel sound signal. Xenakis had his program run twice in order to obtain two mono channels which he then combined into a stereo recording, with a 80 ms time delay in between, in order to produce a pseudo-spatial effect. It would have been much more appropriate to the “mixing in space” approach described above if Xenakis had assigned each of the 16 tracks of *GENDY3* to a separate loudspeaker. This is, however, not what he did for the premiere of *GENDY3*, nor for the premiere of *S709*. Unfortunately, it was not given to Xenakis to elaborate his *GENDYN* project any further. He might have developed interesting ideas for a spatial performance of his automated compositions.

### 3.4 electroacoustic Aesthetics

The electroacoustic practice had a great influence on Xenakis' music as a whole — not the least through his aesthetics of timbre. Xenakis stated many times that for him the richest sound was white noise, “too rich for our ear to perceive

it as sound” ([Var96], manuscript I, p. 90). Xenakis’ most consequent homage to noise – an example which I like to call “apotheosis of noise” – is to be found, as the reader may probably guess, in *GENDY3* (4th section of the piece, between 05’31” and 07’39”). The listener is drowned into a bath of violent noise for more than two minutes. The most interesting inner structure of this – even for Xenakis – extraordinary example of music will be analyzed in part III of this study.

Xenakis’ approach to timbre is closely linked to the notion of spectrum of electroacoustic research. Spectral analysis of sound has its theoretical foundation in Fourier analysis (after French engineer Jean Baptiste Joseph, Baron de Fourier, 1768-1830). Similar mathematical theories have also been developed by Swiss mathematician Leonard Euler (1707-1783) and German mathematician Karl Friedrich Gauss (1777-1855) (cf. [Roa96], p. 1076). Fourier analysis is based on the idea that all periodic signals can be represented as sums of sinusoids, which are the mathematically simplest form of oscillation. Fourier analysis is widely applied in engineering, as it pertains not only to acoustics but to all phenomena where periodic oscillations are involved ([Bra89]).

As the mathematical procedure was tedious and error prone, machines were developed to perform Fourier analysis as early as late 19th century. For acoustic research, analysis was done with analogue filter banks until around 1960. Fourier analysis of digital sound became available on the computer through efficient implementations known under the heading of Fast Fourier Transformation ([Roa96], p. 1076).

From the perspective of Fourier analysis, white noise is the richest signal because it contains the greatest number of sine constituents (all frequencies are theoretically present in a perfect white noise spectrum). In other words, according to Fourier theory, white noise is the most complex sound because the number of its constituents is maximal. The extreme opposite is a pure sine wave because it can be described by one single function.

To define the richness of sound in terms of the number of its constituents is of course a purely theoretical and even debatable method. As for white noise, no ear is able to perceive any richness in it. However, Fourier theory provides a quantifiable basis for assessing the periodicity of sound. Xenakis has often pointed out the importance of degrees of periodicity for his composition of sound, of scales, of rhythms, etc. In his music, Xenakis systematically explores the whole spectral continuum between noise at the one extreme and “pure” tones at the other. His distaste for Fourier synthesis is rooted in the fact that it tends to prefer the pure side over the complex one due to the fact that noise is just too expensive to be calculated by sinusoids.

Xenakis’ ideal of a “rich” sound — which is, as we understand by now, a sound rich in partials (i.e. Fourier constituents) — is therefore inspired by electroacoustic research. Xenakis tried to achieve rich sonorities with instruments through unusual playing modes like *écrasé* (excessive bow pressure) and extreme *sul ponticello* play for strings, or *flatterzunge* and multiphonic techniques for the wind instruments. In his “concrete” pieces, he achieved richness of sound by choosing noisy sound sources like earth-tremors in *Diamorphoses* (1957) ([Del97], p. 39). Xenakis found un-domesticated sonorities with noisy sonic “side effects” in extra-european music and instruments coming from Asia (laotian organ), Africa (marimba) and “antiquity” (bells, cymbals) in *Bohor* (1962) ([KB]).

With UPIC, Xenakis designed jagged wave forms in order to yield rich spectra. Finally, with his GENDYN program, Xenakis generated step-wise signals full of digital artifacts which produced spectra saturated with partials. These signals could never have been achieved through “natural” sound excitation but only be constructed with the computer. Xenakis did not try to minimize these noisy “side effects” of digital synthesis, as any “academic” composer would have done. Quite on the contrary, he exploited them for building his sonic universe.

GENDYN, from its very algorithmic definition, is centered around the idea of periodicity. It explores the continuum between periodicity and aperiodicity in sound. GENDYN sound is constructed by looping through a wave form, as in classical wave table synthesis. Different classes of sounds reaching from pure noise to frozen tones are obtained by different degrees of periodicity, i.e. similarity of this wave form under repetition. If the waveform is gradually altered under repetition, a smooth sonic evolution can be heard. If the changes are violent and disruptive, the aural impression is noise. In-between, a whole field of intermediary sonic phenomena can be obtained by the GENDYN algorithm.

### 3.5 electroacoustic Research

Xenakis is known to have preconceived granular synthesis in music composition. Xenakis cites *Analogique B* (1959) as a product of analogue granular synthesis: a tape glued together from thousands of little splices each carrying a piece of elementary sinusoidal sound (a sonic “grain”). Little is known, however, about Xenakis being the first to try digital granular synthesis on the computer.

“In Paris, when I was still active as a professor at Collège de France, we were in contact for the realization of a device which he [Xenakis] had elaborated. It was the question to put “musical grains” in computer memory which consisted of noise or short sounds [...]. With such grains it would be possible to construct a music at the will of the author, and even to edit it later in a graphical editor after being synthesized. Then, these data would have to be transformed to sound with the aid of a digital-to-analogue converter. This converter was finally constructed by the Centre national d'étude des télécommunications [CNET] after experiments conducted in my laboratory.” [LR81], p. 53<sup>1</sup>

Granular synthesis is as fundamental a theory as Fourier synthesis. Just as all (periodic) sound can theoretically be generated with a sufficient number of sinusoids, all sound can theoretically be generated with a sufficient number of sonic grains. As we have learned from the above quotation, Xenakis found himself *nolens-volens* in the situation of a pioneer. Everything had to be done from scratch: even the D/A converter, nowadays a simple chip built into any home computer. In these conditions the UPIC was constructed, which at times was unique in the world. The UPIC is a computer-based graphical tool for music composition, a kind of Computer Aided Design for music. (Originally, it made use of a huge CAD drawing board taken from automobile industry.) In the

---

<sup>1</sup>Louis Leprince-Ringuet was a colleague of Louis de Broglie in his X-ray laboratory. He took the chair of nuclear physics at Collège de France in 1959.

course of its different realizations, the UPIC has been transformed from a huge setup of bulky machines to smaller and more integrated hardware and finally to a software-only solution ([MRS93]).

At last, with his GENDYN program, Xenakis could finally realize his idea of a stochastic sound synthesis in a way which permitted him to cover the synthesis of a whole musical oeuvre.

From the early days of his artistic career, starting with the composition of *Achorripsis* (1957), Xenakis has strived to formalize the compositional act as a set of rules (cf. [Xen57]). In this sense, GENDYN can be considered the culmination of his artistic life, the achievement of his long dream of a music which is entirely automatically generated. Quite different to the majority of his colleagues in the domain of algorithmic composition, with *GENDY3* and *S709*, Xenakis did not only compute a musical score but the whole sonic gestalt of a piece of music.

The GENDYN technique of music synthesis, instead of conforming to the laws of acoustics and the model of natural sound, follows arbitrary rules established by the composer. This is the deeper meaning of Xenakis' own term "micro-composition": the GENDYN program *composes* the micro-structure of sound. This composition process happens not in the macroscopic time scale of listening time (as in the case of score composition) but in the microscopic time scale of digital sound, which is in the order of microseconds (one-millionth of a second). Surprisingly, this micro-composition of sound generates, in a process of emergence, linking micro-time and macro-time, long "macroscopic" sonic evolutions with all qualities traditionally associated to score composition: pitch, dynamics, timbre, rhythm, in short, qualities which are otherwise attributed to the musical note which, as a unit of composition, is absent in GENDYN.

Instead, GENDYN realizes a continuous space of timbre linking micro- and macro-structure. As an example, violent noise is revealed as being a simple aleatoric melody, but accelerated beyond the threshold of human pitch perception. To give another example, the chaotic, stochastic, microscopic processes of GENDYN sound have a tendency to "crystallize" around "attractors" of stable macroscopic pitches. These pitches form non-octaviating scales, which are reminiscent of the famous sieves developed by Xenakis from the 1960s.

It is hard to imagine how a composing procedure such as the Dynamic Stochastic Synthesis could ever have been thought of without the computer. Xenakis founded the CEMAMu because he had the vision that only the computer would give him the power to realize his algorithmic idea of composition without any compromise directly in musical sound. The universality of the computer, together with the digital representation of sound, gave Xenakis the freedom to design his composition algorithms, bypassing all standard engineering, and calculating the numerical values of digital sound at his wits.

### 3.6 electroacoustic Thought in Composition

Since his "opus 1" *Metastaseis* (1953-54), Xenakis has composed a great number of works on rule paper. This holds specifically for his production up to the 70s.<sup>2</sup>

<sup>2</sup>From 1980, matching his graphic and algebraic sketches with the final score is more difficult. For example, the score of *de Evryali* for piano solo (1973) is nothing but a transcript of the graphics, while it is difficult to see a correlation at all between the sketches of *Khal*

It is true that Xenakis' graphical sketches ask for a technical realization: the typical Xenakian glissando, an element of musical construction of first order (in contrast to a glissando as a special effect or a musical gesture), is mathematically conceived of as a straight line, as the graph of a linear function of pitch over time, and is naturally translated to the movement of a potentiometer or the linear change of a program variable. By the same token, the graphical scripture eases the transfer of macrostructure (notes) to micro-structure (sound) because it is an abstraction of both. This explains the easiness with which Xenakis transcribed the graphs of sound signals he calculated with the computer to melodic lines in the scores of e.g. *Mikka* (1971), *N'Shima* (1975), and *Thallein* (1984). With the UPIC, Xenakis had finally found an immediate way to turn his musical drawings into sound.

In the same spirit, Xenakis applied formalisms taken from sound synthesis to textures in his instrumental scores, like *Cendrées* (1973), *Ikhoor* (1978), or *Mists* (1981). These are the Random Walks. Xenakis has commented upon this technique in a series of lectures around 1980 which have later been published as an article [Xen96]. In *Jonchaies* (1977), Xenakis tried to emulate a sonic maelstrom he had electroacoustically produced in *La légende d'Eer* (1977) with the instruments of the orchestra, a spectacular but not isolated example of a cross-fertilization between electroacoustic and instrumental composition. Even the orchestra pieces of the 90s show a kind of electroacoustic thinking, with the orchestra clusters spread over the whole range of available pitches. This holds even for his late chamber music: in his last quartets *Tetora* (1991) and *Ergma* (1994), the strings are forced to play double stops, generating moving 8-tone clusters through most of the piece.

One can safely say that electroacoustics had a strong influence on Xenakis' artistic life, more than it seems at first glance regarding his relatively small output of electroacoustic compositions. As has been described, electroacoustic thinking has governed much of his production for traditional instruments and made possible his extraordinary multimedia spectacles of *Polytopes* which were way ahead of his time. Xenakis' research activities at CEMAMu and elsewhere have radiated far beyond his own artistic production and inspired generations of computer composers to come.

It is striking that given the importance of Xenakis' activities in music technology and innovation, so little of his work has been preserved. From the *Polytopes* remains practically nothing but a few photos; from the *Diatope*, a kind of *gesamtkunstwerk* of music, architecture and light, only the music has survived. Little seems to remain even of his computer compositions except the recordings. In these circumstances it is a happy exception that at least his *opus summum* in computer composition, *GENDY3*, could be saved from oblivion. But the computer program itself was already lost. Fortunately, I could reconstruct it from a printed listing that Xenakis had kept in a folder on a shelf at CEMAMu. Because *GENDYN* is an example of a RAC composition, it is possible to reproduce and simulate the creative process that had led to the composition of *GENDY3* at any time anywhere in the world, even after the composer's death. The *GEN-DYN* project, i.e. ongoing activities in research and creation with the *GEN-DYN* program, thus are the fulfillment of Xenakis' legacy of a true computer art.

---

*Perr* for brass quintet (1983) and the published score.

### 3.7 Xenakis' Idea of an Automated Art

"We find ourselves in front of an attempt, as objective as possible, of creating an automated art, without any human interference except at the start, only in order to give the initial impulse and a few premises, like in the case of the Demiourgos in Plato's *Politicos*, or of Yahveh in the Old Testament, or even of Nothingness in the Big Bang Theory." (Iannis Xenakis)

Xenakis belongs to the pioneering generation of computer music. However, he practiced more than just "computer music". Xenakis was a major promoter, if not the inventor, of a radical idea of "Automated Art". His interest, however, was not to build a music box or to ease his work as a composer by using machines but to investigate into the possibility of a computable music, much in the sense of definition 5 on page 30. During his whole career, besides his activity as a "normal" composer, he invested considerable time and effort into creating, as it were, a composing automaton.

Xenakis was always interested in systems that auto-generate a whole of music, with all its internal structural subtleties, within a closed design. In his instrumental music, for example, Xenakis often "generated" extended textures by creating a system of compositional rules and setting it "into motion". One of the most notorious of these composing "automata" is the symmetry group formalism he set up for the "Nomos" compositions [Xen92b]. The juxtaposition and contrasting of systemically coherent sections has been observed as one of the main strategies of the composer to create musical form (cf. [Sol96], [Har99]). With *GENDY3*, however, Xenakis created a reference work of pure algorithmic composition, a music "composed" by the computer.

Automated score composition is mentioned for the first time by Xenakis as early as 1957 ([Xen57]. Around the same time, Xenakis also put forward his idea of algorithmic sound synthesis (cf. [Var96]), much in advance to the French electroacoustic music scene of the time. Xenakis is also known to have preconceived Granular Sound Synthesis:

"During the time I held a position at Collège de France in Paris, we have been in contact for a realization of an device which had been designed by him [Xenakis]. It was the idea to put in computer memory 'musical grains' [sic!] produced by noises or short sounds, that means short time cuts of sound pressure curves [...]. With these grains held in computer memory one can design music, and even work on it with the help of a visual editor once the sound signal has been computed. And then, to transform this signal into sound with the help of a digital-to-analog converter. Such a converter was eventually constructed by the Centre national d'étude des télécommunications [CNET] after the experiments carried out in my laboratory." [LR81] (see also [Xen92b], p. xiii).

But what is even more important is how Xenakis used the computer. His approach was to not use it as a tool for computing sounds using standard acoustical models, but rather for creating the acoustic shape of a whole musical composition. His ultimate goal was to entrust the computer the task to actually carry out a compositional process, not of a musical score but the sounding end product. In the case of *GENDYN*, this is done on the level of the "atom" of digital sound, the digital sample.

The GENDYN program produces a macroscopic musical structure (pitch/duration/timbre) by means of microscopic sound synthesis (called “Dynamic Stochastic Synthesis” [Xen92c]). Here, the sound signal is computed by applying stochastic random walks to the geometric shape of a wave form. The resulting modulation of frequency and amplitude covers the ample field between stable, pitched tones and complex modulation noise.

GENDYN was the first composition being entirely based on stochastic sound. This does not mean that Xenakis was not active in this domain before GENDYN. On the contrary, there are evidences that Xenakis constantly tried to realize his dream of an automated music through stochastics for a long time since he formulated this idea in the late 1950s. A module doing a kind of stochastic sound synthesis was added to his ST program (see the program listing in the appendix of [Xen63]). Some examples of stochastic sound produced at CEMAMu could already be heard in the music of *La légende d’Eer* (1977).

Xenakis’ idea of using sieves for sound synthesis [Xen96], dating from the beginning of the 1980s, seems not to have been tested so far. The creation of sieves as such, however, has been sufficiently formalized, and a listing of a sieve-generating program has been published which could serve as a core to computing “sieved” sound ([Xen92b]). If Xenakis had succeeded in using sieves for sound synthesis, he probably might also have thought of using cellular automata in combination with them, similar to the way he combined these two formalisms in *Horos* (1981) (see [Hof02]).

### 3.8 The Dynamic Stochastic Synthesis

In a chapter first included in the 1972 edition of his book “Formalized Music” called “New Proposals in Microsound Structure” ([Xen92a]), later in a dedicated article ([Xen85]), Xenakis discussed several conceptual approaches for composing the sound signal on the sample level. (For a detailed historical account on micro-sound synthesis, see e.g. [Roa02].) In these writings, he describes his non-standard approach to sound generation – yet without using this term.

“We hope to open a new path in micro-sound synthesis research — one that without pretending to to be able to simulate already known sounds, will [...] launch music, its psychophysiology, and acoustics in a direction that is quite interesting and unexpected. [...] This method would use stochastic variations of the sound pressure directly. We can imagine the pressure variations produced by a particle capriciously moving around equilibrium positions along the pressure ordinate in a non-deterministic way. Therefore we can imagine the use of any ‘random walk’ or multiple combinations of them.” [Xen92b], p. 246

In this quote, Xenakis defines the sound signal to be the path of a particle performing a random walk. But this is only one possible way of applying stochastics to micro-composition. In his articles, Xenakis discusses a number of other, more sophisticated, methods to do sound synthesis using random walks. One of these (“proposal b”) corresponds to what he did in GENDYN. (By the way, seeing it put to work in GENDYN makes this proposal much easier to understand.) Therefore we can say that Xenakis knew the concept of what he

realized in 1991 at least 20 years beforehand. There is room for speculation why Xenakis had to wait 20 years before being able to turn this concept of his into a valid composition.

“Explore the method of *polygonal variation*, the term we have given to a series of sound realizations. Roughly, this is a step-by-step construction of a pressure curve that is modified at each step (period) by a stochastic device, modified in the sense of time and also in the sense of pressure values at each sample. Acoustical experiments using a computer and a digital-to-analog converter have shown that, for special values of the mathematical device, there arises a sort of probabilistic resonance that engenders rhythmic multiplicities of timbres, dynamics, attacks and macro-forms.” [Xen85], p. 176

Some lines later in the article, Xenakis goes into more detail.

“Starting from a pressure-time curve [...] one may continue by repeating this curve and at the same time injecting stochastic modification into it after every repetition. This stochastic modification is chosen so as to produce the statistically continuous negation of the original period, affecting the timbre, pitch, rhythm, intensity, and general evolution simultaneously. [...] It becomes the job of the composer to master, with intuition and reason at the same time, the doses of [this negation]. In other words, one establishes an entire range between two poles — *determinism*, which corresponds to strict periodicity, and *indeterminism*, which corresponds to constant renewal.” [Xen85], p. 179-180

In this quote, we can recognize the description of the algorithm of Dynamic Stochastic Synthesis as used in GENDYN (see part II for a detailed description). At that time, Xenakis was already conscious of the compositional capacities inherent in this sound-producing method: a dynamic evolution of sound in gradual or abrupt change, or, in contrast, a frozen sound signal when change would be chosen to be nil. Typical for Xenakis, he envisioned a music stretching between these two poles, identified as the principles of determinism and indeterminism, which for Xenakis had a meaning not only in his enlarged universe of sound but also in the universe as such, a philosophical meaning, an existential meaning.

“This is the true keyboard of musical composition. Thus we emerge in a domain of multiple scientific and philosophic resonances. The continuity and discontinuity of the mathematicians and of the time-space of quantum physicists are such resonances.” [Xen85], p. 179-180

We see that a sound-producing concept, in the case of Xenakis, is more than just a means to make music. It is typical for an explicit culture of computer music, for the non-standard synthesis approach, and last but not least for an ethical approach to music composition that the very process of composing reflects a fundamental attitude toward the world. This is a concept of composition that Xenakis shared – knowingly or unknowingly – with other outstanding computer composers of the explicit computer music culture.

In contrast to most of the other people working in the field of algorithmic composition, Xenakis has never evolved his programming toward an “open system” or toward defining a general programming environment or programming language for musical purposes. This is motivated by the fact that Xenakis' approach may be considered itself as “universal” in the sense that it defines every aspect of a musical composition entirely in terms of stochastic processes. The stochastic indeterminism or “liberty” in the generation of structure compensates in some way for the rigidity and the confinement of the algorithmic composition method. The same holds for his early ST program (1958-62) doing the ST pieces. It has been studied and used mainly by students of his up to the seventies in the USA, and has later been extended by sound synthesis routines but it remained a closed computer program, the formalization of stochastic composition techniques as used by Xenakis, a self-consistent sound world, a microcosm reflecting processes in the Cosmos at large.

## Chapter 4

# The Paris electroacoustic Scene

In this chapter, I would like to draw a rough sketch of the electroacoustic music scene in Paris as of 1995, at least as far as the main institutions are concerned. This is not to suggest any influence onto Xenakis' solitary work in the field of rigorous algorithmic music, but rather to show how different Xenakis' work was from the activities of the more or less established institutions for computer music within his immediate neighborhood. For this purpose, the restriction to the "official" representation of electroacoustic research within Paris seems adequate, as the many small groups and individuals who do exist there may most probably even not have come to his notice. I shall therefore only mention the two major institutions, GRM (since 1959) on the one hand and IRCAM (since 1977) on the other, the CEMAMu, and a small electroacoustic studio, Les Ateliers UPIC (since 1985, renamed in 2000 to CCMIX<sup>1</sup>). The latter took over the former CEMAMu activities in pedagogy and production, and since then the CEMAMu concentrated exclusively on the development of the UPIC system. It is hardly an undue over-simplification to identify the Technology of Writing with the aesthetic mainstream of IRCAM, the Technology of Editing with the GRM, and the Technology of Computing with Xenakis' own research institute CEMAMu.

### 4.1 GRM

"The many formula in vogue at the present time for music with machines — direct synthesizer, direct transformation of instrumental sounds — are seen as a sign of progress, a more advanced stage in musical research than music with loudspeakers. But how is it possible not to see that, in most cases, we are witnessing a regression to a situation where the spectator finds once more everything to which he is accustomed: scores on stands, an interpreter who comes on stage, bows, is applauded, and is recalled at the end. And above all the idea that music can only exist if it is written down somewhere on paper. At best, these experiments are a new and original branch of mainstream instrumental music, one which has little to do with what we call concrete music, whose essence lies in the fixation of sound." [Chi93], p. 53–54

---

<sup>1</sup>Centre de création musicale Iannis Xenakis

The GRM (*Groupe de Recherches Musicales*) incorporates Pierre Schaeffer's concept of "musique concrète", started around 1948 (cf. [Sch52]). In contrast to the traditional way of composing, where scores are written in an "abstract" symbolic language inviting a musician to turn into sound, composing in the GRM tradition means using sounds which are already there, "concrete" sounds, take them out of their original context, and build a composition out of these. The means to do this are sound recording machines (in the beginnings 78 cps turntables, then magnetic tapes, today computers) and heavy sound processing (playback at variable speed, filtering, editing, etc.) in order to forge a work of art out of the concrete material. It is evident that "musique concrète" was the artistic appropriation of a new technology, i.e. the technology of radio production, by the genius of Schaeffer, and later Pierre Henry, Luc Ferrari, and many others.

The promotion of an "acousmatic" attitude to sound is perhaps the most distinctive feature of GRM's music aesthetics. It has obviously left traces in Xenakis' own electroacoustic oeuvre. The term "acousmatic" refers to the teaching practice of the Pythagorean school, famous for its *ακουσματα* (Pythagorean wisdom), where the novices were said to be talked to from behind a curtain, i.e. they learned by intense listening, without being exposed to gestures or mimics (cf. [Sch66], p. 91). This corresponds well to the situation of a radio listener. Schaeffer realized the specific quality of this intense listening experience made possible by radio technology and its consequences for an adequate hearing and an adequate artistic approach (cf. [Dac94]). It has been observed that Schaeffer strived toward nothing less than the foundation of a new phenomenology of hearing [Sol97].

One consequence of musical acousmatic practice is the detaching of the sounds from their origin and physical cause, processing them, putting them into contexts with other sounds, and thus creating a pure listening experience (cf. [Chi93], [Lar97]).

"ACOUSMATIQUE: stems from 'akousma', auditive perception. Term employed by Pythagoras who taught to his disciples hidden behind a black curtain in order to develop their faculties of concentration. In 1955 retaken by writer Jérôme Peignot in order to define the sonic object (*l'objet sonore*): 'Quels mots pourraient désigner cette distance qui sépare les sons de leur origine... Bruit acousmatique se dit (dans le dictionnaire) d'un son que l'on entend sans en déceler les causes'. In 1966, Pierre Schaeffer applies this term to the notion of 'reduced hearing' (*l'écoute réduite*, Schaeffer 1966, *Traité des objets musicaux*)." [Gor96]

For this precise reason, much in contrast to IRCAM, GRM does not pursue any research on the acoustics of music instruments. In its purest form, acousmatic music means "loudspeaker music". The music is entirely produced in the electroacoustic studio manipulating sounds of all kinds and provenance (which is claimed to not matter) to form a coherent composition. No instruments are allowed on stage of a "pure" acousmatic concert (cf. [Chi93]). However, this directive is, it must be said, less and less obeyed by composers working at GRM.

GRM's most significant contribution to the technology of computer music are the "GRM-tools", an efficient implementation of standard and less standard sound manipulation techniques on the computer using a standard hard-

and software environment (MAC/ProTools/DigiDesign). In this sense, GRM tools faithfully follow the tradition of manipulating “concrete” (i.e. existing) sound (in chronological order: Disk Turntables, Phonogène, Tape Machines, the programmable digital SYTER system [Ter98], and then the personal computer [Ges98]). Its distinctive features are very powerful filters and transformations like “brassage”, a kind of granular sample shuffling analogous to tape splicing, but with much deeper impact on the sound because it reaches right into the micro-structure of sound. However, and sadly, GRM tools still do not fully implement the full capacity of the predecessor system SYTER (personal communication of Horatio Vaggione, Fabien Lévy, and Yann Geslin). Another recent software is the “Acousmographe”, a computer-assisted visualization tool for composed sound, based on spectral analysis.

The acousmatic approach explicitly renounces a whole history of musical literacy, of “composing-by-writing”, in that it bypasses (in its pure form) any written documentation which may serve to represent or reproduce it. The deep importance of the acousmatic approach is that it is a border case of a historical development leading from a “music making” culture to a “music listening” culture (cf. [Del96]).

## 4.2 IRCAM

“For an artist, contemporary music cannot really find its expression [...] if he does not mind the past. Presently one uses traditional instruments, and natural singing voices. Now, if I want to enrich this existing language of singers and orchestral instruments by new computer generated instruments or new sounds, I must before and above all make the computer analyze and understand what already exists. If I cannot analyze, model and represent a singing voice or the sound of a violin, it is no use hoping to invent new sonorities which will make sense in a musical and artistic context.” (Laurent Bayle, director of IRCAM from 1992 to 2001 in an interview on LCI “Multimedia”, 10/23/1998)

IRCAM defines as its major contribution to contemporary music history the real-time technology it developed since its beginnings, in order to extend the acoustical and compositional possibilities of the classical instrumental concert. The most notorious achievements in this direction are obviously the compositions of its long time director Pierre Boulez which are accompanied by a more or less sophisticated electronic setup, like *Répons* (1981-1988), or, more recently, *... exploitante-fixe ...* (1991-1995). IRCAM has always felt its vocation in putting to service modern technology to instrumental thought and the traditional concert situation, much in contrast, for example, to the aims and endeavors of the GRM. Within IRCAM, there are departments of research on acoustics of concert halls and acoustics of instruments, as well as a department for psychoacoustics, concerned with exploring the response of humans to instrumental and other sounds [Mac84].

One of the main concerns regarding the concert situation is the “interaction” of musicians with programmed “real-time” systems. IRCAM developed sophisticated environments of machine score following and “real-time analysis” of musician’s acoustic output in order to synchronize and control the electronic part. However, the feedback loop between man and machine is not closed. The reacting of musicians to machine-generated sound does not seem to be part of

IRCAM's live electronic setup, because the human musician is to interpret a fixed musical score. I will elaborate upon the interest in a closed man-machine interaction later in this study.

IRCAM's most significant contribution to the technology of computer music are to be found in the areas of real time sound processing, sound analysis/synthesis, and computer aided composition. (A relatively new branch are IRCAM's efforts in Physical Modeling). Real-time sound processing is achieved with FTS ("Faster Than Sound"), a software engine which replaces the 4X hardware built in 1978 and its successor, the ISPW ("IRCAM Signal Processing Workstation") [LDSS91]. FTS [Puc91] is controlled by the graphical/programmable frontend "jMax", one of its patches being the "Spatialisateur", a sound spatialization device [DBD<sup>+</sup>98]. The major analysis/synthesis engine is called SVP ("Super vocodeur de phase"), embedded in a graphical sound editing environment called "Audiosculpt". Computer Aided Composition is implemented in the Open Music system [Ass99], a fully programmable symbolic/graphic working environment for composers interested in formalization and prototyping of compositional ideas.

Only a part of IRCAM is devoted to actual music creation, and much of the activities are in the field of "pure" research. Certainly, this research "indirectly" has its effect on musical innovation: for example, it is without doubt that without the IRCAM experience, many composers would not have developed their specific style.

Aesthetically, IRCAM is devoted to the classical notion of the composer writing a score for musicians to interpret. Composers who want to work at IRCAM must pass a reading panel, so only composers trained in classical score writing have access to IRCAM's facilities. The keyword of IRCAM's aesthetics may be enunciated by the French word "écriture" which not only means "writing" scores on paper (or into a computer system), but also the musical styles and concepts that go with the technology of writing (cf. [Bor95], pp. 223ff). The quote of a partisan of the acousmatic aesthetics situated in Belgium, Annette Vande Gorne, is symptomatic for the conceptual abyss that separates these two "schools":

"These musics [instruments and live electronics] are often written by composers who are more at ease with instrumental writing techniques [écriture instrumentale] than the morphologic perception of sound and 'timbre', bending the play in favor of the instrument, while the electroacoustic means are only there, at best, to extend its possibilities (which is generally the case with pieces realized at IRCAM), and, at worst, to serve as a contrasting foil, a background against which to show off all the virtuous resources of the instrument. *Rari nantes in gurgite vasto!*" [Gor96]

Conversely, IRCAM always rejected, and still seems to be very critical of GRM's approach of bypassing musical notation and instrumental interpretation in their practice, represented by Boulez' polemic statements about GRM's approach not being pertinent to music composition (cf. [Bor95], pp. 75-77).

It is also Pierre Boulez who coined the phrase that sounds have to be sufficiently "neutral" to be amenable to sophisticated compositional treatment, i.e.

to “écriture”<sup>2</sup>. In this study, we will present a radically opposed approach: creating a musical œuvre by most complex and internally animated, living sound. This approach is certainly much more akin to the aesthetics of GRM than to that of IRCAM.

### 4.3 Les Ateliers UPIC

“We maintain that there is a continuum between the acoustic and the electroacoustic which may be demonstrated acoustically by the use of extended instrumental and vocal techniques that simulate equivalent electroacoustic transformations [...] A unified compositional theory that takes the continuous transformation of sound structures (as opposed to the transformations of series of notes, melodies, themes) as its basis can be a model for unifying the acoustic and electroacoustic domains. Furthermore, whether the transformation is accomplished acoustically or electroacoustically, the end goal is the same: a unified composition on the level of the resulting sound.” [Pap95]

Les Ateliers UPIC (since 2000 renamed to “Centre de Création Musicale Iannis Xenakis”) have been founded in 1986 in order to promote the pedagogical and creative activities around the UPIC system that CEMAMu, Xenakis’ research center, where this system had been developed, could no longer provide. In fact, Les Ateliers UPIC are the only public place where to use the system in a framework of residency programs and computer music courses. In addition to the UPIC, the studios use other software and hardware tools, like GRM tools or, recently, the MARS station, conceived by the same Guisepppe di Giugno that at times built IRCAM’s 4X DSP processor.

The aesthetics of Les Ateliers UPIC are situated beyond the antagonies that exist between IRCAM and GRM in that Les Ateliers UPIC are neither interested to embellish classical music writing with technological gadgets, nor to concentrate on the pure “musical object”. They rather aim at inventing new forms of composing and presenting music which reconcile the electroacoustic with the acoustic. Looking at scores composed within Les Ateliers UPIC, a predilection for techniques composing-the-sound (instead of composing with sounds) becomes evident. Many of the pieces realized in Ateliers UPIC are “mixed pieces” which combine a studio-produced tape and live instrumental soloist(s). But many visiting composers also admit that their experience with electroacoustic techniques has become an integral part in writing for instruments. An important part of this experience is without doubt the systematic exploration of a “continuum” of sonic parameters. The UPIC system – which has for a long time been the major pedagogical tool of Les Ateliers UPIC – makes it indeed rather difficult to compose with discrete scales and rhythms, whereas, on the other hand, it favors the composition of continuous evolution, sound masses and a multitude of shades of sound colors.

---

<sup>2</sup>“I understand that the dialectic of composition better contents itself with neutral objects, not easily identifiable ones, like pure tones or simple tone aggregates, having no inner profile of dynamics, duration or timbre. As soon as one shapes elaborated figures, assembles them into ‘formed’ complexes, and uses them as first-order objects for composition, one is not to forget [...] that they have lost all neutrality and acquired a personality, an individuality which makes them quite unfit for a generalized dialectics of sonic relations.” [Bou94], p. 45.

## 4.4 Xenakis and CEMAMu

“CEMAMu [has] the aim of making it possible for composers to directly transplant scientific thought and mathematics into music. Directly, that is to say, without any sound generator and music instruments. As an example: some of the theories I had developed can also be used in computer music because they are based on mathematics. They were born as a result of compositional necessity, that is, of reasons outside mathematics, but they need mathematical means for their realization.” [Var96]

Iannis Xenakis was affiliated with GRM between 1954 and 1963 when he left the studio on a conflict with Pierre Schaeffer who was violently opposed to his idea of introducing computers into the creation of music. Only years later, in 1972, Xenakis was able to make his own research center CEMAMu operational (it was founded as early as 1966 but did not receive funding then). CEMAMu stands for “Centre d’étude de mathématique et automatique musicales” (“Center for Mathematical and Automated Music”). It was hosted by CNET (Centre national d’études de télécommunications) in Issy-les-Moulineaux near Paris. CEMAMu was closed shortly after the death of Xenakis in 2001.

From 1977, Xenakis fabricated his electroacoustic music at CEMAMu. He did not have equipment for electroacoustic production at home nor in his atelier nearby. There, he exclusively composed music for instruments. In times when he regularly worked at CEMAMu, he split his working day between his atelier and the research center at the other end of town (cf. [Var96]). Even within his own research institute (4 hardware and software engineers), Xenakis’ work was rather solitary. He had a tiny room aside to work in, and he did not seem to share too much of his activities with the engineers who were mainly concerned with the development of the UPIC system.

Xenakis’ preoccupation in his electroacoustic activities since he founded CEMAMu was the use of computers in order to apply conceptual musical thinking, often with a strong abstract formal ingredient, directly to the sounding matter. Xenakis was never interested in standard sound synthesis or digital signal processing techniques, which are at the core of any other computer music studio or music technology research institute. When Xenakis postulated Granular Sound Synthesis in the 1950s it was not the “standard” synthesis technique it became later, but an abstract concept for composing sound, rooted in a corpuscular theory of sound (see [Xen92b]).

Xenakis’ UPIC system, which has since its original ideas constantly been evolved by the CEMAMu team of engineers ([MRS93]), is another instance of putting computer power at the service of conceptual thought in music composition. UPIC incorporates the idea of creating a musical work “from scratch”, starting at the deepest level of sound, i.e. the design of a wave table at the digital sample level. The fact that many aspects of the music are defined by graphical design and controlled in real-time certainly facilitates this task of creating “from nothing”.

Xenakis always promoted the idea of auto-generating machine sound to be used by a composer for his/her music (cf. [Xen85], p. 180). He finally realized this idea in an extreme form of “automated music”, where the composer sets up a computational procedure that is responsible for the entire acoustic shape of a composition, with his ingenious GENDYN program. It is this rigorous aspect of computational sound creation “out of nothing” that much differentiates Xenakis’ practice from that of IRCAM or GRM, where computers are used as tools for

sound analysis, live electronics, and automation of preconceived compositional concepts.

*GENDY3* (1991) owes much to the idea of an acousmatic music because it is pure loudspeaker music. Another attitude Xenakis shares with GRM composers is the fact that the ear of the composer is the absolute and final arbiter. However, one aspect of it goes beyond acousmatic aesthetics. This aspect is Xenakis' stress on the computational aspect of sound synthesis. A "typical" acousmatic composer would not care too much about synthesis because for him it simply does not matter how the sound is produced. Recall that in acousmatic aesthetics, sound finds its compositional value by being detached from its origin. Therefore, the mode of working in the "concrete" studio is empirical in the emphatic sense, as opposed to the conceptual approach of Xenakis.

*GENDY3* is the numeric output of a computer program. This is in stark contrast to the typically iterative, step-by-step process of building up a piece of "concrete" music. On presenting *GENDY3* at the ICMC 1991 in Montreal, Xenakis made the algorithmic nature of the music explicit by describing it in a note which is to be found in the proceedings (later reprinted in his book "Formalized Music" [Xen92d]). Xenakis obviously wanted the listeners of the premiere to perceive *GENDY3* as an example of Rigorous Algorithmic Composition. I would even go so far as to say that the computational nature of *GENDY3* is an integral part of its aesthetic substance. Thinking of *GENDY3* as an experiment to replace a human composer by a computer would not do it justice. Instead, the aesthetic value of *GENDY3* is that of a music idiomatic to the computer, in the sense that an algorithmic concept of composition, realized in the operational form of a computer program, creates a computer music which is artificial in a literal sense, a music invented from nothing, crafted and shaped in its microscopic structure with the help of a computing machine.



## Chapter 5

# A Brief History of Algorithmic Composition

“Machines do not allow their creativity to be frustrated by conventions. [...] The machine is totally devoted to its task. [...] It lives its fate, without doubts or hesitations. This is the ideal that many human persons aspire towards. Now if they loose faith in this ideal, and they want to indulge in neurotic, depressed, and desperate feelings, they should certainly listen to the music of other human persons. But if they want to bring out the best in themselves, they should listen to the sound of machines for inspiration.” [pHH96], p. 156-157

This chapter is not a history of algorithmic computer music. (For more complete accounts on this subject, see e.g. [Loy89], [Hil81], [Dav96], [Ass99], [Ame87], [Bur94a]). The purpose of this chapter is to give a historical perspective to our discussion of different aesthetic and technological approaches to algorithmically generated music. Naturally, this perspective will be focussed on the context of Xenakis’ activities. Specifically, algorithmic composition is understood here in the very particular sense of non-standard conceptual approaches to sound composition, as established during the discussion of the previous chapters.

To begin with, it is interesting to review some accounts on the history of algorithmic composition. Kristine Burns [Bur94a], for example, subsumes under the heading of “algorithmic composition” rule-based composition, automated composition systems and compositions assisted by the computer without any further differentiation. Gérard Assayag [Ass98], on the other hand, sees automated composition as an early historical stage which he presents as a mere precursor of current computer assisted composition technology.

I neither subscribe to Burns’ nor to Assayag’s view. First, my definition of algorithmic music is much narrower than Burns’. Second, I do not regard automated composition as a preliminary stage in the historical development of computer music. The emphasis on the machine nature of algorithmically generated music seems to me much more of a challenge than the imitation or mechanization of human composing.

## 5.1 Prehistory

Kristine Burns [Bur94a] considers automatic instruments and rule-based methods of composing as a preliminary to algorithmic composition. Therefore, according to her, the (pre-) history of algorithmic composition begins in Antique Greece, with the Aeolian harp as an “automatic instrument”. The first rule-based composition procedure named by [Bur94a], p. 35 is Guido of Arezzo’s mnemonic solmization system which has elsewhere been epitomized as a “lookup chart for generating pitches from syllables” ([Sch93], p. 1). Other evidences of rule-based composition systems cited are Athanasius Kircher’s *Arca Musarithmica* (1600) ([Bur94b]), Samuel Pepys’s *Musarithmica Mirafica* (ca. 1670) ([Sch93], p. 1), and Mozart’s *Musikalisches Würfelspiel* (1787) ([Bur94a], p. 36).

## 5.2 The American Pioneers

Mechanic or electronic composing machines aimed at composing popular tunes were designed by John Scott Trotter (1948) ([Bur94a], p. 40), as well as Harry Olson and Herbert Belar (1951) ([Bur94a], p. 40, [Roa96], p. 825). The first computer program composing a piece of popular style music, known as “Push Button Bertha”, was written by Martin Klein and Douglas Bolitho (1956) ([Ame87], [Hil70]). Other programs producing popular tunes were R. C. Pinkerton’s “banal tunemaker” (1956) ([Ass98], p. 9) and J. Cohen’s and J. Sowa’s GENIAC, a “machine that composes music” (1958) ([Roa96], p. 827).

It was, however, the *Illiad Suite for String Quartet*, a sequence of four pieces generated with a program written between 1955 and 1956 by Lejaren Hiller and Leonard Isaacson ([HI58]), named after the University of Illinois and its computer (ENIAC) which was to become the classical reference for the first computer music composition. This is probably because Hiller and Isaacson had the ambition of emulating “serious” music, conforming to rules of classical counterpoint as laid out in Fux’ *Gradus ad Parnassum* ([Bur94a], p. 43). Random numbers, representing musical parameters such as pitch, duration, or articulation (playing modes on string instruments), are subject to a selection by a set of rules coded into the program, each representing a distinct musical style (e.g. from simple and complex counterpoint to “dissonant chromatic music”). A different technique, also aimed at stylistic emulation, was to use Markov chains of  $n^{th}$  order to create “a simple closed musical structure containing simple modulations and a movement towards a final cadence” ([HI58], p. 12-14). These early American efforts have been cursory described as “stylistic control through random sampling and testing” ([Ame87]). The early MUSICOMP program (1963), developed by L. Hiller and R. Baker was a software package which offered the routines developed so far, plus some more, to a growing community of visiting composers at the Studio for Experimental Music at the University of Illinois, among them Herbert Brün who later developed his own music synthesis approaches.

It is against this background of early pioneering work with the computer that Xenakis claims his early “ST” composition program, elaborated between 1958-1962, to have been the first computer program designed for the creation of “absolute” contemporary music ([Var96]), that is, music that challenges the

listener's attention for its musical rather than its experimental or technical value. Indeed, the pieces composed with the ST-program are still performed in concerts up to the present with considerable success, while Hiller's experiments are today of only historical interest.

Another important pioneer in algorithmic composition is James Tenney, who wrote his PLF2 program (1962) at Bell Laboratories, Murray Hill, New Jersey, the same institute where Max Mathews first made the computer produce digital sound (see below). Tenney explored random sequences of tones, mean values and their interpolation, and independent draws within given intervals to define each musical event ([Bur94a], p. 58).

### 5.3 “Project 1” and “Project 2” by Gottfried Michael Koenig

Within this brief history of Algorithmic Composition, the name of Gottfried Michael Koenig shall not be missing. Koenig is the author of two programs of computer composition which have been widely used by himself and others. As may be suspected from the foregoing chapters, Koenig's approach to composing with the computer is of a very conceptual nature. His programs did not only serve to compose music but also, as it were, to instruct composers about what it means to compose. We shall not describe the structure of these programs here. For a detailed account, see e.g. [Roa96], pp. 838-40.

From 1963, Koenig, who had been assistant in the Cologne WDR electronic music studio, sought to model serial compositional thought with the computer. Koenig's comments about this work are of interest here because they reveal a very rigorous approach to Algorithmic Composition, an approach which is very similar to Xenakis'.

“Only very few composers may be inclined to schematize their work. But when out of sheer curiosity they try to compose with the aid of a computer, they are forced to do so [...] precisely because of the regulated combination of musical processes with results that cannot always be foreseen. These contacts between the composer and the musical material forcibly lead to reflection on the craft of composing, and to the formulation of new musical ideas.” (Koenig, quoted after [Roa96], p. 838-9).

Koenig did not only use the computer as a tool to produce music but engaged in a process of theoretical reflection about what it means to compose, being confronted with a machine that is capable of modeling compositional thought.

“Project 2 is also intended to be used for research into compositional theory. It is meant to answer the question as to what in music can be programmed, and to what extent rules of serial or aleatoric music, whether already formulated or capable in individual cases of being formulated, are valid.” (Koenig, quoted after [Sup97], p. 79).

Koenig's statements are quite similar to the statements of Xenakis on his early ST-program. Like Xenakis, Koenig used the computer to investigate into the fundamental possibilities of a conceptual computer art. Like Xenakis, he was

curious how far he could go with formalizing and automating the composition of music. Like Xenakis, Koenig reserved to himself the right to intervene into the automated processes and to review the results of computation. This means that both composers found it interesting to formalize the rules of composition but also found it impossible to fully computationally model their compositional intuition and aesthetic judgment. This is to say that they clearly realized the power and the limitations of computation in the artistic domain.

## 5.4 The “French School”

French engineer and composer Pierre Barbaud is known to be the first European to have created music with the computer. As early as 1960, he created a composition called “7!” (read “seven factorial”). He modelled tonal and serial music styles by implementing rules on the set of the 12 pitches with the help of finite state automata and stochastic matrices ([Bar66]).

Assayag regards the pioneering work of Barbaud and others, R. Blanchard, Michel Philippot, André Riotte and Marcel Mesnage, as the foundation of a “French School” of computer assisted composition. He points out that their interest was to integrate a “general formalization of music theory” ([Ass98], p. 10). Indeed, their contributions seem to stress the systemic properties of composition and their rigorous mathematical analysis and modelization, a preoccupation which seems much in line with French Cartesian thinking. This particular French flavor of computer music research has partly been institutionalized within IRCAM as the “Formes Group” which later became the “Musical Representation” team, led by Gérard Assayag. This group is still one of the most interesting working teams within IRCAM.

Another important proponent for Algorithmic Composition in France nowadays is Paris-based American composer Tom Johnson. His conceptual approach to Algorithmic Composition is perhaps the most radical possible.

“The composer is the automaton itself, and I do not wish to add subjective messages of my own, but simply to interpret, to find the arrangement and colors that allow the automaton itself to be heard as clearly and naturally as possible.” [Joh98], p. F1-2.

On the background of these contemporary activities, Xenakis must be considered a dissident of the French school. He never had the ambition to model existing music theory with the computer. Quite on the contrary, he demanded that part-writing and voice-leading be abolished and replaced by the concept of masses of sonic events, where the sonic events were considered to be independent from each other, i.e. not linked by traditional categories such as chords or voices ([Xen55]). His ST-program is the computer implementation of his original idea of a Stochastic Music, where the sonic events are distributed within the sound masses by means of stochastic laws, i.e. probability distributions. On the other hand, Xenakis never had intent to musically exemplify mathematical formulas, as Tom Johnson humbly pretends to do. His primary interest was the music, and he was willing to sacrifice mathematical purity in favor of the musical end product.

The French School of algorithmic composition, while having no direct influence on Xenakis, might nevertheless have stimulated Xenakis’ own theoretiza-

tion. The specific French cultural climate of music research and experimentation, going back to Pierre Schaeffer's activities (who coined the term "recherche musicale"), compels composers and musicians up to the present day to endow their artistic activities with all sorts of scientific legitimacies. This tendency to produce, in addition to works of art, Ph.D. theses and theoretical treatises, has also affected Xenakis. He published a collection of his seminal writings as early as 1963 ([Xen63]). Musical research, which was at its peak during the 1970s, is still highly funded in France, more than in any other European country (cf. [Bor95]). There are dozens of small research institutes all over the French territory. One of these was CEMAMu, Xenakis' own research institute.

Almost all of the algorithmic composition procedures mentioned so far produced scores to be interpreted by human instrumentalists. But very early, computers also learned to produce musical sound.

## 5.5 Digital Sound Synthesis

Algorithmic composition, because of its historical roots, has always been, and still is, widely associated with computer generation of musical scores. But algorithmic composition can extend to the making of sound itself. Virtually every written account about the history of computer music refers to Max Mathews as the father of digital sound synthesis. In May 1957 at Bell Laboratories, Mathews generated digital sound with the help of an "acoustic compiler" from written instructions. This paradigm adopted by Mathews dominated sound synthesis and computer music for decades after. It was essentially the idea to simulate electric circuitry in computer software to be able to use it in a much more flexible, universal and sophisticated manner. This thinking in synthesizing units makes Mathew's approach a natural continuation of the Voltage Controlled Studio (see below) into the computer era. Ever new generations of "acoustic compilers" were developed, known as the "Music N" family, where "N" stood for various roman version numbers, the most famous being "Music V".

The first piece produced with Mathew's program was "In a silver scale", composed by Newman Guttman ([Pie92]). However, John Pierce, one of Mathew's first collaborators, acknowledged that computers did produce sound before, by means other than digital synthesis, e.g. by "connecting loudspeakers to computer registers" ([Pie92], p. 9).

Probably the very first computer software producing sound for musical purposes, prior to Mathew's digital synthesis, was developed for Australia's first computer CSIRAC between 1951 and 1953. It was only recently unearthed and reconstructed by Australian composer and computer scientist Paul Doornbusch in 1999 [Doo00]. Pierce's argument for Mathew's pioneering role in computer synthesis is that his was "the first deliberate step towards using a computer to produce complex sounds for musical purposes" ([Pie92]). But the same argument holds for the CSIRAC experiments, which therefore deserve to be considered the very first instance of artistic computer sound known so far. CSIRAC was made to play popular tunes, e.g. a Chopin march, and an aria from Händel's *Messiah*, but also a tune specifically written for this machine ([Sin99]). Sound synthesis on the CSIRAC worked by sending digital pulses to a computer register connected to a loudspeaker. Digital to analogue converters which would have permitted to think of sound in terms of waveforms were not existent at

that time. Therefore, the production of musical tones was to a great extent a matter of timing of the digital pulses. This extensive working in the time domain of sound, later to be marginalized by Mathew's successful model of digital oscillators, seems to be somewhat an anticipation of techniques used later by the composers of the Utrecht Institute of Sonology, like Koenig, Berg, Holtzman and others. It can be described as the idea to drive an acoustical transducer in real time by computer action. CSIRAC's output drove a valve amplifier and made it produce buzzing sounds. What's more, the original purpose of these valve amplifiers was to indicate where the current program run was up to — a very early case of using sound for monitoring and debugging computer programs which is researched today under the heading of “program audification” ([Kra94]). Unfortunately, CSIRAC sound synthesis was ultimately abandoned and never developed beyond its mere imitation of “standard” music.

As we saw in this short historical sketch above, there are historic reasons for the separation of score and sound synthesis in early computer music. They simply have been developed separately by different researchers in different places. Hiller and Xenakis created scores with a computer but had them played by string quartets and other ensembles. On the other hand, Max Mathews generated sound with the computer but fed hand-written score data into his program in order to make music.

## 5.6 Voltage-Control and Studio Automation

A first step towards a possible automation of electroacoustic music composition, ultimately leading to the use of computers and the digital music studio, was the development and refinement of voltage control in the analog studio. Originally, voltage control was used for the automation of routine studio work. Standard studio devices like filters, oscillators etc. were equipped with input jacks in order to accept a patch cord for remote control. The control voltage for this remote control then could come from any device producing voltage on its output: oscillators, generators, or even magnetic tapes. More and more custom devices were built this way. A very important application of voltage control, for example, of an oscillator is the control of its frequency by another oscillator. This is called Frequency Modulation (FM), and is a very powerful sound synthesis technique.

The technique of voltage control was used for the design of the first modular synthesizers and hybrid systems (e.g. the CEMS system [Cha97a], GROOVE system [Roa96], and Robert Moog's synthesizers [Roa96]). That means that the voltage controlled components were integrated to form an electronic instrument that could be played through keyboards and other control devices.

Standardization and music industry had it that synthesizers became mere music instruments, for the production and sequencing of sounds. However, much in line with the spirit of heretical use of technology, the idea of voltage control was also taken by composers far beyond its original purpose of studio automation. A prominent proponent for the artistic use of voltage control was the studio of the Institute of Sonology at Utrecht. The idea was to build higher-level systems out of a number of voltage controlled components. This was used not only for the production of sounds but also for the definition of the shape and the timing of musical processes. It is fair to say that algorithmic composition has been practised before the introduction of computers into the electroacoustic

music studio, by custom setups of analog voltage control devices. In this sense, it is possible to conceive of a voltage controlled studio as a specialized analog computer, as evidenced by the following statement.

“The term analogue computer is a better name for voltage-control [...]. Viewed in this way it becomes clear why the patching of devices, namely the program, is the description of the schemes which govern certain calculations. Thereby, composing becomes designing of programs, and programs, in turn, consist of one or several algorithms.” [Rei93].

The patching schema of oscillators in the notorious Yamaha DX7 FM synthesizer are still called “algorithms”. The abstract notion of “patches”, that is, wired interconnections, has been preserved in the design of the most popular and widely used software Computer Assisted Composition systems, e.g. Patchwork and MAX (initially named “Patcher” [Puc88]).

Voltage control has been used in the 1960s and 70s in electronic music studios all over the world, in a time before computers were affordable to these institutions. In few studios, however, the use of it was as systematic and conceptual as in the Utrecht studio. They developed a dedicated device that could be used as a kind of electronic score for a music production process, the “Double Variable Function Generator” (cf. [Sup97], p. 67). Gottfried Michael Koenig composed a series of works called “Funktionensstücke” (“function pieces”) with this function generator, generating voltage step functions in a time scale reaching from seconds to milliseconds, thus covering the continuum of musical time from the macro- to the microsound domain. The music of the Utrecht school in this respect was one of the most innovative and uncompromizing in the world.

Xenakis promoted the idea of Automated Music since the very beginning of his career, as early as 1957 or possibly even before (cf. [Xen57]). And he turned to computers much before they were available for the music studio. His first program for composing music scores, developed between 1958 and 1962, ran on an IBM mainframe computer at IBM headquarters, Place Vendôme, Paris. Much in contrast to other electronic music studios in Europe, French electroacoustic studio practice was dominated by the approach of music concrète, where algorithmic attitudes to composing would not easily fit. Xenakis always preferred brute, natural sounds over periodic functions from oscillators, and it is known that he despised the use of synthesizers altogether. He may not have learned of or even understood the advanced use of voltage control for building cybernetic systems, that is to say, analogue music composing robots, like it was done at Utrecht. Being mathematically trained and analytical in his approach, Xenakis chose the computer to apply compositional thinking to sound. But this is not to say that algorithmic music must necessarily be implemented on computers. It can also be done by setting up a sophisticated voltage control system.

## 5.7 Composed Sound

Ambitions to integrate score composition and sound synthesis into a unified framework of sound composition are comparatively rare. Most electroacoustic composers pragmatically choose among various techniques to assemble their computer music pieces. They seem to prefer to combine the best of computer

assisted composition systems and sound synthesis rather than to struggle with a conceptually coherent unified compositional approach to sound sculpting. In their tendency, composers are encouraged by hardware and software engineers who seem to be rather bewildered at attempts of composers to bend the generation of sound to their idiosyncratic artistic concepts.

### 5.7.1 Lejaren Hiller and John Cage

One of the first to combine both “score synthesis” and “digital sound synthesis” with the computer was probably James Tenney who, starting in Hiller’s group at Urbana, later spent a 2 years residency at Bell Laboratories between 1962-1964. He turned the score output of his computer programs into sound with the help of Mathew’s acoustic compiler. However, the generation of sound and the generation of note events were governed by different concepts not related to one another. Other composers desired to create computer sound and overall structure of a piece from one single compositional approach. Hiller, in his 1967-1968 collaboration with John Cage, produced sound and structure using a specific procedure, namely the I-Ching oracle so dear to Cage, for their joint composition HPSCHD.

### 5.7.2 Herbert Brün

Another example for an integrated approach to computer composition is Herbert Brün (1918-2000). Both instrumental and tape parts of his 1964 composition “Soniferous Loops” were produced using the MUSICOMP library of routines developed by Lejaren Hiller and Robert Baker from 1963. The next piece of Brün’s using MUSICOMP for both score and sound production is “Non Sequitur VI” from 1966. In contrast to the former piece where he post-produced the computer generated tape with analogue techniques, Brün used the audio tape as it came out of the computer resp. the D/A system connected to it [Brü83]. From 1973, Brün went one step further with his SAWDUST program. With this program, Brün was able to compose the microstructure of sound by generating sequences of digital sample values with the help of a small set of structural functions called ELEMENT, LINK, MERGE, MINGLE, VARY and TURN. The overall structure of a SAWDUST composition is thus the result of operations and manipulations on the sample level. In other words, the structure of a musical piece comes about as a result of sound synthesis. This fact is also characteristic for GENDYN.

### 5.7.3 Arun Chandra

A continuation of Brün’s work with SAWDUST is Arun Chandra’s systematic research of the impact of waveshaping synthesis on the resultant transient spectra [Cha94]. Contrary to Xenakis’ GENDYN, the waveforms examined by Chandra are composed of only three distinct geometric shapes (squares, triangles and splines). However, the peak values of these shapes vary (similar to GENDYN) and so does the timbre of the resulting sounds. Chandra observed that the resulting fundamental frequency is correlated to the cycle of repetition of the waveforms and that gradual linear change in the waveform’s shape may result in abrupt, non-linear changes in the resulting timbres.

### 5.7.4 Gottfried Michael Koenig

Quite in the same spirit is the idea of sound synthesis in Gottfried Michael Koenig's SSP system (short for "Sound Synthesis Program"), implemented by Paul Berg at the then Sonological Institute in Utrecht. Like in the case of Brün (and in that of Xenakis), sound synthesis in SSP "owes more to [...] composition theory than it does to signal-processing theory" ([Roa96], p. 326). These composers took the artistic liberty to imprint their idiosyncratic compositional thinking onto the sounding matter without caring too much about standards of acoustical engineering. SSP follows two other composition environments designed by Koenig and aimed at the production of musical scores. Many of the rules implemented in these programs "Project 1" (PR1), since 1963 and "Project 2" (PR2), since 1966, have also been applied for the generation of sound itself. In SSP like in SAWDUST, the composer works interactively with a number of elementary functions acting on samples or collections of samples: ALEA, SERIES, RATIO, TENDENCY, SEQUENCE, and GROUP. All these functions are different ways of creating larger chunks of sequences of samples by specifying a small number of parameters, e.g. minimum and maximum value and the number of samples to choose in between (ALEA), the ratio of occurrence of each of them (RATIO), where minimum and maximum values vary over time (TENDENCY), etc. (cf. [Roa96], pp. 326-27).

"[The sound synthesis program SSP] uses what we call the 'non-standard approach to sound synthesis. That means not referring to a given acoustical model but rather describing the waveform in terms of amplitude values and time values. My first intention was to go away from the classical instrumental definitions of sound in terms of loudness, pitch, and duration, and so on, because then you would refer to musical elements which are not necessarily the elements of the language of today. To explore a new field of sound possibilities I thought it would be best to *close* the classical description of sound and open up an experimental field in which you would really have to start again." [Roa85b], p. 572

### 5.7.5 Steven Holtzman and Paul Berg

Also developed at the Utrecht institute was probably the most radical non-standard approach to algorithmic composition referred to as "Instruction Synthesis" by [Roa96].

"The sound synthesis technique used in our system [...] is non-standard in the following senses: first, the noises this technique tends to generate differ greatly from those of the traditional instrumental repertoire and even from much electronic music; second, in this technique, sound is specified in terms of basic digital processes rather than by the rules of acoustics or by traditional concepts of frequency, pitch, overtone structure, and the like." [Hol79], p. 53

Instruction Synthesis means that sound is produced as the raw effect of machine instructions on a processor's register, rather than as the outcome of a sophisticated numerical calculation. One example of this approach is Paul

Berg's PILE language [Ber79], but there are other examples, one of them even extending to the data processing power of the whole internet ([Nez00], quoted in [Föl02]).

Instruction Synthesis is a recursive process in that it establishes relationships among successive sample values by elementary arithmetic and logic operations (typically the primitives of the ALU<sup>1</sup> within the computer processor, the CPU<sup>2</sup>, using the current result in the accumulator register as an operand (i.e. argument) for the computation of the next result. The current results are interpreted as the current sample values of contiguous sound. Together with a finite list of instructions being executed cyclically, this produces periodic waveforms, where the periodicity can be influenced by the balance between variable and constant values used ([Hol79], p. 55). Aperiodic wave forms are produced by replacing the simple operands by more complicated ones such as a random generator function or arbitrary lookup of values in computer memory. The specific sonic timbres produced in this way are juxtaposed with the help of a superordinated control structure realized in a similar computer "idiomatic" way: generative grammars. In other words, Holtzman strived to generate the scores of Instruction Synthesis, in turn, through programming.

Generative grammars are the productive inversion of the parsing problem. Parsing means analyzing texts in view of their being possible sentences in a given language (e.g. acceptance of a computer program as a valid text in a given programming language). Parsing is also an important concept in linguistic theory.

In spite of the fact that Holtzman stresses the distinction between the two levels of superordinated control structure (by grammars) and sound synthesis (by machine instructions), Instruction Synthesis tends to determine some macroscopic aspects of the music, such as silences, rhythmic structure (i.e. infra-sonic impulses and structures of the generated signal), and also any resulting pitch. This means that macrocomposition comes about as a by-product of microcomposition, which is also typical for Xenakis' GENDYN. This aspect has been described by Agostino DiScipio as the concept of "Sonological Emergence" ([DiS94]).

"In much the same way as subsymbolic paradigms of cognition try to capture how sensorial data are mentally pre-processed to constitute a symbol, and then how symbols are treated as components of higher forms of organization, a holistic approach to composition - understood as a theory of sonological emergence - may find it appropriate to describe its models of sonic design in terms of subsymbolic processes yielding the musical structure we experience by listening." [DiS94], section 3.3

It is possible to regard Berg's or Holtzman's activities at Utrecht as an early example of what later became *program audification* (cf. [Kra94]). Audification is the attempt to make computer programs sound e.g. in order to help a programmer or a system operator to better detect errors or anomalies in a complex software system. Again, we can clearly see here the idea of heresy in bending a (now) standard engineering technique to artistic goals: instead of monitoring

---

<sup>1</sup>*Arithmetic-Logical Unit*

<sup>2</sup>*Central Processing Unit*

existing algorithmic processes with the aid of sound, artists like Holtzman and Berg, and later Xenakis, were interested in creating new algorithmic processes in order to obtain new sounds. But Instruction Synthesis goes even beyond program audification because the sound does more than illustrate program action: it is the direct manifestation of program action in the audible range. The program text can be considered the score of the music in a strong sense (cf. [Roa96], p. 328) and the computer the performing instrument, because the sound is directly dependent on the timing of the instructions and the computer's internal clock frequency, therefore both software and hardware dependent.

### 5.7.6 Xenakis' UPIC

Another instance of a unified approach to computer composition is Iannis Xenakis' UPIC system (short for "Unité Polyagogique Informatique du CEMAMu"), from 1977 [MRS93]. The idea is that everything in composition may be achieved by working out geometric shapes in the time domain: waveforms, amplitude envelopes, and pitch arcs.

This concept, also called "graphical synthesis" ([Roa96], pp. 329-330) is reminiscent of early attempts at optical synthesis started in the early 20th century ([Mor93], see also the picture on the first pages of this study). In this context, a very interesting entry in Burns' list of electronic instruments is a patented machine by Pery Grainger and Burnett Cross (1944) which was to turn projected notation into sound with the help of photo-sensitive graph paper and eight oscillators ([Bur94a]). This seems to somehow anticipate Xenakis' UPIC system.

The figure 5.2 on page 79 shows a snapshot of the workspace of the UPIC system. Many musical objects (lower left: waveforms and envelopes) are attached to lines drawn with either the computer mouse or a graphic tablet on a "score page" (upper left: a very elaborated example with many fine-grained musical events). A control panel (lower right) helps editing the different configurations. The page is played through the play control (upper right) in temporal dimensions to be freely chosen in the range between some seconds to several hours. Any modification of objects, page or controls by the user during playback is immediately reflected in the playback and allows the composer to develop a powerful intuition of composing the micro- and macrostructure of sound.

However, sound generation on the UPIC system has its limitation in that it "only" implements (an extended notion of) wavetable synthesis. In order to produce lively sound with the UPIC, sophisticated "non-standard" techniques had to be developed by those composers who were confronted with this system (cf. [Papad]).

It is interesting to compare the UPIC system with the GENDYN program. They are in many aspects complementary, and it seems as if one concept compensates for the shortcomings of the other.<sup>3</sup>

Let us take a closer look at some aspect of comparison between UPIC and GENDYN.

**CAC vs. automated composition.** UPIC is made for explicit definition of

---

<sup>3</sup>Indeed, it could have been interesting to integrate GENDYN synthesis within UPIC. Thoughts have been given to this possibility (personal communication by Jean-Michel Raszynsky of CEMAMu in 1996) but have, as of now, not been realized.

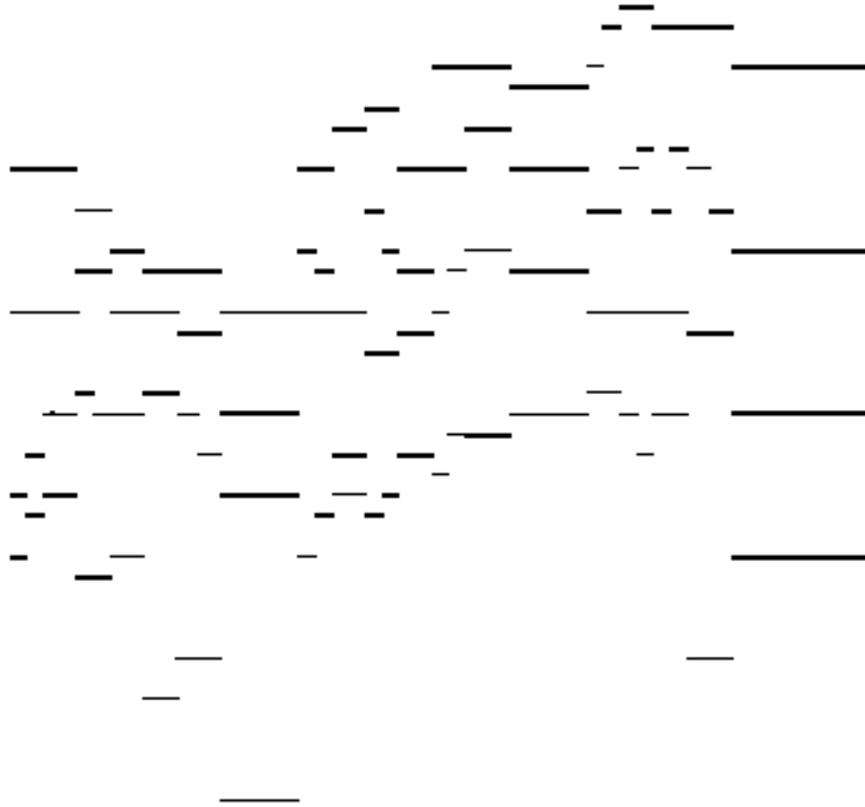


Figure 5.1: A UPIC page representing Bach's chorale "Jesu meine Freude" in "piano roll" notation (every note is represented as a "pitch arc" i.e. a graphical plot of pitch over time). Note the visual poorness of the plotting typical to the concept of a finite set of fixed pitches in classical Western music. Compare this with figure 5.2 on the facing page below.

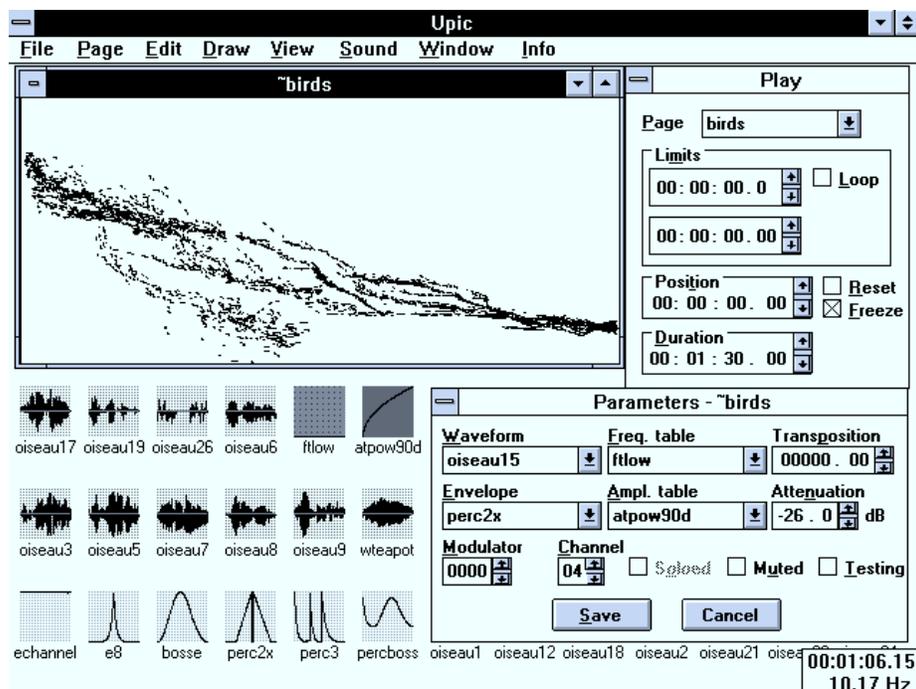


Figure 5.2: The composer's workspace of UPIC 3 on a PC screen, showing the data of a piece by French-American composer Brigitte Robindoré. Note the richness (both in number and form) of the pitch arcs in the page window.

score and sound structures by the unified approach of graphic design. Therefore, UPIC is a (no matter how idiosyncratic) CAC system. In contrast, GENDYN is automated composition: the sound is not designed but is the result of stochastic fluctuations acting on an original waveform (in GENDYN, the original waveform is the degenerated, “flat” waveform of perfect silence). Therefore, composition with GENDYN is implicate: the composer can only define the side conditions of quasi-autonomous stochastic processes.

**Dual paradigm vs. Sonological Emergence.** UPIC is designed for integrated score and sound synthesis by the unified approach of graphic design. But it separates conceptually score and sound synthesis as two hierarchic stages of the definition of a musical piece (composition of a page, then assigning sounds to arcs). In contrast, GENDYN incorporates aspects of score composition (e.g. pitch and dynamics) into the generation of sound. By the algorithmic definition of the sonic micro-structure, (“micro-composition”), aspects on a larger timescale which are usually associated with the score level (“macro-composition”) are also defined through a process of Sonological Emergence (cf. section 5.7.5 on page 76).

**Static vs. dynamic timbre.** With the UPIC, everything is under control: the sonic result is perfectly determined by the graphics, even if the structure of the graphics may be complicated. In contrast, with GENDYN, the synthesis cannot be completely controlled by the composer’s directives because the probabilities integrated into the program give it a kind of independent “life” which escapes all rigid control. As a compensation, GENDYN produces rich spectra which evolve in time, while the UPIC, if not forced otherwise by an experienced composer, has a strong tendency to produce rigid, fixed spectra.<sup>4</sup>

However, the above statements are relativized by one of the most radical uses of the UPIC in recent times: Angelo Bello’s excessive use of frequency modulation with feedback on the UPIC (1997). With the UPIC arcs, Bello would set up a complex FM “algorithm” with multiple feedback loops among UPIC’s 64 hardware oscillators, sending the UPIC onto the road to chaotic oscillation [Bel98]. The chaotic journey of the UPIC oscillators through the feedback loops can then be influenced by the composer/interpreter in real time through UPIC’s graphical control interface, where pitch and intensity control take on a new meaning as frequency and depth controls of the modulation ([Bel01], [Bel97], [Bel99]). In this extreme case the UPIC, quite at odds with his original design as a graphical CAC tool, has the capability of automated composition. In other words, the UPIC has been transformed by Bello from a CAC tool into a RAC tool, by a deliberate act of heresy.

“I realized something about the way I use the UPIC: I could in fact create the FM algorithms with single pixels on the page of the screen (instead of the arcs and lines that are often used). That is, I do not need to scan a ‘score’ with the cursor, as is traditionally

---

<sup>4</sup>Composers who work with the UPIC on a regular basis have found specific techniques to generate complex and vivid sonorities with it, sometimes by making use of the far-reaching capacities of frequency modulation which has been added since the 1991 version (c.f. [Paped]).

done - my cursor remains fixed. The UPIC page becomes a sort of catalogue of algorithms that I can access for sound synthesis. Then perhaps I could organize algorithms to create a more complex score, and then use the cursor. Xenakis' techniques and methods (knowingly or unknowingly to him) have really opened up pandora's box of new ways to consider composing!" (Angelo Bello, personal communication, June 12, 1997)

### 5.7.7 Jacques Chareyron

Another interesting example of a non-standard approach to composing-the-sound is the *Linear Automata Synthesis* (LASy) by [Cha90]. In this case, the samples forming a waveform are made to interact like cells in a one-dimensional ("linear") cellular automaton. The rule of the automaton determines the change of the sample values from generation to generation of the automaton (i.e. iteration of the waveform). The recursive formulation of the automaton's rule is reminiscent of the technique of digital filters. Indeed, one special case of a digital filter, the Karplus-Strong algorithm for the simulation of a plucked string can be modelled as a special case of a cellular automaton working on a waveform's samples. Other rules are less deterministic, and especially one class of automata rules (identified in the theory of Cellular Automata as the universal, irreducible ones) seems most promising for unexpected sound evolutions which may be differentiated enough to catch a musical ear.

## 5.8 Computer Assisted Composition Systems

The term "Computer Assisted Composition", or CAC for short, evokes the analogy to terms like "Computer Aided Design" (CAD) or "Computer Aided Manufacturing" (CAM) in industry. Therefore, it implies the computerization of all manual aspects of composing.<sup>5</sup> The score is digitally represented and all manipulations like editing, arranging, instrumentation, layout and printing are accomplished by software tools working on the digital score representation.

Computer Aided Composition has been defined to meet three requirements (cf. [Cas92]):

**Event specification and editing.** This is the "declarative representation" in the sense of [MWSH93]. Event in the context of CAC means a discernable entity of music action. In a more technical sense as standardized by the MIDI specification (see below), events are timed changes in the state of a performance instrument, like note-on (e.g. a key pressed down) and note-off (key released), etc. It is doubtful, however, if the notion of "event" can be applied to electroacoustic music in general where there might be constant sonic evolution within a continuous space of sonic parameters.

**Procedural specification.** This is the definition of the temporal development of a piece of music, the change of its aspects in time. This can be done explicitly by devising a descriptive (or declarative) representation of the

---

<sup>5</sup>By coincidence, the same term also denotes "composition" of electronic or hyper-structured text.

temporal evolution of the music, or by implicit coding of the musical structure which becomes only explicit when invoking a process (see eg. the description of the Common Music CAC system by [MWSH93]).

**Artificial modeling of musical structures.** This is where a specifically computational aspect might come into play. Since the score (or even the sound) is digitally represented it can be subject to arbitrary manipulations with the help of formal structure-generating models, like random processes, simulation of deterministic chaos, fractal shapes and outlines, generative grammars or any other computational model.

It is important to note that CAC systems are not specifically designed to automate the process of composition as such. The computer might just help executing manual tasks, being guided by the composer. In that sense, working with a CAC tool does not necessarily imply any specifically computational aspect in composition.

However, CAC systems facilitate a computational approach to music composition in that they build on computational primitives that allow flexible recombination and the development of individual solutions. A good CAC system invites the composer to stop being user and take part in the design of the system in a participatory manner. This is where the distinction between CAC and algorithmic composition becomes blurred.

There is, indeed, a tendency to describe techniques in working with CAC systems under the heading of “Algorithmic Composition” (cf. [Ort96]). Yet, I cannot subscribe to this idea. The complexity of a CAC work of art is usually not generated by algorithmic action but by human guidance. But the definition of Rigorous Algorithmic Composition as established above implies that the complexity of the music must be entirely generated by the computer. In Rigorous Algorithmic Composition, complexity and surprise have to be encoded in the algorithm. They do not come from spontaneous decisions taken by the composer.

There seems to be an open agreement within the community of computer composers that the quality of CAC systems does crucially depend on their design. Paradoxically, the question is not that of an optimal design best suited to musical purposes. Such an optimal design cannot exist in contemporary music since there is no more codified procedure or style of composing. The best design of a CAC system is one which is not present. It supports and invites for the building of a personal design by every single user.

Technically, CAC systems are characterized by their “language-oriented” approach. Some examples are Common Music ([Tau91]), PatchWork and OpenMusic ([Ass98]) or SuperCollider ([McC96]). The idea behind a language for representing music is that an infinitude of possible structures can be expressed with the help of a finite alphabet. It is quite obvious that a CAC language designer cannot foresee all possible uses of his/her system. Since a system cannot incorporate all extensions and modifications that may be desirable in the future, large parts of such systems are devised as extendable libraries that evolve with the system’s use over time. To ease the interaction with this abstract representation of musical data, many CAC systems are equipped with powerful graphic user interfaces. There is also a tendency to integrate artificial intelligence features in CAC systems such as repositories where expert knowledge

is accumulated, and efficient retrieving and reasoning procedures in order to activate it when using the system. See [MWSH93] for a broader survey.

It could in principle be possible to use a CAC system for Rigorous Algorithmic Composition. This would mean, though an act of “heresy”, to divert this system from its potential role in “industrial automation” of music production.

“I would like to extrapolate the urge for openness [of contemporary compositional approaches] to such a programming environment and adding the demand for transparency in order to allow for the human to gain insight into all active processes, so he/she would be able to intervene in an interactive way. Such an individual, rhizomatically proliferating system would eventually be more than a mere tool. It could – as an utopia – reflect the current state of individual compositional thinking.” [Ess91]

However, every CAC system makes assumptions about the nature of sonic entities and their structuring in time and therefore presents limitations to non-standard approaches of sonic design. For example, several attempts have been made to implement the core algorithm of GENDYN, Dynamic Stochastic Synthesis, within CAC systems like SuperCollider [McC96], MAX/MSP [Dob97], and others. In most cases, difficulties were encountered when trying to gain access to the sample level of sound synthesis because all these systems have their own assumptions of sound representation which makes it hard to use them for non-standard approaches like GENDYN.

Conversely, it is obvious that GENDYN cannot be an example of a CAC system. It falls short of all three requirements enumerated above.

**“Stochastic Architecture”.** There is no direct event specification, nor can the stochastic duration structure be “edited” (it can only be recomputed until a satisfactory result is reached).

**Stochastic Evolution.** The only procedure to define musical evolution is through the sound itself, as a byproduct of dynamic stochastic sound synthesis; there is no other procedural specification possible.

**Closedness.** No alternative modeling of structure-generating processes is possible. Either the results are accepted or dismissed. This cannot be changed since GENDYN is not an open programmable system.

We have already stated that artistic use of computers obeys different laws than the rules of engineering. This is a reason that speaks against using standard CAC systems in computer composition. Standards for computer composition, e.g. the oscillator/filter/mixer paradigm of the so-called “Music-N” approach now codified in the MPEG-4 Structured Audio [Sch98], codify inherited patterns of standard electroacoustic studio production which are not specific to the technology of computing. Institutions and commercial companies are trying to impose their standards and software solutions onto the computer music community. The computer, in its ability to assimilate almost any existing technology, is helpful to spread and proliferate standard computer music paradigms where instead it should be used to promote idiomatic approaches to computer composition. As we have seen in our discussion of non-standard approaches, a really

valuable use of the computer can only take place where a composer either does his/her own programming or uses existing software in an innovative way never thought of by its designers.

## 5.9 Interactive Music Composition

With the advent of fast computing machinery, computation of music scores and electronic processing of sound can be done by a computer in “real time”, i.e. while playing the music. This can be extended to interactive composition on stage. Interactive composition means that a composer/performer reacts to musical processes generated by the computer, thereby changing these processes or triggering new ones. Within the context of this study, we will not be interested in the specificities and technical details of interactive music systems, nor in enumerating composers doing live composition. Rather, our interest are those implications of this practice which pertain to an aesthetics of a genuine computer art.

In a strict sense, all compositional work with a computer must necessarily be interactive. Even in the case of Rigorous Algorithmic Composition, when there is no interaction while the music is being computed, the algorithm of composition has been worked out in an extensive cycle of software development on the one hand and permanent review of the musical result on the other.

“The composer [...] thinks, he executes, he corrects, he corrects himself, he establishes a dialogue with himself. And his ears. And his thinking. And this, of course, could never had been possible without the computer.” [Del97], p. 86

The idea of an interactive dialogue with an externalized compositional instance has also been put forward by Otto E. Laske who has much reflected on the use of computers in music composition.

“I take computer programs to be objectivations or selected parts of the composer’s own knowledge. I conceive of such programs as the composer’s *alter ego*, without which he remains bereft of the benefit of externalized dialog with self. My view is that dialogue with self in art making is crucial, and that computer programs, rather than being simply tools for achieving results, are dialoguing tools. Moreover, I consider them more as works of art than as tools. This entails that I don’t believe in any linear progress in developing such programs, or in the merit of particularly ‘intelligent’ programs. For me, a ‘stupid’ program that makes a composer ‘intelligent’ is more valuable than an ‘intelligent’ program that hinders him from finding his own voice.” [Las92], p. 181

In his computer music, Iannis Xenakis has lifted, as it were, the task of composing onto a higher level. It is not anymore the writing of individual music pieces but the construction of a sound producing automaton, which is capable of creating interesting music. The fact that the act of composition itself is executed by the automaton does not at all diminish the achievements of the composer.

Quite the opposite, the merit of having delegated one's own compositional decisions to an automaton in a way that it is capable of creating results which are musically valuable, is somehow much more difficult a task.

“The necessary formalization forces the mental penetration of the compositional issues involved. [...] This includes the process of coding musical facts into a more abstract notation which can be processed by the computer. And exactly because this translation step is an interpretational one which entails a semantic shift, it makes it possible to discover new aspects of the musical material which are beyond its inherent significations.” [Ess91]

If a human succeeds in breathing artificial life into a machine, then he is reaching, as it were, into the realm of Creation. The urge for creativity seems to be present in all humans since birth. People try, on a small scale, to emulate a creativity which they themselves owe their own very existence.

The fascination of the computer is to a great extent its potential to make up a kind of *vis-à-vis* with whom one is able to engage in a sensible dialog. But the use of the computer in Iannis Xenakis' music is not an end in itself. The computer helps the composer to assure himself his role as a creative subject in that he renders the creative act, as it were, exterior to himself and transfers it to a *vis-à-vis*, making it objective. The computer helps him to calculate a sounding proof for the validity of fundamental compositional ideas like, in his case, composing by probabilities.

We have seen that in the case of computer music, the culture of writing transforms into a culture of programming. Unlike a written score, a computer program is no direct representation of the music but an algorithmic reduction of it. In order to assess its potentials, the program has to be executed and re-executed while studying its results.

“The composer is not a detached engineer, setting a machine into motion and let it run but he himself is part of the system. His intentions, wishes, aims and aesthetic orientation are reflected by the system as well as his ability to draw consequences from the various constellations generated by the computer which enter the system and possibly modify its behavior. Conversely, the subject's perception is modified by the output of the computer as his 'objective' counterpart. A feedback [...] is thus established; the computer becomes a control instance appointed by the composer and which he interactively communicates with. [...] Thus the subject becomes the central integrating figure, drawing from the collision of the system's components a creative potential and achieving a synthesis which cannot be accomplished by the machine alone. [...] The artistic subject encompasses the whole composition process as a first and a last instance: he/she sketches a model, analyzes its output and thereby gains new insights which may lead to reformulating the approach or to modify the model as such.” [Ess91]

Now, there are different units and modes of interaction with the computer. One extreme is interactive composition in real time. The opposite extreme is what computer composer Curtis Roads calls “the batch approach to automated

composition” ([Roa96], p. 845-6). In batch mode, the “unit of composition and interaction is the entire score” ([Roa96], p. 846). This means that the entire composition process must be re-executed, after e.g. changing program logic or an input variable. This is what Xenakis did with GENDYN.

Xenakis constantly rewrote his program, executed (parts of) it, listened to the results, and rewrote the program further. This procedure corresponds to the batch approach of computer composition. Xenakis produced a great many versions of the music before he generated *GENDY3*. The way Xenakis worked with GENDYN is quite different to what software engineers do. They try to keep programming as general as possible and consolidate a programming version before starting a new one. Xenakis, in contrast, made his program only for one specific purpose, the composition of *GENDY3*, and for each single trial, he made a new altered version of it. Xenakis kept a printed record of his work at CEMAMu (program listings and data printouts) of all those program versions whose results he found interesting. The two mono sound files which make up *GENDY3* are called “S402” and “S403”. This tells us that Xenakis produced about four hundred soundfiles before the release of his first GENDYN composition.

If, however, interaction is done on a smaller scale and during runtime of the compositional algorithm, the composer/performer can intervene in the process of composition, i.e. in “real time”. This is, by the way, what has now become possible with the New GENDYN Program. Xenakis had experimented with interactive composition on the UPIC system, and he might have started to do the same with GENDYN, had a real-time version become available sooner, and had he been willing to engage in activities with the new program.

Interactive composition with computerized systems was started in the late sixties of the 20th century by pioneers like Joel Chadabe and others ([Cha84], [Row92], [Roa96], pp. 828/9).

“Interaction means mutual influence. [...] In musical terms, it means that we influence the instrument that we play and that we are influenced by the sounds that it produces. It means that an instrument has a mind of its own, so to speak, such that it produces musical information that contains surprises.” [Cha96], p. 44

The fact that GENDYN was not destined for real-time interactive use can clearly be seen in its design. For example, the program fixes the number and the temporal order of the sections of the piece. This corresponds to Xenakis’ approach to composing in general. He always stressed the importance of having an overall concept and of fixing things in advance with the help of a representation of the piece written to or drawn on paper. In the case of Rigorous Algorithmic Music, in place of a graphic representation of the piece the composer fixes the computational procedures that lead to the generation of the piece in a program text.

“One must note [the musical idea], one must write it down, if one is to compose. [...] It is absolutely necessary to think it over, and in a more and more conscious way, to be able to correct oneself [...] and thus create a dialogue with oneself. [...] Then, time is without limits, time is frozen, since in design mode one can take all time that’s needed. Whereas while playing, one must be quick. [...] One

must execute, and re-execute, and that's no composition [...] I think this step of analysis and synthesis at a time, which is an intellectual step, but simple and immediate, is a necessary and precious one." [Del97], p. 85

At the same time, we know that Xenakis highly estimated the possibility to do improvisation with the real-time version of UPIC for the staging of *Taurhiphanie* in 1987 [Var96]. He could have used the New GENDYN Program in a similar way. Unfortunately, when the New GENDYN Program became available in 1996, he either did not seem to realize its potential for his own creative goals, or it was too late for him to engage in a new adventure of interactive composition. (He composed his last piece *O-Mega* in 1997 and then remained silent until his death in 2001.)

It is curious to note, however, that two of Xenakis' inventions, the UPIC system on the one hand, and GENDYN on the other, are specifically prone to enable live composition. Several composers have taken the real-time possibilities of the UPIC system, since version 2 (1987, see [MRS93]), further than Xenakis himself did (cf. [Ber92], [Bel01]). The fascination of composers working with interactive real-time systems is well expressed in the words of one of the pioneers of interactive composition:

"I was only partially controlling the music, and the music, consequently, contains surprising as well as predictable elements. The surprising elements made me react. The predictable elements made me feel that I was exerting some control. It was like conversing with a clever friend who was never boring but always responsive. It was, in effect, conversing with a musical instrument that seemed to have its own interesting personality." [Cha97a], p. 287

There are some important features of live composition that make it fundamentally different from composition in non-real time. The live composer cannot go back in time and change what has been already composed. However, he or she can influence, based on the analysis of music that is being played, the composition of music that is to be played. In this way, an interactive feedback loop is established between the composer and the composition program. The dynamics of this feedback loop seems to be one of the main attractions of live composition. On the other hand, performing an interactive composition adds to the difficulties of composition the difficulties of instrumental play: the temporal constraints of time, the mastering of a complex control interface, and the impossibility of review and correction.

In order for the composer/performer to interact with the computer, various custom input devices and controllers have been invented to facilitate control of the musical processes. In the simplest case, the composer/performer uses the classic input devices such as keyboard or mouse. In more advanced research, as is e.g. pursued at STEIM<sup>6</sup>, much effort goes into developing more sophisticated and intuitive human-machine interfaces (cf. [Cha97a], [Cho97]).

Musical instruments, with the help of computing, have developed from mere performing devices to devices participating in the design of the music itself. Therefore, Joel Chadabe speaks of "composing instruments":

---

<sup>6</sup>STEIM stands for *Studio voor electro-instrumentale muziek* in Amsterdam, founded in 1971.

“They make musical decisions, as they produce sound and as they respond to the performer. These instruments are interactive in the sense that performer and instrument are mutually influential. The performer is influenced by the music produced by the instrument, and the instrument is influenced by the performer’s controls. These instruments introduce the concept of shared symbiotic control of a musical process wherein the instrument’s generation of ideas and the performer’s musical judgment worked together to shape the overall flow of the music.” [Cha97a], p. 291

When compositional activity is shared with a computer, what is, then, the role of the composer? Is the composer a kind of Frankenstein who, after bringing life to computer music, is not able to control it anymore? Or can the interaction between creation and computation be more balanced? Can composers and computers be partners in the creation process?

Similar questions arise on the side of the listener. In how far is the fact that music has been composed not only with a computer but in cooperation with a computer relevant to the aesthetic appreciation of this music? To what extent does interaction belong to the aesthetic substance of a musical piece?

Another question when composing with a computer is whether the composer accepts the results of a computation as a whole or just as bricks for further exploitation. In the case of Xenakis’ early *ST* program, the latter was the case. However, when it comes to interactive composition, there is no time for selection because everything is played immediately. (A live composer could possibly argue that he/she foresees undesirable output and simply prevents it from happening.) For his work with the computer, Xenakis always reserved himself the right to chose and alter the results of a computer computation, whereas the purists among computer composers (like French composer Barbaud and American composer Babbitt) argued that for the sake of consistency of their composing methods, the computation had to be accepted as such.

Interestingly enough, in the case of GENDYN, Xenakis took the result of GENDYN computation as it came out of the computer.<sup>7</sup> Therefore, *GENDY3* and *S709*, the two pieces of Radical Algorithmic Composition, could just as well have been composed in real-time on stage.

In part two of this study, we will see that there are theories that interaction extends the power of algorithms beyond the capacity of universal computation. So the limits of computation that apply to Xenakis’ RAC approach may not be valid for interactive composition. We will come back to this idea later.

## 5.10 “Indeterministic” Composition

We have discussed the necessity for a participatory approach when it comes to interactive composition. When the composition process is realized through a process of dialog between Man and Machine, where the machine’s actions cannot fully be predicted, the classical notion of the composer must be changed from the picture of an almighty master to that of a cooperative collaborator who is ready to engage in open-ended processes of exploration.

<sup>7</sup>For the released version of *GENDY3*, Xenakis ran his GENDYN program twice and combined the two resulting sound files to a stereo sound file, with a 80 ms delay between the two.

In this sense, indeterministic models of composition reflect the necessity of an open-ended dialog with an algorithmic system. This dialog will eventually lead to building up a compositional intuition characterized by the specific nature of that system.

In view of algorithmic micro-compositional strategies of composition, the goal of a composer is likely to change from the perspective of the classical composer who preconceives every detail of his/her composition toward a more exploratory attitude. A musical product will be achieved not by imposing compositional thought onto the machine but rather as the result of developing a new compositional intuition in the field of algorithmic sound production by way of a creative dialog with the algorithmic system. In this sense one could speak of the idea of learning how to master a “composition instrument”. It is interesting to parallel this change in the approach to composition with developments in philosophy that led to a modification of the role of the Creator from an omnipotent dictator to a benevolent manager of self-optimizing processes:

“Process thought was introduced into theology by the mathematician and philosopher Alfred North Whitehead, who was co-author with Bertrand Russell of the seminal work *Principia Mathematica*. Whitehead proposed that physical reality is a network linking what he termed ‘actual occasions’, these being more than mere events, for they are invested with a freedom and an internal experience that are lacking in the mechanistic view of the world. Central to Whitehead’s philosophy is that God is responsible for ordering the world, not through direct action, but by providing the various potentialities which the physical universe is then free to actualize. In this way, God does not compromise the essential openness and indeterminism of the universe, but is nevertheless in a position to encourage a trend toward good. Traces of tis subtle and indirect influence may be discerned in the progressive nature of biological evolution, for example, and the tendency of the universe to self-organize into a richer variety of ever more complex forms. Whitehead thus replaces the monarchical image of God as omnipotent creator and ruler to that of a participator in the creative process. He is no longer self-sufficient and unchanging, but influences, and is influenced by, the unfolding reality of the physical universe. [...] This mixture of contingency and necessity corresponds to a God who necessarily determines what alternative worlds are available to nature, but who leaves open the freedom of nature to choose from among the alternatives. In process theology the assumption is made that the alternatives are necessarily fixed in order to achieve a valued end result — i.e., they direct or encourage the (otherwise unconstrained) universe to evolve toward something good. Yet within this directed framework there remains openness. The world is therefore neither wholly determined nor arbitrary but [...] an intimate amalgam of chance and choice.” [Dav93], p. 183–185

If we replace, in the above quote, the word “God” by “composer”, and the term “physical universe” by “composition”, we get a nice wording for a theory of the interactive, indeterministic, participatory approach to composition.

The Dynamic Stochastic Synthesis provides a striking example of what it means to compose in cooperation with a self-organizing process. The settings of the composer, even if manipulated interactively, can never totally dominate the autonomous process of sound signal generation. The stochastic process is “tamed” by the boundaries imposed by the various parametric settings (distribution coefficients, elastic mirrors), but it has its own life. It can be interactively influenced and guided by the composer, but it can never be 100% controlled. The composition process is an interaction between human intelligence, sensitivity, and musical intuition being confronted with an abstract, sound generating process, and algorithmic action being directed in ever new directions by changing boundary conditions.

“The question that arises in all its generality is to know which mathematical construction to specify to the computer so that what is heard will be as interesting as possible - new and original. Without dwelling too long on this, I will cite as an interesting example belonging to a case I was able to discover some time ago by using the *logistic* probability distribution. For certain values of its parameters  $\alpha$  and  $\beta$  and its elastic barriers, this distribution goes through a sort of stochastic resonance, through a most interesting statistical stability withing the sound produced. In fact it is not a sound that is produced, but a whole music in macroscopic form. This form is the result of rhythmic transformations, giving a polyrhythm of events with variable timbre, changing pitches and intensities - in short, rhythmic strands of meeting and jostling sounds. I have used this procedure in the music of the *DIATOPE* at the Centre Pompidou.” ([Xen85], p. 180)

It is interesting how Xenakis immediately noticed the significance of this discovery for his lifelong dream of an “automated music”. Indeed, what can be heard in *La légende d’Eer* is computer sounds that start as noise but quickly turn into pitched glissando, going onto a long evolution of ever-changing pitch, amplitude, and timbre. Only recently such a behavior, where a rich sonic evolution is generated from very few initial data, producing a magnitude of complex sound data, has been discussed in a theoretical and aesthetical context as a phenomenon of “Sonological Emergence” (see section 5.7.5 on page 76). In retrospective, we recognize in the stochastic sounds of *La Légende d’Eer* the very characteristics of sound produced with the GENDYN algorithm, which Xenakis implemented almost 20 years later on his personal computer at CEMAMu.

The pure computation speed radically changes the way of composing as it becomes possible to change the synthesis parameter settings while listening to the sound output. As a matter of fact, Xenakis himself wanted the synthesis parameters to change during sound generation, and in 1994, he devised a version of GENDYN where each parameter is “modulated” by either a deterministic or a stochastic function. However, without the intuitive control of the human ear it is almost impossible to foretell the impact of those changes on the stochastic generation of sound, and the piece Xenakis made at the time, *S709*, must be regarded as suboptimal in this respect.

In Xenakis’ original program, the set of synthesis parameters can be thought of as being “hard-coded” into the program since they are defined before program

execution and not altered until the entire music is calculated. The change of the parameters in *S709* is a programmed change, and the set of parameters that drive the modulation functions must just as well be considered “hard-coded”. This classical notion of “automated music” is of a very different quality than the notion of interactive composing where the composer enters into a feedback dialog with the computer, altering the boundary conditions of algorithmic calculation while the algorithm is in action.

The interactive model of composition, however, where the human and the machine interact and together form a more complex unity, has now become possible with the Dynamic Stochastic Synthesis working in real time on a standard computer. The Dynamic Stochastic Synthesis is powerful enough to generate complex sound with several temporal layers of sonic evolution (microscopic and macroscopic structures). The composer, in reaction to what he/she perceives, pushes the system through a trajectory of states that represents his/her individual way of “playing” the composition algorithm, creating interesting or even surprising results.

However, with the means at his disposal, Xenakis was not able to transform his GENDYN sound-producing automaton into a sound producing robot navigating through an ever-changing terrain set up and changed in real time by the composer.

The notion of algorithmic composition, as it has been realized by Xenakis in his original GENDYN project, is intimately linked with the pure mathematical notion of computation, often presented in the operational model of the “Turing machine”, a computational formalism that conceives of an idealized, abstract programmable machine to carry out programmed mechanic action. Xenakis consciously subscribed to the limitations of such “pure” algorithmic computation. He applied this strength to the creation (in a mechanical sense) of the huge number of sound samples defining the aural shape (and as an emergent by-product also the underlying “deep structures”) of a musical composition. In the interactive approach, however, the notion of the Turing machine computation must be replaced by the model of an interactive Turing machine that reacts to asynchronous events [Weg97]. It can be thought of as a Turing machine with ever-changing input tape or as a robot moving through ever-changing terrain.

In this sense, the composer enters a feedback loop with a composition robot: the program generates musical structures, but the composer can push the robot in ever new directions. This aspect of the Stochastic Synthesis, implicitly inherent from the beginning, has been made explicit with the help of its real-time implementation in the New GENDYN Program.

If such a system is controlled by another such system, or a human, an interaction loop is established which extends the system beyond its own inherent computational power by harnessing external input (cf. [Weg97]).

Consequently, a composer who enters a feedback loop with a system of which he has actively participated in developing establishes a dialog with his complex self. The master-slave relationship in the classical use of the computer is replaced by a more cooperative approach of interaction, challenging human intelligence by contributing genuine computational elements of unpredictability and surprise. It is different from the romantic will to power where the computer is merely used to maximize productivity efficiency in the spirit of industrial automation. The new, interactive approach requires that the composer be willing to explore and conquer new sound worlds, instead of asking for the comfort of having

preconceived compositional thinking executed by a machine, a dream doomed to fail due to fundamental differences between the algorithmic nature of machine action on the one hand and human creativity on the other. It is a situation where “the composer must learn his/her strategy by interacting with a source of structured information, at the level of the micro-structure of music — within and through the sound.” ([DiS95b], p. 42)

## Chapter 6

# Rationalizing Culture?

“It is problematic to depict the ‘rules’ derived from analysis of one musical aesthetic as either musically universal or generative of new aesthetic forms. In fact, the likely effect of applying ‘rules’ derived from one musical genre to composition is to inhibit any possibility of profound aesthetic innovation and to encourage just variants of an extant genre. In this sense, AI-influenced composition represents its ultimate rationalization, the scientific, high-cultural version of what Adorno accused the cultural industries of bringing about: the standardization of music.” [Bor95], p. 319

In the last chapter, we have discussed that computing has transformed musical instruments into performing instruments and even composing instruments in that they interact with a performer/composer and substantially contribute to the musical result so that the difference between composer and performer and even composer/performer and instrument become blurred. If computing does so significantly enhance the contribution of machines to a human’s artistic output, the question of the nature of that contribution moves into focus. Is creativity computable? To what extent can cognitive processes be emulated by a machine? Many composers and researchers work in this direction, trying to implement cognitive models in their computer environments or use computer assisted composition software which has been designed to simulate compositional thinking in one way or the other, either by expert systems, knowledge databases or various artificial intelligence algorithms, neural networks or genetic algorithms.

None of these, however, is the case with GENDYN by Iannis Xenakis. The algorithm behind GENDYN does not model cognitive processes in any way. GENDYN does not model “musical knowledge” or “musical competence”, or emulate any kind of “musical performance”. This kind of computer composition system does not implement any rules of inference, artificial intelligence features or neural networking. Rather, GENDYN is an abstract algorithmic procedure describing stochastic processes, mapped onto the physical sound signal.

The study of the GENDYN music, therefore, does not ask for cognitive analysis techniques. The reason is simply that the best analysis of the music is the GENDYN algorithm itself. Nothing can better explain the music than the algorithm which generated it, entirely determining the music including its sonic realization up to the least significant bit of its CD recording. In contrast to most computer compositions, the music of *GENDY3* is entirely reducible

to the generating algorithm. The act of composition has been entrusted by the composer to the working of an algorithm, and the specification of that algorithm is an exhaustive documentation of the generative process leading to the music. One could say that, in the case of GENDYN, we can enter the composer's "brain", because the composer decided to model the act of creation as an algorithmic process which is so clear-cut that it can even be "understood" by a machine. Certainly, this "brain" is reduced to mechanic action and first-order logic — a deliberate restriction chosen by the composer in order to harness the generative power of a computer.

Still, it would be desirable to apply state-of-the-art methods of psychoacoustics and cognitive sciences to the acoustic products of Stochastic Synthesis. In principle, it would be desirable to be able to predict the effect of a specific synthesis parameter onto the musical perception of the musical end product by a listener. This would require the linking of several areas of research: computer science, acoustics, psychoacoustics, and cognitive musicology (where the latter are combined disciplines themselves).

However, in this chapter, I will raise a fundamental argument against the general utility of such an ambitious research project, even if it could be realized. In my opinion, a highly non-standard artistic phenomenon like GENDYN cannot be, and should not be, investigated with the means of generalizing, universal and standardizing theories of acoustics, psychoacoustics and cognitive musicology.

## 6.1 Computation and Cognition

Cognitive sciences (cognitive psychology, linguistics, cognitive neuroscience, etc.) have for a longer time (roughly since 1970, [Gar85]) been taken as a model to enhance both the analysis of existing and the production of new music. Artificial Intelligence (AI) is a discipline of computer science where cognitive models are being formalized, implemented and empirically verified by testing. Computer music systems are more and more equipped with AI features in order to better serve the needs of computer composers. While AI techniques are employed in the composition of musical scores ever since the very first attempts (cf. [Bur94a]), efforts to integrate AI concepts into systems that do sound synthesis is a comparatively recent field of research [Mir95a].

Three stages are commonly discerned in the application of cognitive models in the computation domain. Early AI techniques of 'symbolic' intelligence have by now more or less been replaced by a paradigm of massive interaction (e.g. neural networks) and, still later, ideas of enaction have been formulated ([VTR91], [McA87]).

Cognitive studies have largely influenced the investigation into the way we hear and aurally perceive. Qualitative statements about the functioning of the hearing apparatus have been made for long, based on personal experiences and scientific experimentation. The cognitive approach, however, permits to verify theories of human sound perception in that it invites to implement cognitive theories of hearing on the computer. Cognitive sciences rely on the powerful support of computer simulation, a method of empirical verification that has so much fueled research in natural sciences.

The generic term "cognitive sciences" brings together different disciplines and practices that are unified by their endeavor to better understand and model

issues of human intelligence and creativity. Modeling is almost exclusively done with machines, i.e. with the help of universal computation by means of computer algorithms. In the music domain, composition, analysis, interpretation, and even perception of music are more and more simulated with the help of computers. It is hoped that by understanding the cognitive processes involved in such a complex phenomenon as is music production and perception there will be gained much insight into human information processing in general.

The key axiom of cognitive sciences is that all cognitive action can be algorithmically formalized. Hence cognitive processes, once formalized, can be studied by explicit simulation with the help of computing machinery.

“Cognitive science [...] considers humans to be information-processing systems. Mental processes are understood as effective computations, to be described by certain rules and representations, which can be formulated and explicated in a formal language. [...] Cognitive science can serve as a new research paradigm in musicology if [this] mechanistic basis is taken seriously. [...] In music research it is for the first time possible to relate research on musical structure, i.e. music theory, with research on music perception, i.e. psychology and physiology of music perception, and to treat them in the same framework of an intersubjective controllable scientific manner in exploring the preconscious structures and processes underlying the perception of complex musical structures. Musicology in the cognitive science research program may be called cognitive musicology [...]” [Sei92], p. 227/232-233.

The above quotation clearly states a voluntary limitation of the cognitive approach to the study of formal systems and mechanic machine action (i.e. algorithmic processes). The question is if this mechanistic view of cognition is able to match the human capacities of reflection, introspection, sensibility, and creative action. It will become clear in the course of this study that formal systems and algorithms possess fundamental shortcomings that make them inferior to human intelligence and intuition. My contention in this study is that machines are wonderful instruments to *challenge* human cognitive action, but they cannot fully put up with it. Algorithms are constructed by human creativity, but human creativity cannot be simulated by algorithms. There is a fundamental difference between human and machine intelligence. I will discuss this issue more fully in part two of this study called “Theory”.

### 6.1.1 Cognitive Musicology

It may have become clear by now that the present study strongly argues against the possibility to implement creative processes of musical production and perception by computational (i.e. mechanical) models. However, in the special case of algorithmic composition, methodologies of cognitive musicology seem appropriate for the investigation into the phenomenon of computer art, exactly *because in the case of rigorous algorithmic composition, the rules of creation are bound to be mechanic*. In other words, the approach of cognitive musicology seems to perfectly fit the analysis of algorithmically created music.

One of the most prominent proponents of a cognitive approach to music creation, Otto Laske, advertises the use of computers to model and verify procedural musical knowledge regarding the performance aspect of music-making and music-listening.

“For the first time in the history of musical research, the computer program provides a medium for formulating theories of musical activity, whereas prior to its emergence only theories of musical artifacts had existed. As well, computer programs inevitably drew attention to the structure of musical decision-making in real time, thus to musical processes; they demonstrate the possibility of formulating musical knowledge in a procedural form, thereby highlighting the fact that musical knowledge in humans exists largely in that form.” [Las88], p. 44

In other words, Laske proposes a simulation of the generative process of music composition, instead of studying the written documentation of the end product.

“The emphasis that cognitive musicology is both a theory of musical activities and musical artifacts [...] is meant to forestall the “knowledge engineering in reverse” accepted up to now, by which one thought to distill theories of musical activity and thinking by way of “analyzing” documents of notated music. Results of this endeavor are unsatisfactory because they never adopted a task-oriented view. These results never distinguish what is true of the composer being studied (if that could be known at all) from the implied idealized listener and analyst.” [Las88], p. 45

It should have become clear by now that I do not adhere to the belief that human thought in general, and compositional thought in particular, could successfully be modeled by algorithmic processes. Since I doubt this fundamental axiom of cognitive musicology, I do not think that this methodology is viable for composed music in general. The special case of Algorithmic Composition, however, is different. Here, the process of composition is algorithmic and an application of the procedural approach as described above is just adequate. As far as the rule-based generation of GENDYN music is concerned, I do follow the methodological approach of cognitive musicology. It means to computationally model the procedural aspect of composition by “implementing a knowledge-based procedure within a specified task environment” which is, in my case, a faithful reconstruction of Xenakis’ program, enhanced by analytical routines which serve my additional needs of analysis. But this is where cognitive modeling stops. I will not attempt to model Xenakis’ possible thinking and acting (e.g. his purported response to the acoustic products which conditioned his working with the GENDYN program). This is because construction and testing of an algorithm as well as aesthetic assessment and critique of its result is a creative action and lies, to my belief, beyond algorithmic capabilities.

The cognitive paradigm, as long as it relies on formal modeling and computing, must find its limits within the limits of formal systems and computation. At the same time, the stress it puts on the procedural and performing aspects of music making opens new pathways of musicological research. In the case of

rigorous algorithmic music (RAC), the cognitive performance of the human is already exteriorized and represented as an algorithmic procedure. A composer who consciously formalizes his processes of creation has voluntarily met the restrictions and limitations of computation. This is why a procedural, computational approach seems to be appropriate because the process of composition is computational itself.

## 6.2 Psychoacoustic Issues

When, right at the beginning of the GENDYN project, I presented the ideas and details of the Stochastic Synthesis at a conference in 1996, I was asked from the audience: “Do you care at all about psychoacoustic issues?” Someone in the audience (not the least, it was Claude Cadoz from ACROE, Grenoble) possibly wanted to hint at the fact that Stochastic Synthesis was beyond serious research in acoustics. In the same spirit, composer/researcher Eduardo Reck Miranda attacked non-standard synthesis methods as being unfit for music creation.

“Early work in computer music, such as Steven Holtzman’s non-standard synthesis (1978), SSP and ASP (Berg 1975, 1980), and Iannis Xenakis’ stochastic sounds (1963, 1971, 1992), has demonstrated the inadequacy of abstract mathematical models that generate musical sounds without accounting for ears’ cultural conditioning.” [Mir95a]

The above polemic is elucidating in that it reflects a profound question which I intend to address in the present study. I would like to contrast Miranda’s view with the artistic conviction of Herbert Brün whose computer music composition program SAWDUST (from 1973), as already mentioned above (see section 5.7.2 on page 74), definitively falls into the category of “non-standard” computer music. Brün is not mentioned in Miranda’s quote, but he was probably the most radical and certainly the most eloquent proponent of an explicit computer music culture that I have defined above (see definitions 3 on page 25 and 1 on page 43).

Brün is very clear about why he uses the computer for the generation of music. He states that the computer helps him to compose “music that you don’t like yet” ([Bur94a], p. 54/55).

“It is one thing to aim for a particular timbre of sound and then to search for the means of making such sound and timbre audible. It is another thing to provide for a series of events to happen and then to discover the timbre of the sounds so generated. In the first case one prefers those events to happen that one wishes to hear; in the second case one prefers to hear those events one wished would happen.” [Brü69], p. 117

As we can see, according to Brün, music composition in general, and computer music composition in particular, means an incessant struggle against “ear’s cultural conditioning”. From the viewpoint of Brün, Xenakis, Holtzman, and the other purported members of the explicit computer music culture, psychoacoustics should not play a normative role in assessing creative musical work. After

all, it is the composer and the public who finally decide on the aesthetic value of the music, not scientific criteria. It follows that computer sound does not need to be “cultured” or “adapted” to a stipulated musical ear at all. Rather the opposite: the task of music production is to question our present condition of listening, in view of the radical alteration of our human condition in the computer age to which humankind has subjected itself.

“There are ‘schools’ which build on the conviction that humans are restricted in their perception, so one studies what they are capable of hearing and one teaches the composers to work efficiently in this sense: ‘Consider that such and such acoustic phenomenon is not audible, so do not use it for structuring your musical discourse.’ For multiple reasons I do not adhere to this way of thinking; it deprives the composer of one of his essential roles: to create the unheard-of.” [Boe90], p. 141

A problem with psychology in general which might also pertain to psychoacoustics is that it establishes its findings in reference to a statistical mean, the representative, the commonly accepted, the average. Yet, art and its reception always deals with the exceptional and has to be exceptional if it is to be art at all, at least in the emphatic sense adopted within the framework of this study. Psychoacoustic investigations, on the other hand, by definition, tend to privilege the status quo of perception. It is interesting in this context that Xenakis, in his writings, liked to state that human nature is subject to change (“changer l’homme”) under the impact of art.

To conclude, psychoacoustics, although highly valuable in the research of human response to music, should not guide music creation. Where they do, artistic production runs the risk of transforming either into an exemplification of psychoacoustic effects or, even worse, to adapt to a kind of “average ear”. To make matters worse, much of psychoacoustic research is confined to instrumental sound or even to tonal music, so an application to electroacoustic composition, where undertaken at all, is inadequate.

Therefore, I shall refrain from trying to “proof” the quality of GENDYN sound against a background of any scientific acoustic model of sound. This would be incompatible with the very subject of this study: a non-standard setup by an independent artistic mind.

### 6.3 Auditory Scene Analysis

“Auditory Scene Analysis”, in analogy to “Scene Analysis” in Computer Vision, is a term coined by [Bre90]. Just like Scene Analysis is a problem in the area of Computer Vision, Auditory Scene Analysis is a problem in the area of Computer Hearing. Scene Analysis is a complex of sophisticated computational procedures in order for a computer to extract meaningful data from visual input, acquired e.g. by means of a digital video camera. This processing is necessary for machines to “see” and “perceive”, e.g. for a cruise missile to follow a moving target. Complex data have to be processed in order to segregate components, extrapolate, recognize and classify groups and components of data. In Computer Hearing, there is the special problem of segregating simultaneous information

mixed in a complex sound, evolving in parallel “streams”. Advances in Computer Hearing already yield practical results e.g. in man-machine interfaces. For example, as I type this text, inexpensive systems are available with the help of which I could just speak it right into my computer.

Much progress has been made in the field of speech recognition, but when it comes to music, research is still in a seminal state. First of all, which “language” should be assumed as a context to machine music listening? Is there a musical “language” at all, and can it be “parsed” by a machine as is contented by [JL83]? As a preliminary, success has been made to differentiate individual “musical streams” within a context of simultaneous, layered sound events ([Bre90]). Such techniques of Auditory Scene Analysis could help the study of electroacoustic compositions where in general, there is no score that would tell us of the multi-layered components of the music. For example, Auditory Scene Analysis could well find out that Xenakis’ computer composition *GENDY3* is made out of 16 parallel streams of sound (a fact which is quite difficult to discover for a human ear, especially if those streams are similar to each other as is the case in some sections of *GENDY3*). Fortunately, since *GENDY3* is RAC music, we have no need to extract its streams from the musical end product but we can just regenerate them, one by one, following Laske’s procedural approach of cognitive musicology as described above (see section 6.1.1 on page 95).

Recall what we said about the function of art, being important for our coming to terms with our own human condition. We see that we are constructing a paradox if we extend the capacity of machine listening to what we might wish to be the essence of acoustic art. If computer music synthesis on the one hand, and auditory scene analysis on the other, enhanced by musicological expert systems and cognitive models of music perception could be made to work together, we could finally dispense of the trouble of thinking about music altogether. Imagine a computer system so sophisticated as to produce high quality music of any degree of complexity, and another computer system able to decode such artifacts in terms of a computed aesthetic assessment. These two systems could then be made to work together. They would certainly be smart enough to improve over time, and to optimize their production and reception potential to a degree surely never reached by humans. Humans would then no longer need to go to concerts or listen to radio programs, all necessary cultural labor being taken over by computers. This kind of computer culture could be just perfect — only the human would be missing.

### 6.3.1 The Emperor’s New Ears?

In view of the attempts to model compositional thinking with algorithmic models (algorithmic composition), to machine-analyze musical scores (cognitive musicology) and machine simulate human listening (auditory scene analysis), the issue of the human factor in art is brought up with increased urgency. If computers are able to implement key activities of artistic communication, the question of the essence of art must perhaps be reformulated. Will art still continue to serve to “discover the essence of what it means to be human”?

Cognitive sciences rely on the assumption “that mental phenomena are based on the neural activity of the central nervous system, which is [...] determined by physical processes and therefore describable by the laws of physics in physical terms.” ([Sei92], p. 229). Since all known laws of physics can be algorithmically

modeled, so must in principle be all mental activity, including music composition, interpretation, and perception. Yet, if this should be indeed the case, then a computer could in principle be programmed to listen and to *enjoy* music! This is, indeed, the claim of the so-called strong branch of Artificial Intelligence, which holds that we shall be able to replace our mortal bodies by computer circuitry, for, as they argue, it would not matter if the computations of life were executed by biological matter or computer hardware ([Mor94], cited in [Pen94], p. 34). Robots could enter a kind of evolution where every parent generation of robots could construct more “intelligent” child robots reaching a state of intelligence that would be far beyond human capacity (cf. [Cas94]).

Automata constructing other automata and reproducing themselves have already been studied by von Neumann ([AB86]) back in the 1950s. This process is certainly conditioned by the fundamental limitations of computation. Therefore, the only way out of this eerie scenario seems to me the supposition that there is some non-computational ingredient underlying physics as relevant to the functioning of human mind and soul (cf. [Pen89], [Pen94] for a more systematic discussion).

The presentation of these interesting, seemingly science-fiction aspects of machine cognition is not to discredit the application of cognitive sciences to art as such. Quite the opposite: I think that the endeavor of cognitive sciences to exactly determine the computational aspect in human thought and action will shed even brighter light onto their non-computational residues. I shall argue that artistic activity is a combination of “mechanical”, rule-based, cognitive elements on the one hand, and intuitive, “holistic”, creative elements on the other. I would like to assert that creative action cannot only be computational, for it excels computation by the fact that it can transcend and go beyond any preconceived formal framework by employing qualities like intuition, “insight” and “understanding”, qualities that have not yet been simulated by machines, and are contended to be in principle beyond computation (cf. [Pen88]).

## 6.4 The Turing Test

In addition to producing sound and musical scores with the help of computers, there are research activities all over the world to make computers interpret classical scores like humans do ([SE98]), improvise jazz music like humans do [RP98], and even listen to music like humans do. The ambitious “Kansei” program strives to model emotionality by machines while making music ([Cam97]). As for myself, I would find it most interesting to combine both a composition and a score interpretation program in a “heretical” manner, in order to “interpret” machine music in an “expressive” manner, like, say, a romantic serenade!

To verify if computers can execute tasks “like humans do”, an objective procedure has been proposed by outstanding mathematician Alan Turing [Tur50]: the Turing Test. If a computer is able to “deceive” a human during a considerable amount of time that it be human (i.e. if a human cannot tell the difference between the actions of a computer and the actions of a human being), then the computer has to be conceded the quality of possessing human intelligence. (For the purpose of our discussion, we might want to replace the word “intelligence” by “musicality”.) To be fair to the computer, human and machine output would be coded in a neutral way. Both computer and human would be behind a hiding

“screen”, letting through only the pure information, being stripped off its form of presentation (voice, gestures, etc.)<sup>1</sup>

It is not easy to argue against the validity of the Turing test. The most popular counter-argument is Searle’s thought experiment of the “Chinese Room”: a human without any clue for the Chinese language is “playing” computer by processing Chinese symbols according to the rules of a sophisticated linguistic program. The human would be attributed the qualities of a Chinese scholar by the judgment of a Turing test (cf. [Pen89]).

My personal viewpoint, for the purpose of the present study, is not directed against the Turing Test as such. Rather, I would like to direct it *against* any computer music system that would successfully make me believe that it produces “human” output! It should be clear from the foregoing discussion that the successful passing of the Turing test by a “composing robot” will testify against its artistic value, by virtue of our definition of Explicit Computer Art (cf. definition 1 on page 43).

**Definition 6.1: The “Inverse” Turing Test.** Algorithmic computer music passing the Turing Test is to be rejected on the grounds of Explicit Computer Art.

I personally think that a Turing Test reveals more about the human testing the machine than about the machine itself. If a human cannot discern machine output from human output, then it is his/her ability to “humanize” the machine, to project a conscious identity into a mechanism where in fact there is none. To me, the Turing test seems to testify human self-deception more than machine intelligence. We shall see later that this phenomenon of human projection even pertains to the listening experience of a piece like *GENDY3*.

At the same time, I want to be very clear about the fact that I do not argue against research into artificial intelligence. Quite on the contrary, I find it most important and even urgent to investigate the still obscure limit between computation and human thought. Since computers have become part of our everyday reality, and we are concerned by computers even in sensitive areas of our lives, mankind should have an increased interest in studying the “moods” and “minds” of computers! Now, the domain of the arts is where we are very sensitive to smallest shades of expression, behavior, and structural detail, due to our century-long cultural conditioning. Art can therefore be a prominent research area for determining the computational potential and limits of computers.

There is also a moral issue about the Turing Test applied to machine made music. The computer is not bound to the same moral framework (if any) as we are supposed to be. (If this were the case, this would make switching off the power supply of a computer a case of manslaughter.) However, art does have to do with ethical questions. For example, if I listen to music made by a human (e.g. a jazzman), and I am touched by it, I build an emotional relationship to that human, to his/her person, history, social background, etc. If, however, I want to have a somewhat detached listening experience, I listen to computer music. I am asking: What is the use of training a computer to produce expressive human output? In general, an expressive musical experience is to be preferred over

---

<sup>1</sup>With the advent of virtual reality techniques, this “fairness” is becoming more and more obsolete: the computer is more and more able to present itself to the Turing tester in a “human” guise.

an inexpressive one (if that inexpressiveness is not an expressive experience by itself). But a computer system should not try to imitate human expressiveness, if it is not to abuse the feelings of the public. It is hard to accept the idea of human listeners establishing emotional links to simulated musicians. It seems preferable, then, that the listeners be simulated by computers as well!

## 6.5 Calculating the Incalculable

In this chapter, I have tried to sketch current research in the field of computer music striving to implement cognitive processes and artificial intelligence in order to model some of the most noble activities of humans: the production and reception of art. Computers are nowadays made to compose, interpret, listen to and analyze music in the way humans do. I have tried to extrapolate these tendencies toward their ultimate consequences and demonstrated that they are paradoxical in nature: in trying to be faithful to conventions and rituals of existing culture, they destroy its very essence, which is human self-awareness and self-reflection through cultural artifacts.

One reaction to this could be to dismiss technology and computers altogether from the domain of the arts. Should music and art not be the last reservation of what is “incalculable” in the emphatic sense of the word? This would, however, lead to a cultural atavism. Art in our times cannot ignore computers. On the contrary, art has to extend to computing in order to remain authentic since computing is the most prominent feature of all Western societies since World War II. A good way to use computers artistically is the explicit way, described in the foregoing discussions.

But in how far can the creation of music be so emphatically associated to computation? Can music be “calculated”?

Composers of all epochs have played with rules and systems, sometimes self-imposed, sometimes consciously or unconsciously inherited from their environment or history. Some music deliberately plays with the aspect of being easy to predict — sometimes only in order to create a moment of surprise when the music suddenly takes a different turn than expected.

Calculation does not necessarily mean mechanical music. Quite on the contrary: the construction of, e.g., “sounding fractals” (cf. [Joh97]) seems to show that the fascination of music is to some extent its navigation within a space between predictability and unpredictability. Using the complex effects of even simple rule-based (i.e. computational) systems is one way to systematically explore this field. One of the main theses of this study is that computation, although it does not replace or even emulate human creativity, is a powerful means for generating *surprise*. Surprise is a virtue of human behavior, but not exclusive to humans. “Lively” sounds and “innovative” sounding structures do not necessarily have to imitate human music. I contend that it is much more interesting to generate lively and innovative music in a novel way, transcending inherited aesthetics, than to imitate human music making.

Let us see in the next chapters how computation can contribute to artistic creation in a constructive way.

## Chapter 7

# Creativity, Communication, and Originality

“Music is not a means of communication. It is meaningless material, used for open-ended processes of aesthetic reflection by a multitude of culturally diverse audiences whose interpretations are totally arbitrary. [...] To engage in a musical project that would acknowledge this state of affairs would mean: to avoid choices, to transcend styles, to present *everything*. It is probably no coincidence that such a project has not been formulated before. It could not have been carried out by a spontaneous individual composer. What we need is a well-considered division of labor between human and machine, which exploits the complementary strengths of human persons and digital computers. I would like, therefore, to propose a new method of music creation, which blurs the distinction between composing and theorizing. Human composers become formal music theorists: they develop mathematically precise definitions of classes of pieces (“styles” or “genres”), rather than individual works. And they become programmers: they collaborate with computer software to develop implementations of these definitions. Computers thus end up storing algebraic definitions of large and complex classes of unthought, unheard compositions, and they run algorithms that generate random instances of these classes. The task of the human theorist is then, to use his wild and erratic cognitive powers to continually extend, refine and generalize these definitions. In this way, computers and humans work together, toward the realization of all possible music.” [pHH96], p. 156-156

### 7.1 Issues of Creativity

In the preceding chapters, the relation of computer art and technology has been discussed and the notion of an explicit conceptual computer art enunciated. Issues of creativity and computation, and their mutual relationship have been discussed. In this chapter, we will have a look at the other side of the artistic communication process, the listener and her/his special role in the special case of Rigorous Algorithmic Composition (RAC).

Xenakis’ work with the computer in general and GENDYN in particular is an approach to composition which is “artificial” in the emphatic sense of the word: a self-imagined and self-constructed sound world “ex nihilo”, completely independent from any preconceived framework of acoustic research or compositional

procedure. However, although this music is conceived of in a completely novel way, listening to it still stirs sensations that point to a “composing subject” behind the music. The listener cannot help but feel a certain intentionality behind the acoustical and musical phenomena that are to be heard in a GENDYN piece. These are e.g. effects of music dramaturgy like moments of “retardation” and “suspension”, of “surprise”, of “finale effect” and more. All these can, by virtue of the procedural approach to analysis discussed above, be reduced to technical features of the GENDYN algorithm. (A detailed discussion of these features is to be found in part III of this study.) Therefore, there cannot be any intentionality behind these phenomena on the side of the composition process because it is mechanical. So, the felt intentionality must somehow be hosted on the side of the listener. The sensation of a composing subject behind the music must be a projection created by ourselves.

We shall have to take recourse to a fairly exotic aesthetic theory in order to reconcile the seeming contradiction between mechanical production on the one hand and creative modes of listening on the other. This is the aesthetic theory of Radical Constructivism. This theory, as it turns out, is in perfect correspondence to Xenakis’ own radical ideas of “originality” in music composition, leaving complete freedom to either the conceptual framework on the side of the composer and to the perceptual framework on the side of the listener.

The creativity on the side of the composer, i.e. the constructor of the composing algorithm, is therefore twofold: First, it happens during the invention and construction of the algorithm of composing. The invention of algorithms is a creative act which can not be taken over by machines. Second, while checking the result, the composer is his own listener. As such, he adds to the creativity of the creation of the (mechanical) production process the creativity of the reception of its results. Since in this special case, composer and listener are one and the same person, a feedback cycle is established which feeds back cognitions about the result to the creation or alteration of the composing algorithm.<sup>1</sup>

As should be clear from the foregoing discussions, I consider the match of machine computation against human creativity, both on the side of the composer and of the listener, as the most interesting aspect of Rigorous Algorithmic Computer Music. Machine computation has the power of surprise, induced by its very complexity. (The notion of complexity shall be discussed in depth in part II of this study.) Human creativity has the power of transcending any preset framework, to envisage something not-yet-considered. As such, and by definition, it must always be beyond any formalization.

“If an experiment is made only in order to reveal and communicate the information content of a system, in order to tell all that is worth knowing about the system, we are dealing with a scientific experiment. But if the experiment is done with the intention to create means of communication in order to convey an account on the thoughts and states which transcend the self-definition of the system and proclaim analogies to other systems, then the experiment is to be regarded as a creative undertaking.” [Brü71], p. 90, quoted in [FGG96], p. 236

---

<sup>1</sup>This is at least what happened in the case of Xenakis while other composers, such as e.g. Tom Johnson, seem to less wish that this feedback cycle should take place.

There are aspects of creativity that go beyond computation. We will show in part II that computation is closely linked to formal systems which are finite and closed by definition. Art seems to have an ingredient of uncomputability which differentiates it from the pure experiment.

## 7.2 The Aspect of Communication

It might be interesting to find out if a composer's intention could be unequivocally communicated through sound. In order to clarify this, psychoacoustic aspects about the listening process would certainly have to be considered. In the case of Iannis Xenakis, however, such an investigation would be quite at odds with his pronounced artistic goals. Xenakis was not interested in how his music was perceived by the listener, nor that the listener might catch a certain "message" through it. He did not intend to communicate anything through his music. In his opinion, music was to supply the means for each listener to experience something about him/herself. In connection with one of his major electroacoustic works, *La Légende d'Eer* (1977), Xenakis wrote:

"Music is not a language. It is like a rock with scratches and scars etched onto it and which can be deciphered by Man in thousand different ways, without one of them being the best or the truest. Because of these multiple interpretations music stimulates like a crystal catalyzer all kinds of phantasmagories." [Xen78]

Xenakis conceives of his music as a "catalyzer" for processes to happen within the listener which are not conditioned by the music.

"All music [...] is like a kind of catalyzer which makes that you *by yourself* reach mental states which concern you. By the very effect of existence and presence of what you just hear. It is like a drug [...] which enables you to realize yourself. In different ways, and also in multiple ways." [Del97], p. 138. (Emphasis added by me)

Let us now return to the above question whether it might be objectively possible to find out if a composer's intention could be communicated through sound. Suppose we would be able to analyze the sound by acoustical and psychoacoustical treatment and use cognitive methods in order to simulate a listening experience with these data (recall our discussion in the previous chapter about auditory scene analysis and machine listening). Then it would be debatable if the listening conditions described by Xenakis above were reproducible in such a scientific experiment. And if, on the other hand, they were, it would indeed be questionable if such conditions were acceptable as a basis for objective scientific research.

## 7.3 Autopoiesis and Radical Constructivism

We have seen that the act of composing, if executed algorithmically, cannot be, by itself, an act of creativity. So how can it be that listening to algorithmically created music still conveys a sensation of a "composing subject"? How can it be

that a listener “feels” a musical intention or even expressivity when there is no subject that could be associated with it? In the context of traditional genres, convention has it that most people listening to funeral marches will understand the meaning of the music. In case of an algorithmic music, however, which is, as in the case of GENDYN, the result of a chaotic, self-organizing process governed by probabilities, there is no cultural context nor convention, and if we happen to receive the impression of a funeral march, say, then there is no correspondence for our sensation on the side of the generation process (which is, again, chaotic and self-organizing).

If musicality is to be a creative achievement, i.e. non-algorithmical, then it is not to be found in the mechanical execution of a composition algorithm, i.e. not in the formalized process of composition. It can only reside on the side of the listener, in the specific way in which the a-priori meaningless acoustic substrate is perceived. A fundamental ingredient of this perception seems to be categories of hearing which can be called musical in the context of music listening. Examples are the perception of regular pulsations in terms of rhythmical patterns, or the adaption of non-temperated pitch to culturally established scale structures.

Algorithmically generated music brings a question into sharp relief which applies to music in general: the question of the listener’s creative role. If we follow the unfolding of a creative process of a “composing subject” when listening to algorithmically composed music, then we follow the unfolding of a creative act within ourselves. Then the listening process is creative by itself, and the impression of a “composing subject” behind the acoustic substrate is a projection of our own. The machine only offers complex acoustic data for us to make sense of. The seeming intentionality of the music is not machine-generated but our own human contribution during the listening process.

The extreme case of GENDYN teaches us a lesson: the human reception of a work of art is probably to be rated as important as the artistic production itself. In the case of *GENDY3* the production is entirely carried out by a machine, and therefore cannot contain artistic qualities by itself. As a consequence, the artistic qualities of the product must “be” on the side of the listener who projects them onto the acoustic substrate. However, the acoustic substrate must possess enough structural complexity in order to justify such a projection.

The tendency to humanize machines becomes particularly evident when machines are put to do artistic work. The source of any “musicality” that seems to be in the artifact generated by the machine is in fact the listener himself. GENDYN is a good case study for this, because it does not attempt to model “musicality” by emulating a “musical mind”. Instead, GENDYN produces complex output by means of abstract algorithmic processes. Yet the music produced is sometimes no less appealing to a musical ear than “real music”. Moreover, sonic evolution, when attentively followed several times by repeated intense listening, takes on the qualities of an intentionally designed (i.e. composed) musical process. Yet, this process has not been produced by a musical spirit. It is our own musical mind (and that of the composer himself as his own first and prominent listener) that makes complex structures “musical”, regardless if they have been designed by a musician or not.

During the process of listening, there is a reciprocal interaction between sonic events and the listener’s creative participation which combine to form a “musical substance”. This is the reason why stochastic music becomes less stochastic from the second hearing onward.

### 7.3.1 Radical Constructivism as an Aesthetic Theory

Radical Constructivism is a theory in the domain of aesthetics which is influenced by Humberto Maturana's seminal studies in biological science and philosophy. The theory which Maturana derived from his biological findings basically states that a subject's perception of its environment is built in a complex process wherein the subject "constructs" its own perception, as the outcome of a closed "autopoietic" process, triggered by perturbations and impulses of the outside world. [Rot87]. As a consequence, the aesthetic theory of Radical Constructivism declares that every human constructs for him/herself the experience of a work of art in his/her own individual way. In this sense, author and recipient of a work of art do not communicate.

"According to H. Maturana and F. Varela, the cognitive and phenomenological domains of any *observing system* (or subject of cognition, a listener, a composer) are always distinct from the cognitive and phenomenological domains of any *observed system* (the object of cognition, another living system, or an "inanimate" object — the musical artifact, in our case). No direct relationship can exist between the two, and no input to an observer may ever come directly from an observed. [...] Such a relativistic, solipsistic view is intimately bound to the experience of a music where timbre is perceived as designed form. The richness and variety of the cognitive processes in the observer and the structural richness and variety in the morphology of the artifact are the only means we have to account for the intense experience occasioned by such music." [DiS95b], p.43–45

Xenakis' music can be considered a paradigmatic example for the aesthetic theory of Radical Constructivism. His idea of a "Symbolic Music" ([Xen92b]) means abstracting from the musical meaning of the musical vocabulary of "tones" (as e.g. a constellation of pitch, duration, and intensity), notwithstanding their individual expressive interpretation by a listener, in order to concentrate on their mutual formal relationship. As a model for this procedure Xenakis chose mathematical logic, where a formal system, consisting of a set of signs and rules to manipulate their combination and transformation, is assigned an interpretation in the "real word" of e.g. numbers or musical events only in a separate, and subsequent, step (cf. [Eic94]).

The strict separation of formalized systems and their interpretation has promoted much of the advance in Science. The stunning scientific models of our physical reality (curved universe, anti-matter, etc.) are in fact only attempts to interpret the various mathematical formulae of theoretical physics. The construction of such "exotic" worldviews like e.g. non-euclidean geometry, which goes beyond the Kantian conception of space perception given "a priori" to the human, is only possible if we give up familiar concepts and conventions and understand that they are no less constructed than the unfamiliar ones. Xenakis has repeatedly declared that one of the goals of his art was to transform the human mind in order to discover new cognitive faculties [Xen79].

"Instead of confronting [the listener] with a message which he has to get at any price - otherwise missing the work of art by not understanding it - a work of art should be constituted in an open

way so as to permit meaningful cognitive interactions. [...] It is the responsibility of the composer to create a ‘code’ such that meaningful constructions [‘sinnstiftende Konstruktionen’] on the side of the listener become possible.” [Ess92]

This aesthetic approach goes in line with the observation that in the course of the 20th century, the notion of “truth” in art has been replaced by a multifaceted offer for individual “experience” (cf. [BE94])

### 7.3.2 Opera Aperta

It seems as if Xenakis, consciously or unconsciously, made the autopoietic autonomy of Radical Constructivism the central aesthetic feature of his art. This is even more true for his electroacoustic music, and in particular those pieces with synthesized sound. Since Xenakis’ sound synthesis is non-standard, these sounds neither refer to instrumental sound nor any other existing sound but offer a completely new and unheard-of sound world. This music is free from evocative associations and demands an active participation on the part of the listener in constructing a novel acoustic experience.

Just as there are different possibilities to reduce complex information to an algorithmic description (we will discuss this in depth in part II), there are different ways for a listener to make sense of a musical artwork. As a consequence, a work of art may be conceived in a way sufficiently open to allow for a variety of intelligible cognitive interactions on the side of the listener. Here we come to one of the deeper aesthetic aspects of a stochastic approach to composition as it has been promoted by Iannis Xenakis from the 1950s onward. Indeed, the idea of Stochastic Music is a cogent realization of the concept of an open work of art [Eco96] because here, openness not only applies to the process of composing with probabilities but also to the very process of listening.

We have stated that with Rigorous Algorithmic Composition, the aspect of creativity must be situated on the side of the listener, since it cannot be contained in the generative process of the music. However, this does not mean that there is no creativity required on the side of the composer. First of all, the non-standard approach demands that the generative process be unique and fully invented, instead of implementing standard models of sound synthesis and composition. Even more than that, since composers of algorithmic computer music are, in general, their own first and prominent listeners, creativity enters algorithmic composition by virtue of a closed feedback loop between the human as a creative designer and the algorithm as a faithful executioner of the algorithmic processes of composition.

“Form [...] is not merely a vessel, but a process that starts with the composer’s inspiration and proceeds by way of design, execution, correction, and performance to the listener’s ear and judgement. [...] In doing so, [...] the composer – the first interpreter of the material – [...] enters territory that is at least as close to interpretation as to composing.” [Koe83], p. 31

## 7.4 A Radical Notion of “Originality”

Much has been achieved in computer music research on how computer software could be evolved to better serve the needs of musicians and composers, resulting in rich programming environments and comfortable music languages. Yet for GENDYN, Xenakis did not use any level of abstraction whatsoever from low-level programming. He even did not define any layer of functions to build a composing environment of his own. Xenakis programmed as if there had been no development in algorithmic computer music composition since his early ST-program (1958-62) [Xen92b].

There are probably two reasons for it. First, Xenakis’ approach is more that of an artist than that of a software engineer. His programming was not meant to be a contribution to the scientific community of computer music. He was more interested in producing something unique than something for general use. Second, it seemed to be Xenakis’ ambition to create everything from scratch. He did not even think of using any existing tools or packages. His idea was to realize musical imagination by algorithmic action. Mathematics was the means to achieve this: Xenakis was able to model his musical thought in terms of mathematical processes that could be formalized as a computer program and thus be turned into sound and music. This attitude is a very pure and solitary approach to computer composition. There have been similarly radical approaches taken by Herbert Brüin in his SAWDUST program (1973, see section 5.7.2 on page 74) or Gottfried Michael Koenig in his SSP sound synthesis program (cf. section 5.7.4 on page 75).

It follows that Xenakis was never ready to accept the role of a “user” in the computer music domain. As a composer, he did not want to use anything he had not designed himself. As a consequence, he did not even use my New GENDYN Program when offered, although this would have dramatically eased his research on GENDYN sound. As a consequence, *GENDY3* and *S709* were to remain his last compositions in the electroacoustic domain<sup>2</sup>.

This extreme attitude of Xenakis’ has to do with his rigorous convictions about “originality”. Xenakis took this notion very far in the sense of creation in general, not only in the domain of arts, but also as the evolution of Nature, the creation of the universe, etc. [Xen94b]. Because of this general notion of creativity, of making something new which has not yet been there, his casual references to religion and science are not as far-fetched as they may seem at first glance. Xenakis was not interested in anything ready-to-use because this would have deprived him of the chance to create everything by himself. Not being creative for him was synonymous with death.

To give an example, a classical music programming language like Max Mathews’ Music V (whose concepts still survive in state-of-the-art programming environments as e.g. C-music or Csound [Bou99]) would not be interesting to Xenakis because its primitives already contain certain assumptions on the structure of musical sound and its control. For example, wave generating functions (oscillators) produce entire wave forms, while Xenakis wanted to create and control virtually every single sample of his music.

---

<sup>2</sup>We do not count EROD here, produced for the Bath festival in 1997, because Xenakis did not keep it in his *catalogue des œuvres*.

## 7.5 Universality and Limitation of Xenakis' Approach

The above considerations, when applied to the special case of Rigorous Algorithmic Composition, imply that a composing algorithm must somehow bear the signs of the composer's understanding of art. This is, indeed, the answer to the questions sketched at the beginning of the present part, concerning the use of standard computer technology for the creation of highly individual artworks. The analysis of such works, like e.g. *GENDY3*, has to address the issue of how computer technology can be bent to produce such an original music.

In contrast to e.g. Berg's and Holtzman's concept of Instruction Synthesis, Xenakis' Stochastic Synthesis does not target a specific computer hardware (cf. section 2.3 on page 41). The idea that stands behind his artistic concept is more general: the idea of universal computation. It is conceived of independently and abstracted from the actual functioning of a computing device.

While Xenakis' GENDYN is not idiomatic to a specific computer hardware it is idiomatic to computation as such. Therefore, Xenakis' algorithm is more universal than Instruction Synthesis or similar concepts in that it relies on an abstract mathematical model rather than on a concrete machine model. Stochastic Synthesis is not the sonification of a specific *van Neumann* architecture of a computer but rather the sonification of mathematical models which can be implemented on any computer hardware.

The universality of the programming approach is not to be confused with the universality of the program itself. While Xenakis always contended that e.g. his ST-program would be usable by other composers due to its generality and extensive parametrization, the algorithmic structures of his programs are fixed. Xenakis' programs are not toolboxes with basic elements to build new "algorithms", nor do they seem to be designed for further development. They are self-contained, stand-alone constructions. Realizing a different aspect of composition would mean to develop a new program. This is actually what Xenakis did with GENDYN: for the composition of the follow-up piece *S709*, Xenakis rewrote the program to an extent that it could never generate *GENDY3* again. Every program version is there, like a "Software Sculpture" ([Bis91]) which stands for itself.

Is the Stochastic Synthesis universal in the sense that it is capable of creating all possible sound? This is certainly not the case. In spite of the fact that this comparatively simple algorithm creates a considerable variety of rich sounds in the vast continuum between perfect noise and frozen pitch, it nevertheless produces a limited class of sound signals. A short consideration may serve to underline this statement. GENDYN waveforms are, in general, breakpoint functions with sample values interpolated in between the breakpoints. Therefore, the interpolated values take on only exactly one of the many possible values: the one which is derived by the interpolation formula. This means that for each configuration of breakpoints all values in between which do not coincide with the interpolated values, are excluded. There are GENDYN sounds where interpolation is nil, i.e. where each sample value is defined by a stochastic process and therefore could cover every conceivable waveform. However, these GENDYN sounds are always noises because by definition of Stochastic Synthesis (as we shall see later in this study) these sequences of breakpoints show no inner

correlation and thus no periodicity. To conclude, non-noise sounds which are not reducible to breakpoint functions cannot be synthesized by the Stochastic Synthesis algorithm. Moreover, GENDYN samples are restricted to a set of 200 distinct quantization values, yielding true step functions which can only be generated by the computer. GENDYN is a true subset of synthesized sound, rich and strange in nature but without the capability to model more than its own unique sound world.

We have now laid the foundations for a deeper discussion of GENDYN, both in its theoretical and its musical aspects. With all the considerations and background knowledge discussed in the present part, we are now ready to embark on parts II and III of this study. Let us first dig deeper into the theoretical ramifications of GENDYN: sound, computation, complexity and programming.



Part II  
Theory



In part I of this study, called “Introduction”, I have attempted to situate Xenakis’ work on GENDYN within its historic context, its aesthetic implications, and its technological background. We came about some deep questions concerning issues of creativity and perception, mind and machine, and about the very nature of what might be an algorithmic music in a strong sense.

This present part is dedicated to supply the theoretical foundations of what will be discussed in the next part: analysis. In the present part, we need to shed some more light upon the formalisms and technicalities which at the same time enable and restrict algorithmic composition. We will surely have to discuss the theory of computation in the first place. We will see that computation is strongly linked with the history of mathematics and philosophy, and naturally leads toward a discussion on the nature of matter and life, computability and creativity within the universe, and how this possibly relates to the creation of music or music making machines. We will have a closer look at sound and how it relates to computation. Last but not least, the GENDYN project concerns questions of software development and programming language paradigms. All that is the theory behind the New GENDYN Program. With the help of this program, a procedural analysis of *GENDY3* was made which will be described in detail in part III of this study.

First we will have a look at how it is possible to model sound in a computer. We will discuss standard and non-standard approaches, i.e. solutions which rely on known physical and acoustical phenomena of sound and solutions which borrow from application domains alien to sound. We will contrast these approaches with Xenakis’ radical algorithmic approach to sound composition, where sound is computed according to rules of no existing theory, but rules created by the composer, which means nothing less than that in Xenakis’ case, the theory of sound is part of the composition.

We will then discuss the fundamental notion of an algorithm and how it relates to the practice of computer programming. We will see an example of a very simple but ingenious algorithm and the way it is executed with the help of a small set of simple rules which can be carried out by a machine. This example will serve as a preparation to the formal description of Xenakis’ Dynamic Stochastic Synthesis algorithm as a procedure of generating a lists of numbers which can be interpreted as the sound of a piece of music.

After the presentation of the entire GENDYN algorithm, I hope the reader will have the patience to follow me through the rest of the theoretical discussions concerning both the power and limitations of algorithmic action, the relation between computation and simulation, determinism, chaos and indeterminism, and the different notions of algorithmic, computational, and structural complexity.

Then, at the end of this part, the concept and realization of the New GENDYN Program will be described. This will conclude the discussion of the theory of Stochastic Synthesis. At the end of this part, we will finally be prepared for part III: the analysis of *GENDY3*, generated by the GENDYN program.



## Chapter 8

# Models of Sound

If Xenakis' late computer music does so heavily rely on sound synthesis, to an extent that one can almost identify the composition of *GENDY3* with the synthesis of its sounds, we have to throw an eye on the question of sound computation. How is it that a computer can generate sound? To what an extent is sound at all computable? Are there any sounds that cannot be computed? How far does a given model of sound computation carry us? Can such a model be universal, i.e. is it capable of generating all possible sound?

### 8.1 The Shannon/Nyquist Sampling Theorem

Because we are looking at sound computation by computer, it is understood that we restrict ourselves to digital sound synthesis. There is no loss of generality here, since the quantization in amplitude and time of sound with limited bandwidth is known to be universal, thanks to the Sampling Theorem (cf. [Nyq28] and [SW49], quoted after [Roa96]). This means that sound can be represented as a stream of digital numbers, called samples, of the sound's amplitude measured (sampled) very densely in time, and quantized to finite binary numbers.

The fidelity to natural sound depends on the number of samples per second of sound and the precision of their quantization. The current consumer standard is still 16 bit amplitude quantization and 44.100 samples per second per audio channel. Yet this standard is exceeded by today's professional audio by a factor of two and more, and there are reasons for this, because more audio bandwidth and coding precision decrease the error signal inherent in the sampling process.

So when we look at digital sound synthesis, the problem of generating sound with a computer reduces to generating an appropriate stream of digital samples. We can stay with this and state that sound synthesis is the computation of a sample stream. However, this is a very formal approach, and a bit too general to really fit to music. After all, not every possible sequence of numbers corresponds to something we can perceive as sound. Let us take a very simple example of a sample stream which would not represent any sound: a sequence of equal numbers. So it seems that there is more theory to digital sound than just numbers.

## 8.2 Sound Computation

Formally, a synthesis model, be it standard or non-standard, has the following function: it reduces the complex temporal structure of sound to a finite set of control parameters. Normally, the size of the set of control data will be considerably smaller than the size of the sound data. Viewed from the perspective of information theory, one can speak of a “data reduction” or “compression” of the sonic phenomenon. The instance which permits this data reduction is the process of computation which generates the bulk of sound data from the control data through the executing of an appropriate algorithm.

Such an algorithm usually implements a model of sound. Models of sound can be taken from acoustics (additive, subtractive, granular synthesis), physics (physical modeling) or other. In fact, through a given synthesis model the amount of data required to describe a complex acoustic signal is reduced from its theoretic minimal sampling frequency (by convention 44,1 kHz or more) to a relatively small number of synthesis control data. This is the opposite of sampling where existing sound is digitally recorded into the computer. Sampling may still be described as a synthesis method but where every single sample of the digital sound must be viewed as a parameter to the sound, hence no data compression involved.

Yet if sonic complexity is not just input into the computer through sampling, but to be entirely defined through computation, it has to reside within a computer program. The computer program, then, can be viewed as the algorithmic reduction of the sound. Sound is generated by the unfolding in time of this algorithmic complexity into sonic complexity. Put the other way round, computer sound is algorithmically reducible to a computer program. A related phenomenon in image processing is fractal compression of images by means of reducing them with the help of iterative equations.

The idea of GENDYN is that a whole musical structure, reaching from macroscopic score aspects such as duration, pitch and timbre, to the tiniest detail of transient sound micro-structure, representing some hundreds of megabytes of sound data, unfolds as the outcome of an algorithmic process which is described by a few lines of code. In other words, GENDYN algorithmic action transforms a few kilobytes of control data into megabytes of sound data without any human interaction except for the making of the program itself.

But usually there is nothing to be gained without a penalty. One has to pay for the reduction of sonic data by using synthesis models. They are able to produce only a subset of all possible sound. Sometimes this subset is comparatively small. Hence the originality of the sonic impression becomes limited: sound becomes “standardized”. For example, Frequency Modulation using sinusoids usually yields very characteristic sound and marks every piece produced with its help with a stamp “made by FM”.

As a tradeoff, frequency modulation using sinusoids is amenable to a closed mathematical theory linking the choice of synthesis parameter values to the shape of the resulting frequency spectrum. In other words, sinusoidal FM sound becomes fully predicable. This allows for a conscious planning and manipulation of the synthesis data on the side of the composer — one who understands the mathematics involved — without the need of trial and error usually necessary for “untheoretized” synthesis procedures.

In contrast, no closed theory exists (yet) for more complex synthesis models

of frequency modulation using signals other than sinusoids. The same holds, of course, for Xenakis' Stochastic Synthesis and many other non-standard synthesis procedures. As a consequence, sonic results are much less predictable. In these cases, the composer is forced to develop an intuition by empirically working with the synthesis during extensive trials.

A sensible classification of the different approaches to computerized sound can be made by looking at the object of sound modelization. According to this classification, one can differentiate approaches which model the physical source of sound, the acoustics of sound or something completely independent of both. Consequently, these three approaches are called Physical Modeling, Spectral Modeling, and Abstract Modeling, respectively (cf. [Smi91]). Another technique of creating sound is sampling, which is, strictly speaking, not a sound synthesis but a sound transformation method.

### 8.3 Physical Modeling

Physical Modeling concentrates on the modelization of the physics of sound excitation. It is the science of vibrating bodies, their propagation, resonance and emanation of sound. Virtually all possible methods of sound excitation have been modeled so far: percussion, friction, plucking, resonance, air vibration, turbulence phenomena, etc. (cf. [Roa94]). Drums, clarinets, trumpets, violins, guitars and many other instruments have been simulated with considerable success, including their ability to produce squeaks and other incidental sounds, just as with real instruments. Attempts to model the physics of the human voice have even a much longer history dating back to the 1970s (e.g. VOSIM synthesis by Werner Kaegi and Stan Tempelaars, cf. [Kae86]).

In a way, any musical instrument can be conceived of as a material system being subject to the laws of physics. Playing musical instruments means triggering and controlling algorithms describing the laws of physics concerning sound excitation. It seems as if there were no theoretical limit to simulating the excitation and emanation of sound by a musical instrument modeled within the computer (cf. [Ort96], p. 40). Note however, that there are voices who doubt that all natural physical processes can be simulated by algorithmic action (cf. [Ros88]).

Physical Modeling has generally been advertised as a means to transcend the limits of instruments. Being a simulation in a computer, a physically modeled instrument is not bound to restriction imposed by Nature. People often speak of vibrating strings as long as the San Francisco bridge and clarinets as big as power plant chimneys. Another promise of Physical Modeling is to construct hybrid instruments coupling model components that would hardly work together in reality.

However, even if the virtual instruments differ from real ones, they are still instruments and so, by its very approach to simulate instrumental sound, Physical Modeling is trapped by standard compositional thinking of instruments playing notes. It seems that it is very difficult for physically modeled music to escape the traditional instrumental paradigm of composition (cf. [Sup97], p. 61–62).

Another critique to Physical Modeling is from a more social-economic viewpoint. Physical instruments can be replaced by perfect computer simulations in

commercial music production studios in order to avoid the cost of hiring human musicians. Unless — they offer their skills in mastering these new instruments.<sup>1</sup>.

## 8.4 Acoustic Modeling

In this section, we will look at two important theorems of acoustic sound modeling. These theorems have been originally conceived to analyze, i.e. to describe existing “natural” sound. In computer music applications, they are used to synthesize not-yet-existing sound, i.e. they can serve as a theory of sonic composition.

### 8.4.1 The Fourier Theorem of Spectral Decomposition

The Fourier theorem of sound is universal in the sense that any sound signal can theoretically be conceived of as a combination of elementary sine waves. Sine waves do not exist in nature. They are a good example for the triumph of mathematical reductionist theory, because they work so well, up to a certain point. The paradigm used is that the nature of sound is a wave phenomenon. Because of this, Fourier analysis works perfectly only with sounds of infinite length, which of course do not exist in reality. Any “practical” temporal constraint (as is inherent in music) affects the exactitude of the Fourier decomposition and reveals the uncertainty of wave phenomena: the shorter the signal, the less precise its frequency spectrum (cf. [Bra89]). This fundamental uncertainty relation inherent in the duality of wave and particle phenomena has gained a certain notoriety by the findings of quantum theory.

An application of the Fourier theorem in the domain of sound synthesis is additive and subtractive synthesis. Additive synthesis works by mixing sine waves together to form complex spectra. Subtractive synthesis works by reducing noise to the desired spectrum through filtering. These synthesis paradigms are universal only in theory: theoretically, they can produce any possible sound. However, the sheer amount of control data (frequency, intensity envelope, phase) of the many independent sinusoidal components in a spectrum is difficult to handle and results in reduced and idealized setups, with a tendency to concentrate on the combination of partials. As a result, composers working with spectral analysis and synthesis of sound tend to be biased toward a “vertical” frequency thinking (cf. [Sup97]). However, advanced and sophisticated wavelet analysis, pitch-tracking and the addition of noise components as well as the combination of Fast Fourier Transformation and its inverse (translating spectra back to the time domain) have much enhanced the classical Fourier approach in order to handle the complexity of time-variant sonic evolution (cf. [RDF93]).

Nowadays, sound can be resynthesized from spectral analysis data with extremely high fidelity. Moreover, its spectrum can be arbitrarily changed underway, resulting in novel sounds. (An important application of these techniques is IRCAMs software *AudioSculpt*). Because of its relatively early and comprehensive theoretization as well as the abundant applications in mathematics and physics, sound synthesis relying on Fourier’s theorem was historically the first

---

<sup>1</sup>“[...] a proliferation of [...] Physical Modeling instruments might *increase* the job opportunities for traditional instrumentalists [...] Acoustic musicians can now ‘plug in’ and gain access to the strengths of digital music.” [Bai]

and can still be regarded as a “standard” way of doing sound synthesis. Its first applications even existed before sound recording techniques had been invented: they were electric instruments like the Telharmonium (patented by Cahill in 1897, see [Roa96], p. 134), and the Hammond organ ([Roa96], p. 136).

### 8.4.2 The Gabor Theorem of Sound Granulation

Another universal theory of sound is based on the view of sonic phenomena as a result of quantum sonic effects. In analogy to Einstein’s postulation of an elementary sound quantum, the “phonon” ([Var96], manuscript, p. 184), Dennis Gabor developed the notion of acoustical quanta and experimented with sound “grains”, succeeding in time-stretching and pitch-shifting effects as early as in the 1940s, much before these manipulations became available through magnetic tape machines or later vocoder techniques ([Gab58]). Gabor’s claim for universality of granular sound representation has later been mathematically verified ([Bas68], quoted after [Roa96]). The compositional use of sound granulation is Granular Synthesis, first postulated by Iannis Xenakis ([Xen60]), and applied in his tape composition “Analogique B” (1959), realized at GRM (see section 4.1 on page 59).

In practise, the elementary signals used for Granular Synthesis of sound tend to be less elementary than the sonic quanta according to the theory. Still, many sound grains have to be defined in order to produce sounds of longer duration. The method of Granular Synthesis naturally leads to time-variant spectra, i.e. time evolution is an integral part in Granular Synthesis compositions, and not a secondary control issue, as it tends to be in the Fourier paradigm. The composer is forced to take the horizontal time dimension of sound into account (cf. [Sup97]).

Granular Synthesis has first been implemented on the computer by Curtis Roads [Roa78], but attempts have been made by Xenakis himself in the late 1950s ([LR81], p. 53). Just as with the Fourier paradigm, the large number of elementary sonic grains in a sound call for a large set of control data to be specified by the composer. Most implementations therefore employ statistical approaches to assign grain properties. Recently, the statistical approach has been complemented by deterministic chaos gained from the simulation of complex dynamic systems with the aid of cellular automata ([OHK91], [Mir95b], [Mir99]).

## 8.5 Sampling

Recording natural sounds is an easy way to attain natural complexity with composed sound. It is far easier to realize complex sonic evolution with recorded sound than to generate it within the computer “from scratch”.

Therefore, very many composers fall back upon recording (“sampling”) of natural sound in order to input the wanted complexity into their sonic design. This is true for very diverse approaches to electroacoustic composition, reaching from acoustic photography (composition of “soundscapes”) to artificial assemblies of sound images and heavily processed sounds whose natural origin is impossible to recognize.

Strictly speaking, the practice of sampling does not imply any “model of sound” because sound is already “there”. However, sampling often stretches

over to sound synthesis because of the many transformation techniques applied to the sampled sound. This is to say that heavy transformation of sampled sound can yield results very similar to purely synthetic sound. The difference between a transformational and a generative approach to sound is that in the first case, sonic complexity is input into the computer while in the second case it has to be produced by the computer.

Sampling has a long history, if understood in the wide sense of instruments “speaking with the voice of another instrument” ([Dav96]) and a short history if understood in the narrow sense of processing recorded digital samples on the computer.

Compositional use of recorded material dates back to Dada experiments with gramophone disks by Stefan Wolpe around 1920 ([Dav96], p. 9). Perhaps the first compositions with existing sound were the three studies made by Paul Hindemith and Ernst Toch from 1929-30 ([Dav96], p. 9). In 1939, John Cage operated gramophone recordings of oscillator test tones in his *Imaginary Landscape No. 1*. Other compositions using recorded material were edited on optical film track (e.g. Walter Ruttmann’s *Weekend* (1930)). An extensive analysis of this piece can be found in [Fri02].

Composing with recorded sound was driven to perfection by Pierre Schaeffer and his collaborators at the *Studio d’Essais* at Radio France from 1948 (later to become the famous *Groupe de Recherche Musicale*). Schaeffer developed an original aesthetics around working with recorded sound: the idea of *musique concrète* or of an Acousmatic Music (see section 4.1 on page 59). Here we can indeed speak of a very pure attitude toward working with the acoustic material of sound. In order to obtain their concrete sounds, the composers of the Concrete School sometimes created novel instruments (cf. [Sch52]) and tools for sound transformation, eventually leading to an important computer software toolbox called *GRM tools* (see section 4.1 on page 59).

Transformation of sampled sound is a very common technique in electroacoustic composition. It means the combination of sampling and (re)synthesis techniques. Common techniques of transformation include spectral modeling of sampled sound in phase vocoder applications like IRCAM’s AudioSculpt, or granular (re)synthesis of sampled sound through e.g. the “brassage” tool of GRM tools. Some composers like e.g. Paris based Argentinian composer Horatio Vaggione have developed such artistic skill in the transformation of sampled sound that the results are not manipulated sounds but autonomous sonic artifacts.

Sampling is even available in Xenakis’ UPIC system. Originally, all sound was thought to be drawn by hand. From version 3 there is the possibility of sampling “natural” sound into the system instead of drawing it. Very impressive results can be obtained by scattering grains of sampled sound over the audible range with the help of the UPIC drawing facilities, realizing a kind of Granular Synthesis within the UPIC system.

## 8.6 Abstract Models

In contrast, in Xenakis’ GENDYN the complexity of sound is created within the computer, instead of being sampled into it. One can fairly say that GENDYN and other non-standard synthesis models use the Shannon-Nyquist sampling theorem “in reverse”. Instead of representing existing sound in a digital format

by sampling (as is done e.g. by a recording on a CD) they create new sound by synthesis of digital samples.

**Definition 8.1: RAC Sound** In a radical algorithmic approach to sound synthesis, the complexity of the sound signal is provided by the generating algorithm itself, instead of exploiting the complexity of already existing sound or using existing models of sound excitation or transformation.

Sound synthesis by “Abstract Models” implies synthesis models detached both from physical sound source and from acoustics. This means that these sounds are abstract computed structure; they are thought to have no physical, natural cause, and no preexisting model. Abstract modeling of sound can be viewed as an avant-garde approach to sound synthesis, similar to abstract painting or abstract film which are detached from visual reality.<sup>2</sup>

As a consequence, there is no restriction whatsoever concerning a preconceived theory of hearing, a “natural” disposition of the human ear toward certain classes of sonic phenomena, or a “human scale” of listening.

Sound as a sequence of numerical data can be conceived of as the result of arbitrary algorithmic computation (for example as the computation of a real number with sufficiently many digits after the decimal point, as in the case of the infinite number  $\pi$ ). In this sense the output of a Turing machine (see below, section 10.2 on page 161) can be interpreted as digital sound. Indeed, approaches like “Instruction Synthesis” (see section 5.7.5 on page 75) are very close to such a concept.

It would be impractical to define every single sample value in order to obtain interesting musical evolutions on all super-ordinated time scales. This would mean that every single sample value would have to be considered a “parameter” of such a sound (cf. [Jaf95]). Therefore, just as with standard synthesis techniques, abstract synthesis uses synthesis models in order to reduce the task of sound generation to the control of a few synthesis parameters. The difference to standard acoustic or engineering models is that abstract models have to be developed by the composers themselves. In turn, they are invited to endow these models with their individual compositional flavor, i.e. to extend the impact of composition even to the sonic model of their music.

### 8.6.1 Modulation Techniques

Synthesis models like Amplitude Modulation (AM) and Frequency Modulation (FM) are not inspired by acoustics but by electroacoustics. AM and FM are standard techniques of radio broadcasting where a carrier signal (carrier for short) is modulated by another signal (modulator for short) in order to be transmitted over long distances. Transposed to the audible range, AM and FM are very effective procedures to achieve a wealth of possible sounds with comparatively little computational effort. In its simplest case, FM needs only two oscillators: one carrier and one modulator ([Cho85], quoted in [Roa96], p. 226). In contrast, Additive Synthesis, in theory, employs an infinitude of oscillators in order to produce one single sound.

Amplitude and frequency are (the) two fundamental aspects of sound, and therefore it is no wonder that they have both been chosen for modulation tech-

---

<sup>2</sup>A good source for abstract art is *iota*, <http://www.iotacenter.org>.

niques. For example, a sine wave, Fourier's "atom" of sound, is exhaustively described by its amplitude (the absolute value of both its positive and negative peak) and frequency (the number of repetitions of the sine wave form within a second). Amplitude and frequency are somehow dual aspects of sound (or wave phenomena in general). Sound can be represented as amplitude (sound pressure) varying in time (time function), or alternatively, as a bunch of its constituent frequencies evolving in time (spectrum). Both representations can be transformed into one another by the Fourier resp. the Inverse Fourier transformation.

AM and FM can be explained as generalizations of two fundamental musical principles of tremolo and vibrato, applied within the electroacoustic domain. The difference of tremolo and vibrato to AM and FM is that with the former, the frequency of modulation stays below the threshold of 20 Hz (lower audible range). Therefore, both tremolo and vibrato do not alter the structure of the carrier's spectrum. (As an aside, musical application of tremolo and vibrato is of course a much more complex phenomenon than just the variation of amplitude and frequency of a musical tone.) In contrast, AM and FM do produce spectra through the addition of partials, the so-called sidebands, to each of the carrier's harmonic. With AM, two sidebands are produced while FM can produce many sidebands, depending on the depth of modulation (cf. [Roa96], p. 230). By varying the modulation depth over time, transient spectra can be produced, approaching the complexity and liveliness of natural sound. Both AM and FM are discussed here because Xenakis' Stochastic Synthesis can be understood as a combination of the two, using stochastic signals as modulators and complex wave shapes as carriers.

### Amplitude Modulation

In electroacoustic music, amplitude modulation has been one of the first synthesis techniques used because it could easily be realized with the hardware equipment of the early electronic studios. Therefore, amplitude modulation, or its sister technique ring modulation, determined much of the sounding of early electronic music. With amplitude modulation, depending on the frequency ratio of carrier and modulator, harmonic or inharmonic spectra can be obtained. Circuitry implementing ring modulation (4 diodes switched together in a ring circuit, hence its name [Roa96], p. 220) could even be used to alter instrumental live sound, a technique which was widely used in instrumental avant-garde compositions of the 1960s.

In Xenakis' UPIC system, where the waveforms can take on shapes of arbitrary complexity, AM is possible as the modulation of a waveform by an envelope. Therefore, a generalized form of amplitude modulation can be realized, with modulator and carrier not being restricted to sinusoid signals.

### Frequency Modulation

In its simplest form, FM can be implemented with just one carrier and one modulator signal. However, in practice, FM implementations are done in a slightly more complicated way. It has been found that FM yields better results when the modulator is in turn modulated either by itself or by the carrier, i.e. modulation is fed back to the modulator. This setup is called Feedback FM. For example, the famous Yamaha DX series synthesizers achieved their unique sounds with

the help of a specific Feedback FM technique (cf. [Roa96], p. 245), having a strong impact on the sound of both popular and academic electroacoustic music in the 1980s.

Another variation of FM is to use other wave forms than sinusoids for carrier and modulator. For example, in Xenakis' UPIC system, where the waveforms can take on any shape drawn by hand, FM is possible between any two waveforms (from version 2, 1983), and with as many waveforms as wished. As a consequence, the resulting timbres are far richer than could be produced with sinusoidal FM. Since in the UPIC, any waveform can modulate any other, feedback is also possible. The result is, for want of a better term, Complex Feedback FM. The UPIC bank of oscillators, when being intermodulated by feedback FM, go onto a road of chaotic oscillation, producing long evolutions of transient sonic phenomena, almost a music in its own right (cf. [Bel01], [Bel97], [Bel99]).

With Complex Feedback FM, sound synthesis transgresses the threshold to automated composition, in the very sense of Sonological Emergence (see section 5.7.5 on page 76). This means that by using a sound synthesis technique, Complex Feedback FM, a sonic evolution can be created that has sufficient internal differentiation as to form the substance of a whole section of music, if not a whole musical piece. Since the processes are chaotic, working with these sonic evolutions has a strong aspect of interactiveness (since the composer must share control with the sound synthesis) and non-deterministic composition.

This is not much different from working with GENDYN. After setting up the parameters to the GENDYN program, control is yielded to the computer. However, parameters could be altered during synthesis. It is curious how the two inventions of Xenakis', although conceived of as complementary (fixed, hand-drawn UPIC waveforms vs. dynamic, ever changing GENDYN ones) can both fulfill Xenakis' lifelong dream of an "Automatic Music", in spite of the fact that Xenakis himself was seemingly not aware of this being possible with his UPIC system.

### 8.6.2 "Non-standard" Synthesis Models

Non-standard-synthesis is, as synthesis with Abstract Models in general, conceptually dependent on the Shannon-Nyquist sampling theorem. Since it does not realize any physical or acoustical model of sound, it can only be realized through a digital representation of sound. Non-standard synthesis thrives on the fact that digitized sound can be computed. The sound signal, then, is not the end product of a physical or acoustic theory of sound but an artistic object in its own right. With a non-standard approach to computer synthesis and composition, a composer invents the theory of composition and also the model of sound for every one of his/her compositions.

Non-standard sound is artificial in an emphatic sense: it cannot exist in nature, not even theoretically, because such sounds do not correspond to the sound excitation modes of vibrating bodies. Non-standard synthesis often involves sound signals which are acoustically non-sensical. For example, GENDYN sound signals are step functions which cannot be produced other than by digital computation.

Computation of digitized sound is the most unbiased and unflavored technology one can use for artistic purposes. This is why Xenakis was interested in

computing sound from the mid-1950s, resulting in the foundation of CEMAMu and ultimately the creation of GENDYN.

Composers with a non-standard approach to computer composition, like Holtzman, Berg, Koenig, Brün and Xenakis make use of the Sampling Theorem in a conceptual, creative way. They do not use the Sampling Theorem in order to process existing sound. Instead, they use the Sampling Theorem in reverse, in order to create new sounds, changing its descriptive function into a productive one. This can by itself be considered a heretical use of sound technology. Holtzman, Koenig, Berg, Brün and Xenakis did not care if the sounds so produced corresponded to pleasant, natural or even plausible sounds. They were interested to create coherent compositional processes which would manifest themselves in new sounds and sonic structures. In the case of Holtzman and Berg, it was sequences of arithmetic and logic operations on binary numbers (section 5.7.5 on page 75). In Koenig's case, it was techniques taken from serial composition applied to the molecules of sound, i.e. sound waveform segments (his SSP sound synthesis program [Roa96], pp. 326-27). In Brün's as well as in Chandra's case, it was abstract operations on waveform segments (his SAWDUST program, cf. [Roa96], pp. 324-26). In Xenakis' case, it was stochastic processes working on the shape of a wave form period. The latter is what will be presented now in detail.

### 8.6.3 The Model of Dynamic Stochastic Synthesis

In a narrow sense, Dynamic Stochastic Synthesis, or DSS for short, is a model for sound synthesis with the help of a computer (digital sound synthesis). However, as with many non-standard abstract models for sound synthesis, it is more than just that. DSS does not only generate sounds but long-term evolution of pitch and dynamics, properties which are usually not considered aspects of sound synthesis but of score composition. Therefore, DSS does more than only deliver raw sonic material for further compositional use. The sonic result of DSS is already the end product of an act of composition. Xenakis' GENDYN program does nothing more than running several instances of DSS simultaneously in order to create the music.

Indeed, DSS is inspired by a compositional idea. This idea is the concept of variation, taken in its broadest sense as a constant change of something. Xenakis understood this idea as a universal principle which, according to him, does not only apply to music.

“The variation of [a] theme is a kind of repetition: the not quite faithful reproduction of a unit (the theme). Between the two, there is a little change. If the first variation is followed by others, eventually we get so far from the theme as to make the link between the last variation and the starting point unrecognizable. [...] The distance produced by more or less faithful repetition is once again an aspect of the question of continuity-discontinuity.” [Var96], manuscript, part I, p. 83

The unit of variation in the case of DSS is a wave form period. A wave form period is a tiny chunk of a sound signal which is repeated in time. The simplest example of such a wave form is a sine wave or square wave. A simple

sound synthesis would work by just producing successions of sine or square or any other simple geometric waves (“wave-table synthesis”). In the case of DSS, however, the process is a bit more complicated. We will see in a moment how it works.

### The Concept of Polygonal Variation

One fundamental composition principle of Xenakis’ which is faithfully reflected in GENDYN is his concept of all music being driven by the principle of repetition and alteration ([Var96], [Xen96]). In GENDYN it is applied to a very abstract notion of “variation” within the sound structure itself, i.e. the micro-structure of sound, the shape of a wave form repeated in time. Xenakis called the application of the variation principle to a wave form “Polygonal Variation”. He first applied this principle in instrumental music, for the definition of pitch contours, e.g. in *Ikhoor* (1978), as indicated by himself (cf. [Xen85], Appendix B). In GENDYN, however, he applied Polygonal Variation to the “micro-composition” of sound.

Polygonal variation, in Xenakis’ terms, means the variation of a polygonal shape. This polygonal shape, in the case of Xenakis, is not a closed geometric form but an “open polygon”, i.e. a collection of straight lines which are connected at their end points (vertices) to form a kind of jagged mountainscape. Such an open polygon is defined by the set of the geometric positions of its vertices. When the polygonal form is to describe a melodic line, the vertices are identified with “notes” (a pitch value positioned in time). When, as in the case of GENDYN, the polygonal form is interpreted as a wave form, the vertices are the breakpoints of a wave shape, and the straight lines are the linear interpolations between these breakpoints. In both cases, the horizontal dimension (the extension of the mountainscape) is time (musical time resp. micro-time), whereas the vertical dimension (the “peaks” of the mountainscape) is pitch (frequency) in the one case, and digital amplitude in the other.

The concept of the open polygon, taken as such, is independent of its interpretation as a “melodic line” or a “wave form”. This is very typical for Xenakis’ attitude to creative processes. Taking one step back, he found abstract principles underlying different aspects of composition and even different artistic domains. A famous example is e.g. the application of ruled surfaces to both architecture (the *Philips Pavilion*) and composition (*Metastaseis*) which is well documented in the Xenakis literature. This application of a geometric principle (approximation of curved surfaces by bunches of straight lines) to both architecture and music initiated the famous Xenakis glissandi (straight lines in the domain of pitch vs. time). For sound synthesis, Xenakis applied the straight lines to the physical sound signal (a sort of sound pressure glissando in micro-time).

This abstract thinking in geometric shapes, regardless of the domain of application, went in line with Xenakis’ postulation of a universal “Theory of Shapes” (cf. [Var96]). Xenakis was convinced that geometry be the unifying principle from ontogenesis onward up to the cosmic scale. He reduced time, physical laws, biological evolution and even cultural history to being mere epiphenomena of a deep underlying shape-generating force ([Xen92e]). Curiously enough, in the light of physics’ current String Theory (for a description see e.g. [Gre00]), some of Xenakis’ speculations take on a very topical if not prophetic taste.

We see by these reflections that GENDYN, the application of DSS to music composition, is not only a radical approach to sound synthesis and algorithmic

composition, but a kind of practical philosophy in the domain of arts, since every aspect of this music is reduced to the emerging shape of a wave polygon. But how is this polygonal shape altered, and how is it defined in the first place? This question is the subject of the next section.

### The Concept of Random Walks

We have understood that in GENDYN sound composition, everything is reduced to the shape of a wave form undergoing a process of “Polygonal Variation”. Therefore, we now have to understand the aspect of variation of this polygonal shape.

We have seen that the polygonal wave shape is completely defined by the geometric positions of its breakpoints in the 2-dimensional space of amplitude (y-axis) vs. time (x-axis). “Polygonal Variation” means repeating the polygonal wave while displacing its breakpoint positions in both the time and amplitude dimension. Therefore, the polygonal wave shape is deformed under repetition. This is what differentiates DSS from wavetable synthesis as realized e.g. in Xenakis’ UPIC system: With UPIC, a wave shape is repeated unchanged. This accounts for a rigid, frozen timbre of the resulting sound even if the shape is very complex. With DSS, the shape may be simple, consisting only of a few breakpoints, but the dislocation of these breakpoints over time produces a rich and ever-changing spectrum. In this respect, GENDYN sound is similar to natural sound, since sound in nature (including sound by “natural” music instruments) is always changing in time. At the same time, GENDYN sound is artificial because polygonal shapes cannot be produced by natural sound excitation; they are mathematical, digital artifacts. This is the reason why GENDYN sound is both rich and strange, to pick up a wording coined by the composer himself.

The change (the “variation”) of the polygonal form is not arbitrary but stochastic. This means that it is governed by stochastic processes (i.e. Random Walks) which are a mathematical concept of randomness. We can speak of a kind of dynamic wavetable synthesis where the wavetable is stochastically altered in shape. This is the reason for the term “Dynamic Stochastic Synthesis”. In the following, we will discuss how both concepts, the concept of Polygonal Variation and the concept of Random Walks, together form the model of DSS.

The displacements of each breakpoint in both amplitude and time dimension each follow a Random Walk. This means that there are 2-times-the-number-of-breakpoints Random Walks active for the deformation of a single wave form over time. All these Random Walks are uncorrelated and completely independent of each other. Therefore, the polygon’s deformation does not obey any linear geometric transformation. It is non-linear.

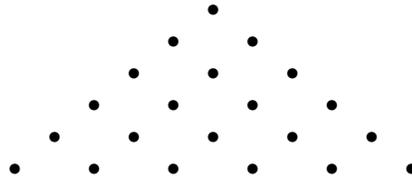
One can think of the waveform’s breakpoints as billiard balls performing Random Walks in an abstract space, the 2d space of amplitude and (positive) time increment (since there cannot be any movement backward in time). When a billiard ball moves up in the direction of increasing amplitude, its associated breakpoint forms a peak within the waveform polygon. When a billiard ball moves to the right in time, its associated breakpoint moves apart from its preceding breakpoint, expanding the wave’s length (and decreasing the resulting frequency of the sound).

The billiard ball picture is a graphic picture for Brownian Motion, a phenomenon observed by Scottish Botanist Robert Brown (1773–1858) around

1827. Brownian Motion is the erratic movement of macroscopic particles in a liquid. It is caused by the fluctuation of the collision forces by the liquid's microscopic molecules bombarding the particle from all sides, according to the kinetic theory of gases and liquids. According to this theory, the phenomena of temperature and pressure are statistical phenomena which can be reduced to the massive chaotic movement of a very great number of atoms or molecules (Statistical Mechanics). With the help of Brownian Motion, Einstein proved the existence of atoms and even measured their weight in a seminal paper of 1905.

The mathematical theory to describe Brownian Motion is the theory of Random Walks. We will consider here only one-dimensional (1d) Random Walks, i.e. Random Walks along a line, just forward and backward, on the y-axis of amplitude and the x-axis of time because these two coordinates determine the positions of the waveform's breakpoints.

The easiest model of a 1d Random Walk is one with equal unit steps forward or backward, with the same probability (e.g. by flipping a coin), a stochastic process. At each step, the coin decides whether the next step will be forward or backward. A nice simulation of such a stochastic process is the Galton board. A Galton board is a board with pins on it which are arranged in rows so that at each row, a pin ball being thrown down from top is deflected either to the left or to the right side.



The pin balls are collected in bins at the bottom of the Galton board. It is clear that the bins in the middle will contain more balls than those at the extremities of the board, if the probability of right and left deflection are equal. Each distinct path through the rows is unique but there are more paths leading to bins in the middle than to bins on the extreme left or right. Indeed, there is only one path that leads to the leftmost resp. rightmost bin. In Statistical Mechanics, one speaks of several micro-states (unique combinations of left-right deflections) building one macro-state (the end result, the ball ending up in a specific bin).

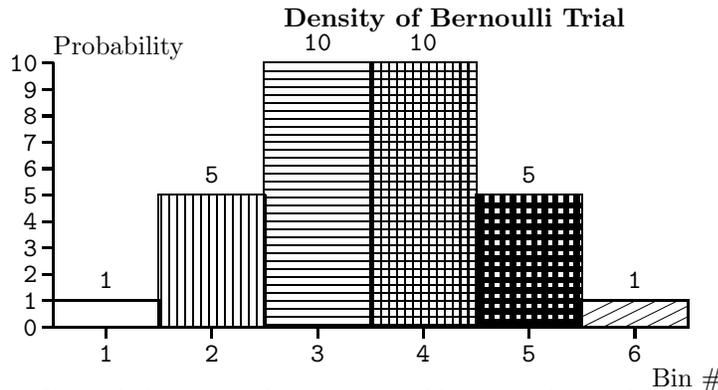
An easy way to view the number of paths that lead to each bin is to construct the Pascal triangle: put a 1 on the top row, and in each subsequent row put the sum of the two diagonally adjacent numbers from above (with the borders assumed as 0's). This is a good example of a recursive algorithm: It is called recursive because its rule is defined in terms of itself (it takes recursion to its own definition):  $Pascal_{i,j} = Pascal_{i-1,j-1} + Pascal_{i-1,j+1}$ , where  $i$  is the row number and  $j$  is the column number of the Pascal "matrix".

$$\begin{array}{cccccccc}
 & & & & & & & 1 \\
 & & & & & & 1 & 1 \\
 & & & & 1 & 2 & 1 & \\
 & & 1 & 3 & 3 & 1 & & \\
 & 1 & 4 & 6 & 4 & 1 & & \\
 1 & 5 & 10 & 10 & 5 & 1 & & \\
 \dots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots
 \end{array}$$

The Pascal triangle is a good example of how complicated formulae can be easily computed using a simple algorithm. One can use the Pascal triangle to count the number of paths leading to each bin under the Galton board. The “complicated” formula for calculating each individual number of the Pascal triangle is the one giving the coefficients of the binomial expansion of  $(a + b)^n$ :

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} \quad (8.1)$$

The binomial expansion is nothing else than a simple combinatorics of terms because the binomial expansion is the sum of all products that can be formed by systematically choosing either an  $a$  or  $b$  from each bracket. For example, the expression  $(a + b)^2$ , i.e.  $(a + b)(a + b)$ , yields  $aa + ab + ab + bb = a^2 + 2ab + b^2$ , i.e. all four possible combinations of  $a$ 's and  $b$ 's. We see in this simple example the coefficients of the third row of the Pascal triangle. Thinking of the  $a$ 's as denoting a step to the left and of the  $b$ 's as denoting a step to the right, the combinatorics of  $a$ 's and  $b$ 's also describe all the possible paths of a Random Walk of a pin ball through the Galton board ([Fow02]).



If one is interested in the probability by which a pin ball lands in a specific bin, one is to consider the combined probability of the left-right-decisions that lead to that bin. Since we assume each pin as being a “fair coin”, deflecting the ball to the left or right with equal probability, a specific path through  $n$  rows must happen with probability  $(\frac{1}{2})^n$ . Remember that in general, more than one path (i.e. micro-state) leads to a bin (i.e. macro-state), the exact number being given by the combinatorics discussed above. Therefore, the distribution with which the pin balls lands in each bin at the bottom of the Galton board is the *binomial distribution*

$$b_n(k) = \binom{n}{k} \left(\frac{1}{2}\right)^n, k \in \{0, 1, \dots, n\} \quad (8.2)$$

For example, the probability of a pin ball landing in the leftmost bin is

$$b_5(0) = \frac{(5)!}{(5-0)!0!} \left(\frac{1}{2}\right)^5 \quad (8.3)$$

$$= \frac{(5 \cdot 4 \cdot 3 \cdot 2 \cdot 1)}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1} \left(\frac{1}{2}\right)^5 \quad (8.4)$$

$$= \frac{1}{32} \quad (8.5)$$

The probability of a pin ball landing in the rightmost bin is given by

$$b_5(5) = \frac{5!}{(5-5)!5!} \left(\frac{1}{2}\right)^5 \quad (8.6)$$

$$= \frac{1}{32} \quad (8.7)$$

The probability of a pin ball landing in one of the center bins is

$$b_5(3) = \frac{5!}{(5-3)!3!} \left(\frac{1}{2}\right)^5 \quad (8.8)$$

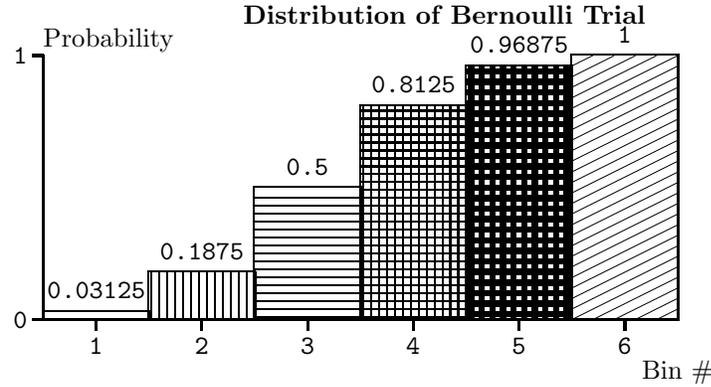
$$= \frac{(5 \cdot 4 \cdot 3 \cdot 2 \cdot 1)}{(3 \cdot 2 \cdot 1 \cdot 1)(2 \cdot 1)} \left(\frac{1}{2}\right)^5 \quad (8.9)$$

$$= \frac{10}{32} \quad (8.10)$$

If we consider systems with greater numbers of constituents than pin balls, i.e. a liquid or a gas, then the binomial distribution transforms into the Gauss distribution with its famous bell shape. However, the same considerations apply to these massive processes. Therefore, a diffusion process can be modeled as a Random Walk. This is what Einstein did for Brownian Motion and this is what Xenakis did for clouds of string glissandi in his early orchestral works, and, of course, in his ST-program of 1962, and finally for GENDYN sound synthesis in 1991. Let us see how he did this.

### Computer Simulation of Stochastic Processes

We have seen in the foregoing section how complicated mathematical formulae (the binomial coefficients) can efficiently be computed by simple algorithms (the Pascal triangle). When we want to model stochastic processes, like random walks, with the computer, we are not interested in an analysis which will tell us the probability of a certain event to happen but we want to have this event occurring with a given probability. In other words, instead of putting in an event and getting out its probability we want to put in the probability and get out the event.



To this end we can use the distribution formulae of stochastic processes. They show the distribution of all possible events by summing up their probabilities from 0 (certain not to happen) up to 1 (certain to happen). For example, it is 100% certain that the pinball will land in *any* of the bins under the Galton board. If we are interested in the question with which probability the pinball will end up in any but the rightmost bin, we have to subtract from these 100%  $\frac{1}{32}$  or 3.125% to yield 96.875%. In other words, we have summed up the probabilities of all events but the last to happen.

That is, in the above example, we count the probabilities of all bins containing the ball except the last. Now, if we are to *produce* a stochastic process, i.e. simulate a ball going down a Galton board, we put in a number between 0 (certain not to happen) and 1 (certain to happen) with equal probability. A computer can compute such a number with the help of a recursive algorithm that will be discussed later in this study. Now we lookup the point on the y-axis corresponding to that number and see to what bin it corresponds (we draw a imaginary horizontal line from the point on the y-axis to the right and see which of the bars it hits). If we do this often enough, we get a faithful simulation of the Galton board game. Xenakis did the same in GENDYN, except that he did not simulate the Galton board game but diffusion processes (a set of bell shaped curves).

## Chapter 9

# Algorithms and Computer Programs

“The computer, which has arisen through the wealth of achievements of the human mind through the millenia, cannot create anything new. Several mathematicians, for instance the Nobel price-winner Simon Newell, tried to create theorems with the computer. Recently it was demonstrated that this is not possible. [...] Beneath the level of consciousness there lies all this fantastic amount of intuition that ultimately leads to a rational expression, but without this intuition it is impossible to create anything.” [Var96], quoted after the manuscript, p. 148/49

In this chapter, we discuss the fundamental notion of an algorithm and how it relates to the practice of computer programming. We will see an example of a very simple but ingenious algorithm and the way it is executed with the help of a small set of simple rules which can be carried out by a machine. This example will serve as a preparation to the formal description of Xenakis’ Dynamic Stochastic Synthesis algorithm as a procedure of generating a lists of numbers which can then be interpreted as the sound of a piece of music.

Often, in common language, “algorithm” and “program” are used as synonyms. However, there is a big difference between the two. An algorithm is just a description of an procedure to solve a given problem. Therefore, it is very general. Algorithms exist from the times of the Ancient Greeks. A famous example is Euklid’s algorithm of finding the greatest common denominator of two natural numbers. The term “algorithm” is derived from the name of an Arab mathematician at the court of Bagdad, “Al Chorezmi” (783-850).

An algorithm is independent of the processor which is to execute it. The processor can be a human (e.g. a mathematician solving a set of equations) or a machine (a dedicated automaton or an electronic computer).

A program, on the other hand, is a possible implementation of an algorithm on a computer, in a specific programming language. In order to be executed, a program has to be compiled to a specific machine code and run under a specific operating system on a specific computer hardware. Computer programs only exist since there are computers, that is, not before the 1930s.

## 9.1 What Is an Algorithm?

Let us first give a short definition of an algorithm.

**Definition 9.1: Algorithm** An algorithm is a finite description of a method for finding a solution to a given problem. This description is so precise and exhaustive that it does not require any further knowledge or insight on the side of the processor.

The above definition is quite a general one, corresponding to the generality of the notion of algorithm itself. What is so general about an algorithm? Well, it is the fact that an algorithm is independent of the domain of application (mathematics, process engineering, information technology, art), the language in which it is formulated (English, Chinese, Flow Chart, mathematical equations), the strategy chosen (“divide and conquer”, “brute force”, “genetic programming”), or the nature of the processor executing it (human, machine, physical system, cellular automaton).

The very idea of algorithm is intimately linked to the problem of logical foundation of mathematics and sciences, as well as to the notion of the axiomatic method and formal proof. Preconceived by the ancient Greeks, it was formulated by the mathematician David Hilbert as a challenge to the mathematical community around 1900. The mathematician Gregory J. Chaitin states this “Hilbert program” in the following way:

“Hilbert’s idea is the culmination of two thousand years of mathematical tradition going back to Euclid’s axiomatic treatment of geometry, to Leibnitz’s dream of a symbolic logic, and Russel and Whitehead’s monumental *Principia Mathematica*. Hilbert wanted to formulate a formal axiomatic system which would encompass all of mathematics. [...] A formal axiomatic system is like a programming language. There’s an alphabet and rules of grammar, in other words, a formal syntax. [...] Hilbert emphasized that the whole point of a formal axiomatic system is that there must be a mechanical procedure for checking whether a purported proof is correct or not [...] In other words, there is an algorithm, a mechanical procedure, for generating one by one every theorem that can be demonstrated in a formal axiomatic system. [...] That’s the notion that mathematical truth should be objective so that everyone can agree whether a proof follows the rules or not.” [Cha97b], p. 14–16

The mathematician Allan Turing, answering Hilbert’s challenge, devised a formal system capable of automating arithmetics in 1936, his famous Turing machine ([Tur36]). He proved that is was inherently incomplete: not everything which is a true mathematical statement can be automatically proven: there are simply not enough conceivable Turing machines! The famous incompleteness theorem established by Kurt Gödel in 1931 is implied by Turing’s findings. The Turing machine is a simplified thought model of a computer. Six years later, the first physical computer was constructed ([Roj95]). Today’s computers are faster, smaller, and more comfortable to use, but in principle they are no more powerful than this first one, because all algorithmic action is bound to the specifications

and limitations established by Gödel and Turing. Alonzo Church's Lambda Calculus, invented at the same time as Turing's machine, is an equivalent formalism equally expressing all the power and limitation of computation.

In order to be executed on a computer, an algorithm has to be implemented by a computer program. The world is full of computer programs, but algorithms are comparatively few. One and the same algorithm can be implemented in infinitely many ways, depending on the programming paradigm and the programming language chosen. For example, the programming can be oriented toward a procedural, declarative, or object-oriented paradigm of computer programming. Moreover, a computer program is directed toward translating the algorithm's problem solving strategy toward the specific capabilities of the program execution environment, i.e. it has to be adapted to the language primitives. A notorious example for this dependency is the handling of character strings, which can be extremely different in different programming environments, if one thinks of e.g. string processing in LISP, Java or C.

An algorithm is a short and dense description of a problem solving method. A computer program, on the other hand, tends to be bulky, because programming has to consider control of periphery (disk access, keyboard input, the computer screen) and limited resources (RAM memory and disk storage). An extreme example is Xenakis' GENDYN program. Xenakis wrote about 15.000 lines of program code to implement his algorithm in the BASIC procedural programming language, which takes only a few lines to describe in his book [Xen92b]. However, the very same algorithm can be implemented by a few hundred lines of code in an object-oriented style, the programming paradigm chosen by the author for the New GENDYN program. An implementation in a declarative programming language would yield an even even smaller program. An example for this is shown below.

But let us first go into more detail about the notion of "algorithm". We have seen that the notion of algorithm is quite an abstract one. However, despite this abstractness, some properties of an algorithm are very distinct.

- An algorithm is always finite, in contrast to its execution by a processor, which may run forever (consider, e.g., the main process of an operating system).
- There is a clear-cut correspondence between problem and solution which is laid out beforehand, i.e. it is known that a solution, when found, is valid for a given problem, prior to executing the algorithm.
- An algorithm is unambiguous and explicit. The processor does not need to guess and try — instead it must slavishly follow the instructions of the algorithm in order to accomplish its task.

## 9.2 An Example: the Sieve of Eratosthenes

As an example, let us look at a very classical algorithm. This algorithm is called the sieve of Eratosthenes. It computes a series of prime numbers. In fact, it is a constructive definition of what a prime number actually is. (As a matter of fact, most algorithms can be regarded as constructive definitions or constructive

proofs because they simply construct the theorems that are to be proved or described.)

A prime number is a number which cannot be divided by any other number greater than 1. For example, 17 is prime because there is no number, except 1, which divides it without leaving a residue. The Sieve of Eratosthenes finds all prime numbers in ascending order by deleting the multiples of all numbers except 1 from the list of all numbers. The algorithm is called a “sieve” because it lets all the non-prime numbers fall through, retaining only the prime ones. In other words, the sieve function transforms the list of natural numbers into a list of prime numbers.

An algorithm can be informally described as a human-readable text or formally as a set of mathematical definitions. Let us start with a textual description of the Sieve of Eratosthenes, before giving a formal one.

1. Take 1 as a prime number.
2. Take 2 as a prime number.
3. Delete all multiples of the last prime number found from the ordered list of natural numbers.
4. Take the next number in the resulting list as a prime number.
5. Repeat steps 3 to 4 forever.

In this simple example, we can easily recognize the distinct properties of an algorithm that we have listed above.

- The algorithm of the Sieve of Eratosthenes is finite, in contrast to the solution it computes which is infinite. (There are infinitely many prime numbers.)
- Prior to the execution of the algorithm, it is already clear that the result will represent the desired solution. This is because the construction of the list (deleting all multiples) corresponds to the very definition of a prime number (to be no multiple of another number except 1 and itself).
- The algorithm is precise and exhaustive. At every point of execution it is clear what to do next. The steps are sufficient to produce the result.

In addition to these properties, The Sieve of Eratosthenes has the property of being effective, i.e. every step is a progress toward the complete solution. In other words, the action described by the algorithm approximates the solution ever closer as the algorithm is executed. This is a property that is often demanded for an algorithm to have, but that we did not mention in our definition of an algorithm. Indeed, there are algorithms that do not necessarily approximate a wanted solution but proceed rather randomly. Examples are genetic programming and randomized algorithms. Such algorithms are used for problems where the path to the solution is not known beforehand but has to be explored by the algorithm itself.

The Sieve of Eratosthenes nicely prepares the discussion of Xenakis’ GENDYN algorithm. Just like the sieve, GENDYN computes a (huge) list of numbers (i.e. the digital samples of the sonic appearance of a GENDYN composition).

Similar to the sieve, the GENDYN algorithm can run forever, if not stopped by a super-ordinated control structure (“stop after having computed 20 minutes of music”).

### 9.3 GENDYN and Dynamic Stochastic Synthesis

There have been different descriptions of Xenakis’ GENDYN algorithm: by the composer himself [Xen92c], by his then-assistant Marie-Hélène Serra [Ser93], and by composer Agostino DiScipio [DiS98]. All of these descriptions do well present the principle of Stochastic Synthesis. However, they do not cover the actual functioning of Xenakis’ GENDYN program, i.e. the program which produced the music of *GENDY3*. In the context of this study though, we are interested in exactly this algorithm, i.e. the one defining the operational behavior of the program which computed *GENDY3*. Unfortunately, the description of this algorithm does not exist. Even the program which computed *GENDY3* does not exist anymore. It was not anymore on Xenakis’ computer at CEMAMu, and no backup of it could be found at CEMAMu.

Fortunately, Xenakis had printed and filed the program text in a folder that was still sitting on a shelf in his room at CEMAMu when I came. On the first page of the program listing, there is a note “15-11-91 pour S403” in Xenakis’ handwriting. The program has 877 lines (printable on 85 densely filled A4 pages). In the same folder, there was another document which is a plotting of *GENDY3*’s time structure, entitled “S403 (idem S402)”. This document is annotated “données du 5-11-91 pour GENDY3 de Metz”. Therefore, both the plotting and the listing refer to the two mono sound files, “S402” and “S403”, which were combined by Xenakis for the stereo version of *GENDY3* premiered on the 17th of November 1991 at the *Rencontres Internationales de Musique Contemporaine*, Metz (France), and later published on CD [Xen94a]. Note that both plotting and program text are different from the excerpts published in [Xen92b].

Once I knew that I had found the original program, I only had to compare the listing with a later version of the same program which I could copy from Xenakis’ computer, and change it back to obtain an electronic version of the original program. Then, I executed the program to obtain *GENDY3*.

We have already stated that the text of Xenakis’ GENDYN program is extremely bulky and obfuscated, and not a good documentation of the process of computing *GENDY3*. So what we are interested in is not so much the program text than the algorithm which is implemented by it. Since there was no such description of the algorithm, I had only one chance: infer it from the program text! This procedure is inverse to standard software engineering: normally the algorithm comes first and then the implementation by a computer program. Inferring an algorithm from an existing program text is therefore called reverse-engineering. In other words, the idea is to study the program in order to find out how an algorithm specifying the operational behavior of this program should have looked like. We will now describe this algorithm.

## 9.4 The Algorithm of Dynamic Stochastic Synthesis

Let us take a look at the algorithm of Dynamic Stochastic Synthesis as it has been implemented by Xenakis' GENDYN program, in the version which computed *GENDY3*.

This algorithm is made up of two parts, which Xenakis actually implemented in two separate programs, "PARAG" for defining the duration structure and the sound synthesis parameters for each of the sections of *GENDY3*, and "GENDYN" for the sound synthesis proper. I call the first program the "control structure" of GENDYN. The control structure determines the multiplicities of sounds (up of 16 DSS processes running in parallel) and both the activation/deactivation of synthesis and the lengths (durations) of the sound/silence patches in time.

Let us start from bottom to top and first describe the GENDYN algorithm of stochastic sound synthesis. GENDYN is a waveshaping synthesis, where the wave shape is formed of a series of breakpoints and their interpolation (see Xenakis' own sketch as found in [Xen92b] in figure 9.1. The wave shape is constantly altered by displacing the wave's breakpoints both in the time dimension (x-axis) and the amplitude dimension (y-axis) by stochastic processes (Random Walks). It is important to note that there is a "primary" and a "secondary" Random Walk both for the displacements on the x-axis and the y-axis. (This is not documented in Xenakis', Serra's or DiScipio's descriptions of DSS.) To my knowledge, such a combination of Random Walks has no parallel in mathematical theory.

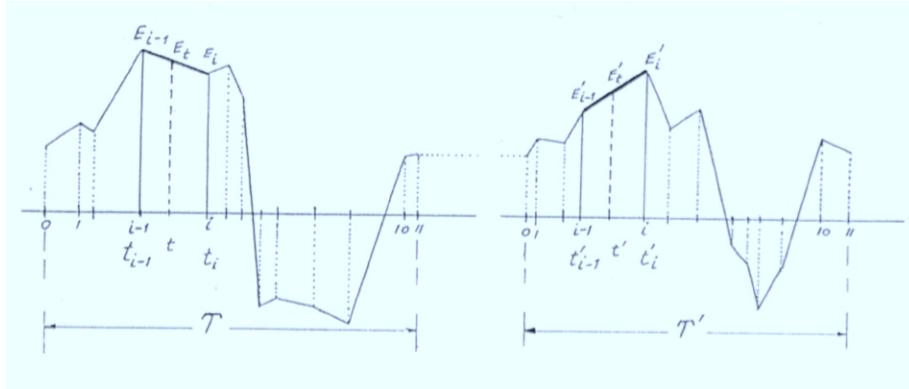


Figure 9.1: The construction of a GENDYN waveform out of segments which are defined by height (amplitude) and width (spacing in time) both of which are subject to stochastic fluctuation (drawing by Xenakis in his book *Formalized Music*).

1. Generate a random number distributed according to a specific stochastic distribution.

2. (“Primary” Random Walk) Add this number to an accumulator (a memory to sum up values).
3. Keep the sum within a finite numeric interval (given by a pair of elastic barriers). Steps 1 on the facing page to 3 are mathematically known as a “Random Walk” in one dimension (up and down a line).
4. (“Secondary” Random Walk) Sum up the result of step 3 in a second accumulator.
5. Keep this sum within a finite numeric interval (given by a second pair of elastic barriers).
6. The result of step 5 yields a time increment (number of sample points between a foregoing breakpoint and a current one in a wave form). Repeat steps 1 on the facing page to 5 using another two accumulators and barrier pairs to compute the corresponding amplitude value of that breakpoint.
7. Repeat the process described in step 6 for a given number of breakpoints, each using a different set of accumulators.
8. Interpolate between the waveform’s breakpoints so generated, i.e. derive the value of all intermediate samples between the breakpoints by a linear formula.
9. Repeat the above process in order to create successive wave forms.

Reading the above description of Xenakis’ algorithm, a software architect will inevitably see before his/her eyes a finite set of units of computation which can be used at different places in the overall procedure. For example, all random walks are equipped by pairs of elastic barriers. There are random walks for time increments (x-axis) as well as for amplitude values (y-axis). The computing unit, the mirror pair, will always function the same way regardless of acting on y-axis or x-axis values. As a consequence, the whole of Dynamic Stochastic Synthesis can be broken down into a combination of elementary computing units.

It should be noted that Xenakis himself did not do this analysis when writing the GENDYN program. Instead, he imagined the functioning of his algorithm as a complicated automaton, being built of as many parts as there are computation steps in his algorithm. To give an example: GENDYN computes up to 16 simultaneous tracks of sound. Each of these tracks is computed by a different piece of code, in spite of the fact that these 16 code pieces do exactly the same thing! As a consequence, the length of Xenakis’ program code is blown up by a factor of 16.

This concludes the description of stochastic sound synthesis proper. There is a second part of the algorithm which controls how these stochastic GENDYN sounds are combined together to form sets of parallel evolving tracks of sound (called “sequences”), and how several of these sequences make up a GENDYN piece. I call this part of the algorithm the “control structure” of GENDYN. It can be seen in figure 9.2 on the next page. See also section 14.6.2 on page 218 for a more detailed description.

One of the first goals of the GENDYN project was to check if the music of *GENDY3* can be reproduced by executing Xenakis’ algorithm. The answer is

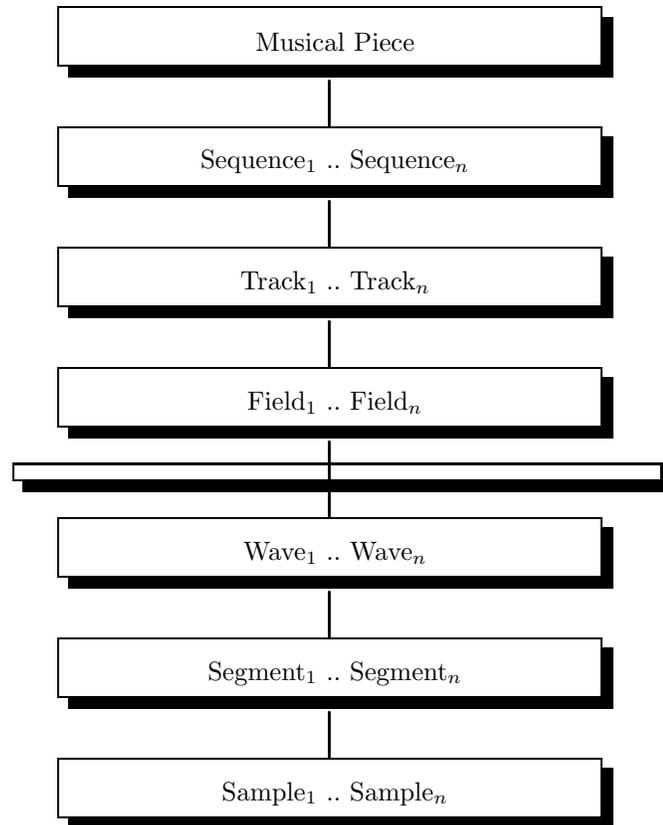


Figure 9.2: The computation of a GENDYN piece breaks down into a hierarchy of nested entities of ever smaller size down to a single sample.

“yes”: all structural information of these 20 minutes of music are inherent in this algorithm. The mechanical process of execution just unfolds something which is already in it. We shall see in the following what this “something” actually is. To conclude, Xenakis did not add anything to the computational result. All of *GENDY3* is in the computation. It is 100% “computable music”.

A second point is that the unfolding of the music by algorithmic action is perfectly reproducible. This means, running the program over and over again would always produce the same music. This is because Xenakis did not have computation depend on volatile information such as system data or execution time, although he easily could have done so.

Therefore, we are confronted with a singular instance of a rigorous algorithmic music which comes into being by simple mechanical action. This means that structurally, the music is fully explained by the generating program. In fact, the best “analysis” of *GENDY3* is the text of the program itself! However, sadly enough, the text of Xenakis’ original program is nothing to be easily understood by a human. (The computer, of course, does not care. It does not have this human problem of wanting to understand.) One motivation for making the New GENDYN Program, therefore, was to create the same music with a human-understandable program.

Of course, being a human, we are not content just having a human-understandable program text. Then we ask why the program is the way it is and why it achieves this sonic result, how sound relates to algorithmic action, and in how far Xenakis had control over all this. But in order to approach these questions we first have to dig deeper into the theory and practice of computation. First we shall shed some light onto the different ways an algorithm may be realized by a computer program.

### 9.4.1 The procedural paradigm

This is the programming approach Xenakis chose to implement his algorithm. It is the classical sequential programming paradigm where control flow is partitioned into functional entities (i.e. procedures) each fulfilling a well-defined task. Procedures are invoked or called either by the main part of the program or by other procedures. This is the theory but practice is often different. Procedural programming environments do not enforce nor particularly promote partitioning the overall problem. Nor do they very well support sophisticated data structures. It is true that all kinds of data structures can be build using pointers, but this is quite cumbersome and error-prone. In any case, the BASIC programming environment used by Xenakis does not even provide pointers so Xenakis had to model his data by using flat arrays (simple memory chunks of data).

In Xenakis’ procedural programming, there are only a few but very large procedures, called with a great number of parameters, and working by side effects on a large set of global variables. Moreover, since the procedural programming paradigm is more centered around the manipulation of data than the modeling and description of the data itself, the units of computation work on rather unstructured flat data like arrays and sets of variables which are not very explicit. Xenakis even went so far as to replicate the code of many procedures, renaming local variables, as if it was his ambition to “unfold” these procedures in order to avoid the overhead of procedure invocation by the BASIC runtime environment. As a matter of fact, he mapped the  $n$  parallel voices he represented as boxes in

his chart ([Xen92b], pp. 298–299) directly onto  $n$  different procedures, whereas the boxes actually represent only the runtime picture of  $n$  instances of a single procedure, namely the one computing a track of sound. As a result, Xenakis had to develop and maintain a computer program blown up to the size of about 1.000 program lines! In order to get a feeling for what that means, take a look at the code listing fragments in [Xen92b], pp. 300–321.

As a better example of the procedural approach to implementing the Xenakis algorithm, let us look at an example which is more structured than Xenakis' own implementation. In the seminal article by Marie-Hélène Serra, a then-member of CEMAMu, the algorithm is presented in terms of iterative procedures. I will quote from this article ([Ser93]), reversing the order from the simple to the complex.

“In the program, we use the random generator in order to get a uniform random number that we call  $z$ [...]. Then we compute  $f_x(z)$  and  $f_y(z)$ , where  $f_x$  and  $f_y$  represent the inverse functions of the desired stochastic distribution. [...] We assume that the numerical sound is made up of a series of  $J$  successive polygonal waveforms. The polygonal waveform [...] is defined with  $I$  vertices of index  $i$ . In the following, we note the coordinates of the vertices as

$$(x_i^j, y_i^j), \text{ where } 0 \leq i \leq I \text{ and } 0 \leq j \leq J$$

The coordinates of each vertex in waveform  $j + 1$  are obtained by adding a stochastic variation to the coordinates of the corresponding vertex [...] in waveform  $j$  [...]

$$\begin{aligned} x_i^{j+1} &= x_i^j + f_x(z) \\ y_i^{j+1} &= y_i^j + f_y(z) \end{aligned}$$

In this (much abbreviated) excerpt from Serra's paper, a few observations can be made. First, in contrast to Xenakis' own description, there is no notion of two random walks working in a series. Second, in Serra's description, the length of the waveform's segments may become negative<sup>1</sup>, for she explicitly states that  $f_x(z)$  as well as  $f_y(z)$  can be both positive or negative and she defines the segment's length as

$$l_{i+1}^j = x_{i+1}^j - x_i^j$$

In fact, the segments must be always positive, for time cannot flow backward. What is true is that a primary random walk may yield negative values when a secondary one assures that in the end the sum be always positive.

---

<sup>1</sup>In Xenakis' programming, negative lengths are prevented by the pragmatic fact that the mirrors acting on the length values force them to be positive.

As an aside, for the creation of *S709* in 1994, Xenakis simplified his algorithm to work with only one random walk for each amplitude and time. This does partly explain the fact that the sonic evolution in this piece is musically less interesting, compared against the structural richness of *GENDY3*. Is it that Xenakis had already simplified his algorithm shortly after *GENDY3* had been produced or that Serra had only simplified her description of Xenakis' *GENDY3* algorithm? Unfortunately, no statement could be obtained from either of these persons.

It is straightforward to proceed from this description of Serra's to a procedural implementation. The time and amplitude coordinates of the wave form breakpoints are stored in two arrays  $x$  and  $y$ . The indexes in the equations are realized as counters in iterative loops addressing the elements of the arrays. In the example below, the index  $i$  is shifted by 1 to simplify the wraparound of the wave form.

```
x[0..I] := 0;
y[0..I] := 0;
for (j from 1 to J) // compute a wave form
  for (i from 1 to I) // compute breakpoints
    x[i] := x[i] +  $f_x(z)$ ;
    y[i] := y[i] +  $f_y(z)$ ;
  end for // end of wave
  y[0] :=y[I]; // wrap around
end for // end of sound
```

We can use destructive assignment here because we do not need the coordinates of past waveforms anymore once the new wave form has been computed and written to disk (or sent to the loudspeaker).<sup>2</sup> Now we interpolate the  $y[i]$  with the help of the  $x[i]$ :

```
for (k from 1 to x[i]) // interpolate segment
  y := y[i-1] + (y[i]-y[i-1])/x[i]*k; // x[i] > 0!
end for
```

This is the core of the synthesis algorithm in its procedural implementation. Of course, there is much more to program: the drawing of the random numbers, their distribution with the help of the inverted stochastic functions, and the realization of the control structure (the architecture of the piece). Note that there are no complex data structures modeling waves, breakpoints, and so on; we have only manipulations of values in linear arrays. We will see that this is different in object oriented programming where the definition of complex data structures is greatly encouraged.

```
for all sequences // the sections of the musical piece
  for all tracks // 1 to 16 simultaneous tracks of sound
```

---

<sup>2</sup>Xenakis, not realizing this, kept past waveforms in a separate array and alternated between these two arrays. This is one of the idiosyncrasies in Xenakis' programming style that had to be emulated by the New GENDYN Program in order to obtain the exact same sonic evolution as in *GENDY3*.

```

for all fields // patches of sound and silence with stochastic duration
    generate and concatenate wave forms
end for // fields
    pause / unpause sound synthesis
end for // tracks
    mix tracks together to obtain a master recording
end for // sequences

```

Pasting the above loops together, we get the structure of a possible procedural implementation of Dynamic Stochastic Synthesis. This implementation is not how Xenakis' own program looks but how it could work in principle. In reality, Xenakis' program operates in a much more complicated way. We will see some aspects of this, as far as they become relevant for details of the music, in the analysis part of this study.

CAC systems implementing the procedural programming paradigm are the first and pioneering Music N systems, where N stands for a roman number (I to V and beyond). They mainly consist of libraries implementing sound synthesis and transformation routines inspired by the analogue electroacoustic music studio (frequency generators, filters, etc.) We have already stated in part I of this study that for the non-standard approach to synthesis, such as Xenakis', such a programming environment is of little help since it already implements a given theory of sound (additive or subtractive synthesis or other), whereas in non-standard approaches, the theory of sound is an integral part of composition, and must be invented and programmed for every musical work to be created.

## 9.4.2 The declarative paradigm

Declarative programming is a generic term for two programming disciplines: functional programming and logic programming. Functional programming is modeled after mathematical functions and their evaluation. Logic programming is based on the theory of resolution of logical clauses. It is possible to conceive of logical programming as a specific form of functional programming, where all functions yield a boolean result (a true/false value). However, logical programming is more powerful because the logical predicates take on the form of relations instead of functions. While a function maps a set of input arguments to only one result value, a relation is more general: there may be more than one result value. Moreover, there is a powerful feature called backtracking which means that the runtime systems tries by itself to find all solutions which fulfill the programmed relations; the programmer (ideally) does not need to worry about algorithmic strategies to tell the computer how to obtain these solutions.

The execution of both a functional and a logic program works through term rewriting: the match of a left hand side of a rule in a start term is substituted by its right hand side, where the variables of the rule are instantiated according to the pattern matching of the left hand side. For functional languages this term rewriting procedure is canonized by the Lambda Calculus ([Chu41]) and called  *$\beta$ -reduction*. For logical languages the term rewriting procedure is *resolution*. For this to work mechanically, the logical clauses have to be in a special form (the so-called *Horn-clauses*). An even more powerful form of rewriting is graph transformation where multi-dimensional structures (graphs) are rewritten

according to graph transformation rules. It is possible to simulate the execution of functional and logical programs through graph rewriting (cf. [Hof94b]).

One of the most ancient programming languages implementing the functional paradigm is LISP (*LIS*t *PR*ocessing *L*anguage). In its more modern dialects, LISP has been used to realize such successful CAC systems as Common Music [Tau91] and Common Lisp Music [Sch], PatchWork ([Lau96]) and OpenMusic ([Ago98]). Modern functional languages have become more elegant and readable but in principle, they do not supersede the computational power of pure lambda calculus. Functional code is relatively easy to write and debug because typically, the result of a functional expression does only depend on its arguments, regardless when and where the function is evaluated. (This is called referential transparency). Because of this, functional code also easily lends itself to parallel execution on multi-processor hardware.

### 9.4.3 The object-oriented paradigm

Object orientation is a relatively modern software engineering approach to procedural programming. It aims at the encapsulation of data structures and procedures (called “methods” in the OO jargon) into objects. Encapsulation means both combining the data and the methods that work on them and separating those from other data and methods which form their own objects. Programming action works by the interoperation of independent objects with clear-cut relationships described by the object’s services much in the sense of free-lance contractors. Objects are created, used and destroyed at runtime following their design-time class description just like their real-world counterparts are created, used and destroyed, e.g. windows and widgets in operating systems, or random walks and barriers in the case of Xenakis’ stochastic synthesis.

Object orientation, as a contrast to classical procedural programming, is very helpful in assuring re-usability, programming maintenance and evolution of large programming systems. But object oriented programming, if it does not follow a certain discipline, can also become quite complex and obfuscated. This is where proven design patterns and customizable components come into play. They promote a programming discipline where different components work together along clear cut interfaces much like objects do on a lower level. This skilled design of an object oriented programming system is also called “programming in the large”. Programming in the large means the design and implementation of components which may in turn consist of many objects.

One such design pattern is the Model-View-Controller (MVC) pattern which is used to link parts of a program by loose coupling. This allows for removing or replacing (i.e. “plugging”) some parts without affecting the rest of the system. As the name suggests, there are three parts (or roles) involved in a MVC pattern. The Model encapsulates the data and provides methods to alter and retrieve them, but hides the internals of data storage. The Controller acts as the instance invoking actions and transformations on the data (e.g. changing a data value by dragging a scrollbar). The Model then calls the View which updates itself by retrieving (if necessary) the new values from the Model. Arbitrarily many different kinds of View components can register with a Model, and arbitrarily many different Control components can be used to manipulate the data. The MVC pattern is invaluable for the loose coupling of a graphical user interface (GUI) with data such as e.g. text documents or databases, or, as in our case,

the New GENDYN Program.

Object-oriented design has been used for the encapsulation of sound-producing or sound-processing entities e.g. in the KYMA system [Sca87] and in several other OO-music environments developed with the help of the venerable OO-language Smalltalk (Siren, Squeak, and others ([Pop91]).

The object oriented approach was the programming approach chosen for the New GENDYN program. It accommodates well the interaction of clearly separated entities such as the sequences, tracks, sounds, mirrors, random walks, and filters described in Xenakis' algorithm. At runtime, the segments, waves, sounds, tracks, fields, sequences, and pieces of Xenakis' algorithm all have their direct counterparts in specific classes which are instantiated into a hierarchy tree of data and procedure calls operating on them. As a result, a GENDYN piece is built by plugging together the action of all these objects in order to cooperate for the goal of sound production (see figure 12.1 on page 189). OO programming with optimization options as they are offered by the C++ programming language, such as in-line function calls, seemed the best compromise between execution efficiency on the one hand and structured program design on the other.

In the New GENDYN Program, the parameter sets of the synthesis are held separate from the objects of the synthesis engine. This is to encapsulate the (fixed) parameter data with the access methods used by the GUI on the one hand, and the (changing) synthesis state variables with the synthesis methods on the other. Parameter classes and synthesis classes are linked together by inheritance: the synthesis objects are extended by the parameter sets. Controllers (the "knobs" of the graphical interface) are implemented in a separate GUI component (actually implemented in a specific GUI programming language, Visual Basic). The GUI also realizes the View (diagrams and animations) of the synthesis by polling the engine in regular intervals for its synthesis data.

There are, of course, alternatives to this way of implementing Stochastic Synthesis. One could have envisaged modeling the parallel tracks as concurrent threads. The problem to solve would then have been to synchronize the sound chunks of all threads on mixdown to the sound output. Indeed, it would have been tempting to parallelize the sound engine, and to assign a sound synthesis object to different processors on a parallel computer or in a workstation network. However, to get the sound output of the individual objects synchronized and assembled together would probably have eaten up all gain of processing speed.

## 9.5 Specification, Implementation, and Execution of a Computer Program

We have seen that an algorithm is not a computer program. An algorithm is an exact description of a mechanical solution process. It need not be executed by an electronic computer. However, an algorithm can well be implemented by a computer program. In order to do so, more work has to be done than just inventing the algorithm. For example, in order to execute the algorithm on a computer, the structure of the data to be processed has to be defined. The algorithm will be still the same, but the actual computer program will look differently, if the data to be processed are represented e.g. as a list of numbers,

or a hierarchical branching structure, or some other data structure. We will see that Dynamic Stochastic Synthesis can be conveniently defined to work on lists of numbers, but lists have not been used in neither Xenakis' own implementation nor in the object-oriented implementation of the New GENDYN Program.

Implementing an algorithm by a computer program has more problems to solve than just data structures. First of all, how is the result of computation to be conveyed to the user? Is it to be a printout, then the program has to drive a printer. Is it text on a screen? Then the program has to care about writing text to the screen. Is it graphics? Then the program must bother with drawing routines offered by the operating system or some higher-level graphic system. Is it music? Then we have the problem to get the sample data into the digital-to-analogue converter (DAC). As for Xenakis, he just wrote each computed sample into a data file on the computer's hard disk. In the New GENDYN program, in addition to writing sound files, samples are written into a ring buffer for real time sound output to the DAC. Buffering, scheduling, multitasking and doing I/O are things necessary to make a real program function in the real world of operating systems and computer hardware. However, all these are not fundamental to the algorithm itself. Therefore, the idea of a list of samples is a convenient abstraction from these complicated and fuzzy details of making a program work on a real computer.

Let us, as a warm-up, specify the algorithm of the Sieve of Eratosthenes as a recursive function in an extended version of the Lambda Calculus:

$$\begin{aligned} \textit{Sieve} & : \text{list}(\text{number}) \rightarrow \text{list}(\text{number}) \\ \textit{Sieve} & = \lambda \textit{prime} :: \textit{nums} . \\ & \quad \textit{prime} :: \textit{Sieve}(\textit{Filter}(\textit{prime}, \textit{nums})) \end{aligned} \quad (9.1)$$

$$\begin{aligned} \text{where } \textit{Filter} & = \lambda n. \lambda m :: ms . \text{if } m \bmod n = 0 \\ & \quad \text{then } \textit{Filter}(n :: ms) \text{ else } m :: \textit{Filter}(n, ms) \end{aligned} \quad (9.2)$$

In this simple function definition we can see a number of notational conventions. First, a function definition is marked by a so called "Lambda Abstraction" in the form  $\lambda x.f(x)$  where  $x$  is the parameter of the function and  $f(x)$  the function's body containing the parameter as a variable. The body can contain the application of other lambda abstractions, as is the case with *Sieve* which uses a function named *Filter* to actually drop all the non-primes off the result list. Another convention used here is pattern matching where the structure of the argument is examined and its components extracted into parameters to be used in the function body. For example, *Sieve* splits the list of natural numbers into its first element *prime* and the rest of the list *nums*, as does *Filter*. Last but not least, there is a convention called "currying" which expresses multiple parameters to a function as nested lambda abstractions, as is the case of *Filter* which takes two parameters, a number  $n$  and a list. In this and the following examples, the constructor  $::$  which prepends an element to a list, is meant to be "lazy", i.e. it does not evaluate its arguments if not "forced" to do so by a calling function through pattern matching. This allows for the easy definition of infinite lists without falling into the "black hole" of infinite recursion, as would be the case with so-called "strict evaluation". For example, the list of natural numbers is "lazily" defined as follows:

$$\begin{aligned} & \text{Nat}(1) \\ \text{where } \text{Nat} &= \lambda \text{number.number} :: \text{Nat}(\text{number} + 1) \end{aligned} \quad (9.3)$$

The function *Nat*, when called, does not deliver more than one (more) element of the list of natural numbers because the recursive call to *Nat* in the argument of the constructor  $::$  is not made until required by the pattern matching mechanism of the calling function. In the same spirit, *Sieve* applied to *Nat*(1) does not deliver any prime number until “forced” to do so by a main function which would typically format the list of prime numbers as a character string on a computer screen.

Let us now describe  $\beta$ -reduction of functional terms in our simple example.

“To  $\beta$ -**reduce** the expression  $(\lambda x.E)A$  we return a modified form of  $E$  in which all free occurrences of  $x$  in  $E$  are replaced by  $A$ .” [FH88], p. 123

Is it important to note that this is (almost) the single rule of program execution in the pure functional calculus, and that it works purely mechanically, i.e. by “blindly” rewriting string symbols. In the above definition, all free occurrences of  $x$  (i.e. which are not bound by inner lambda abstractions) are literally replaced by the expression  $A$  without thinking twice. Therefore, this process can be executed by a machine. A few more (mechanical) rules decide which part of an expression to reduce first if there are more than one possibilities, and when to stop the process.

Now, as an example, let us see how given such an algorithm, a computer (human or machine) is guided through a sequence of purely mechanical steps to produce what is specified by the algorithm. Let us look at the sieve of Eratosthenes in action until finding the first prime number greater than 3.

$$\begin{aligned} & \text{Sieve}(\text{Nat}(2)) \\ &= \text{Sieve}(2 :: \text{Nat}(3)) \text{ by 9.3} \\ &= 2 :: \text{Sieve}(\text{Filter}(2, \text{Nat}(3))) \text{ by 9.1} \\ &= 2 :: \text{Sieve}(\text{Filter}(2, 3 :: \text{Nat}(4))) \text{ by 9.3} \\ &= 2 :: \text{Sieve}(3 :: \text{Filter}(2, \text{Nat}(4))) \text{ by 9.2} \\ &= 2 :: 3 :: \text{Sieve}(\text{Filter}(3, \text{Filter}(2, 4 :: \text{Nat}(5)))) \text{ by 9.1} \\ &= 2 :: 3 :: \text{Sieve}(\text{Filter}(3, \text{Filter}(2, 5 :: \text{Nat}(6)))) \text{ by 9.2} \\ &= 2 :: 3 :: \text{Sieve}(\text{Filter}(3, 5 :: \text{Filter}(2, \text{Nat}(6)))) \text{ by 9.2} \\ &= 2 :: 3 :: \text{Sieve}(5 :: \text{Filter}(3, \text{Filter}(2, \text{Nat}(6)))) \text{ by 9.2} \\ &= 2 :: 3 :: 5 :: \text{Sieve}(\text{Filter}(5, \text{Filter}(3, \text{Filter}(2, 6 :: \text{Nat}(7)))))) \text{ by 9.1} \end{aligned}$$

Looking at the pattern of function evaluation in the above example, we clearly realize how the algorithm works: every number which is added to the right of the resulting list of numbers must be a prime because it has passed a row of filters assuring that no multiple of any number may ever come through.

Therefore we “see”, by an act of understanding, that Eratosthenes’ algorithm is correctly defined. It is important to note that a computer does not check an algorithm for correctness when executing it. The computer cannot “see” that the algorithm is correct. A machine cannot generally decide the correctness of an algorithm: this is one of the limitations of machine action and can be reduced to the halting problem to be discussed below (cf. [GL88]).

Another observation we can make by looking at the example is the fact that we have to apply the Sieve function to an ordered list of natural numbers in order to produce a list of prime numbers. If we applied the Sieve function to an unordered list of natural numbers, the result would not be prime numbers at all. Therefore, we see, how important it is to correctly specify an algorithm. If the input specification (ordered list of natural numbers) is not met, the result will not meet the output specification (prime numbers) either.

## 9.6 Functional Specification of Dynamic Stochastic Synthesis

Before discussing universal computation, let us see, as an example for a specific computation, how Dynamic Stochastic Synthesis can be implemented using the declarative paradigm of functional programming. Like any other program in any other programming paradigm, it uses the features of universal computation, namely the fundamental concepts of recursion, branching, and function composition.

Recursion means that a function is defined in terms of itself, i.e. during evaluation of the function, the function calls itself repeatedly to approximate the desired end result of computation. In order for recursion to stop, branching is required to break off the process of recursion. Last but not least, functions have to be composed to build up an entire program. Let us see how we can do Stochastic Synthesis with that.

The following functions are defined on lists of numbers. The original list is a list of random numbers and the resulting list is the music. Since random numbers can be algorithmically computed, we even do not need to feed a list of numbers into the algorithm. The algorithm, therefore, creates the music all by itself, “out of nothing”.

$$SOUND : \rightarrow \text{list}(\text{number}) \tag{9.4}$$

The idea is to first generate a list of numbers that is random, i.e. pure noise and then to transform it into music. Let us begin with the definition of that noise.

There are many sophisticated algorithms to generate such numbers, but the one given has been used by Xenakis in GENDYN for the generation of the amplitude values. (For the random numbers controlling the time increments, he used the default random generator built into his BASIC programming environment which is more sophisticated but which we do not know, since the BASIC runtime system is not open source.)

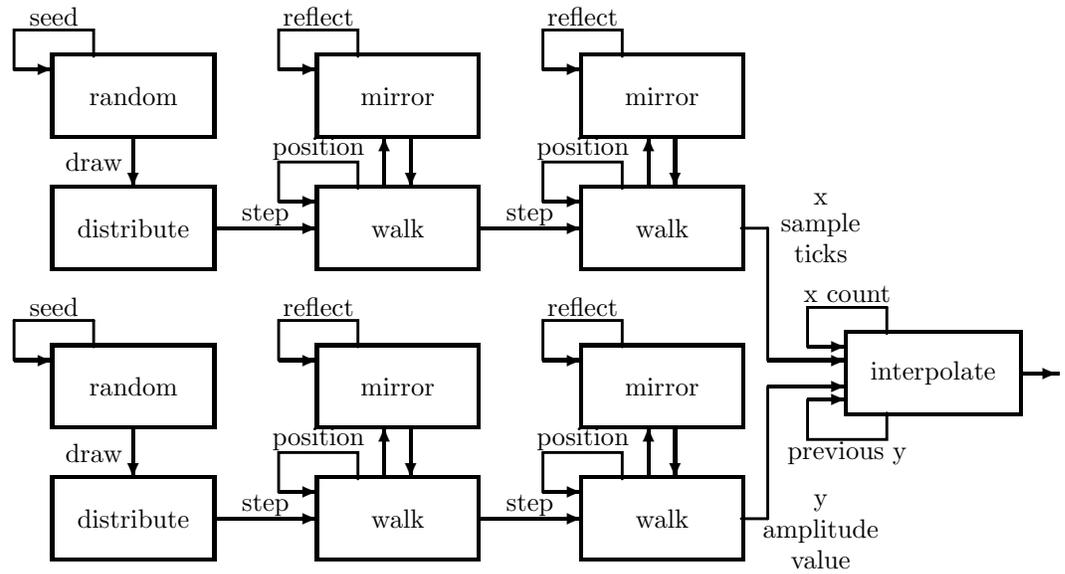


Figure 9.3: The DSS algorithm as a circuit of recursive functions

A simple random generator typically works with a numerical “fold-and-stretch” algorithm: it starts with an initial value (the “seed”) and then recursively blows it up (multiplies it with a great factor) and folds it back into a small number interval delimited by a modulo  $M$ . This procedure shuffles the numbers much in the same way as the baker kneads his dough: he rolls it out to a long shape and then folds it back onto itself, rolls it out again, folds it back, and so forth. The result is a well mixed dough, or algorithmically speaking, fairly good pseudo-randomness.

Such a random function generates a list of numbers whose elements are randomly distributed over the number interval of  $M$  (typically 1). It is required that the values  $a$  and  $c$  be chosen sufficiently large to obtain good pseudo-randomness. Note that the list of random numbers is infinite: there is no bound defined in the function. In the lazy evaluation paradigm, the random function delivers one and only one random number at each time it is called. The random function takes a seed as an input and feeds the result of computation (the next random number) back into itself.

$$\begin{aligned}
 \text{random} & : \text{number} \rightarrow \text{list}(\text{number}) \\
 \text{random} & = \lambda \text{seed} . \text{seed} :: \text{random}((\text{seed} \ a + c) \bmod M) \quad (9.5)
 \end{aligned}$$

So now, we have a list of random numbers between 0 and 1 with equal probability. Next, we transform these probabilities into the step values that we need to feed into the random walks. To do so, we pass the uniformly distributed probabilities through (inverted) distribution functions that will return step values for our random walks, distributed along a given range in a characteristic way,

according to the specific distributions function chosen. Recall that we want to simulate a stochastic process, and in order to do so, we use the distribution function in the reverse sense.

In other words, the inverted distribution function “distorts” the uniformly distributed random numbers gained from the random function in a specific way characteristic to the mathematical distribution function used. The set of inverse distribution functions used by Xenakis is given in appendix A on page 315.

$$\begin{aligned} \textit{distribute} & : \text{distribution function} \times \text{list(number)} \rightarrow \text{list(number)} \\ \textit{distribute} & = \lambda f.\lambda y :: ys . f^{-1}(y) :: \textit{distribute}(ys) \end{aligned} \quad (9.6)$$

The function specifying the random walk is nothing else but a summation of all values passed to it (i.e. like an accumulator in computer hardware). For each accumulation step, the sum is passed through another function called “mirror” which keeps the value within a predefined interval (this is the concept of elastic barriers in Random Walk theory). In other words, the random walk transforms a list of numbers into a list where each element of the list is the sum of all elements to the left of it. Note that the random walk starts at 0 (this is not defined in the algorithm but follows from Xenakis’ implementation).

The random walks in Xenakis’ program also act as agents for chaos by folding and stretching. As a matter of fact, at the same time they amplify the incoming step values by summing them up and they fold them back onto a limited interval by the reflecting effect of the barriers. Arithmetically, both the random number generator and the barriers are implemented using the same operation, the modulo division.

$$\begin{aligned} \textit{walk} & : \text{list(number)} \rightarrow \text{list(number)} \\ \textit{walk} & = \lambda \langle \rangle . \langle 0 \rangle \\ & \quad \lambda xs :: x . xs' :: x' :: \textit{mirror}(x' + x) \\ & \quad \text{where } xs' :: x' = \textit{walk}(xs) \end{aligned} \quad (9.7)$$

The mirror function limits its argument to a fixed interval by “reflecting” (i.e. subtracting) it back inside the interval if it exceeds either the interval’s upper or lower bound. This process can be repetitive in case the excess is greater than the size of the interval.

$$\begin{aligned} \textit{mirror} & : \text{number} \rightarrow \text{number} \\ \textit{mirror} & = \lambda x . \textit{mirror}(\textit{lower} + (\textit{lower} - x)) \text{ if } x < \textit{lower} \\ & = \lambda x . \textit{mirror}(\textit{upper} - (x - \textit{upper})) \text{ if } x > \textit{upper} \\ & = x \text{ otherwise} \end{aligned} \quad (9.8)$$

Finally, the actual sound samples are gained by linear interpolation between the amplitude values produced by  $n$  random walks,  $n$  being the number of the breakpoints of the wave form. This means that between the  $n$  random walk values,  $x - 1$  more values are inserted, the number  $x$  being taken from a second set of  $n$  random walks.

In other words, the interpolating function scans amplitude values and adds a number of intermediary values in between them that go stepwise from the previous amplitude value to the next.

Let us first define the multiplexing of the  $n$  amplitude and time random walks into a wave form. The idea is to collect, for both amplitudes and time values, the values of  $n$  lists into a single list (which lists the elements “cycled through” in order).

$$\begin{aligned}
 \text{wave} & : \text{list}(\text{list}(\text{number})) \times \text{list}(\text{list}(\text{number})) \\
 & \rightarrow \text{list}(\text{number}) \times \text{list}(\text{number}) \\
 \text{wave} & = \lambda \langle x \rangle . \lambda \langle y \rangle . (\langle x \rangle, \langle y \rangle) \\
 & \lambda \langle x_1 :: xs_1, \dots, x_n :: xs_n \rangle . \lambda \langle y_1 :: ys_1, \dots, y_n :: ys_n \rangle . \\
 & (\langle x_1, \dots, x_n \rangle :: xs', \langle y_1, \dots, y_n \rangle :: ys') \\
 & \text{where } (xs', ys') = \text{wave}(\langle xs_1, \dots, xs_n \rangle, \langle ys_1, \dots, ys_n \rangle) \quad (9.9)
 \end{aligned}$$

Now we are ready to take these two lists of amplitudes and timing values and produce a contiguous sound out of them. Note that interpolation starts with zero amplitude (this is not defined in the algorithm but follows from Xenakis' programming).

$$\begin{aligned}
 \text{interpolate} & : \text{list}(\text{number}) \times \text{list}(\text{number}) \rightarrow \text{list}(\text{number}) \\
 \text{interpolate} & = \lambda \langle x \rangle . \lambda \langle y \rangle . \langle 0 \rangle \\
 & \lambda x :: xs . \lambda y :: ys . y :: ys' :: y' :: ys'' \\
 & \text{where } y' :: ys'' = \text{interpolate}(xs, ys) \\
 & \text{and } ys' = \langle y' + \frac{y - y'}{x}(x - 1), \dots, y' + \frac{y - y'}{x} \rangle \quad (9.10)
 \end{aligned}$$

The GENDYN sound synthesis algorithm is just a combination of the functions described above:

$$\begin{aligned}
 \text{SOUND} & : \rightarrow \text{list}(\text{number}) \\
 \text{SOUND} & = \text{interpolate}(\text{wave}(x :: xs, y :: ys)) \\
 & \text{where } x = \text{walk} \circ \text{walk} \circ \text{distribute} \circ \text{random} \\
 & \text{and } y = \text{walk} \circ \text{walk} \circ \text{distribute} \circ \text{random}
 \end{aligned}$$

where the operator  $\circ$  denotes function composition, defined as

$$f \circ gx = f(g(x)) \quad (9.11)$$

As follows from the definition, the *SOUND* algorithm has no termination. It generates an unlimited stream of sound. Therefore, in order to create a piece of music which has a beginning and an end, this algorithm must be controlled by another one which switches it on and off. In addition, GENDYN combines multiple sounds in a sort of multi-part counterpoint by switching on and off a

set of sounds “in parallel”. As a matter of fact, this control structure is nothing else than yet another random walk.

$$\begin{aligned} \text{STRUCT} & : \rightarrow \text{list}(\text{number}) \\ \text{STRUCT} & = \text{walk} \circ \text{distribute} \circ \text{random} \end{aligned} \quad (9.12)$$

where the inverted stochastic distribution function is the exponential function, which delivers only positive values. It can therefore be used to compute the successive positive steps in time (i.e. positive duration values measured in sample ticks), and the random walk sums them up to a chain of durations.

Parallel to the random walk computing durations, a list of 0's and 1's is built.

$$\begin{aligned} \text{TOSS} & : \rightarrow \text{list}(\text{number}) \\ \text{TOSS} & = \text{distribute} \circ \text{random} \end{aligned} \quad (9.13)$$

The distribution function here is the discrete uniform distribution with only two values, 0 and 1, like in a coin tossing game.

The control algorithm pairs the duration structure and the structure of 0's and 1's and when finding a 1 it produces sound for the respective duration, otherwise it pauses synthesis during that time (i.e. it produces silence). A track consists of a finite number of such sounding or silent durations, called “fields” by Xenakis.

$$\begin{aligned} \text{TRACK} & : \text{list}(\text{number}) \times \text{list}(\text{number}) \rightarrow \text{list}(\text{number}) \\ \text{TRACK}(\text{fields}) & = \text{TRACK}'(\text{fields}, 0, \text{STRUCT}, \text{TOSS}) \\ \text{TRACK}' & = \lambda 0. \lambda \text{ticks}. \lambda \text{as}. \lambda \text{bs}. \langle \rangle \\ & \quad \lambda \text{fields}. \lambda \text{ticks}. \lambda a :: \text{as}. \lambda b :: \text{bs} . \text{if } \text{ticks} \geq a \\ & \quad \text{then } \text{sample} :: \text{TRACK}'(\text{fields} - 1, \text{ticks} + 1, \text{as}, \text{bs}) \\ & \quad \text{else } \text{sample} :: \text{TRACK}'(\text{fields}, \text{ticks} + 1, a :: \text{as}, b :: \text{bs}) \\ \text{where } \text{sample} & = \text{if } b = 1 \\ & \quad \text{then } \text{SOUND} \text{ else } 0 \end{aligned} \quad (9.14)$$

A multitude of  $n$  such tracks working in sync yields a GENDYN sequence.

$$\begin{aligned} \text{SEQ} & : \rightarrow \text{list}(\text{list}(\text{number})) \\ \text{SEQ} & = \langle \text{TRACK}_1(\text{fields}_1), \dots, \text{TRACK}_n(\text{fields}_n) \rangle \end{aligned} \quad (9.15)$$

A piece simply consists of a succession of sequences. A succession of  $m$  sequences yields a GENDYN piece.

$$\begin{aligned} \text{PIECE} & : \rightarrow \text{list}(\text{list}(\text{list}(\text{number}))) \\ \text{PIECE} & = \langle \text{SEQ}_1, \dots, \text{SEQ}_m \rangle \end{aligned} \quad (9.16)$$

Finally, the GENDYN algorithm converts the piece structure into a stream of samples.

$$\begin{aligned}
 \text{GENDYN} & : \text{list}(\text{list}(\text{list}(\text{number}))) \rightarrow \text{list}(\text{number}) \\
 \text{GENDYN} & = \lambda \langle \rangle . \langle \rangle \\
 & \quad \lambda p :: ps . \text{GENDYN}'(p) :: \text{GENDYN}(ps) \quad (9.17) \\
 \text{GENDYN}' & : \text{list}(\text{list}(\text{number})) \rightarrow \text{list}(\text{number}) \\
 \text{GENDYN}' & = \lambda \langle s_1 :: ss_1, \dots, s_n :: ss_n \rangle . \text{MIX}(\langle s_1, \dots, s_n \rangle) :: \\
 & \quad \text{GENDYN}'(\langle ss_1, \dots, ss_n \rangle) \\
 & \quad \text{where } \text{MIX}(ss) = /_0(+, ss) \quad (9.18)
 \end{aligned}$$

where  $/_0(+, ss)$  denotes the reduction of list  $ss$  by recursively summing up all of its elements.

This concludes the declarative specification of Xenakis' GENDYN algorithm. In a functional programming language, the above equations would be coded in the specific syntax of that language, yielding a computer program of only 50 program lines.

Now, after having seen how the Xenakis algorithm can be implemented and executed on a computer, let us take a step back and dwell a bit on the notion of computation.

## Chapter 10

# Computation and Computability

Human sciences and arts have only lately come in contact with formal reasoning, one important example being computer linguistics. Xenakis was possibly the first in Europe to think of applying computers to music composition, at a time when the technological equipment of the electroacoustic studios did not involve much more than tape machines and oscillators. It has been shown elsewhere [Hof94a], [Eic94] how Xenakis' extended manifesto "Formalized Music" aims at having musical thought and creativity take profit of formal reasoning the same way mathematics did since the turn of the 19th century. In mathematics, an important by-product of these endeavors was the computer. In the case of Xenakis, the by-product of "Formalized Music" was the design of computer programs doing automated composition, like ST (1958-62) and GENDYN (1991).

In order to automate a process, it must be specified in a rigorous way, suitable for mechanical execution by a machine. Formalization and automation are two sides of the same coin. Once a process is formalized it can be handed over to a machine in order to be automated, by mechanically applying the formal rules of execution. On a computer, these rules are specified with the help of a computer program<sup>1</sup>. A computer is nothing else than a multi-purpose automaton: when fed with one program, the computer behaves like the automaton described by the one program; when fed with another program, the computer behaves like the other automaton. This feature is called universality: a universal computer (i.e. all of our programmable computers) is able to imitate any other automaton. Here we consider automata that do not interact with their environment. Xenakis was mainly interested in constructing automata that were autonomous.

The computer, a universally programmable automaton, had been theoretically postulated around 1936 by Allen Turing, Alonzo Church and Emile Post [Gan88], and physically came into existence around 1948 by unified international effort [Roj95]. Computer hardware has, since those times, undergone a significant evolution, an evolution which today allows algorithmic composition to work "faster than sound". In contrast, the very notion of algorithmic com-

---

<sup>1</sup>Here, we do not differentiate program and runtime system; both are considered here as the software controlling the computer.

putability has not changed at all - everything computers can do may just as well be done by an appropriately programmed Universal Turing Machine (UTM, a mathematical thought model of a computer, see below).

It will be shown that unlike many of his colleagues, Xenakis was perfectly aware of both the power and the limitation of universal computation as applied to algorithmic composition. Even more so: the very essence of what algorithmic action means is part of the aesthetic substance of his algorithmic compositions. Thus, in contrast to a wide-spread ambition to “humanize” the machine and to render its output so as it is difficult to tell that it comes from a machine, Xenakis makes algorithmic complexity (in the sense of e.g. [Cha75]) of machine action the very subject of his artistic discourse.

## 10.1 Formal Systems

The very idea of computation is intimately linked to the problem of the logical foundation of mathematics and sciences, as well as to the notion of the axiomatic method and formal proof. Preconceived by the ancient Greeks, it was formulated by the mathematician David Hilbert as a challenge to the mathematical community around 1900. The mathematician Gregory J. Chaitin states the “Hilbert program” in the following way:

“Hilbert’s idea is the culmination of two thousand years of mathematical tradition going back to Euclid’s axiomatic treatment of geometry, to Leibnitz’s dream of a symbolic logic, and Russel and Whitehead’s monumental *Principia Mathematica*. Hilbert wanted to formulate a formal axiomatic system which would encompass all of mathematics. [...] A formal axiomatic system is like a programming language. There’s an alphabet and rules of grammar, in other words, a formal syntax. [...] Hilbert emphasized that the whole point of a formal axiomatic system is that there must be a mechanical procedure for checking whether a purported proof is correct or not [...] In other words, there is an algorithm, a mechanical procedure, for generating one by one every theorem that can be demonstrated in a formal axiomatic system. [...] That’s the notion that mathematical truth should be objective so that everyone can agree whether a proof follows the rules or not.” [Cha97b], p. 14-16

It was the ambitious program of Hilbert’s to create a kind of meta-calculus, named “metamathematics”, through a rigorous formalization of the mathematical language. In view of a fundamental crisis prevailing within the mathematical community around 1900, it was to put mathematics on solid axiomatic grounds and to raise mathematical proof technique beyond any discussion. In our present understanding, Hilbert’s aim was to automate mathematical reasoning.

The decidability problem is the question if, for every formal system  $F$ , there exists a formal decision procedure  $D$  that would decide if a given well-formed string in  $F$  was a theorem in  $F$ . If this were the case, formal systems could be constructed automatically by a machine. The machine would just produce strings in  $F$ , in a systematic manner, beginning with very simple ones and then going to more and more complicated ones, and run the formal decision procedure on each of them, keeping those as theorems which pass the test. Together

with an appropriate well-defined interpretation of the symbols of the formal system as mathematical entities, one would have a means to automatically generate true statements. This would mean to mechanically discover mathematical truth. Although for some specific formal systems, specific decision procedures do exist, there is no general procedure which would solve the decidability problem for every formal system. Therefore, in general, the validity of theorems is not formally decidable. Princeton mathematician Kurt Gödel was the first to prove this.

Gödel showed that Hilbert's dream of automating mathematics was in principle unachievable due to the very nature of formal systems themselves. However, research about the issue has proven tremendously fruitful, and a profound theory of computation has been developed by just the same people who proved its limitations. In the end, Hilbert's program has fostered so much research about formal languages and computing that it may be said to have inaugurated the science of computing as such.

Chaitin makes it clear that he does not find Gödel's proof depressing, as did many scholars of Gödel's time. He is actually quite happy about Gödel's proof. Gödel's findings are for him an urge to do empirical mathematics, beyond the limitations of formal reasoning. The same might hold for algorithmic composition: there must be a residue in creation which is in principle not reachable by machines but only accessible to humans, even when using machines for creation.

### 10.1.1 Syntax, Semantics, Interpretation

Graphically speaking, a formal system is a means to express infinitely many explicit facts of the real world in an implicit and finite manner through formalization. Formalization is finite because it has only finitely many axioms and rules of inference. Formalization is implicit because not all true statements are explicitly listed within the formal system but must be derived from its finite set of axioms through the iterative application of the system's finite number of rules of inference.

First of all, the alphabet and the vocabulary of that formal language must be defined. This is called the "rules of formation" ([FBHL84]). Then the set of axioms and the rules of inference must be defined. These are the "rules of transformation".

The rules of formation build on three sets: the set of symbols  $\Sigma$  (the "alphabet"), the set of terms  $T$  and the set of formulae  $F$ . Symbols are either variables or constants or auxiliary symbols (like brackets and commata). They are the "atoms" of the formal system. With the help of the symbols, expressions (or "words")  $\Sigma^*$  can be formed by combining them into arbitrary strings of symbols. A subset of all possible expressions is the set of terms whose formation must obey effective rules such as e.g.:

1. Each variable is a term.
2. If  $\alpha$  is a term, then  $S(\alpha)$  is a term.
3. If  $\alpha$  and  $\beta$  are terms, then  $T(\alpha, \beta)$  is a term.
4. These are all terms.

There is no sense trying to understand what these definitions “mean”. They are only there in order to construct valid sentences in the formal language or, for a parser, to accept a given sentence as valid (to be parsed with success). As we have said, formal systems are made to be meaningless! Nobody can prevent us, however, to immediately see syntactic analogies to things we understand, such as counting and adding numbers like “ $+(1, 1)$ ” or establishing logic terms like “ $\wedge(a, \neg(b))$ ”. We also recognize in this syntax the possibility to build complex terms representing all kinds of data structures like lists, trees, graphs, etc.

The third set of formation is the set of formulae. Again, these form a proper subset of  $\Sigma^*$ , described by effective rules such as e.g.:

1. If  $\alpha$  and  $\beta$  are terms, then  $\alpha = \beta$  is a formula.
2. If  $\phi$  is a formula, then  $\neg(\phi)$  is a formula.
3. If  $\phi$  and  $\psi$  are formulae, then  $\phi \rightarrow \psi$  is a formula.
4. These are all formulas.

Anyone with background in logic programming will recognize here the syntactic schema of logical expressions. Now let us proceed to the “rules of transformation” by giving an example for a set of axioms and a set of rules of inference.

The set of axioms is a subset of the set of formulae. To be sure, all ambition is concentrated on the aim to keep the set of axioms as small as possible. Axioms for propositional calculus could be e.g.:

1.  $\neg\neg\chi \rightarrow \chi$  (double negation)
2.  $(\psi \wedge \chi) \rightarrow \psi$ ,  $(\psi \wedge \chi) \rightarrow \chi$
3.  $\psi \rightarrow (\psi \vee \chi)$ ,  $\chi \rightarrow (\psi \vee \chi)$
4.  $(\psi \rightarrow \chi) \rightarrow ((\chi \rightarrow \psi) \rightarrow (\psi \leftrightarrow \chi))$

With the help of the formal system so defined, we can deduce formal theorems by repeatedly applying the rules of inference to the set of axioms. A chain of inferences from axioms to a theorem is called a proof. A computer can produce such proofs in a systematic manner by blind manipulation of symbols according to the above rules. Hence, this notion of formal proof is an effective one. We can see by the proof chain, once it is computed, that the end product, the theorem, follows from the axioms of the formal system.

Finally, the set of rules of inference may contain such rules as

1. From the set of the two formulae  $\phi$  and  $\phi \rightarrow \psi$ ,  $\psi$  is immediately derivable (rule of detachment or modus ponens).

Now we have a formal system which is purely syntax. How do we relate its theorems to true statements in the real world? We have to provide the formal system with an interpretation to reflect its structure in the real world, i.e. to obtain an interpreted calculus. The interpretation has the function of providing each sentence of the formalized theory with a meaning such that it turns into something that is either true or false.

The interpretation assigns logic signification to the symbols, fixes a universe of discourse  $U$  which will serve to instantiate the constants and variables of the formal theory, assigns mathematical relations to the predicates of the calculus and mathematical functions to its operations. If the axioms of the formal system become true under the interpretation, and its rules of inference are truth-preserving, then the interpretation is called sound, and a model of the formal theory exists.

In other words, an interpretation of a formal system is the assignment or mapping (i.e. well defined correspondence) from the syntactic structure of the formal system to the semantic structure of the problem domain, i.e. the pairing of theorems and truths. The idea is that provable theorems can be interpreted as truths resp. that truths are amenable to formal proving by mechanical manipulation of syntactic structures.

### 10.1.2 Consistency and Completeness

We can ask for the properties of the aforementioned interpretation mapping. It is desirable that it be total and surjective. A mapping is total if it is defined on all elements. For the interpretation mapping this means that all theorems are mapped to true statements, i.e. that all theorems are true. We then call the formal system consistent. If the mapping is surjective, which means that all elements in the problem domain are reached by the mapping, then for every true statement there is a theorem in the formal system. We then call the formal system complete.

### 10.1.3 Gödel's Proof

In a seminal article of 1931 [Göd31], the mathematician Kurt Gödel showed that the Hilbert program was in principle not feasible. He proved that any formal system capable of modeling positive integer arithmetics is inherently incomplete. There are mathematical statements for which no general procedure can decide whether they are true or false. Hence Gödel's theorem is called "incompleteness theorem": formal systems which are capable of formalizing arithmetics and which are consistent, are necessarily incomplete. This means that there can be no general mechanical procedure to find all the theorems even in basic mathematics. In other word, mathematicians cannot be replaced by computers, and the dispute between the adherents of various competing mathematical schools cannot be "objectively" settled. Formal systems are not capable of formalizing the whole of mathematical truth. Provable facts form a proper subset of mathematical truth.

But Gödel's theorem has more implications beyond mathematics itself. It has repeatedly been cited to show that computers, which run mechanical procedures, i.e. programs, cannot step out of their formal framework and consequently run into dead ends without a chance to escape, contrary to humans who are able to see beyond the limits of an established formal framework [Pen89]. This is a belief shared by Xenakis himself, who was, unlike many computer composers, perfectly aware of the limitations of computation (see e.g. [Xen79], [Xen96]). Gödel's proof is a strong argument against the tendency to project anthropomorphic qualities onto computers like e.g. the capacity of intelligence, understanding and

creativity. Therefore, the findings of Gödel are a foundation for any discussion on the specificities of machine art.

Gödel's theorem reads: "For every consistent formalization of arithmetic, there exist arithmetic truths unprovable within that formal system." [Cas94], p. 139.

The punch line of Gödel's proof goes like this (cf. [Dav93], p. 102-103). Suppose a systematic procedure exists proving all true statements in the formal system. Call it P. Then consider the statement S: "P cannot prove S to be true". Suppose P were to arrive at the conclusion that S is true. This means that S will have been falsified, because P will just have done it. But if S is falsified, S cannot be true. Thus, if P answers "true" to S, P will have arrived at a false conclusion, contradicting its infallibility. Hence P cannot answer "true". We have therefore arrived at the conclusion that S is, in fact, true. But in arriving at this conclusion we have demonstrated that P cannot arrive at this conclusion. This means we as humans know something to be true that P can't demonstrate to be true. And this is exactly what Gödel's theorem says: there will always exist certain true statements that cannot be proved to be true.

To summarize, formal systems capable of expressing self-referential statements of the kind "This statement is unprovable" are necessarily incomplete. For this statement is only provable if it is false, which means there would be an inconsistency in the system. Therefore it is an unprovable truth and hence the system is incomplete. The difficult task in Gödel's proof was to formulate the paradox as an arithmetic statement, i.e. as a statement within a formal system capable of formalizing arithmetics.

"It is important to realize [...] that the limitation exposed by Gödel's theorem concerns the axiomatic method of logical proof itself, and is not a property of the statements one is trying to prove (or disprove). One can always make the truth of a statement that is unprovable in a given system itself an axiom in some extended system. But then there will be other statements unprovable in this enlarged system, and so on." [Dav93], p. 103

It must be said that, in spite of Gödel's proof which showed the principal limitation of formal systems, the merits of the axiomatic method, of formal systems and algorithms have since then become more than evident in physics, mathematics, and sciences in general. Mathematicians have learnt to live with incomplete formal systems and some even judge Gödel's findings as purely theoretical. However, Gödel's theorem is an important example showing that human thought and creativity cannot be entirely emulated by machines because humans are capable of ways of reasoning which will bring any mechanical emulation to a deadlock.

This weakness of formal systems goes in pair with a terrible strength: since they solely rely on mechanical manipulation of "meaningless" symbols, they offer themselves to computerization. In fact, a computer program is nothing else than the execution of a formal mathematical proof. An input is transformed into an output through a succession of finite steps, each performing a well defined operation, just like the steps in a mathematical proof. For example, a loop in a program, or, equivalently, a recursive definition can be conceived of as the execution of the mathematical proof by induction.

- There is an anchor of induction before the loop is executed.
- There is the inference from step  $n$  to step  $n + 1$  when the loop is executed.
- Finally, the proof is complete when the loop is exited.

The above is, of course, only true if the loop is correctly implemented. The discipline of program verification aims at ensuring the formal correctness of computer programs. Other disciplines help to develop the code of a program directly from the specification of what the program is supposed to achieve in a formal way. As a matter of fact, there is an equivalence between the output of a computer and the theorems of a formal system (cf. [Cas94]).

## 10.2 Models of Computation: The Turing Machine

The mathematician Allan Turing, answering Hilbert's challenge, devised a formal system capable of automating arithmetics in 1936, his famous Turing machine [Tur36]. He used it in order to address Hilbert's question about decidability in formal systems.

“Gödel's theorem was a devastating setback for the formalists program, but the idea of a purely mechanical procedure for investigating mathematical statements was not completely abandoned. Perhaps undecidable propositions are just rare quirks that can be sifted out of logic and mathematics? If a way could be found to sort statements into decidable and undecidable, it might then still be feasible to determine for the former whether they were true or false. But could a systematic procedure be found to infallibly recognizing undecidable propositions and discarding them? [...] The problem was [...] addressed [...] by Alan Turing, while he was still a young student in Cambridge.” [Dav93], p. 103

A Turing machine is an extension of a deterministic automaton (cf. [BDG88]). Formally, a deterministic automaton is a quintuple

$$A = \langle Q, \Sigma, \delta, q_0, F \rangle$$

where  $Q$  is the finite non-empty set of internal states;  $\Sigma$  is a finite alphabet;  $q_0 \in Q$  is the initial state;  $F$  is the set of accepting final states, and  $\delta : Q \times \Sigma \rightarrow Q$  is a partial transition function of  $A$ .

The automaton starts in the initial state  $q_0$  and reads a symbol, say, from an input tape. It then transforms, based on its current state and the symbol read, a state transition to another state, as defined by the state transition function  $\delta$ . This process is repeated until the automaton reaches a final accepting state  $f \in F$ . Such an automaton is able to accept valid words  $S \subset \Sigma^*$  built out of the symbols of the alphabet  $\Sigma$ . In other words, the automaton is able to solve the parsing problem for the decision procedure in our formal system. More specifically, a deterministic automaton is able to parse regular expressions, i.e.

is able to recognize tokens in a computer program text such as string literals, number literals, identifiers or keywords.

A possible extension of a deterministic automaton is a (multi-tape) Turing machine

$$M = \langle Q, \Sigma, \delta, q_0, F \rangle$$

where  $\delta : Q \times \Sigma^k \rightarrow \Sigma^{k-1} \times Q \times \{R, N, L\}^k$  is a partial function called the transition function of  $M$  (R: “move right”, L: “move left”, N: “no move”)

In addition to the deterministic automaton, a Turing machine is able to write symbols down on its tapes (one tape is read-only, the others are read-write). Furthermore, while reading and writing, the read/write head may move one step backward and forward (or remain motionless) on the tapes. We clearly see that the concept of tapes adds a kind of memory to the Turing machine where it can store intermediate results or from where it can read a program. Therefore, the Turing machine is able to act as a universal computer: it can read the program of any automaton, including that describing another Turing machine, on its input tape and then simulate this other machine. We shall soon see that this very concept of universality leads to its own limitation, namely when a Turing machine reads itself as input.

### 10.3 Undecidability and The Halting Problem

Turing first proved that his machine was capable of universal computation, i.e. that it could be “programmed” to solve any problem that can be solved by a logical machine. Then he showed that his machine could not solve all conceivable problems. In the case of his Turing machine, Gödel’s incompleteness translates to the Halting Problem: there is no general procedure which would decide (or foretell) if a given Turing machine will ever halt on a given input.

The punch line of Turing’s proof goes like this. Suppose that the Universal Turing Machine could solve the halting problem. What would then happen if the universal machine was to attempt to simulate itself? Consider the case where the problem is not decidable. The universal Turing machine, given as an argument to itself, would not halt. The universal Turing machine simulating the other one, however, would then have to halt with the answer “no”. This is a logical contradiction: the same Turing machine cannot halt and not halt on the same input, so the proposition cannot be true, i.e. there is no general procedure to decide if a given computation will successfully terminate.

In a word, Gödel and Turing had used a “flaw” in first-order logic that enables one to formulate paradox statements like “This statement is not true”. If it is, it is not, and if it is not, it is. The Gödel version of it reads: “This theorem is unprovable.” And the Turing version of it reads “This Turing machine does not halt.” The point is that we as humans understand the trick while machines don’t.

“Turing had exposed a variant of Gödel’s theorem about undecidable propositions. In this case the undecidability concerns undecidable propositions itself: there is no systematic way of deciding whether a given proposition is decidable or undecidable. Here, then, was a clear

counterexample to Hilbert’s conjecture about the mechanization of mathematics: a theorem that cannot be proved or disproved by a systematic general procedure.” [Dav93], p. 107

The incompleteness of formal systems is reflected in algorithmic action by the fact that the halting problem cannot be algorithmically decided. This means that there exist well-defined problems which are not computable. In fact, formulated from the perspective of number theory, the majority of numbers is uncomputable, i.e. the majority of numbers cannot be computed by a finite algorithm: there are simply not enough conceivable Turing machines (cf. [Cha97b])! There is an equivalence between formal systems and Turing machines: The output of a Turing machine can be regarded as a theorem, and the decision problem is equivalent to the halting problem (cf. [Cas94], p. 137). In fact, many undecidable problems can be formulated as halting problems (see [GL88], section 3.1.4.). One of these is the problem to decide on the equivalence of two algorithms. This cannot in general be achieved by a machine. But if a machine cannot even decide if two algorithms do the same, it can neither decide, in general, if a given algorithm fulfills a given specification, nor can it construct, in general, algorithms that realize a given specification.

**Definition 10.1: Creativity and Computing.** The design and analysis of algorithms is a creative action which cannot in general be algorithmically simulated. These are aspects of creativity that transcend any predefined formal framework.

After all, computers would be by far less reliable, were they to become ‘creative’ while executing programs, or putting questions back at the programmer: ‘Why am I to compute this here?’

The design and analysis of algorithms is one of the most distinguished human activities. Attempts have been made to have computers find algorithms for themselves, e.g. by specifying evolutionary processes in “Genetic Programming”, but this only shifts the algorithmic design to the specification of these evolutionary processes.

## 10.4 Universal Computation

The Turing machine is a simplified thought model of a computer. Alonzo Church’s Lambda Calculus, invented at the same time as Turing’s machine [Gan88], is an equivalent formalism equally expressing all the power and limitation of universal computation. As a matter of fact, the Church-Turing hypothesis, believed by most mathematicians today, states that universal computation depends rather on some principles than the actual design of a computing machine, so computability can be defined by what a Turing machine or Lambda calculus can achieve. No computer, sophisticated as it might be, can surpass such thought computers in power. Recently, cellular automata have also been found of being capable of universal computation (cf. [Wol94]), hence, computability may also be in the reach of even simple organic or anorganic systems.

In order to realize universal computation, a set of primitive functions is needed with which to build general recursive functions. A zero element (zero

function) and the successor function are necessary to realize arithmetics. Projection and function composition are necessary to access variables containing values and to chain sequences of computing operations. Last but not least recursion has to be provided in order to process input of finite, but variable magnitude. We have seen examples of any of these computational primitives in the definition of the GENDYN algorithm above.

In a *van Neumann* computer, these primitive functions can be implemented by providing an accumulator and memory, along with the instructions “clear” (to reset the accumulator), “increment” (to add to it), “load” (to load a variable’s value from memory), “store” (to store the value back into memory), and “branch” (to test the accumulator and jump to a specific instruction to be executed next). A mechanism has to be provided to access the instructions of a program in order and to perform the branch. Recursion is implemented by repeated branching back to the first instruction of a loop depending on the successful test of the loop’s condition. The first physical machine implementing all of these features and therefore capable of universal computation was the Mark 1 built at Manchester from 1946 to 1948. 12 years after Allen Turing conceived of a universal computing device, his dream ultimately became true (cf. [Roj95]). The computers of today, though incomparably more reliable and fast, are in fact nothing better than the first universal computer when it comes to the fundamental mathematical notion of computing. They are subject to the same limitations as any other universal computer.

## 10.5 Turing Music

Since we have seen that Xenakis’ music is entirely computed by a machine, we can characterize Xenakis’ computer music as a “Turing Music”. This is an adequate naming because it explicitly points at both the strength and the limitation of Rigorous Algorithmic Composition.

A Turing music is restricted to computable numbers. We can conceive of music’s information theoretic representation as a sample stream, or, alternatively, as (the decimal expansion of) a number. It immediately follows that not all conceivable music is computable. Since the majority of numbers is uncomputable, so must be the majority of sonic phenomena. Rigorous algorithmic music can only cover a tiny part of all conceivable music and sound.

Given this fact, we may wonder why GENDYN sound is so rich and complex. Must a Turing music not be dull and dry, given the fact that a Turing machine is so hopelessly primitive? In order to answer this question, we have to look at the “dark side” of computation: the notion of chaos. Xenakis exploited the structure-generating capabilities of classical computation to its extremes by implementing a chaotic model, namely the simulation of stochastic processes.

Let us, to begin with, have a short look at the notion of “Stochastics”. In the following chapter, we will then see how the concepts of stochastics, computation, randomness, dynamic systems, chaos and complexity relate to one another and form a possible background for the world-view and the composition theory of a composer of Automated Art like Xenakis.

## 10.6 Stochastics

“Humans exhibit many impressive physical and mental capabilities, but one of their most characteristic features is the absence of any overall structure that exploits the capabilities in a systematic way. Their behavior therefore often looks erratic and confused. Human consciousness is a passive, association-driven process – a Brownian motion through cognitive space. Many humans find this a rather bewildering experience, and they have difficulty harnessing it to any useful purpose.” [pHH96], p. 155

Stochastics means the analysis of probabilities in random processes and their statistical description or algorithmic simulation. However, in the colloquial use of the computer music community, this term is often used to describe the incorporation of random elements into a deterministic process. Stochastics help calculating the incalculable, the fundamentally unforeseeable. This mathematical theory of probabilities has been developed in the context of gambling and fortuitous events.

The use of stochastics is still quite common in algorithmic composition, and has been crucial in its beginnings. Again, these beginnings, and still the preponderant practice today, are characterized by the computation of score level data that are eventually mapped to notes and other macroscopic patterns, i.e. basically used for score composition. Stochastics have played a prominent role since the very beginnings of computer music (cf. [Loy89]). Xenakis himself used stochastic distributions as early as 1962 for his score-computing program ST [Xen92b]. Other proponents of a stochastic approach to computer composition were Lejaren Hiller, and Charles Ames (by virtue of his COMPOSE system [Bur94a], pp. 94-96).

Tutorial articles of how stochastic techniques can be used for automated composition have been provided by a wide variety of authors, cf. [Ame91], [Lor80], and [Jon81], among others. Few composers, however, have applied stochastic processes directly to the sound signal, as Xenakis kept trying since his first experiments in “Microsound” in the 1970s.

In the domain of digital music synthesis, the problem of composition may be formally viewed as the task to produce an appropriate sequence of sample amplitude values. Given the multitude of sample values to be determined (e.g. 44.100 values per second and channel for CD sound quality), one is obliged to take recourse to various synthesis models that help reducing the amount of sound information to a set of synthesis parameters and leave the job of deriving the raw sound data to the computer. Using probabilities is a way of generating a rich variety of musical structures out of some input data and a limited set of rules (i.e. an algorithm) that can be given to a computer to execute. The use of (pseudo-) randomness in the computer is a most effective procedure for producing highly complex output with comparatively little algorithmic complexity (only a few lines of code). Stochastics are a means to generate surprise in a controlled way.

In science, probabilities are used to describe statistical phenomena, i.e. phenomena that are either too complex to describe in detail (the movement of molecules in a gas) or exhibit an intrinsic statistical behavior (elementary particles) [Hof94a]. The interest may either lie in describing macroscopic features of the statistical whole (temperature, overall radioactivity) or the microscopic, individual behavior of one or some of its members (the moving of a molecule, the spontaneous disintegration of an atom). The macroscopic features of a statistic

whole are linked to the behavior of its microscopic components by stochastic laws. These laws give, for example, the probability for a certain molecule to move with a certain velocity at a given temperature, or for an atom to disintegrate within a certain time interval at a given radioactivity rate.

The laws of quantum mechanics describe the deterministic evolution of a wave function, that is, there is nothing probabilistic about these laws at the level of the quantum world. They allow for a superposition of complementary states that each evolve in parallel as the wave function unfolds in time. However, when crossing the border from the quantum micro world into the macro world of human observation, this potentiality of superposed states is translated into an actualization of only one of these states each with a given probability. Therefore, the quantum laws, while deterministic in nature, describe evolutions of probabilities of macro-world states, that is, stochasticity is intrinsic in their impact towards the macro-world.

“A stochastic system is, roughly speaking, one which is subject to unpredictable and random fluctuations. In modern physics, stochasticity enters in a fundamental way in quantum mechanics. It is also inevitably present when we deal with open systems subject to chaotic external perturbations. In modern physical theory, rationality is reflected in the existence of fixed mathematical laws, and creativity is reflected in the fact that these laws are fundamentally statistical in form. The intrinsically statistical character of atomic events and the instability of many physical systems to minute fluctuations, ensures that the future remains open and undetermined by the present. This makes possible the emergence of new forms and systems, so that the universe is endowed with a sort of freedom to explore genuine novelty. [...] Although each individual quantum event may be genuinely unpredictable, a collection of such events conforms with the statistical predictions of quantum mechanics. One might say that there is order in disorder. The physicist John Wheeler has stressed how law-like behavior can emerge from the apparent lawlessness of random fluctuations, because even chaos can possess statistical regularities.”  
[Dav93], p. 191-193

The use of stochastics in Xenakis' latest computer music has nothing to do with introducing chance into composition. It is a way of harnessing stochasticity in order to create rich structures within rigorously algorithmically produced (“automated”) sound. The drawing of a random number by the computer is not a chance operation, as in the case of Cage's music but a means to actualize a complex state of possible musical evolution to an audible one following a given probability.

The stochastic synthesis as conceived by Iannis Xenakis uses stochastic laws to generate the successive sample values of digital sound. The overall aural aspect of the work is shaped in a way characteristic to the stochastic laws, but due to probability fluctuations, the evolution of the music is always new and rich in detail.

“I could gain some experience in sound synthesis where randomness is acting very fast, that is to say, very dense, because every single sample has to be computed. And when it is finally computed

one gets the impression of the whole form of this stochastic law which is coherent and which is perfectly perceptible also for the ear. This is a remarkable fact. It means that form and randomness are not really a contradiction. Rather, randomness is a form itself, from the beginning, and it also creates forms which take part in non-symmetry, non-periodicity - because that is what we understand by randomness - as they take part in periodicity. That means that one creates the two of them from something more complex. Like the inverse: if you take laws which are very rigorous but at the same time very complex, you end up with something which looks like randomness. [...] These are actually two complementary notions, randomness and non-randomness, absolute determinism and absolute non-determinism; and I think that they are two aspects of the same thing, much like particles and waves are two aspects of the same phenomenon.” [Del97], p. 70

We have by now gained some insight into the concepts of sound synthesis, computation and the notion of stochastic processes, and how they can be harnessed for the creation of rich structure in a computational framework. We are now ready to embark on a discussion of how these concepts may work together to enable the creation of works of art by computational means. In the following chapter, we will discuss dynamic systems and chaos, randomness and complexity, in order to understand the nature of computation in Xenakis’ *GENDY3*.



## Chapter 11

# Randomness and Complexity

In recent times, the word “complexity” has become ubiquitous in discussions about contemporary music. There is even a trend in contemporary composition called “New Complexity”. In contrast to a more or less diffuse use of the word “complexity” in most discussions on music, I shall attempt here to establish a precise meaning to this term which is of a great importance in our discussion on a rigorous algorithmic music. In fact, four different definitions of complexity will be presented and compared. The reason to do so is not so much my interest in the definitions as such but more the aspects of complexity they cover. We will be interested in the notion of complexity in respect to what it tells us about randomness, and in how far randomness may entail or not degrees of complexity. We have already seen that computable randomness is a small subset of real-world randomness. (There are simply not enough Turing Machines to compute all sorts of randomness). Since computed randomness is nothing else than pseudo-random complexity, we shall better speak of (deterministic) chaos.

### 11.1 The Complexity of Chaos

We have seen that universal computation, and hence algorithmic composition procedures, give rise to a complexity which makes it sometimes impossible to predict the results. Algorithmic computer action can be extremely complicated and intricate. For example, there is no general algorithmic procedure to predict the behavior of a computer program, e.g. if it halts after a finite number of steps or not. This fundamental theorem about the complexity of algorithmic action has been established by Allan Turing together with his definition of universal computation. In this section, we will discuss another form of complexity which also escapes prediction. This is deterministic chaos.

Even simple algorithms, when they simulate chaotic processes, are able to produce output which is unpredictable. Very simple formulae, recursively iterated in a simple program loop, can produce patterns which seem to be completely random. Such formulae have been extensively studied by all those interested in fractals and chaos, among them many computer music composers. Algorithms making use of such formulae would have to be called effective in the

sense of information theory: they produce a wealth of output information given only a few bytes of input, which is the formula itself along with its parameters.

We understand the term chaos here as deterministic chaos, i.e. a randomness which is (re-)producible by computation. We have seen that this is (fortunately) the case with Xenakis' GENDYN program. Chaos means that a process can be exactly simulated when initial and border conditions are exactly the same. In Nature itself, this is more or less impossible but with computation, these conditions can in principle be fulfilled, as was the case with the GENDYN project. However, when the starting and border conditions are only similar, the deviation of the process even after a short time will be dramatic. This is called chaos: even tiny deviations at the start end up in dramatic deviations after a short time interval.

The output of algorithmic processes producing chaos is irreducible. Although these processes are perfectly deterministic, they cannot be described other than by simulation. That is, no prediction of the system's future state is available. It has to be calculated by just following the evolution of the process. Therefore, it cannot be reduced to a calculation more effective than the process itself.

Chaotic systems are potentially interesting for composers because of their virtue to deliver complex output, which is a necessary, if not sufficient, condition for musically interesting acoustic structures. Chaotic systems can behave surprisingly, and surprise is an aesthetic category in art. For example, the process of listening to music largely works in terms of (satisfied as well as frustrated) expectations on the side of the listener.

## 11.2 Cellular Automata

Steven Wolfram ([Wol94]), in his studies on Cellular Automata, has shown that they can be classified into four classes due to the amount of complexity they exhibit. It has been already noted that the fourth, most complex class is capable of universal computation, i.e. is equivalent to Turing machines. One class below are CA that produce chaos. So we see that even if algorithms are effective, i.e. if they do not run into the halting problem but successfully compute a finite output, they may still be chaotic. So let us now have a look at Cellular Automata.

A cellular automaton is digital in three ways. It evolves in discrete time steps, as if being driven by a "clock", not unlike a computer. Its space coordinates are discrete. The in-and output of the automaton are discrete. A distinct point in space at a specific point in time having a distinct value can therefore be represented as a colored "cell". A cellular automaton with one space coordinate forms a line of unit cells. The value of each cell at a specific time point is determined as a function of its adjacent predecessor cells at the previous time point. This (recursive) function is called the rule of the automaton. Each rule defines a different automaton.

Cellular automata, being discrete dynamical systems, exhibit the same qualities of dynamical behavior. There are four classes of them. The automata of the first class soon reach a stable state no matter what initial state they started from. They are attracted by a point attractor. In terms of hydrodynamics, one can speak of a "laminar flow". The second class comprises automata which, starting from arbitrary initial states, always evolve to small sets of periodic

states. They correspond to torus attractors. In terms of hydrodynamics, they represent convection flows. The third class is made up of automata whose evolution is “chaotic”. In contrast to the automata of the two previous classes, they are sensitively dependent on their starting conditions. They are characterized by strange attractors. In terms of hydrodynamics, they represent turbulent flow. Certain automata of class four are capable of universal computation.

Cellular automata have been employed in the past by a number of composers, e.g. [Cha90], [HKO91], [OHK91], [Mil90], [Mir95b]. They have mostly been applied to score composition “only”, an exception being [Cha90], [OHK91], and [Mir99]. Xenakis used CA to combine scales of durations and scales of pitch (“Sieves”) in order to create complex temporal evolutions of orchestra clusters in his orchestral works *Horos* (1981) and *Ata* (1986).

Xenakis used CA in his orchestral pieces as if to demonstrate chaotic algorithmic action while composing ([Hof02]). It is almost sure that Xenakis found “his” automaton in a scientific article by Steven Wolfram dealing with the strength and limitation of universal computation in simulating physical reality ([Wol84]). Here the Xenakis automaton is listed as an example for the computer simulation of turbulent flow, hence a chaotic dynamic system.

CA demonstrate fundamental notions such as universal computation, chaos, and incompleteness in a graphical way. They also make tangible the fundamental equivalence between formal systems and dynamic systems (cf. [Cas94], p. 147), for they are both artificial dynamic systems and information processing machines. Xenakis’ occupation with cellular automata, therefore, reveals more than just the using of a mechanism to create complex textures. Cellular automata allow to study both the universal features and the fundamental limitations of algorithmic action of automata as such, including the computer. Using cellular automata for music composition therefore means, at least in case of Iannis Xenakis, to profoundly reflect about the nature of a machine music, i.e. about Automated Art.

## 11.3 The Dynamics of Time

According to Xenakis, the perception of musical time is, like its physical definition, closely related to the conception of space. Time can only be conceived through the discrimination of events. (As Xenakis puts it, time as a continuous flow is imperceptible.) In the human mind, these events form points of reference. They are related by a network of temporal intervals. In this way, time is mentally represented as a discontinuous spatial structure. This structure can be represented on a time axis (cf. [Xen92e]).

Conversely, if a time structure is to be realized in sound, it has to be “imprinted” onto the continuous time flow. It can be experienced, then, as a temporal movement from one point of the structure to the next one. Through this movement an irreflexive, asymmetric ordering relation is established on the set of time-points just the same way as it is established on the set of natural numbers. Time can thus be “counted”.

A final step in the compositional process as conceived by Iannis Xenakis consists, as we have seen above, in establishing a temporal ordering on the points of a duration structure which itself is outside time. It is only through this temporal ordering that the direction of the musical evolution inside time - its “dynamic”

behavior - is fixed. These dynamics can cover the ample field between order and chaos. They can be conceived as a discrete chain of transformations of system states. The succession of states  $f(0), f(1), f(2), \dots, f(n)$  gives the temporal evolution of the piece. In both of Xenakis' algorithmic composition procedures, CA and GENDYN, it is fixed by the general recursive scheme:

$$f(t) = f(f(t-1)) \quad (11.1)$$

The term "recursive" means that the determination of a state has recourse to the determination of previous states, which in turn take recourse to previous states, and so on. To avoid infinite regress into the "past", the recursive chain has to be "anchored" in an initial state  $f(0)$ . This initial state sets the very "beginning of time". In the *Horos* CA, this is a single brass sound, d4 flat. The recursive state transformation is used to evolve cluster configurations in time according to a simple local rule which yields complex output because of parallel, reciprocal action between the tone "cells". Each individual cluster configuration is derived from its predecessor state by applying the same local CA rule to all of its constituents. The recursive formula of a totalistic CA with range 1, as used by Xenakis in *Horos*, has the following form:

$$f_i(t) = f_{i-1}(t-1) + f_i(t-1) + f_{i+1}(t-1) \quad (11.2)$$

where  $i$  is the spatial coordinate of a cell.

This rule is in fact nothing else than a discretization of the differential equation for diffusion processes in physics (cf. [Hof95]). Cellular automata are used by Xenakis as a means for in-time composition: they are one solution to the problem of projecting structures "outside-time" (unordered sets of intervals resp. durations) into time.

The recursive formula of GENDYN sound synthesis is (recall our image of the billiard balls of section 8.6.3 on page 128):

$$f_i(t) = f_i(t-1) + random(z) \quad (11.3)$$

Therefore, the dynamics of GENDYN are of a stochastic nature, hence the term Dynamic Stochastic Synthesis.

## 11.4 The Stochastic Synthesis Viewed as a Dynamic System

The key idea of Stochastic Synthesis is its nonlinear wave shaping, where the wave shaping "function" changes stochastically from wave period to wave period. Consequently, it is not the wave form as such that defines the aural result (although its jaggedness and quantization may contribute to a special sonority) but the dynamic behavior of its deformation over time. It is this change that makes up the specific quality of GENDYN sound by continually transforming its spectrum.

This differential aspect links Stochastic Synthesis to the study of dynamic systems and to notions such as "phase space" and "attractors". A phase space

is spanned by the set of degrees of freedom of the system under consideration. An attractor is a subset of that phase space which geometrically characterizes the long-term dynamic behavior of the system. Is the long-term behavior of stochastic sounds governed by attractors? If so, their shape will look differently from the fractal images of deterministic systems since probabilities are involved. A look at the trace of a random walk particle in the phase space of amplitude value vs. time offset can help to understand some of its dynamics. The graphics that will be discussed in part III have been generated by the New GENDYN program (see figures 14.13 on page 237 to 14.16 on page 239).

Dynamic systems are characterized by a recursive law taking the present state of the system as the departing point for the computation of a future state. These laws take the form of differential equations. In their discrete form, as being modeled e.g. in a computer, these laws can be simplified to so-called difference equations, replacing differential operators by simple arithmetical ones.

Dynamic systems have been much studied in music composition. Rarely though have they been applied to sound synthesis. This might be related to the problem of controlling the dynamics of a dynamic system in a musically convincing way. A complex dynamic system takes much of the compositional control away from the composer, but we have already seen in part I of this study (section 5.10 on page 88) that this fact is rather beneficial in the cooperative approach to music computing, where the composer concentrates on driving the complex sound producing process on a higher level.

## 11.5 Randomness in GENDYN

The GENDYN algorithm, like a Cellular Automaton, represents a dynamic system. It consists of  $n$  times  $m$  Random Walks, where  $n$  is the number of breakpoints per waveform, and  $m$  is the number of tracks which are synthesized simultaneously. One can observe the same complexity classes for GENDYN as for CA: point-like, periodic and chaotic attractors. A point-like attractor, for example, would be any evolution toward a steady state, like a GENDYN sound that dies out because all of its breakpoints are eventually captured by one of the barriers. A periodic attractor is typical for GENDYN sounds with fixed pitch which run through repeating patterns of amplitude modulation. A chaotic or strange attractor would be one of those capriciously jumping pitch movements which go on for a long time without ever repeating themselves.

GENDYN synthesis is chaotic “by nature”. The faintest deviation (e.g. a rounding error in the seventh digit after the decimal point) results in a dramatically diverging synthesis path. The stochastic processes of Random Walks are not characterized by coresponding to a statistical average but, on the contrary, they have a tendency toward extremes (cf. [Fel68]). Pairs of reflecting barriers (which are part of the mathematical Random Walk model) help auto-organize these extreme tendencies into more or less stable configurations. The barriers also may force frequency or amplitude or even both to periodic behavior (stable pitch resp. static wave form), by reducing the chaotic fluctuations to a small degree or even to zero.

Chaos is already present at the start of GENDYN computation by using pseudo-random numbers, i.e. chaotic (deterministic, but complex) number sequences. In fact, pseudo-random number generators do nothing else than folding

(by reduction modulo  $n$ ) and stretching (by multiplication) numbers, similar to the creation process of a fractal structure. This chaos of pseudo-random numbers is further amplified by the cumulative effects of the Random Walks together with the effect of the reflecting barriers.

The Stochastic Synthesis works by running independent Random Walks in parallel on each breakpoint on each of a number of wave forms (one for each synthesis track). However, these Random Walks influence each other by drawing their increments from the same global random number generator. There are many drawings even for a short chunk of sound (up to 44100 per second if breakpoints are narrowly spaced), so the period of a pseudo-random sequence is quickly exhausted. (Xenakis had the ambition to use for the amplitude random walks an implementation of his own, the “Lehmer” random generator, which seems to have a rather short period.) However, when synthesized with other tracks in parallel, the random numbers are drawn at “random” because of the concurrent drawing of all tracks. The values drawn influence the rate of drawing, because the smaller the time spacing between breakpoints, the earlier the next random number is about to be drawn in order to proceed. This mutual impact is clearly audible during the last few seconds of *GENDY3* (18’22”-18’45”): a few remaining tracks (with fixed pitches) enter into repetitive amplitude modulation patterns because they start to run through the same amplitude values again and again. These repetitions are disturbed any time another track stops or resumes synthesis because the random number sequence, then, starts to be shared between the remaining tracks in a different way.

## 11.6 Complexity

Now we are in a situation where a clarifying word about complexity is due. There are many different notions of complexity, scientific and not-scientific, and we shall make clear to which kind of notion we adhere to in the framework of this study. We are less interested in quantifying complexity — i.e. in measuring degrees of complexity — than in the different qualitative notions of complexity that different definitions of complexity imply. We will see that complexity is a fundamental notion for discussing algorithmic music in general and Xenakis’ rigorous algorithmic composition in particular.

**Classical information-theoretic complexity** of a message, or its mean information content, is basically a logarithmic measure of each of its symbols’ probability of occurrence, a definition in analogy to thermodynamic entropy.

**Algorithmic Complexity** of a given set of data is defined as the size of a minimal algorithm needed to compute it.

**Structural Complexity** or information content of an organized structure is equaled to the logical depth, i.e. the number of computational steps needed to arrive at a certain complexity level. It can be related to the execution time of the minimal algorithm needed to compute the data. Others have defined structural complexity as negative entropy or the inverse of entropy.

**Computational Complexity** of a given problem is the expenditure of time and/or space to solve it computationally.

We will see in the discussion of the term complexity that it has always been associated to the notion of information. The information content of messages has been one of the the main interests in the Theory of Communication. The notion of information is also a quality which has been applied to describe the level of organization which is to be found in complex systems such as living organisms, intricate dynamic systems, complex computations, and possibly also art. We will see that the different aspects of information, on the one hand as the content of a communicated message, i.e. its value in a given communication context, and, on the other hand, as an abstract structural quality, leads to almost contradicting definitions which seem at first sight mutually exclusive.

The idea of information which conveys a certain quality to complexity is reinforced by the fact that most, if not all physical, chemical and biological processes can possibly be regarded as computational processes, either because they can explicitly be simulated by computer programs or because (this is a rather indirect argument), it is found that matter and its processes of interaction under the guidance of natural laws is thought to be capable of universal computation (i.e. the universe can be regarded as its own simulation [Ros88]).

## 11.7 Information Theoretic Complexity

We have to discuss communication theory here, and if it was only for the interest Xenakis himself showed for it. Communication theory was prominent in its application to art and music in the 1960s and 1970s in most European centers or creation. Its application to music composition has been discussed by a variety of composers, the most prominent among them being Lejaren Hiller, Stockhausen, and Xenakis (cf. [Mos91]). Xenakis came into contact with this theory under the name of “information theory” through the work of Abraham Moles who was associated at the time to Pierre Schaeffer’s GRM, a studio where Xenakis produced his early electroacoustic compositions. In his writings, Xenakis referred to communication theory at several occasions and linked it indirectly to his music (see [Xen92b]).

### 11.7.1 Randomness, Entropy, and Information

A central notion for the discussion that follows is the notion of entropy.

The Second Law of Thermodynamics states that in a closed system — a system sealed off from its environment to prevent exchange of matter and energy — there is a quantity of disorder that inevitably increases, called entropy. The Boltzmann/Gibbs kinetic model of thermodynamics explains it: consider the molecules of a gas in a container. The temperature of the gas corresponds to the mean kinetic energy of the molecules bouncing off each other and the inner walls of the container. Each micro-configuration of the molecules is equally probable: a configuration where the molecules describe an ordered pattern is equally probable as a configuration where the molecules are spread out all over the container. Boltzmann’s Ergodic Hypothesis states that the system wanders randomly through all these possible micro-configurations. But since the overwhelming majority of microstates of the system correspond to the macro-state of a more or less uniform distribution of kinetic energy, disorder is indeed what is observed. “Thermodynamic equilibrium” is the most probable macro-state.

Let us denote the number of possible micro-configurations of energy states of the particles in a given gas at a given temperature by the variable  $W$ . Then the entropy  $S$  is a function of  $W$  ( $k$  being Boltzmann's constant):

$$S = k \log W \quad (11.4)$$

So the argument is a statistical one: due to the great number of the system's constituents, the spontaneous emergence of an ordered configuration, although theoretically not impossible, is so rare that it is never observed in practice. A consequence of this Second Law of Thermodynamics is that one cannot construct a *perpetuum mobile*: the structured energy lost by being converted into disordered heat due to friction etc. cannot be regained. Order can only be produced at the expense of an increase of disorder in the environment.

A considerable (and in fact the most prominent) part of Xenakis' composition theory, the invention of "Stochastic Music", is directly founded on the statistical aspect of thermodynamics. Thanks to Xenakis' GENDYN algorithm, we can even listen to The Second Law of Thermodynamics in action. When a GENDYN waveform is built only out of a few breakpoints, we perceive fluctuating pitch because the fluctuation of the waveform segments' lengths constantly change the resulting frequency. However, with many breakpoints, the fluctuations of each segment is canceled out on average, and a fixed frequency results. Another observation also pertains to thermodynamics: A GENDYN wave starts with a highly ordered state (e.g. perfect silence where the amplitude of all breakpoints is zero), but reaches "thermodynamic equilibrium" of sound after a short time, i.e. all initial structure is quickly destroyed.

### 11.7.2 Entropy as Information Content

In their seminal work [SW49], Claude E. Shannon and W. Weaver laid the foundation of communication theory, in view of the increasing need for high-speed and long-ranging transport of information to other parts in the world. They looked for a quantification of message information content, in order to assess the best coding schema, in the presence of transmission disturbance factors like noise or crackling. In analogy to thermodynamics, they used the notion of entropy as a measure for the content of information that has to be communicated over a transmission channel:

$$H_n = -\sum_i^n p_i \log p_i \quad (11.5)$$

where  $p_i$  is the probability of occurrence for the  $i$ th out of  $n$  symbols of a given alphabet. In other words, Shannon and Weaver defined the information content of a message to be the statistical rarity or surprise value of the occurrence of its symbols (cf. [Sto90], p. 46).

### 11.7.3 Xenakis' Hypostasis of Noise

Xenakis himself seemed to adhere to this classical definition of complexity as information content when he stated in his conversations with Bálint András Varga:

“How can we construct a line? Once again, with the help of a rule. Such as a sine wave or a descending straight line or a complex line. The freest line corresponds to noise in the pressure versus time domain. [...] A noise [...] is too rich for the ear, we cannot perceive it as repetition, so we put it in the noise domain.” [Var96], manuscript part 1, pp. 100-101.

We know well that this hypostasis of noise is problematic. It is true that in information theory according to Shannon, white noise contains a maximum of information, but everybody knows that musically speaking, white noise is just as poor as a pure sinusoidal sound. Strikingly, Xenakis blames the ear as not being evolved enough to “understand” the white noise. Xenakis’ ultimate homage to noise is probably section 4 of *GENDY3* where the listener is plunged into violent noise during a couple of minutes (see part III: analysis).

We have seen that the notion of information theoretic complexity is founded on the notion of probability of an event. However, there is a fundamental anomaly implied in this definition of complexity: the highest complexity (or information content) is noise (maximal entropy). Maximal Information Theoretic Complexity corresponds to white noise, i.e. to a perfect randomization of all structure, to the “death of heat”. What notion of information or complexity is this? It certainly contradicts all intuition about life and art. In these vital domains, complexity and information is rather the result of a struggle against the nivellating forces of entropy.

“How can a random string contain *any* information, let alone the maximum amount? Surely we must be using the wrong definition of ‘information’? But if you think about it, the  $N$ -bit strings could each label a message [...], and receiving a particular string singles out which of the  $2^N$  possible messages we could get that we are actually getting. In this sense, the string contains a lot of ‘information’. Receiving the message changes your circumstance from not knowing what it was to now knowing what it is; and the more possible messages you could have received, the more ‘surprised’, or enlightened, you are when you get a specific one. If you like, the difference between your initial uncertainty and final certainty is very great, and this is what matters.” [Fey99]

Feynman is clear about the fact that information-theoretic complexity makes statements about the quality of the coding schema of a given information, and not about the information itself. After all, Shannon and Weaver were interested in optimal encodings of messages for transmission over disturbed channels, not in the value of the messages as such. Therefore, as powerful a notion as it may be, information theoretic complexity cannot really explain complex phenomena as e.g. life or artistic expression. Alternative notions of complexity have been established partly as a response to this, as we shall see in the following section.

## 11.8 Structural Complexity and Information

### 11.8.1 Complexity as Negative Entropy

The model of negative entropy has been developed by scientists in order to explain the extraordinary tendency of living organisms to keep and even increase their structure and inner organization in spite of the general tendency of the universe to ever increase its entropy. It has been found that the phenomenon of biological life does not contradict the laws of physics and especially the second law of thermodynamics, but manages somehow to perpetuate a state far from thermodynamic equilibrium, i.e. maximal entropy.

Therefore, we see that the complexity of living organisms finds itself at the opposite end of the entropy scale, in comparison to information theory which identifies entropy and information content. This is reflected in a notion of negative entropy.

“How are we to describe the wonderful ability of living organisms to defer their decay into thermodynamic equilibrium (i.e. death), in the language of the statistic theory? We said: ‘It thrives on negative entropy’ by seizing a stream of negative entropy, as it were, in order to compensate for the increase of entropy caused by its own living and to keep a stable and rather low entropy level. If  $D$  is a measure for disorder, then the inverse  $1/D$  can be regarded as a direct measure for order. Since the logarithm of  $1/D$  is equal to the negative logarithm of  $D$ , we can write Boltzmann’s equation like this:

$$-(Entropy) = k \log(1/D) \quad (11.6)$$

Therefore, we can replace the awkward expression ‘negative entropy’ by a better one: entropy with a negative sign is by itself a measure for order.” [Sch44], quoted after the German edition, pp. 128-129

Schrödinger goes on by saying that the input of order needed by living organisms is assured by the high degree of organization present in the organic substances which serve them for food. At the front end of the food chain, the plants draw their negative entropy from sunlight. (As an aside, the perspective of entropy once again underlines the deep connections between the scales in our universe, going from single cells to the cosmic scale, our solar system, the galaxy in which our sun was born, etc., much in the pathways of Xenakis’ own philosophic reasoning.)

Schrödinger’s negative entropy has later been associated with the notion of information. In the history of science, Schrödinger’s thoughts are considered to have considerably fostered research on the material conditions of life, which eventually led to the discovery of the DNA double helix structure by Watson and Crick, in 1953. We today know that the DNA structure is a combinatorial reservoir of genetic information, and that the body is unfolding and processing that information much in the kind of a biological “program”.

“The increase of order during evolution has been shown to be compatible with the laws of physics by studies of Manfred Eigen

[Eig76]. Eigen was able to show the following: If a system with the property of self-reproducing with low error rate is constantly fed energy and matter, then there is an exchange quantity in this open system which tends toward a maximum. This quantity is closely related to Schrödinger's negative entropy. It can be labeled 'information'." (Ernst Peter Fischer: "Was ist Leben?"—Mehr als vierzig Jahre später", preface to the German edition of [Sch44])

### 11.8.2 Complexity and Information as Inverse Entropy

A theory of information, based on Schrödinger's ideas on negative entropy, has recently been developed by [Sto90].

"Schrödinger's equation is my point of departure. To this I add [...]: information  $I$  is a function of order. [...] information and organization [are] in a direct linear relation [...]:

$$I = ce^{\frac{-S}{k}} \quad (11.7)$$

[Sto90], quoted after the German edition.

A graphical example for the connection of (negative) entropy and information is Maxwell's daemon ([Szi29]). It is a thought experiment of an agent acting against the Second Law of Thermodynamics. Imagine a gas container with a membrane dividing it into two halves. The Maxwell daemon uses his information about the velocity of individual gas molecules to sort the faster ones onto one side and the slower ones onto the other side of the membrane. In this manner, he is able to invert the irreversible process of diffusion (cf. [Sto90], p. 39).

But can the notion of entropy (and its inverse: negative entropy) really capture the organizational "deep structure" of patterns and messages, like the observational data of the universe, the signs of life, or creations of living beings, like e.g. art and music? It seems as if one still had to search further for other notions of complexity.

## 11.9 Algorithmic Complexity

Let us consider two strings of binary digits

0101010101010101

001011010100111010

Both have the same information theoretic complexity (this can be checked by applying formula 11.5 on page 176 given above) but obviously exhibit a very different quality in their patterns. The first one is a regular repetition of a substring 01, the second one may have been obtained by tossing a coin. Information theoretic complexity does not reflect the structural quality of these two patterns because the symbols 0 and 1 in both strings exhibit equal probability. (Each of these two 20-bit strings is a possible outcome among  $2^{20}$  different combinations of 20 1's and 0's., cf. [Cha75]).

Algorithmic complexity is a notion that better reflects the qualitative characteristics of regularity and irregularity. According to its definition, the first string can be generated by a small program with a loop:

```
[1..5].each{print "01"}
```

The second string takes a longer program to generate:

```
{print "001011010100111010"}
```

i.e. there is no algorithmic reduction at all.

We have seen that computable solutions form a proper subset of the problems that can be formally specified. This leads to an equivalent algorithmic information theoretic statement which says that no program can calculate a string more complex than itself ([Cha75]). For example, a given number cannot be proven to be random. This would imply to prove the non-existence of an algorithm able to reduce that number to a program smaller than this number.

Applied to the notion of scientific theories, this means that there are aspects of reality which cannot be reduced to a simple application of Natural Laws. Scientific theories are a proper subset of physical reality. Much of reality is irreducible in this sense, and can therefore only be studied through explicit simulation and empirical observation. This imposes, in Chaitin's view, an empiric attitude to mathematics and physics, in contrast to the formalistic approach of e.g. Hilbert and his disciples.

The notion of Algorithmic Complexity directly pertains to Xenakis' GENDYN, and therefore plays a prominent role when discussing GENDYN's theoretical implications. The whole of the complexity of a music like *GENDY3* can indeed be algorithmically reduced to the 50 or so specification lines of DSS in section 9.6 on page 149. More generally: the algorithmic complexity of a RAC composition is precisely the size of the smallest algorithm used for its generation.

A corollary of this theorem is that minimal algorithms are perfectly random — they cannot be produced by other algorithms. In trying to find the complexity of a piece of music like *GENDY3*, a minimal or at least close to minimal algorithm must be found that is capable of producing the whole of the music. One of the aims of section 9.6 on page 149 was to describe this algorithm, and one of the goals of the GENDYN project was to implement it in order to proof that this algorithm is a valid reduction of *GENDY3*.

GENDYN is a rare case of computer music whose algorithmic complexity is indeed measurable, precisely because this music is algorithmically defined. It is questionable if the notion of Algorithmic Complexity could be applied to music in general. It should be quite clear that music produced with human interaction is not reducible to an algorithm, even if there have been some successful attempts to do so (cf. [Kin91], [Che95] [RM88], [RM89]).

## 11.10 Computational Complexity

Computational complexity measures the hardness of a problem to be solved by computation, with regard to expenditure in computing time and/or memory space. Problems can be proved to be hard by a certain factor linking the input size of a problem to the expenditure to solve it: constant, linear, polynomial or

exponential. Solving time and memory space can often be traded off against each other. Consider the entries of, say, a telephone book. If we provided as many sorting “bags” in memory space as there are possible telephone book entries, all entries could be sorted in one go, i.e. in a time linear to the input size. In the following, however, we will only consider the time factor.

**A constant complexity** is basically the application of a closed formula: regardless the size of the input, a solution to the problem can be found in a time  $k$  not dependent on the size of the input.

**A linear complexity** means that computation time will linearly grow with the size of the input, e.g. by the factor  $2n$ ,  $n$  being the input size. This is close to human intuition: more potatoes will take more time to peel.

**Polynomial complexity** can be expressed by a polynomial in the size of the input; a simple polynomial, for example, would be a quadratic or cubic expression  $n^2$  or  $n^3$ . This corresponds roughly to the number of nested iterations that are necessary to process the input: for example, if, for treating each element of a list, all the rest of the list has to be looked through, the polynomial will be of a quadratic degree.

**Exponential complexity.** This is the “worst” case. Time to solve the problem is expressed as a power of the input size, e.g.  $2^n$ . Such a complexity rapidly grows beyond all measures. If the problem has a considerable size, a solution may take longer than the universe is assumed to persist, regardless of how fast the computer program is thought to execute.

What is the computational complexity of Xenakis’ DSS algorithm? We can find this out by examining its structure. In spite of the many hierarchical levels of computation (samples, breakpoints, waves, fields, tracks, sequences), there is essentially a single loop calculating sample after sample, by drawing random numbers, passing them through the stochastic formula, then through a series of two random walks, and finally assigning them to the time and amplitude coordinates of the wave form’s breakpoints. Computation is therefore “only” linear, with a considerable, but bound amount of computation per sample. Even if we assume that every single sample were to be a breakpoint, there will be exactly as many iterations of the program’s main loop as there are samples to be computed. Therefore one can fairly say that the execution time of GENDYN grows linear with the size of the music to be computed.

This is the reason why Dynamic Stochastic Synthesis, given an efficient implementation, and a fast computer, can execute in “real time”, i.e. the computation of the samples is faster than the rate of “playing” them back through a digital-to-analogue converter, regardless of how long this goes on.

## 11.11 Complexity as Logical Depth

We have had a look at algorithmic complexity and we had a look at computational complexity and we found both unsatisfactory as an explanation for the felt complexity of Xenakis’ music. So how about a combination of the two? Here is what I found in a book which discusses the complexity of Creation. The title of this book is “The Mind of God”:

“Even though chaos is rather common, it is clear that on the whole the universe is far from being random. We recognize patterns everywhere and codify them into laws which have real predictive power. But the universe is also far from being simple. It possesses a subtle kind of complexity that places it partway between simplicity on the one hand and randomness on the other. One way of expressing this quality is to say that the universe has ‘organized complexity’. [...] There have been many attempts to capture mathematically this elusive element called organization. One is due to Charles Bennett, and involves something he calls ‘logical depth’. This focuses less on the quantity of complexity or amount of information needed to specify a system, and more on its quality, or ‘value’. Bennett invites us to think about the state of the world as having coded information folded up in it, information about the way the state was achieved in the first place. The issue is then how much ‘work’ the system had to do — i.e. how much information processing went on — to reach that state. This is what he refers to as logical depth. The amount of work is made precise by defining it in terms of the time taken to compute the message from the shortest program that will generate it. Whereas algorithmic complexity focuses on the length of the minimal program to yield a given output, logical depth is concerned with the running time for the minimal program to generate that output. Simple patterns are logically shallow, because they may be generated rapidly by short and simple programs. Random patterns are also shallow because their minimal program is, by definition, not much shorter than the pattern itself [...]. But highly organized patterns are logically deep, because they require that many complicated steps be performed in generating them.” [Dav93], pp. 136–138

The notion of logical depth is, I think, the definition we need in order to understand the complexity of *GENDY3*. I, personally, feel most comfortable with this definition of complexity, in relation to Xenakis’ composition program. The sonic results of its action are neither simple (i.e. rigidly periodic) nor perfectly random (i.e. white noise), but something in between, so they are not logically shallow. But where does the complexity come from, since it cannot be found in the length of the algorithm, which is comparatively short, nor the computational complexity, which is, fortunately, quite low, that is, a linear one? If neither the algorithmic complexity of the *GENDYN* program nor its computational complexity can account for the actual complexity of the sonic structure produced, then we have to look for other criteria.

The fact that the program is relatively simple but heavily iterative points at the model of dynamic systems and chaotic action. An individual step in the execution of the program might be relatively straightforward, but it heavily depends on the accumulated result of all the small steps of the program executed so far, so the computation of the last sample of *GENDY3* is in fact the result of computing all foregoing some billion samples of the piece. So, there is indeed an ingredient of heavy work required for the sonic structure to emerge: it is equivalent to the evolution of a chaotic dynamic system.

Let us recapitulate the argumentative path that we have followed in order to find the notion of complexity appropriate for the discussion of Xenakis’ algo-

rhythmic approach to music. We started with the classical notion of information theoretic complexity which pairs complexity with thermodynamic entropy - a concept which was known to and adhered to by Xenakis himself. It also stands behind Xenakis' concept of a Stochastic Music developed in the late 50s. But we saw that this concept has an anomaly: maximal complexity leads to complete randomness (white noise) which is without any structure and musically meaningless.

We have also seen that this statistical notion of complexity only applies to a statistical whole - as a choice out of the set of all possible messages constructed out of a reservoir of symbols. It does not imply a qualitative statement about the message itself: as we have seen, the alternation of 1010101010 has just the same information theoretic complexity as has 1101001010, although one of them is very structured and the other is not. Therefore we tried algorithmic complexity which covers exactly this aspect: it makes a statement on the internal structure of data. Because this structure is defined in mechanical terms of computation, it ideally applies to Rigorous Algorithmic Music, i.e. to GENDYN.

Yet, there is something unsatisfying about the reduction of 20 minutes of complex sound to just a few hundred lines of program code. There is a feeling that there must be an aspect that has been missed by equaling the complexity of *GENDY3* with the length of the program text behind it. The idea is to have a look at the process computation itself and how it builds up the complexity of an artistic edifice such as *GENDY3*.

So we discussed the computational complexity of *GENDY3*, the expenditure of time relative to the length of the piece. We found that the computation of *GENDY3* is linear in respect to its length. Linear computation is one of the lowest categories of computational complexity (as a matter of fact, a desirable one), so again we left the discussion disappointed.

The last notion of complexity we looked at finally gave us a good feeling. It is the notion of complexity as the notion of logical depth, i.e. the amount of recursive recourse to computationally antecedent results. Even though the program itself may be minimal, and its execution linear, a considerable amount of complexity in the end result is achieved through an interweaving of the many little computational steps, similar to the computation of chaotic dynamic systems. As a consequence, the result of such a computation is not reducible to simpler terms, it can only be simulated by another computation which can in principle be no more efficient. It is as though the music would compute itself, for every structural detail depends on a cumulation of the structural details computed so far. If the universe and its biological inhabitants were to be simulated by computation at all, they would be of such a complexity.

An artificial example of such a complex computation are the toy universes of cellular automata: seemingly primitive in the iterative application of a simple formula, therefore algorithmically extremely compressible, linear in execution, and yet not reducible to any simple description because everything is linked and nested to a considerable logical depth which can ultimately reach up to universal computation. We have already stated that universal computation as such is irreducible in a strong sense.

It is important to note that the notion of logical depth, of complex organization and order is in direct contradiction to the notion of entropy. According to Schrödinger and others, organization and the evolution of order, differentiation and integration, and most notably biological development thrives on negative

entropy, i.e. these systems avoid entropy by increasing the entropy of their environment. The notion of information, which adds a qualitative feature to the rather neutral concept of complexity, has been defined in inverse exponential relation to entropy by [Sto90] (see equation 11.7 on page 179). This entails that entropy can become negative indeed, when it comes to living systems and to information processing, in contradiction to the Second Law of Thermodynamics.

“The rest of the universe may be declining and tend toward a state of maximal entropy, yet on our planet (with the help of the sun) the entropy decreases constantly! This is true not only for biological systems but also, and to a higher extent, for the cultural evolution, the technological culture and the evolution of human information systems. Only in closed systems can entropy never decrease. In open systems not only can entropy decrease but even fall under absolute zero [...]” [Sto90], quoted after the German edition, p. 51

Information is a factor that not only characterizes higher-order life, but may also play a substantial role in the physical foundation of our universe. Austrian physicist Anton Zeilinger has pointed out that information lies at the bottom of the subtleties of quantum mechanics. He explains Heisenberg’s uncertainty relation with the limited amount of information which systems of elementary particles can contain. They do not have “enough” bits of information to determine both, say, location and momentum of an electron ([Zei03]).

The logical depth of chaos seems to be an important ingredient to the state of our world in general. As a member of the Santa Fe Institute for Complexity Studies puts it:

“A world without strange attractors and, hence, without chaos would be very impoverished in the number of mathematical theorems that could be proved. This conclusion, in turn, implies that whatever real-world truths might exist, the overwhelming majority of them cannot be the counterparts of theorems in any formal logical system. Of course from this perspective we might already be living in such a world. But the existence of strange attractors allows us to hold out the hope that the gap between proof and truth can at least be narrowed - even if it can never be completely closed.” ([Cas94], p. 148-149)

The same author presents an interesting definition of complexity on a meta-level of computation, i.e. he looks at the number of possible theories which can explain a given phenomenon, in a computational sense, i.e. though formalization. This definition seems particularly appealing to a discussion of complexity as related to phenomena of art and music because it brings into account the perspective of interpretation of a given formalism. In this definition, complexity is defined as the potential of a system to be described by a multiplicity of incongruent theories.

**Definition 11.1: Complexity As a Multitude of Possible Theories**

“The complexity [of a system  $N$ ] as seen by an observer is directly proportional to the number of inequivalent formal descriptions of  $N$  in a formal system  $F$ .” [Cas94], p. 276–277

This recalls our discussion of formal systems, and how they relate to computing. Computing can be regarded as a manipulation of symbols (representing bits of information) within a formal system  $F$ , connecting axioms and theorems both expressible within the formal language of  $F$ . The theorems of  $F$ , in turn, are related to true statements within a (more or less) “real-world” system  $N$  (such as mathematics or computer music) through the process of interpretation. Just the same way as we are interested in constructing formal systems  $F$  and algorithms linking the theorems and axioms of  $F$  to express mathematical truths, we can be interested to find formal representations and generating algorithms for musical artifacts.

It is possible to construct different formal systems to express the truths of mathematics, or parts of it, like arithmetics, for example. This is the task of mathematical basic research since the turn of the last century. In the same way, musical structures can be generated by computer programs representing algorithms working on a system of symbolic music representation, be it notes or just sample values. In art, in contrast to mathematics, the idea is not to prove any truth — “truth” in art is nothing that can be logically proven, but the potential of an artifact to refer to a multitude of constructing frameworks is certainly a sign of aesthetic substance. Music in particular, as art in general, asks for theorization because its aesthetic substrate is amenable to thought. This recalls the famous dictum of Hanslick’s: “Composing means the action of thought in an intellectual medium [Komponieren heißt: Arbeiten des Geistes in geistfähigem Material]” [Han54].

It is interesting to see that Xenakis seems to have felt the need for such a quality of complexity when it came to composing by rigorous algorithmic means. GENDYN is one example; Xenakis’ use of a cellular automaton for the generation of orchestra clusters in *Horos* (1981) is another.

## 11.12 Xenakis’ Concept of Automated Art revisited

The execution of an algorithm can be understood as purely syntactical symbol processing in the framework of a predefined, closed formal system. One possibility to define artistic creativity, however, would be the artist’s ability to transcend any preconceived system, to establish new relations between instances and to build new systems that can subsequently be subject to yet another act of transcendence. Viewed in this perspective, algorithmic and artistic modes of creation seem to be mutually exclusive. It has even been doubted if all what happens in the physical world can be described by the computable laws of physics [Ros88].

It seems as if those who know the mathematical foundations of computation the best, tend to assume a fundamental uncomputability of those processes that the specific human mental activities are based upon: consciousness, self-reflection, and intellectual insight [Pen89]. Often this is reduced to one or the other form of Gödel’s theorem, which states a principal limitation in modeling the world with the help of formal systems. Gödel’s theorem has been referred to by Xenakis himself as a proof for the limitation of all algorithmic action concerning creative issues (cf. [Xen96], p. 148-149). Only few other computer-composers have so profoundly been reflecting the foundations of their practice.

One of them was John Myhill. He was a composer-scientist working with Herbert Brün whom we cited as one of Xenakis' brethren in spirit.

"In 1952, the mathematician and philosopher John Myhill from the Urbana School formulated the assertion that the whole edifice of musical thinking can scientifically be formalized 'crystal clear' but that not all aspects of this edifice would be computable. In other words: According to Myhill, finite automata, and therefore also computers, are incapable to simulate every aspect of the outer world." [Sup97], preprint, p. 202, quoting [Myh52]

What does this mean for Xenakis' concept of Automated Art? This concept is certainly not the dream of an artificial artist. An automaton cannot "create" in the artistic sense of the word. This is implied by Gödel's, Turing's, and Church's findings, and explicitly recognized by Xenakis. Automated Art, in Xenakis' view, can therefore only mean to use the processing power of a computer to extrapolate the ramifications of artistic thought, all the while reserving to the human the role of creative decision. The computer, for Xenakis, is neither a tool for computer aided composition, nor is it a wonder machine for the realization of an artificial new cyber art. It serves as a powerful instrument for the artistic verification of fundamental compositional ideas.

In the case of CA, the idea is complex temporal evolution of dense orchestra clusters, typical for his late instrumental style. In the case of Stochastic Synthesis, the idea is nonlinear wave-shaping in order to create unheard-of sonic evolutions beyond any established acoustic theory. It is striking to see that in both cases where Xenakis relies on rigorous automatisms in composition, he makes use of parallel reciprocal action, chaos and emergent processes. CA are classical examples for the study of these phenomena, and the reader is referred to the literature (cf. [Wol94]). For GENDYN, this is less obvious, and has neither been put forward by the composer himself ([Xen92c]), nor by his colleague of the time at CEMAMu ([Ser93]). Yet, sonic self-organization of the sounding matter is indeed the most prominent feature of Stochastic Synthesis.

The "automation" of creative action is a natural consequence of the attempt to formalize musical thought. But Automated Art cannot be a substitute for human creativity. Its true value is only revealed when it is harnessed by human ingenuity. Its rich potential serves to stimulate and challenge artistic invention, as well as to confront the listener (and the composer himself in the first place) with a different acoustic reality. Therefore it is not only legitimate but important to break the rules and to change the specification wherever it seems appropriate in order not to be trapped by machine logic. This is, I think, the lesson that can be learned from Xenakis' project of Automated Art.

## Chapter 12

# Software Engineering Aspects of the GENDYN Project

The New GENDYN Program is a new implementation of Iannis Xenakis' composition algorithm Dynamic Stochastic Synthesis in a distributed real-time environment. The New GENDYN program is backward compatible to Xenakis' original concept of a self-contained sound producing automaton creating music "out of nothing". However, the step toward real-time processing extends the Stochastic Synthesis into the domain of "interactive composition", turning the GENDYN into a stochastic composition instrument interactively controlled in real-time by a musician/composer.

In this chapter, I will focus on describing how Dynamic Stochastic Synthesis has been ported to a new computer environment, from procedural programming under DOS to distributed objects under Windows.

The GENDYN re-engineering project mirrors in some way the evolution of software engineering from the sixties to today. Xenakis was more a composer than a software engineer. His programming concepts had not much changed from the times of his pioneering ST-program, written in FORTRAN on an IBM mainframe in 1962 [Xen92b]. Control flow in his GENDYN program jumps along GOTO statements. The calculations are overloaded with a number of state transitions on a large variable set. Large subroutines work with side-effects. This programming is very hazardous indeed! However, the algorithm is more or less correctly implemented (with minor deviations described in section 14.10 on page 224). The fact that Xenakis got such a rich variety of sounds out of his unnecessarily complicated program makes the composer's achievement even more impressive.

### 12.0.1 Object-Oriented Design

Objects are autonomous units of computation that are dynamically created during program execution. In the current implementation of the stochastic synthesis, the task of computing a sequence is carried out by a sequence object, the task of computing a track is carried out by a track object, and so on. The objects

cooperate for the generation of a musical piece i.e. a sequence object employs a collection of track objects and so on. The wave form polygon of each track, the random walks working on it, and the barriers controlling the random walks are also modeled as objects. They are referenced through a super-ordinated “sound” object (see figure 12.1 on the facing page). It passes each vertex of the polygon to the random walk objects to have it displaced in random walk space. A random walk object employs a distribution object which delivers a stepwidth for each random step according to a random number and a given probability distribution. The random walk object then limits the displacement value as well as the random walk space with the help of two mirror-pair objects.

The use of objects for the implementation of the GENDYN algorithm has at least two advantages. First, synthesis data are rigorously structured in small separated entities that are clearly organized on a higher level. Second, the computation is divided into small independent routines that cooperate in a straightforward way for the generation of a musical piece. Since data are closely tied to the routines working on them, there is no need to access global data structures which makes the implementation simple and easy to understand. It will also facilitate future development.

## 12.1 Distributed Computing

The new GENDYN program is divided into two components: a C++ sound server “engine” and a graphic interface client written in a 4GL language (currently Visual Basic). Client and server are coupled with the help of distributed objects (currently using Microsoft’s OLE library). Sound computation has been sped up by a factor of about 60. This makes it possible to change parameters of the synthesis and listen to the sound in real time. A more detailed description can be found in [Hof96] and [Hof98].

The separation of computing logic and control logic into two separate components combines the advantages of C++ speed on the one hand with the flexibility of Rapid Application Development on the other. (Moreover, in a non-multi-threaded operating system, such as was at times Windows 3.x, running interface and sound engine as two separate tasks were the only possibility to realize a fair balance between user interaction and computation. Otherwise holding the mouse clicked on a parameter control would bring sound computation to a halt.) The distributed design allows for an independent development of the “purely algorithmic” and the interactive part of the program. For example, it would now be easy to reintroduce the idea of time-variant parameters that Xenakis tried out in his own program, as mentioned in the introduction. It could easily be done by “scripting” the graphical control interface, without further complicating the sound engine itself.

The load of interactive graphic processing may be assigned to a different computer in a (possibly heterogeneous) network or even across the Internet, if standard protocols (like CORBA or SOAP) are supported.

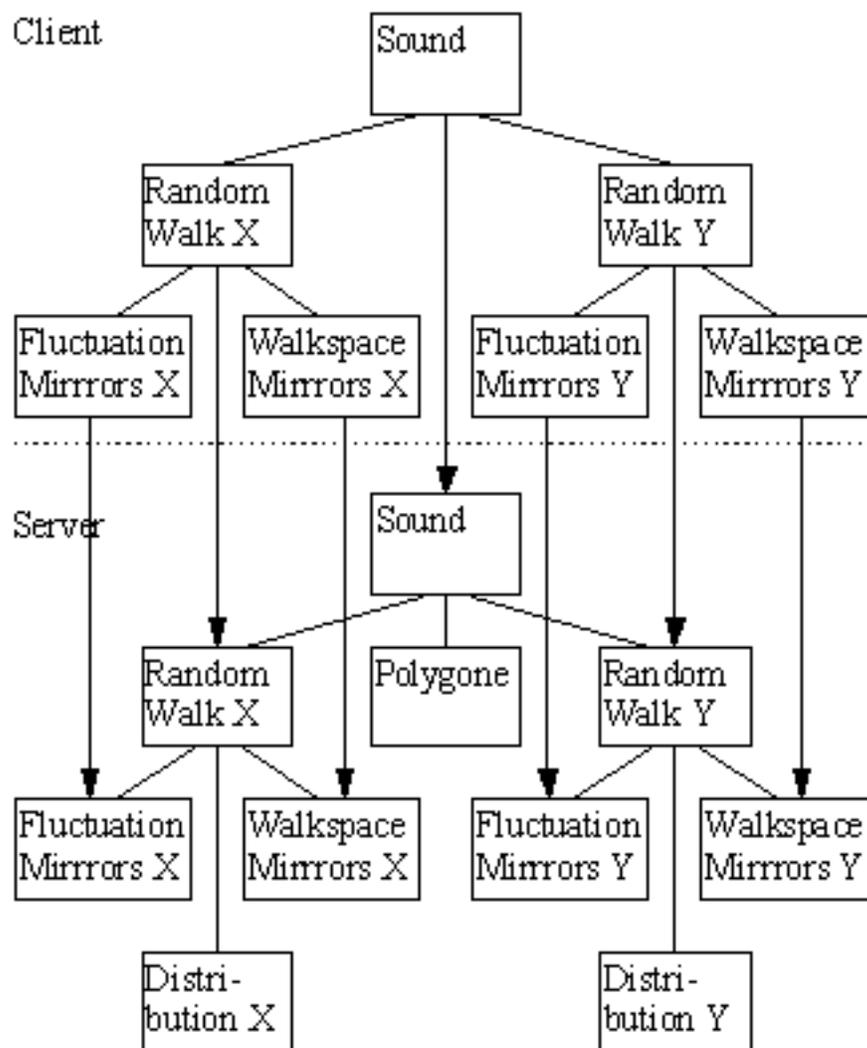


Figure 12.1: The sound synthesis objects along with their primary parametrization.

## 12.2 Distributed Objects

Interactive soft- and hardware systems tend to be no longer developed as closed systems with a rigidly defined overall functionality but composed of small autonomous entities flexibly cooperating for the fulfillment of a super-ordinated task. To achieve the overall function, software components are plugged together in a specific way. Such systems are easy to be reconfigured, scaled, extended, and adapted to evolving needs. Components are self-contained, functional entities with a well-defined behavior and interface.

Typically, a component hides implementation details to the outer world, but not the parameters of its functioning: it can be fully automated and monitored through the control of remote components. Since components are autonomous computing entities, they are even more reusable than software objects. Not only are they reusable by the developing programmer but also by the user himself. The user (=artist) therefore gains considerable independence and emancipation from the system designer, much in the sense of the participatory design paradigm.

If components implement local-remote transparency and multi-threadedness, computation, control and monitoring can be flexibly assigned to different hardware platforms and/or input/output devices, without changing the code. If portable code is used (e.g. a subset of C++ or Java), components can be made to run on different hardware architectures, and be controlled by various means, e.g. through a Web browser. The access from the outside (the idea of the “studio on-line”) [HBLB98], is, then, nothing than a mere side-effect of the transparent distribution in the component approach.

## 12.3 Software Engineering Issues

Xenakis’ implementation of Stochastic Synthesis was a Quick Basic program of 5813 lines of code (with parts of a similar version being published in [Xen92b]), along with a collection of 11 Quick Basic programs defining the parameters, each comprising about 846 lines of code, forming together about 15119 lines of code.

In the New GENDYN Program, the synthesis is implemented in 1549 lines of code (500 of which are the core algorithm, 700 I/O and the rest miscellaneous control functions). Added to it are 6 Visual Basic components for the graphical user interface: 1267 lines for the application window and menus (`gendyn.frm`), 400 for docking to the sound synthesis engine (`genintfc.gbl`) 500 for the main dialog (`genpiece.frm`), 800 for the sequence dialog (`genseq.frm`), 1400 for the sound/track dialog (`gentrack.frm`) and 110 for various control (`mdiintfc.bas`), summing up to 4477 lines of code.

Besides, there are about 1100 lines of “gluing code” in order to make synthesis and user interface work together. In sum, the new software has 7126 ( $1549 + 4477 + 1100 = 7126$ ) lines of code. However, using the Visual Basic compiler and the C++ compiler, the software is compiled into over a megabyte of machine code, without counting various dynamically linked system libraries that would add at least another megabyte to it, whereas Xenakis’ BASIC program compiled to an executable file of about 60 kilobytes.

The difference in a nutshell between the original GENDYN program and

modern programming techniques can be seen in the fact that the new GENDYN program comprises less than half the size of source code of the old program (7000 vs. 15000 lines), in spite of the fact that it features more functionality: a complex graphical interface with real-time control of the synthesis. At the same time, its binary file is fifteen times as large (about 1 MB, without counting the dynamically linked system libraries). That means that there is a lot more support by the programming and operating systems today used to implement additional functionality (like e.g. remote procedure calls and graphics).

## 12.4 The Design of the New GENDYN Program

The component approach is used to separate the interface and the synthesis engine of the New GENDYN program into two processes (tasks), each implemented in a different language: C++ for synthesis, for sake of execution efficiency, and a RAD language for the interface (currently Visual Basic). The communication between the components is established on a “high level”, by remote object creation and invocation, rather than by “low-level” communication through streams or sockets. The Object Request Broker negotiating the object-to-object messaging is currently Microsoft’s OLE, but the design can easily be adapted to a CORBA or another architecture.

The New GENDYN program implements the Xenakis algorithm as a hierarchy of small objects, each fulfilling a specific task, and cooperating for the production of sound. Enumerated from low to high level, objects implement distributions, elastic mirrors, random walks, polygons, sounds, tracks, sequences, the piece and global control of I/O and playback. The objects encapsulate the generation routines with the variables and the parameters of the synthesis. There are actually two layers of objects: one encapsulates the synthesis state variables, the other the synthesis parameters, in order to keep them logically separate.

All objects are controlled in real time by the graphical interface, containing the control counterparts of the synthesis objects. At program startup, each control object is linked to its remote partner in the synthesis engine. The engine runs as a background process (implementing the Windows MFC “idle loop”), computing sound as a succession of sample chunks. Sound is only buffered for output to the DAC device; all computation is done on a sample-by-sample basis. In order to speed up computation, the C++ inline declaration feature is extensively used for the routines within the synthesis calling tree (up to 7 levels deep into the object’s hierarchy).

During synthesis, the user interface process queries the sound synthesis engine on a regular basis to display the current state of synthesis in a graphical way: the distribution functions are displayed as graphs (each drawing of a value plots one point of the graph’s curve at a time), and the random walks as billiard balls moving in 2D space. Moreover, either the waveform or a pitch curve (the evolution of the wavelength as a representation of the sound’s fundamental) can be plotted. All plotting is done by pulling the current states of the synthesis variables in a timer loop (i.e. there is no callback from the synthesis engine).

The synthesis object hierarchy can be either remotely created by the interface or by the engine itself on parsing a data file. While the stubs of the remote object creation and invocation are offered by the Visual Basic runtime system (they are implemented in the Visual Basic Object Adapter), the C++ stubs have to

be coded by the server application. (Fortunately, the MFC framework does most of the low-level coding.)

What may be interesting from the point of view of current Object Request Broker implementations is the fact that in the GENDYN server, I designed the skeletons as classes which inherit from the synthesis classes, a design that allows almost 100% transparency. Transparency, in this context, means that one cannot tell from looking at the code that parts of it execute on different computers or within different processes on the same computer. As a consequence, one cannot tell from the signature nor from the implementation of the synthesis classes that they are remotely controlled by a user interface running in another process by the same reason that one cannot tell from a class that its functionality has been extended by another subclass. This is different from e.g. Visigenic's CORBA implementation, where the skeletons either are base classes of the implementation classes (default) or delegate the incoming calls to the implementation objects ("tie-mechanism"). Only for the server to make its own objects controllable, object creation has to be implemented in virtual methods that can subsequently be implemented by the skeletons. Thus the synthesis engine can be used stand-alone in "command line" mode without changing a single program line, because the skeleton classes form an "outer shell" to the synthesis classes.

Part III  
Analysis



This part is dedicated to the analysis of *GENDY3* (1991), a piece of music entirely generated by computer. *GENDY3* is the one and only piece composed and published with the 1991 version of Xenakis' GENDYN program. *GENDY3* was premiered at the *Rencontres internationales de musique contemporaine*, Metz, France, in November 1991. The audition of a former, unpublished variant had taken place a few months before at the Computer Music Conference, Montreal [Xen91]. *GENDY3* was later (1994) released on CD [Xen94a].

As we have seen in part I of this study, any assessment of electroacoustic music cannot be dissociated from its technological aspects. Artistic creation in the electroacoustic domain is necessarily conditioned by the technology used. This is all the more true in the case of *GENDY3* which is a music "idiomatic to the computer", as we have stated. As a consequence, the computer is part of the aesthetic substance of the music. This becomes evident through the presentation of *GENDY3* by the composer himself: its first variant was performed at the ICMC Montreal, accompanied by an article [Xen91] stressing the importance of stochasticism and computing.

The foregoing parts I and II of this study had the aim and purpose to lay the foundations for a proper appreciation of Xenakis' ultimate achievement of an "Automated Music". Part I discussed background, context, and aesthetic issues. Part II was an elaboration on the various theories touched by the subject. Having discussed the various ramifications of the GENDYN phenomenon, we can now concentrate on analyzing the work of art as such. Where necessary, we shall draw upon the results of the two foregoing parts, without the need to digress too much into details.

I shall leave no doubt that I consider *GENDY3*, and not *S709* as being the piece of reference for the GENDYN program. An analysis of the latter piece, produced with an altered version of GENDYN three years later, by the end of 1994, is therefore not included in this study. *S709* consists of two sections with identical parameters (compared to 11 uniquely parametrized sections of *GENDY3*), uses only up to 10 simultaneous sounds, misses the secondary random walks which are a prerequisite for crystallizing musical structures within the chaotic sonic processes and, last but not least, it introduces an oscillation on the parameter values which does not function well in the piece. Knowing both pieces technically as well as musically, I consider *S709* inferior to *GENDY3*. It simply does not "sound". When looking at the sound signal generated by *S709*, one clearly sees that it is flattened by a large negative amplitude offset. This explains well the distorted and somewhat "choked" sonic ambiance of this piece.

Musical analysis of *GENDY3* will proceed in several distinct steps. First, the GENDYN program will be looked at under the aspect of its specific musical features: on the one hand, the features built into the algorithm of Dynamic Stochastic Synthesis, and on the other, the features added through Xenakis' specific computer implementation of that algorithm. We will see that some of the seeming idiosyncrasies of Xenakis' programming are in fact crucial to the musical quality of *GENDY3*. I would like to call this sort of analysis a "static analysis", because it can (if only theoretically) be performed by looking at the "static" text of the program listing without the need of running it.

The next analytical step will be what I would like to call "runtime analysis" of the GENDYN program. This means that in order to perform this kind of analysis, the GENDYN program has to be looked at during synthesis. This is the essence of what I mean by procedural analysis applied to algorithmic

composition: inspecting the various details and interdependencies on the simulated genesis of a musical artwork (see our discussion in part I, section 6.1.1 on page 95). Only by a runtime analysis can we discover the dynamic behavior of the computational processes which make a GENDYN composition so unique. We will see that a few undocumented, arbitrary-looking licenses in Xenakis' implementation reveal themselves to be fundamental to the musical quality of *GENDY3*.

The seemingly random details of the specific way Xenakis put his algorithm to work are a possible instance for speculations about the question of whether it was artistic craftsmanship, intuition or just plain luck that made Xenakis achieve such a powerful musical result out of his programming. Did Xenakis consciously know that the details of implementation were so decisive for the quality of his music or did he just have a good instinct taking the "right" decisions? Was it in the end only by incident that Xenakis was able to produce such a rich piece of music?

There are evidences that Xenakis designed the various pitch structures of *GENDY3* very carefully, quite contrary to the appearance of randomness and arbitrariness suggested by the intense use of stochastics. At the same time, Xenakis inevitably depended on the intrinsic specificities of Dynamic Stochastic Synthesis regarding the generation of pitch. In this part, I hope to shed some light onto the interdependency between Xenakis' artistic freedom in using his GENDYN program on the one hand and the specific requirements of the synthesis he had to meet and cope with on the other.

## Chapter 13

# Computer Music Analysis

The analysis of computer music is, in general, an extremely intricate and difficult matter. This is because virtually all tools of “classical music analysis” fail to meet the specificities of electroacoustic sound and computer generated structure. In the same sense that atonal music made classical harmonic analysis obsolete, and aleatoric composition the notion of classical counterpoint, electroacoustic music, in its turn, was to put an end to the very notion of musical tones and tonal systems. In a way, electroacoustic music is “atonal” music in a verbal sense! It is true that electroacoustic music can emulate tonal systems or create new ones, but electroacoustics in general are a means to transcend the very notion of fixed frequency and harmonic spectrum, and are therefore able to go beyond the notion of musical tones. For advanced electroacoustic composition which extends to the inner structure of sound, the notion of musical tones and their combinations must be replaced and generalized to the more abstract concept of sound in continuous spectral evolution.

### 13.1 Problems in Analyzing Computer Music

Let us first take a glance at the various difficulties one is confronted with when dealing with the analysis of a computer music piece. Some of these problems will be discussed in depth in the sections to follow. Then, we shall see how these problems apply to the special case of GENDYN, an instance of Radical Algorithmic Computer music (RAC), as defined in part I.

**The technology.** One obstacle in dealing with electroacoustic music is inherent in its nature: it is enabled and conditioned by technology and specific techniques of its handling in the production of a musical artwork. In general, an analyst, if not in close contact to the composer, cannot really know the exact working conditions that have governed the creation of a musical piece. He or she would need to have the same software at his/her disposition, the same working environment, the same choice of program settings as the composer, etc.

**The opaqueness of computer operation.** In general, when computing sound, the computer outputs the sound, but in general no further information on its computation. This is because a computer composer is

only interested in the musical output, and generally not in producing documenting information about it.

**The sheer amount of sonic data.** If the digital numbers defining *GENDY3* were to be printed on paper, the printing would reach from Middle Europe to the North Pole. In order to grasp the essence of a piece of electroacoustic music, one has to find telling abstractions from the raw sonic data, and, if possible, some key structuring the confusing topography of the electroacoustic landscape into a sensible pattern of explanation.

**The absence of a “musical text”.** Since the age of musical literacy, the idea that the essence of a musical artwork could be treated under the form of a written text document has conditioned musical analysis ever since. Yet the art of electroacoustic music is rather an “aural” than a written culture. Musical texts tracing the time structures of electroacoustic pieces are comparatively rare. The program text of an algorithmic composition does in general not represent the time structure of the generated music.

**Missing documentation of the tools involved.** Most often, the conditions under which an electroacoustic oeuvre has been produced are not documented. This is especially the case with custom hardware and software for which exist no manuals nor technical specifications.

**The strangeness of sounds and structures.** It is difficult to establish a framework of aesthetic common grounds and conventions when the artistic phenomenon is so hard to verbalize. This makes description, communication and interpretation of electroacoustic music very difficult.

## 13.2 The Absence of a “Musical Text”

It should be fairly clear that our classical discrete and symbolic music notation is hardly suited to model the continuous evolution of electroacoustic sound. We have to resort to other techniques of representation in order to get an analytical handle on electroacoustic music. As an example, complex sonic spectra can be represented with the help of Fourier Transformation techniques, yielding faithful and detailed sonagrams. Musical analysis may draw upon these graphs as a support for exact timing and differentiation of sonic characteristics (cf. [Bre94]). Yet the specific impression electroacoustic music exerts on a human listener can hardly be grasped by sonagrams. Only think of the huge problems in speech recognition, when trying to “read” a sonagram of a spoken sentence with all its transients, diphones and individual speaker characteristics. Even worse, try to imagine the voice of such a speaker drowned in the background noise of a cocktail party. Even specialists in the audio analysis field state the inadequacy of our technical tools to the problem of analyzing complex sonic events (cf. [Bre90]). The situation, one can imagine, is even worse with computer music. It is questionable if there can be any “objectifiable” means to get hold of essential aspects of computer music artifacts.

There have been efforts to narrow the gap between sonagramming and symbolic representation in order to yield a kind of “listening score” for electroacoustic music (compare the Acousmographie of INA/GRM). This technique can indeed serve as a good support for studying and communicating analyses of

electroacoustic music (see, e.g. [Del02]). It cannot, however, stand in for the highly subjective human perception of sonic phenomena. In general, electroacoustic music does not imply, nor is it reducible to a musical text. In other words, electroacoustic music cannot be studied in the reading room of a music library.

However, there have been attempts to create written accounts of electroacoustic works. They can be classed into three categories: realization scores, performance scores, and study scores.

### 13.2.1 Realization Score

A realization score is closest to the traditional function of a music notation: a symbolic text indicating, explicitly or by convention the actions that have to be taken by an interpreter or operator to produce the desired musical result. If the interpreter or his/her instruments are changed, the result will be different, even if the score is faithfully followed. A known example is the realization score of Stockhausen’s *Studie I* (1953). Yet, since the Elektronische Tonstudio des Westdeutschen Rundfunks of those days does not subsist anymore, various attempts of reproducing Stockhausens’s piece by realizing its score have proved unsuccessful (cf. [Sup97]).

### 13.2.2 Performance Score

A performance score is the record of performing actions: specific to a certain interpretation and hence less representative to the work as such, which is to be conceived as the sum of its possible plausible interpretations. A MIDI piano roll notation is a good example for a performance score: it records, among other parameters: pitch, duration, attack and sustain, i.e. the parameters of interpretation by means of a given music instrument (e.g. a keyboard). Such a performance score contains much more information for reproducing a given piece of music than a classical music score. For example, live performances by Liszt and Busoni can be rendered today more or less faithfully by means of the pianola recordings they had been making in their lifetime. Yet, even these realizations are heavily dependent on the specificities of the technical environment that is employed while recording and playing, e.g. the pianola, or a MIDI expander.

### 13.2.3 Study Score

Last but not least, the third kind of score to be discussed here is a score not for producing or performing but for pedagogical purposes. A study score is primarily aimed at explaining the music, rather than reproducing it. For example, different symbols in different colors may underline the existence of different structural elements which may go unnoticed when sounds get complex. Some of these elements may appear elsewhere in the piece and create references to the complex section which would be more difficult to be perceived without the graphical hint. The structures highlighted by a study score need not necessarily coincide with the structures the composer designed. A study score of an electroacoustic piece reflects a possible interpretation on the side of an (ideal) listener. A good (and rare) example is the score of Ligeti’s *Articulations* published by Schott [Lig70].

### 13.3 The Program Text as a “Potential Score”

“Computer Assisted Composition completely renews the idea of notation, which becomes dynamic, including the genetic mechanisms of the work. Scores become potential scores, and constitute in a way their own analytical descriptions, which is the seed for an upcoming outbreak in contemporary musicology.” [Ass98], p. 16.

The above statement, referring to Computer Assisted Composition (CAC) must be even more true when applied to Rigorous Algorithmic Composition (RAC, see definition 5 on page 30). In CAC, a score — we clearly see that the “score” in the above statement is a “realization score” of the composition process — contains algorithmic procedures, such as spectral analysis, calculation of rhythmic or harmonic patterns, or computed transformations, assembled according to the wits and talents of a composer. A CAC system keeping record of the compositional structures, events, and generative procedures of such a musical piece, then, is a description of that piece “in potentia” - the piece could also be different if parameters were changed or inputs replaced. Still, the rough design of the piece is evident from the graphical output of the CAC system - one of the goals of a CAC, much in the spirit of musical literacy, is symbolic music representation.

But if an entire music is created by a single run of a single algorithm, such as is the case with RAC, the potential score is the program text itself. We recall that RAC not only means algorithmic score composition but algorithmic *sound* composition, whereupon the music is defined up to the last bit of its information theoretic representation, according to the sampling theorem (as described in part II of this study). The realization score, i.e. the program, then, is no intermediate symbolic instance left to further interpretation by a human being but the object of a mechanical program execution. That is, the program’s directives are transformed into algorithmic action of the processor, unfolding the music and generating the sound in real time, on-the-fly.

The composition *GENDY3* can be regarded as a singular case where music composition, interpretation and musicological analysis converge, because the only analysis model capable of describing the whole of the music is the algorithm used by the composer himself (n.b. the algorithm, not the program text used by the composer!). “Interpretation” is to be understood in this context as the execution of machine code by the computer, i.e. interpretation in the (computer) scientific sense. In computer science, interpretation of program code is defined as an unequivocal mapping of code statements to machine action (i.e. there is no ambiguity or “interpretation” in the creative sense of the word). Much of the machine action induced can therefore be foretold by a “static” analysis of the code. However, some aspects of program execution can only be revealed through an actual run (or simulation) of the program. The former aspects are those “algorithmically reducible” to a static description, the latter are the “irreducible” ones that can be equated with “chaos” (i.e. high deterministic complexity) (cf. [Cas94], pp. 147-149). We shall have a closer look at both static analysis and runtime analysis in the next chapter of this part.

There is another, and as I find, even deeper meaning to the term “potential score” when RAC is combined with stochasticism (defined in part II), as in the case of Xenakis’ *GENDYN*. Werner Heisenberg used the term “potentia”, referring to Aristoteles, when speaking of quantum statistics as a “ten-

dency towards a specific action” [Hei79]. In GENDYN, this tendency can be regulated between perfect certainty (e.g. fixed pitches and frozen spectra) and almost complete “freedom” (Brownian noise). Both extremes are demonstrated in *GENDY3*: section 4 at 5’27” is a combination of 5 noisy tracks (see figure 6), whereas the last section of *GENDY3*, section 11, is just one big cluster of 15 fixed pitches, perforated by stochastically distributed pauses.

Composing a computer program is different from composing a piece of music. RAC means devising a computational procedure in a way that it may, when executed by a machine processor, result in a desirable musical output. The computer program “is” not the music but just a description of the generating procedure. One can say that the algorithm contains the music “in potentia”. This is not very different from a musical score which is not the music either but only a code of symbols and conventions which work in reciprocal action with a human’s cultural background and musical skills in order to be turned into music. (This human interpreter need not be a conductor, it can also be a musician silently reading the score.) The difference is that the human interpreter can be creative in re-imagining the intended piece while the computer processor follows an interpretation schema laid out beforehand at design time of the programming language used.

To resume, we assert here that the computer program of a RAC piece be equivalent to a music (realization) score. Just as a score is interpreted by a human musician, a computer program is interpreted by a machine processor (the computer hardware, the operating system and the runtime system of program execution) in order to produce the sound of the music. The difference between the human interpreter and the machine is the nature of interpretation. The interpretation of program code by a computer is algorithmically controlled, following the model of the mathematical interpretation of a formal system: the isomorphic mapping of syntactic structures to meaning in the real world (see part II: Theory) which is in our context the operational behavior of a computing processor. The human interpretation of a musical score, in contrast, is a creative process in view of an artistic goal, on the background of historic knowledge, social context, human emotion and communication needs. We have argued in part I that machine activity and human activity are necessarily different, even if the end product could be judged by a human as being equivalent. As a matter of fact, we have seen that contemplating a work of art is a creative act by itself, and that it entails the possibility of projecting this very creativity back onto the work of art, as if this creativity was “right in there”.

To conclude, the algorithmic structure of a program text can indeed be considered an objective substrate for the musical artifact produced with its help. With rigorous algorithmic computer music in the conceptual sense as practised by Xenakis, the computer program, together with the laws of acoustics and sound information theory, can be regarded a possible theory for the entire sonic phenomenon of the musical piece. In other words, the music, in its physical shape, can ultimately be reduced to a few hundred lines of programming code, in the sense of Algorithmic Information Theory (cf. part II: Theory). Rigorous algorithmic computer music therefore urges a new analytical approach exploiting the same computer as a tool for analysis that has served for the same music’s synthesis. This approach has been called for by some proponents of computer music (section 6.1.1 on page 95). It seems even more appropriate to the situation where the relationship between generating program and musical artifact is the

closest possible, as in the case of Rigorous Algorithmic Computer Music.

In order for a program text to become functional for the purposes of music analysis, it is necessary that one can “read” it. We will discuss the problems inherent in this statement in the following sections.

### 13.4 Problems Analyzing Algorithmic Computer Music

Even under the lucky circumstances of Rigorous Algorithmic Computer music described above, computer music analysis has still a number of obstacles to face. We have stated that the computer program can be equated to a (mechanical) score. However, such a kind of score is difficult to read. This is not only because the computer idiom of that score is hard to understand for a human but also because comprehension of the score would mean to simulate the algorithmic activity described by the program. We know that such an algorithmic action can be very complicated and far-reaching – depending on the complexity of the algorithm. Algorithms can be so complex that they become practically intractable – a computer would have to run longer than our universe is estimated to exist. There is even no way to decide by computation, at least in general, whether a given algorithm will yield a result at all, or simply go on running forever.

Many aspects of algorithms can be clearly formulated but are computationally undecidable. The question if a given algorithm will ever halt, if two algorithms are equivalent, or even if a program fulfills a given specification or not, are computationally undecidable ([GL88], pp. 82-89). These incompleteness features are the reverse side of the power of universal computation. Now, if the potential score is nothing else than a computer program, it might be very hard to understand its action, due to the kind of computational complexity it may imply.

We have seen in part II of this study that GENDYN belongs to complexity class three out of four, following the systematization of computational complexity classes established by [Wol94]. The deterministic pseudo-randomness of GENDYN is the way to algorithmically implement stochasticity. This is the reason why GENDYN can be simulated at all (exactly same starting conditions lead to same results). At the same time, it is also the reason why GENDYN is hard to simulate: slightly diverging starting conditions lead to dramatically different results (“weak causality”). GENDYN may not be reproducible on another computer architecture than the PC or with a different mathematical library.

We cannot study a GENDYN composition in all of its potential ramifications because its computation is chaotic and rapidly spreading all over computation “phase space”. Such a system cannot be reduced by a theory less complex than the system itself. But we can study the system by observing it under simulation, extracting some aspects and tendencies which characterize its dynamics and its overall behavior. For, even if the exact computational behavior is chaotic, for musical purposes, a mere tendency (“potentia”) proves to be sufficient, because musical aspects as timbre, loudness and other musical features are by nature statistical phenomena formed by masses of sonic data. We can recognize families of characteristic GENDYN timbres and link them to specific configurations of parameter settings, even if the absolute sequences of numeric samples forming

these sounds seem random.

In contrast to such statistical qualities as timbre, the simulation, and therefore the analysis of GENDYN's pitch movement proved to be much more difficult. Pitch is, as we have seen in part II, an emergent by-product of stochastic sound synthesis. For the study of pitch in GENDYN, we need to produce identical GENDYN output – at least identical for the ear's capacities of pitch recognition. As we have seen, GENDYN pitch depends on the cumulative processes of 2nd order random walks along with barrier reflections, which is a heavily chaotic process. In addition, the slightest change in pitch would change the order of concurrent drawings by different tracks from a shared random number sequence, i.e. pitch deviations auto-amplify themselves even further.

To resume, there are a number of obstacles to face when analyzing computer music. Some of them can be tackled by our procedural analysis approach to RAC music. Let us have a second look at the problems raised at the beginning of this chapter and relate them to the procedural approach of analyzing a GENDYN composition.

**The technology.** In the case of GENDYN, all technological knowledge is incorporated into a computer program which was made available by the composer. Therefore, we are in a lucky position to be able to reconstruct the “laboratory” of Xenakis' last electroacoustic project.

**The opaqueness of computer operation.** Because of the fulfillment of the above requirement, documenting material can be produced by plugging into Xenakis' laboratory documenting tools like visual animations, graph plottings, and auxiliary programs that compute e.g. all possible pitch scales that are available to GENDYN synthesis. This makes the algorithmic processes much easier to understand.

**The sheer amount of sound data.** Fortunately, there is a category that explains much of the sonic world of GENDYN: the organization of frequency in time, a category that controls stationary pedal tones, glissandi textures and even perfect noise. It is directly related to GENDYN's prominent algorithmic feature: the (stochastic) Frequency Modulation of a wave form by random walk processes.

**The absence of a “musical text”.** Fortunately, Xenakis makes it comparatively easy to us getting a grip on the time structure of *GENDY3*. Its clear sectioning into 11 parts, the more or less obvious distinction of a number of layers of sound, and the (re-) appearance of characteristic sonic phenomena like buzzes, glissandi, drones and noises allows us to get a clear picture of the piece, and to find telling descriptions of its musical aspects. Because the program is there, the timing of the piece can be tracked with a precision of a sample tick — all durations within the piece can be pin-pointed with an accuracy of microseconds.

**Missing documentation of the tools involved.** Xenakis used only standard technology: IBM Personal Computer, BASIC programming language, 16 bit D/A conversion. Any PC in the world can do the same.

**The strangeness of sounds and structures.** Unfortunately, this is what this study has to struggle with, like all other studies in electroacoustic music.

We see here that some of the problems in analyzing computer music can be tackled more easily in the case of RAC. For example, the technology of the standard PC and its software development tools are largely known. The opaqueness of computer action could be enlightened by adding graphical animations and plottings to the computational processes. The amount of sound data could be handled by finding a graphical representation, the pitch-time graphs, which preserved essential features of the sonic phenomena while abstracting from the transient, accidental ones.

A key to understanding the different kinds of GENDYN sounds is (a generalized notion of) pitch movement. (For a detailed discussion, see sections 8.6.1 on page 124 and 14.4 on page 214.) Now, there is no need to try to get this pitch information from the *GENDY3* recording. It would not be very reliable to gain this information from sonagramming, given the complexity and unstableness of GENDYN sound spectra. We can get the pitch information much easier and more accurately from the program itself. It suffices to identify the variables within the program which correspond to the length of the wave form. We then obtain detailed data of (an extended notion of) the whole polyphony of pitch movement in *GENDY3*, at the “source” of sound creation. This saves us browsing through a plethora of sonic data within which this information is buried and neutralized to an extent that makes it practically impossible to be retrieved from the musical end-product.

The pitch-time plottings made by the New GENDYN Program are in fact nothing else than a study score concerning the pitch/duration aspect of the processor’s interpretation of the realization score, the program. Please note that different processors may yield different study scores out of the same realization score (=program). The complete GENDYN pitch/duration study score, with its information regarding the pitch structure of the music, is included here in this study (see 15.1 on page 277 and the following pages). It can be compared to Xenakis’ own hand-drawn pitch/time diagrams on ruled paper in some of his instrumental pieces or his UPIC pages (which are, in that case, not study scores but realization scores). In the *GENDY3* study score, musical phenomena are clearly represented such as the spacing of clustered tones, the long-term contour of glissando evolutions or the rough shape of pedal tones oscillating over a small pitch interval, informations which cannot be gained from the GENDYN realization score (i.e. the program) because its interpretation is so complex that it must be left to a machine processor.

Yet nothing can really alleviate the struggle to communicate the specific musical sensation that is created by an audition of *GENDY3*. Here, we have to take recourse to language and metaphor, hoping to be able to convey a sense of the quality of a possible listening experience to this kind of computed sound (see chapter 15 on page 261).

### 13.5 Analysis through Re-Synthesis

A naive view on the business of composers and musicologists could be that musicologists analyze musical works which have been synthesized by composers. In this section I try to argue that the dividing line between musical analysis and synthesis is not so clear-cut as it may seem at first glance. In fact, analysis and synthesis are complementary procedures both within the field of music pro-

duction and within the field of music analysis. In the field of music production, composition is often taught by examples, i.e. by looking for principles of composition in model works. Composers use the experience and knowledge gained in previous compositions on the conception of new ones. In electroacoustic and computer music, the complementarity of analysis and synthesis is even more obvious. Sound synthesis procedures are based on a thorough analysis of either acoustical, physical, mathematical (or in the case of non-standard synthesis: conceptual) models.

The notion of music “synthesis” can be extended to the generation of musical structure above the timbral level, i.e. entities represented symbolically in traditional score notation. In this case, too, synthesis may be influenced by a foregoing or complementary aspect of “analysis”. There have always been many attempts and research activities concentrated on synthesizing harmony, counterpoint and melody, often departing from a close analysis of those structures in existing music. The algorithmic composition of music scores is often directly or indirectly driven by an exhaustive machine analysis of existing music or musical structures, from Hiller’s early statistical experiments in grammar-based composition up to the training of “composing” neuronal nets.

Transformational approaches naturally combine an analytical and a synthetic stage (cf. Analysis-Synthesis Systems like Audiosculpt, FFT and FFT-1, or Wavelet synthesis). Dedicated CAC systems support an interactive, prototyping way of working even on a symbolic, i.e. rather high and abstract level. Virtually any composer working in the field of electroacoustic music studies the analysis of natural sound, given compositional models, existing styles and/or “grammars” of musical production.

Conversely, a similar symbiosis between analytic and synthetic procedures exists in the field of music analysis. Music analysis is more than mere book-keeping: the endeavor to reveal compositional thought before the background of compositional problems and possible alternate solutions. Therefore, music synthesis could and should play a larger role in the analysis business in general, and in the analysis of computer music in particular. If we speak of analysis, we have to consider the fact that musical analysis is not confined to the pure dissection and distillation of musical data. In understanding a musical artwork, a process of re-creation is necessarily involved, the construction of a theory which explains the “observed data” of the musical artifact. It is less important that the theory constructed matches exactly the composer’s theory (if there was any) but that this theory be coherent and plausible and contribute to a deeper understanding, an intensification of the artistic experience, and possibly enhanced effect of the work of art onto the contemporary world or even onto posterity. The Italian composer Marco Stroppa has formulated this point in a seminal article about the analysis of electroacoustic music:

“What does it mean to analyze a piece? In fact the term itself (from the Greek *αναλινω*, to dissolve) is only part of the story. [...] The most important and the most creative of challenges [is the] logical and coherent synthesis [...] to reconstruct a unified totality, which may or may not be the same as the one at the outset.” [Str84], here p. 176/177]

To conclude, analysts of computer music find themselves confronted with an acoustic product whose creation is most often opaque to them. Rarely is there

a documentation of the creation process comparable in expressiveness with a traditional musical score. Most of the time, the analyst must try to gain as much information and insight from the acoustic end product.

In the rest of this chapter, I will argue that music analysis should more draw upon procedures and techniques of (re-)synthesis of the musical fabric in order to get better results in its analysis of even any results at all. This is because in many aspects, the process of composition is irreversible; it cannot be traced back by studying the musical artifact but only by following the process of creation i.e. to try to repeat (parts of) the synthesis process.

## 13.6 Reconstruction of the Composition Environment

The resynthesis approach described above is crucial to a detailed analysis of algorithmic computer music. So far, it was hardly possible to separate the many layers of complexity that sum up to a computer music piece distributed as a sound recording on tape or on CD. But if one succeeds in regenerating the individual layers of complexity with the help of the algorithmic formula, one can inspect them in detail and see how they combine together to form the whole of the music. To put it in computer science jargon: the resynthesis approach is a “bottom-up” procedure reproducing the details of a composition that are impossible to obtain “top-down” from the recording. In the case of *GENDY3*, the layers of complexity can be identified with the (up to) 16 parallel “tracks” of complex sound produced by the Dynamic Stochastic Synthesis algorithm.

Detailed analysis becomes then possible through explicit simulation of the algorithmic generation process that led to the original composition. As a consequence, the same music can be computed as many time as wished, comparable to a mathematical formula which always yields the same result.

In order to exploit this reproducibility, however, the exact algorithm and its original input data used by the composer have to be known. This was achieved, in the framework of this study, by a close analysis of the composer’s original working environment. It was found that the program that Xenakis used for the generation of *GENDY3* represents a possible implementation of Dynamic Stochastic Synthesis, albeit a rather obfuscated one.

The idea of synthesis for analysis purpose, striving to cover as much of the aspects of the original work as possible, is called by Stroppa “archaeological reconstruction of the choices and conditions of work of the composer”. It is exactly this notion of “archaeological reconstruction” that has been the first and hardest part of the GENDYN project. The resynthesis of *GENDY3*, a major work of art of Iannis Xenakis, serves as a practical proof for the validity of an analysis through programming, of implementing a possible analytic theory (in the sense of algorithmic complexity as discussed in part II) on the genesis of *GENDY3* by devising a different, more concise (and more efficient) algorithm. This undertaking can be viewed as a kind of “computer archaeology”, where one is left with a few traces of a computing process that happened in the past, wishing to restore the conditions to let the same process happen at present and in the future.

## 13.7 Related Work

Resynthesis of composed musical works from the analysis of their lawful structure has been used as a powerful testbed for the strength of analytical models by e.g. [RM88], [RM89], [Kin91], [Che95]. However, so far, these attempts of reconstruction did not yet extend to timbral design. Indeed, the sonification of the reconstruction models for demonstration purposes was achieved with the help of standard MIDI expanders [Kin91] or custom software [Che95]. A reconstruction of Xenakis' *GENDY3*, however, has to include the algorithmic design of sound. Every bit of its sample representation must be matched by the resynthesis process.

## 13.8 The Algorithm as a Composition Theory

Similar to Solomonoff's model of scientific induction, where scientific theories are understood to be the most concise algorithmic reduction of observation data ([Cha75]), music analysis is generally interested in compositional theories underlying the score data, with or without the ambition to "reverse-engineer" the composition process (cf. [Lév03], p.16). In the case of electroacoustic music, the scope of analysis is broadened to encompass the wealth of sonic data.

In this situation, RAC offers a lucky chance. Since the entire musical phenomenon is completely determined by a single algorithm, the conceivably best theory of composition is the very algorithm itself. Therefore, we do not have to look far for a theory of *GENDY3*. It suffices to get hold of Xenakis' program. The analyst can then proceed to make this program the subject of a further analysis of "second degree". For example, one can establish a meta-theory of the algorithmic composition process by analyzing the ideas and the conceptual models behind the algorithm. In the framework of this study, this meta-theoretical reflections have been concentrated in parts I and II. In this part, we can therefore devote our attention to phenomena which are the realization of these concepts in music.

Even more, once we have understood the theory of the program, we can explore its implications in compositional practice and try to understand the role of the composer. For example, we can test the musical capacities of *GENDYN* by exploring different parameter settings than the composer. In this way, we can see if the result obtained by the composer can be said to be a rather typical or rather accidental output of the program. In other words, was the compositional outcome predictable and the result of systematic labor or was it just plain luck?

In *GENDY3*, there are examples for both. The configurations of sounds with stable pitch is the result of extended and systematic work, as is the choice of timbre, while glissandi contours and stepwise pitch movement have to be "lucky strikes". For example, the expressive glissando arc which opens the composition of *GENDY3* (see figure 15.1 on page 277) is extremely atypical for *GENDYN*, since normally, glissandi tend to oscillate back and forth around an equilibrium position of medium pitch.

However, the definitions of "typical" and "accidental" are dependent on the findings of analysis. For example, the intervals of stepwise pitch movement were "accidental" for Xenakis. A sketch by Xenakis shows that he had only a very approximative notion of the interval structure of moving pitch. (see "desc." and

“aigu” in figure 14.25 on page 251). In contrast, these interval structures are now “typical” for us since we know their underlying law (see below). We could even use them by design in a composition, while Xenakis was in a situation of either having to approve or to reject their accidental appearance in his music.

## Chapter 14

# Analysis of *GENDY3*

The analysis of the composition process that led to the music of *GENDY3* is what could be termed an undertaking in “Reverse Engineering”. While the standard process of software engineering starts with a detailed description of the task to fulfill, then a more or less formal specification of the software system, and last but not least, its implementation in a chosen runtime environment, the analysis of *GENDY3* had to start, in reverse order, with the program that computed *GENDY3* and work the way back to reconstructing its underlying specification and conceptual framework. Even worse, when starting the analysis project, the program which had computed *GENDY3* was not there anymore.

The few pages of program text reprinted in [Xen92b] do not belong to the program which computed *GENDY3*. Even worse, they represent but a small part of the program. The original program would have filled the entire book. Luckily, I was able to excavate and identify the original program of *GENDY3* at CEMAMu in 1995. It comprises some 10,000 lines of program code. Polemically speaking, the first task in the analysis of *GENDY3* was to make sure that this program dealt with Dynamic Stochastic Synthesis at all. The second task was to make sure that this was the program which actually computed *GENDY3* back in 1991.

As it turned out after a close study, the *GENDY3* program of Xenakis’ can be regarded as a possible implementation of an algorithm very close to the one described in [Xen92b]. The differences to the algorithm presented in Xenakis’ book, and the specific way this different algorithm was implemented by Xenakis is partly responsible for the specific musical and sonic quality of *GENDY3*. A detailed description of the reconstructed *GENDY3* algorithm along with a formal (functional) specification has been given in part II of this study. In this part, we will be interested in the musical consequences of the specific way Xenakis realized his Idea of Dynamic Stochastic Synthesis.

As an answer to the second question, if the program text found at CEMAMu belonged to the program that computed *GENDY3* back in 1991, there are two evidences: first the music computed with the found program sounds exact the same way as *GENDY3* released by Xenakis in 1991. Second, Xenakis annotated the printout of the program in a way which connects it unmistakably to the premiere of *GENDY3* in 1991 at the *Rencontres internationales de la musique contemporaine* in Metz, France.

As a matter of fact, the computation of *GENDY3* proceeded in two distinct

steps: first the global and local duration structure of the piece (its “architecture” or form) was computed. This task was implemented by a set of auxiliary programs called “PARAG001” through “PARAG011” for each of the 11 sequences of *GENDY3*. These programs also had the function to define the sets of sound synthesis parameters for each sequence, as well as the order of the sequences in time. All this information was hard-coded into the PARAG programs. Each time Xenakis wanted to try other values for his parameters, he had to edit the PARAG programs and execute each of them. The PARAG programs would then write all informations to a bunch of disk files. Then, Xenakis would start the GENDYN synthesis program which would read the files from disk and compute the sound according to the precomputed architecture and the synthesis parameters given by the PARAG programs.

The analysis will proceed in two steps. Much about the composition process and henceforth the music can be said even before starting the program, just by studying its logical structure. This has become easier through the New GENDYN Program: it is designed to faithfully reproduce the composer’s workspace as far as it could be reconstructed from the study of the programs PARAG and GENDYN. In other words: where PARAG offers a possibility to type a numeric value into the program text, I created a knob in the user interface of the New GENDYN Program. Therefore, by looking at the graphical user interface of the New GENDYN Program with the parameters of *GENDY3* loaded, one sees the internal structure of the piece, from the global sections and their timing down to the settings of the individual sounds. This is what I call “static analysis”, i.e. the demonstration of all static properties of *GENDY3* before creating any sound.

In the second step, I shall describe the properties of *GENDY3* that become only evident when observing the generative procedures of the piece on starting the synthesis. In order to do so, I have added animations to the graphical interface which display snapshots of various aspects of the synthesis: the drawing of random numbers, their distribution over a numeric interval, primary and secondary random walks, and the resulting wave form polygon. By looking at these graphical animations, I realized why it was possible that there were steps within the glissando movement of pitches, that these steps were fixed and eventually, that their scales were governed by a combinatorial law. In other words, having understood how *GENDY3* works, we can now foretell how GENDYN music works in general. We are now in a situation where a composer could set up a plan of pitch and scale configurations beforehand and realize a GENDYN piece based on this plan.

But let us first follow the course of events and finish the story of the musico-archaeological excavation campaign of *GENDY3*.

## 14.1 The *GENDY3* Excavation Campaign

As soon as it was clear that the key to *GENDY3* was indeed the version of the GENDYN program found at CEMAMu, the next question was in how far and in what aspects the specificities of Xenakis’ programming was responsible for the musical results. At the beginning of my research, I had started with my own implementation of Dynamic Stochastic Synthesis, so I knew how different the output of a program could sound from *GENDY3* even if every detail of the

GENDYN algorithm, as described by Xenakis, was realized in such a program.

In the course of the project, it turned out that the actual music produced by the GENDYN program was extremely sensitive to the tiniest detail of implementation, down to the twists of floating point arithmetics within the mathematical libraries. Fortunately, Xenakis had produced his music on a PC running a BASIC programming environment under DOS made by the same well-known software company that also manufactured the C++ programming environment I used for the New GENDYN program. Seemingly, at this time, both environments shared the same mathematical runtime libraries. Otherwise the exact reconstruction of the GENDYN music would probably not have been possible.

On the other hand, in Xenakis' program, there were things added to the algorithm that were nowhere else documented than in the program code itself. In fact, one cannot expect that anyone would have taken the effort to document such details because they do not at all apply to the principles of Stochastic Synthesis. Yet they do influence the music, and therefore they had to be tracked down, emulated and incorporated into the New GENDYN program.

It soon became clear that over the course of his experiments with Dynamic Stochastic Synthesis, Xenakis had constantly evolved and rewritten his program. The program I found on his computer was not the same one with which *GENDY3* had been computed in November 1991 but a later, quite different version, similar to the one which must have computed *S709* in December 1994. Similar, because Xenakis might have even touched it afterward. CEMAMu staff helped me to search backup tapes but we could not find a backup of the 1991 version of the program.

Fortunately, in a paper folder, Xenakis had collected printed listings of different versions of his program, and one of these listings was marked "S403", stating that this version of GENDYN had generated output sound file no. 403. Another printout showed a graphical plot of the time structure of *GENDY3*, bearing a handwritten note "S403 = S402 stéréo", and on the same sheet: "données du 5-11-91 pour GENDY3 de METZ", the location and date of the premiere. I could then be sure to have found the correct version of the program. I then compared the printed program text, line for line, with the GENDYN version I had copied from his computer. Some of them were different, and I changed these back to the *GENDY3* original.

The systematic numbering of the sound files suggests that Xenakis had created and dismissed 401 trial versions before he premiered *GENDY3*, and another 306 before he released his second and last piece with GENDYN, *S709*. I did not investigate further into this matter, trying to find out if these 700 or so sound files were complete pieces or just chunks of sound. I felt there was enough work waiting for me with what I had found. The reader is encouraged to make good for this default.

In the following sections, the issues of analysis will be presented not in the archaeological order of the excavations but in a systematic order. Therefore we will start in section 14.6 on page 216 with a description of the "ideal" algorithm of Dynamic Stochastic Synthesis such as it can be distilled from a detailed analysis of Xenakis' program text. Next, the specific way of implementation Xenakis chose will be discussed in section 14.9 on page 223. Then, in section 14.10 on page 224, various features and licenses willingly or unwillingly added by Xenakis will be considered along with their impact on the musical result. Finally, in section 14.12 on page 234, the analysis of the program will be concluded by a

description of its dynamic “runtime” properties which are so crucial for timbre and sonic evolution of GENDYN sound in general and for the music of *GENDY3* in particular.

## 14.2 Restoration and Conservation of *GENDY3*

Xenakis’ program is certainly not the most concise way of implementing the GENDYN algorithm, nor is it the most effective. The program’s operation is cumbersome and opaque. There is a series of auxiliary programs called PARAG, one for each GENDYN track, whose program text Xenakis had to edit in order to change synthesis parameters. He then had these programs write his parameter values into binary files for the GENDYN program to read. After starting the synthesis with GENDYN he had to wait days for the result while seeing nothing of what the program did except a plotting of the raw sound signal. Another auxiliary program was to read the time structure of the sequences out of the binary files and to print a plot of the durations for all the sequences of the piece. This plot was found in the same folder where Xenakis had kept the program listings of the many versions of GENDYN and some more notes and annotations.

One of these is a sketch by Xenakis’ hand which gives evidence that the composer thoroughly investigated into the interdependencies between parameter settings and resulting (fixed) pitch. He seemed to have synthesized the 16 tracks one per one (see the sound file enumeration in the leftmost column of figure 14.25 on page 251), keeping track of their parameters (3rd and 4th column) and he wrote down their pitch (curiously about a semitone below chamber tuning – what was his reference instrument? There was no piano at CEMAMu). At the same time, the tools at Xenakis’ disposal did not permit a deeper and more systematic study of moving GENDYN pitches. It would therefore have been impractical, if not prohibiting to use Xenakis’ own program for RAC analysis. This is the motivation for the GENDYN Project.

The goal of the GENDYN Project is to document and elaborate upon the unique GENDYN concept of Automated Music conceived of by Xenakis. The GENDYN project aims at enhancing GENDYN with state-of-the-art computer technology in order to keep this unique RAC approach operational for both analytic and creative purposes. To begin this, during my research stay at CEMAMu in 1995/96, I implemented the GENDYN algorithm as an interactive sound synthesis tool with a graphical interface, which I called, for want of a better name, the New GENDYN Program [Hof00]. The idea was to turn Xenakis’ “black box” into a realtime composition tool with immediate sound feedback and visual control over the operation of the algorithm and its parameters. In order to prove that the New GENDYN Program was a valid implementation of Xenakis’ algorithm, I made it re-synthesize *GENDY3*. This turned out to be much more difficult than I had thought when starting the project. As a result, with the New GENDYN Program it was now possible to reproduce the artistic creation of *GENDY3* and analyze this process in detail.

One can regard the New GENDYN Program as a documentation of both the GENDYN algorithm and the specific way Xenakis implemented it. It is important to note that concerning the creative goals, the new program is in no way superior to the original one. However, it is much more practical to use. With the New GENDYN Program, it is possible to monitor the compu-

tational composition process by graphic display of the algorithm's actions: the drawing of random numbers, their specific distribution over a given range, the random walks driven by those distributions, and the resulting wave form. It is also possible to alter the input parameter configurations while listening to the sound output. This "real-time" control makes the analysis work much easier and encourages experimenting with alternate settings. One can isolate superposed layers of sonic evolution, and prepare a graphic "score" representation of the music as a collection of simultaneous pitch curves over time. In addition, the real time interaction with the new implementation program helps to develop a kind of intuition for the algorithmic composition process, its strengths and its limitations, and to explore the "decision space" within which the composer navigated and found his specific, individual path, leading to a major work of computer art like *GENDY3*.

This dissertation study is an important part of the GENDYN Project. But the GENDYN Project should not end with the dissertation. Since its beginning in 1995, results have been presented at various conferences in France, Germany, and elsewhere. Some interested composers have tried out the New GENDYN Program at Ateliers UPIC in Alfortville near Paris (later CCMIX in Romainville) and will continue to do so. But only Australian composer Paul Doornbusch deliberately used it the "pure" way Xenakis did, by having it synthesize a whole piece. He called it, in reverence to *GENDY3*, *G4* (released on [Doo02]).

The GENDYN Project should continue, in the hand of more people interested in the legacy of Iannis Xenakis. And, certainly, GENDYN should produce more music, possibly much different from what Xenakis experienced himself. The New GENDYN Program will be freely available to musicians and scholars. Others may elaborate on it or integrate Dynamic Stochastic Synthesis into their composition environments. There are also some ideas to develop DSS further which are, of course, beyond the scope of this study.

### 14.3 The New GENDYN Program as a Tool for Analysis

Finding a working model or algorithm for the (re)construction of a musical work is not an end in itself but rather the starting point for the question why it has been made the way it has. One way to approach this question is to alter the model in order to explore possible alternatives to the solutions chosen by the composer, and to listen to the aural results. This is because the choices and "style" of a composer may not only be found in the specific way he or she set up his/her compositional procedure (i.e. the program) but also in the way he/she actually employed it for composition.

In this context, it can be observed that the GENDYN implementation of Dynamic Stochastic Synthesis is a faithful reflection of Xenakis' fundamental concept of separating the process of composition into two stages: composition outside-time and composition in-time (cf. [Xen92b]). The GENDYN algorithm defines a number of sections (called "sequences" by Xenakis) containing a collection of sound tracks each being characterized by a specific set of sound synthesis parameters. That is why the sections of *GENDY3* sound differently: the synthe-

sis parameters assigned to their tracks are chosen different. (This is not the case for *S709* composed three years later in 1994: the set of synthesis parameters does not change between the two sections). Each section has an identification number (called  $\psi$  in Xenakis' program), reaching from 1 to 11 (i.e. there are 11 distinct sections in the piece). The identification number reflects the order in which the sequences have been created ("outside time") by the composer, much before he decided upon where to place them in temporal order within the piece.

In *GENDY3*, the 11 sequences are played in the order 10, 1, 2, 9, 3, 8, 4, 5, 7, 11, 6. When Xenakis played a prototype version of *GENDY3* at ICMC Montreal in September 1991, the number and order of the sequences was a different one: 1, 2, 3, 7, 4, 5, 8, 6. Obviously, Xenakis used the time between the conference and the Metz premiere to compose three more sequences 9, 10 and 11. Fortunately, as it turned out, because sequence 9 is the "apotheosis of noise", unparalleled in his entire oeuvre (except, maybe, the infernal noise generated by activating virtually all of a church organ's pipes toward the end of his unique organ piece *Gmeeorgh*). Xenakis interpolated his "apotheosis of noise" in between sequences 2 and 3 which share a number of similar sonorities. Obviously, the interpolation of the noise serves to break up this sonic continuity.

Interestingly enough, despite the fact that duration structure and sounds of both versions are the same, the prototype version of *GENDY3* sounds completely different from the Metz version published on CD. This is because a simple re-ordering of sequences already changes the order of drawing random numbers from the random number generator, which in turn changes the starting conditions of the random walks which changes everything — at least as pitch is concerned. The timbres, in contrast, are more or less preserved since they do not depend on the specific course of the random walks. They rather depend on the parameter settings which Xenakis does not seem to have changed much between September and November 1991.

The ordering in time is a compositional decision that comes after the definition of the sonic characteristics of each of the sections. This is represented algorithmically by a function that assigns successive "in-time" numbers (called  $\phi$  in Xenakis' program) to the "outside time" numbers  $\psi$ , i.e. 1 is mapped to 10, 2 to 1, etc. This "reordering" in time (which in theory would also permit repetition) of larger musical building bricks is a typical feature of many (if not most) of Xenakis' compositions. One may call it a "collage" technique. But why did the composer choose this specific ordering of the sequences in the published version of *GENDY3*? Without going into too much detail, it can be said that there is a certain dramaturgy involved in the succession of the "sonic characters" of each sequence. For example, the "apotheosis of noise" seems to have its ideal place near the end of the first third of the piece. Also, a certain effect of "finale" can be stated for the concluding sequences, with particularly brilliant and saturated sounds that somehow evoke a sensation of an organ's "plein jeu".

## 14.4 Creating a *GENDY3* Study Score

In general, algorithmic non-standard composition and synthesis programs neither take as input nor output any "score" information that may be associated with the sound data produced. In general, the algorithmic composer is more

interested in the sound itself and its temporal evolution than in a symbolical representation of the sound events. In the case of the original GENDYN program, there existed a separate tool which pictured the “patchwork” of sound durations controlling the activation/deactivation of the up to 16 simultaneous synthesis tracks over the course of the GENDYN piece. (An excerpt is included in [Xen92b]). These graphics, in representing only the duration structure of the music, conveyed only a very restricted aspect of the piece, given the richness and complexity of GENDYN sound. This sonic aspect – which is a primary one in Dynamic Stochastic Synthesis – stayed hidden to the “reader” of such a “score”.

The resynthesis approach discussed above, in contrast, is able to add more information to such a graphical representation. It is possible, for example, to graph the dynamic evolution of fundamental frequency, which is the key to both the pitch macrostructure and the sonic microstructure of *GENDY3* together with the durations into a sort of UPIC-like “pitch-arc” score. Think of it as Xenakis’ duration score with its straight horizontal lines changed into curves to follow the pitch evolution of the sounds, from delicate curves to scattered patterns, depending on the dynamics of pitch movement (see e.g. figures 15.1 on page 277 and 15.8 on page 283).

As has been already stated, much of GENDYN sound is determined by frequency modulation in its broadest sense: the movement of frequency in time. Therefore, plotting the time-frequency graphs of every track of *GENDY3* can somehow take on the quality of a study score. This score can then serve as an orientation to further analysis. In addition to the duration/frequency structure of the piece, it also depicts information about the segmentation (horizontal architecture) and layering (vertical architecture) of the music.

With the exact timings of the duration structure, the analysis of exact pitch and the study score, it would be in principle possible to create a music score in traditional symbolic notation — but with very complicated a-metric patterns. But the question is: who would be willing to read such a score and to what end? For the purpose of this study, the UPIC-like plot of duration and pitch should suffice, and “classical” notation only be consulted to display scales of pitch when they underly the movement of pitch (see e.g. 15.12 on page 287). The reader, then, is invited to match the scale structure against the pitch curves of the tracks where this scale is said to occur.

All pitch/duration plottings in this study are generated with the New GENDYN program. They are part of a complete pitch/duration plotting of the entire piece of *GENDY3* (11 sections each comprising up to 16 layers of sound). Their duration structure is identical to Xenakis’ own plots but with pitch information added in the vertical dimension. This pitch information is a trace of the algorithmic activity of *GENDY3* concerning frequency modulation. More specifically, it is proportional to the logarithm of the length of the current wave form period according to the formula:

$$\text{pitch} \approx \log_2 \left( \frac{f_s}{n} \right) \quad (14.1)$$

where  $f_s$  is the sampling frequency (44100 in the case of *GENDY3*), and  $n$  is the length of the wave form in sample ticks.

## 14.5 Creating *GENDY3* Scale Notation Examples

The study of *GENDY3* with the help of the New GENDYN Program also revealed the lawful (self-)organization of scale patterns which emerge for some sound synthesis configurations of GENDYN. I found it appropriate to notate these as scales in classical notation, extended by microtonal annotations in Cent (because GENDYN pitch does not fit into the tempered system). Based on the observations made with the help of the graphical animations in the user interface of the New GENDYN Program, a formula could be established which holds to enumerate all possible (i.e. “potential”) GENDYN scales for a given sampling frequency. This formula was implemented in an auxiliary analysis program which generates Music $\TeX$  notation output suitable for the  $\TeX$  compiler to turn into music examples (see e.g. figure 15.12 on page 287).

That is, a theory of all possible GENDYN scales can be established and matched against the scales used in *GENDY3*. This is comparable to matching Xenakis’ famous sieve theory against the scales he used in his pieces since the 1960’s (for a reference, see e.g. [Gib01]).

## 14.6 Analysis of the algorithmic structure

The logical structure of the Xenakis algorithm, such as it is realized in its *GENDY3* version, is depicted in figure 14.1 on the next page. The chart can be broken down into two parts, an upper and a lower one. The lower part is the sound synthesis part of the algorithm. The upper part is what can be regarded a control structure which assembles patches of sound in the horizontal dimension (time) and vertical dimension (harmony), thereby creating the overall “architecture” of the musical piece. Let us discuss the figure from bottom to top. We will therefore start with GENDYN sound synthesis.

### 14.6.1 Sound Synthesis

The wave form at the bottom of the chart is constructed out of a series of breakpoints. These breakpoints are linked, sample per sample, by linear interpolation. Each breakpoint is defined by a time offset to the preceding breakpoint (i.e. the number of sample values to be interpolated in between them), and an amplitude value. These time offset and amplitude values are taken from the momentary positions in two associated random walks. They are represented on top of the wave form by the mathematical sign for sum because a random walk is basically the summation of random steps (sometimes depicted as the path of a “drunken sailor”). In the case of GENDYN, random walks are in one dimension only. In other words, at every step of a random walk, the direction (forward or backward resp. up and down) and the width of the step to be taken are chosen at random. In figure 14.1 on the facing page, there are three breakpoints to the wave form (the first and the last in the figure are taken to be the same, for the waveform is “wrapped around” and cycled through in time). Consequently, there have to be three time random walks and three amplitude random walks associated to each. Note that a random walk does not produce the positions of successive breakpoints but the successive positions of one and the same breakpoint as the

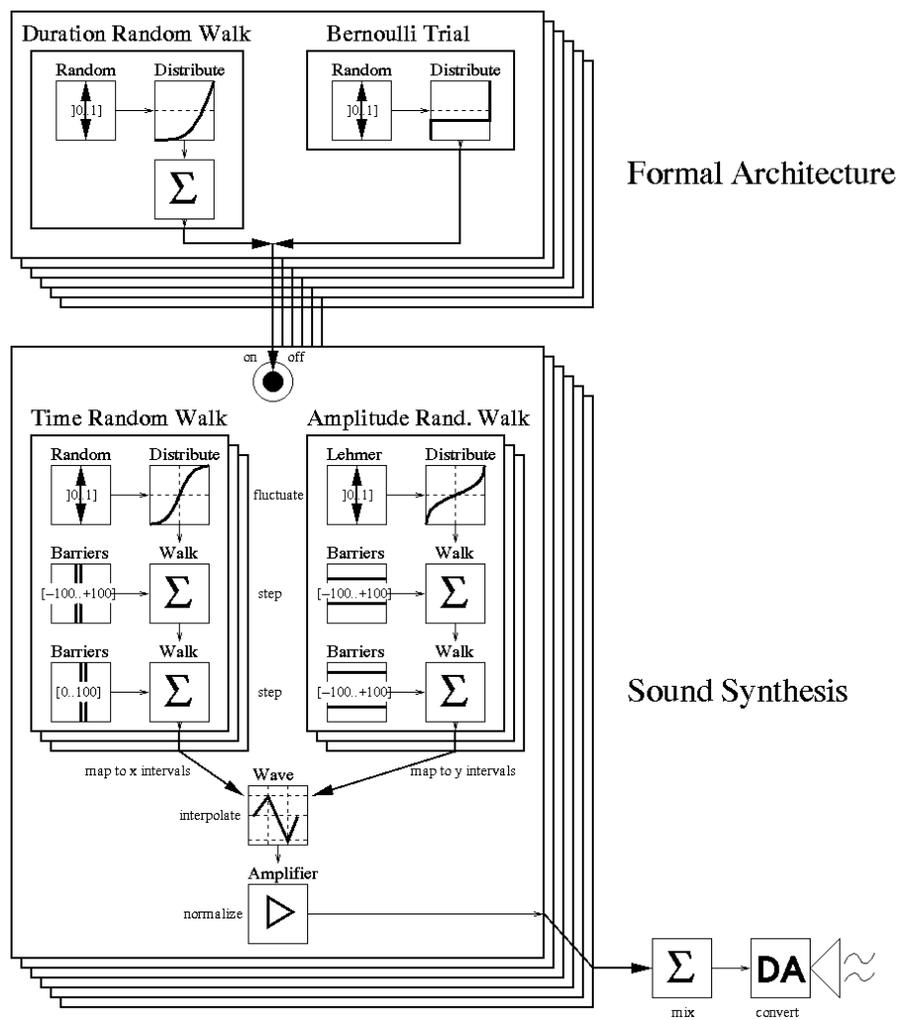


Figure 14.1: Chart diagram of Dynamic Stochastic Synthesis as realized in *GENDY3*.

wave form is cycled through in time. In other words, there are as many pairs of (time and amplitude) random walks as there are breakpoints in the wave, and they evolve independently “in parallel”. The result is a nonlinear stochastic distortion of the shape of the wave form in time.

The successive stepwidths of the time and amplitude random walks are controlled by yet another pair of random walks, drawn on top of the aforementioned ones. (The “patch connection” between the random walks in the figure means that the “output” of each upper random walk, i.e. its momentary position, is taken by the lower one as its current step input). Let us call the upper ones “primary” and the lower ones “secondary”. (Note that the primary random walks are not part of Xenakis’ presentation of his algorithm of Dynamic Stochastic Synthesis in [Xen92b].) The stepwidths of the primary random walks follow a certain probability distribution around and including zero (depicted in the figure by one distribution graph for each). All random walks are confined by pairs of barriers which reflect excessive values back into the value range in between them. The random walks for the time offsets between the breakpoints are constrained to stay within a range of positive values since time cannot flow backwards. This concludes the description of the sound synthesis aspect of GENDYN.

### 14.6.2 Control Structure

Now we turn to the aspect of the “score” structure of GENDYN. There are several instances of sound synthesis going on in a GENDYN program, each producing a track of complex sound and mixed together in order to yield a complex sonic evolution. The control structure takes care of this as well as of the segmentation of the tracks into sounding and silent fields. Note that the computation of pitch and timbre, classically attributed to the aspect of “score” composition, are already taken care of by the synthesis itself. The control structure does only handle the layout of the synthesized sound patches in time.

There are six “tracks” of sound drawn in figure 14.1 on the previous page, each represented as a card in a stack of cards, but there are 16 of them in *GENDY3*. (Conceptually, there is no fixed limit to their number.) These sound tracks are played simultaneously, and mixed down to a master output by a “mixer” in the lower right part of the diagram. In GENDYN, sound is chopped into patches of sounds and silences (called “fields” by Xenakis) by a control depicted on top of the synthesis module (and called “duration random walk” in the figure). Each track is controlled by an independent duration random walk. At each (positive!) random step, sound generation is switched on and off by a probabilistic yes/no decision (called “Bernoulli trial” in figure 14.1 on the preceding page). All fields, whether sounding or silent, have different random duration (from fractions of a second to up to 30 seconds and more). This creates a kind of stochastic rhythmical “counterpoint” between the tracks. The sum of the durations of the fields defines the duration of a whole track. The superposition of a number of tracks so generated is called a “sequence” by Xenakis. A sequence has a typical duration between some seconds and some minutes. It is simply as long as the longest of its tracks. An arbitrary succession of such “sequences” structured this way (11 in the case of *GENDY3*) forms a GENDYN piece. For the composition of *GENDY3*, Xenakis assigned different sets of sound synthesis parameters to each sequence. This created different sounds and different field density, and hence, sections of different sonic character within

the *GENDY3* composition. Different probability values, barrier positions, and choices of distribution formula also have strong impact on the timbre and pitch evolution of the generated sound.

A hierarchy of intermediary logical levels thus helps breaking down the problem of generating a musical piece into the more specific problems of iteratively generating a sequence, a track, a field, a wave form, a wave segment (see figure 9.2 on page 140). In other words, the problem of generating a linear succession of sample values is solved by designing a hierarchy of abstraction levels on top of the raw sound data.

Let us now turn to the user interface of the New GENDYN Program and see how it supports to develop an intuitive understanding of Stochastic Synthesis in general and of the *GENDY3* composition in particular.

## 14.7 A Look at Synthesis in the New GENDYN Program

We can watch the various stages and levels of GENDYN sound computation at work with the help of the New GENDYN Program. As has been stated, this implementation of Xenakis' *GENDY3* algorithm features, on top of the synthesis itself, a number of plots and animations which help to understand the nature and the dynamics of the synthesis process and hence of the compositional procedure.

First, on the level of the sequence architecture we have a plotting of the duration structure (the tracks with their sounding / non sounding fields) on the screen, in the same way as Xenakis generated it on paper for his own purposes. However, in our case, this plotting is interactive in the sense that at every (re-)computation of this structure, the plot updates itself to show a picture of the new structure. Therefore, it is possible to see typical characteristics of the GENDYN distribution of sounding patches over time and tracks while playing with the parameters controlling the distribution formula. One can thereby also observe properties of this structure that are common to all results of computation, e.g. the property that tracks start synchronous at the beginning of a sequence but end asynchronous depending on their individual length.

Second, the sound synthesis procedure is pictured with the help of five small animation windows on top of the dialog controlling the sound synthesis settings of a track. The first two are the graphs of the distribution functions driving the steps of the (primary) time and amplitude random walks. The primary (see figure 14.2 on the following page) and secondary (see figure 14.3 on the next page) random walks are pictured in the next two windows, where each combines the aspects of (horizontal) time and (vertical) amplitude fluctuation by painting a billiard ball at the point in the plane where the momentary horizontal and vertical positions intersect. The fifth, rightmost window (see figure 14.4 on the following page) is the picture of the computed wave period, which is built by using the momentary positions of the billiard balls of the secondary random walks to define the sample offsets and amplitude values of the wave form's breakpoints.

We may conceive of the random walks as billiard balls moving in a two-dimensional space. The segment width is assigned to the horizontal and the

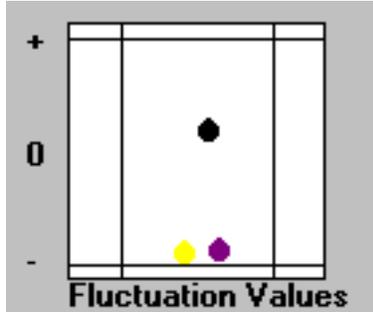


Figure 14.2: A snapshot of  $2 \times 3$  primary random walks: 3 for the stepwidth in time and 3 for the stepwidth in amplitude for the secondary random walks, plotted as the position of 3 billiard balls in a 2d plane.

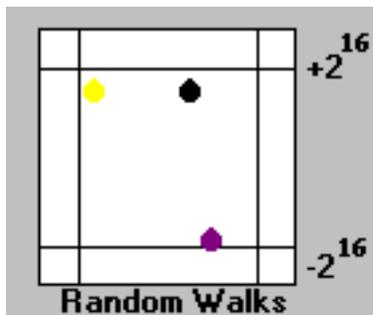


Figure 14.3: A snapshot of  $2 \times 3$  secondary random walks: 3 for the time increments and 3 for the amplitude value of 3 wave form breakpoints, plotted as the position of 3 billiard balls in a 2d plane.

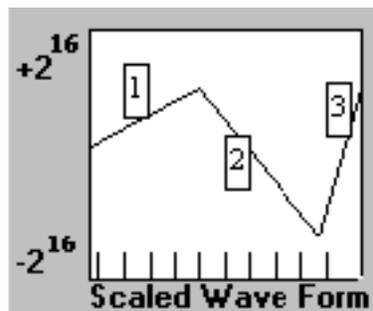


Figure 14.4: A simple wave polygone with 3 vertices forming 3 segments, scaled to fit the window. Sample ticks are schematically shown underneath.

amplitude to the vertical axis (see figure 14.4 on the preceding page). With each repetition of the polygonal wave form, its vertices (= “balls”) perform random steps in a vertical direction (changing its amplitude) as well as in a horizontal direction (changing the width of the segment). Each vertex of the wave form polygon performs an independent random walk, so there are as much random walks in parallel as there are segments in the wave form polygon (see figure 14.5). Since these random walks take place within the context of a closed wave form period, a periodicity is maintained in spite of the erratic movement of the “sound particles” (cf. [Xen96]).

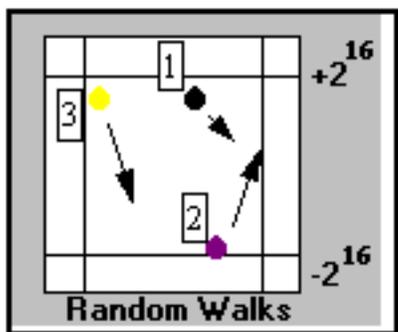


Figure 14.5: The movement of the polygon vertices (balls) in a random walk space.

The probability values for each random step are drawn from a set of specific probability distributions. Each probability distribution has a characteristic variance of probabilistic values within a given range. If driven by different distribution functions, the “billiard balls” move in different ways. A distribution with a steep gradient will inhibit, a distribution with a smooth gradient reinforce their “diffusion” in random walk space (see figures 14.13 on page 237 to 14.16 on page 239). Gradient and value range of each function can be influenced by a coefficient parameter.

## 14.8 “Static” Analysis of *GENDY3*

Let us first look at the static properties of the GENDYN algorithm. They hold for all music generated with the help of Dynamic Stochastic Synthesis as it has been defined by Xenakis.

**Exponential rise of pitch.** The linear change of the length of the generated waveforms corresponds to an exponential shift in pitch. Wavelength is the inverse of frequency, and frequency perception is logarithmic. One of the most striking results is the extended glissando of the first section of *GENDY3*. It accelerates as it rises until it starts to turn virtual pitch “somersaults” when approaching its climax (cf. figure 15.1 on page 277).

**Tension, suspense and surprise.** Sonic event density is maximal at the beginning and minimal at the end of a sequence. Sequences start with all

their sound tracks synchronously switched on but asynchronously switched off due to probability fluctuations in the computing of their length. As a consequence, contrast in timbre between sequences due to the different parameter settings of the sounds they contain is marked and reinforced by an abrupt change in overall dynamic envelope.

**Sound brilliance.** There is a strong presence of high partials due to the “staircase”-like form of the sound signal computed in the restricted numeric interval of  $[-100 \dots +100]$ . Even more so, in *GENDY3*, the individual sound tracks are confined to an interval not greater than  $[-20 \dots +20]$  (a quantization of little more than 4 bit). The result is a specific “digital” noise aura added to all sound (see figure 14.6).

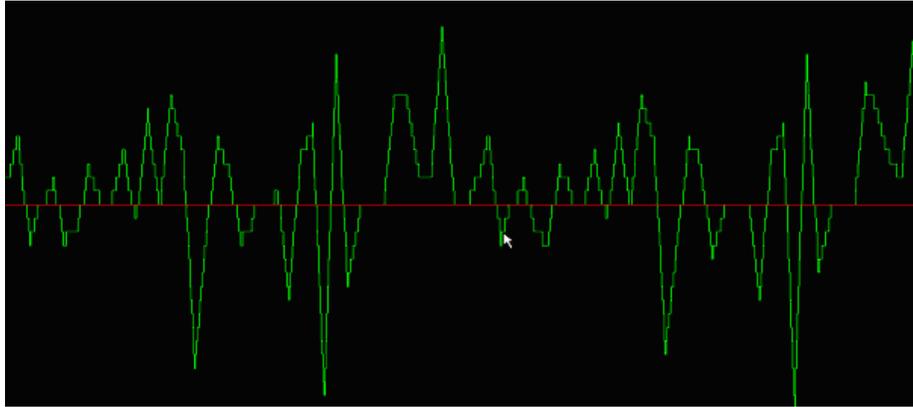


Figure 14.6: The sound signal of an arbitrary GENDYN sound, produced with the New GENDYN Program. One clearly sees the stepwise movement of sound pressure curve, as if produced with a “bad” analog-to-digital converter.

**“Dropout” effects in sustained sounds.** These “dropouts” are actually composed, as a result of a steep envelope curve at the beginning and the end of each sounding section of a track (called “field” in Xenakis’ terminology). Whenever two sounding fields follow one another, sound is discontinued at the border between the two, as a result of this sustain/attack attenuation.<sup>1</sup> (see 14.7 on the next page).

**Pitch transients.** In Xenakis’ implementation, the initial positions of all random walks are at the origin (i.e. zero). For amplitude this means that sound rises from silence. For the time values it means that all sound starts with the shortest possible waveform, i.e. where pitch is as high as possible, and then descends until it reaches a kind of “stochastic pitch equilibrium”. This explains the beginning of the pitch curve of the “solo voice” in the first sequence of *GENDY3* (cf. 15.1 on page 277). Random walks are continued from one sequence to another, so each sequence’s track starts with

<sup>1</sup>These composed “dropouts” caused a caring technician at Saarländischer Rundfunk to dismiss one of his DAT players as “incompatible” to the *GENDY3* tape only to discover that the other players did not do any better on this tape!

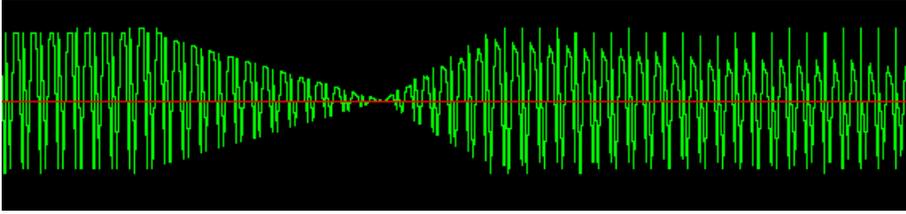


Figure 14.7: A continuous sound with a “composed dropout” which occurs because Xenakis provided for a fade-in of sound fields even when they were preceded by a sounding field. The example is the sound signal of a small excerpt of figure 14.30 on page 255.

the wavelength reached at the end of the foregoing sequence. If this start wavelength is off the stochastic equilibrium of the new track, the attack of the sound will be characterized by a more or less dramatic balancing movement of pitch. This may explain e.g. the “anomalies” in the spectrogram of the noisy sequence  $\psi = 9$  observed by [DiS98] (see figure 14.24 on page 249).

## 14.9 Analysis of Implementation Features

There are a few peculiarities in Xenakis’ design of the algorithm to be noted:

1. Even though the random walks define a quantized digital wave form, they run in a continuous space (real numbers). Therefore, their movement is much more delicate than it could if random walk space had also been quantized. This allows for a much more sophisticated dynamic behavior and, consequently, results in richer sound.
2. The fact that the random walks forming the wave are “second order” (i.e. they are driven by another set of random walks) makes their dynamic behavior much more interesting (see below figure 14.13 on page 237).
3. Xenakis designed a special random generator for the amplitude probabilities, after the Lehmer formula, with a different (much more periodic) time behavior (see figure 14.1 on page 217). Xenakis gives this formula as

$$x_i = (x_{i-1}a + c) \bmod M$$

in [Xen96], p. 154. This equation can be recognized (albeit not easily) in the GENDYN program (compare the listing in [Xenakis 1992], p. 319).

4. The random walk spaces are arbitrarily restricted to the abstract numeric interval  $[100..100]$ . This is adequate for the time offsets between break-points which should not get too large (they are measured in sample ticks, where the sampling rate can be chosen at will; Xenakis chose 44.1 kHz for both *GENDY3* and *S709*). But for the amplitude values this means that, when they are later normalized by a constant factor to 16 bit (i.e. to the interval  $[-32768 \dots 32767]$ ), they take on only 200 different values.

5. The interpolation between the amplitude breakpoints is also done within the same interval  $[-100 \dots +100]$ . (Note that in figure 14.1 on page 217, the sound signal is normalized to 16 bit only after interpolation.) Therefore, GENDYN sound has a “quality” of less than 6 bit. The resulting “quantization noise”, however, gives GENDYN sound its specific brilliance.
6. A sequence does not stop with the end of the longest track but with the end of the last sounding field. This is equivalent to saying that all silence at the end of each sequence is cropped by the algorithm. It is as if Xenakis had “edited off” the silences between sequences which would inevitably have occurred due to the stochastic distribution of sounding and silent fields over the duration of a track. The musical result is as if the sequences were cut hard against each other after having been generated. Yet this effect is not an action of postproduction but an intricate part of the algorithm generated along with the music. This effect stresses the contrast of sonorities between adjacent sequences: the new sequence breaks “tutti” into the last fading out “solo” of the foregoing sequence.

## 14.10 Analysis of Xenakis’ Programming “Licencies”

In addition, there are peculiarities in Xenakis’ implementation of the GENDYN program that have nothing to do with the realization of the algorithm itself, but add additional features to its operation.

1. Due to a program bug (undeclared variables), the Finite Impulse Response filter acting on successive time offset values has a delay line of length zero (instead of length 3). If this filter is active, all time increment values are simply reduced by a constant weight factor of one third. Since time increments are integer values, this reduction increases the coarseness of the wave form’s time structure. If applied to a high noise, the result is a sizzling modulation of that noise impossible to obtain by other means.
2. Due to an intricate interaction between if-then-else cascades, control flow (with GOTO’s) and variable state transformation, sounding fields are one sample value longer than muted ones, making it more difficult to retrace the “history” of the successive random number drawings. Simulation of the original composition procedure is further complicated by the fact that the original implementation uses alternating arrays to store the current random walk positions. Therefore, when a new active sound field is started, all random walks resume with positions stored in the first of the two arrays, regardless which array was assigned the position computed by the last preceding sound field.

These odds in Xenakis’ programming have no algorithmic value whatsoever, and it would be nice if one could avoid to pay any attention to them. Yet, since computation of GENDYN sound is a chaotic process in the precise sense established above, it is sensitive to the tiniest details of its programmed realization. Therefore, there is no choice but emulating these odds of Xenakis’ original implementation within the New GENDYN Program, in order to obtain the same

music. However, in contrast to Xenakis’ original program, these odds are now made explicit and well-documented within the program. One can dispense of them as soon as there is no longer any need to recompute the original music of *GENDY3*.

This concludes the analysis of the GENDYN program, as far as the general structure and the basic operational behavior of the program is concerned. In the following, we will turn our attention to a specific instantiation of this program, namely the one that was used for the computation of *GENDY3*. *GENDY3* was composed by feeding this version of the GENDYN program with a number of a predefined set of synthesis parameters.

In order to do so, we will have a look at the New GENDYN Program just after having loaded those parameters and before the synthesis is started. The parameters govern many aspects of the composition such as its division into sections (11 in the case of *GENDY3*), the number of simultaneously sounding tracks (up to 16 in *GENDY3* and varying from section to section), the duration of these tracks and their subdivision into sounding and muted patches, as well as the characteristics of its sounds.

Xenakis’ original program, after loading the parameter files, would start computing the music right away. In contrast, the New GENDYN Program, after reading the synthesis parameters, waits for the user to check and fine-tune the parameter settings with the help of its graphical user interface (which Xenakis did not have). Therefore, we are now in a situation where we can look at a graphical display of the *GENDY3* parameters (and test them interactively one by one by executing parts of the synthesis, of selected sections and / or selected tracks), whereas Xenakis was in a situation where he had to either accept or reject a result that he could hardly foresee.

Let us first have a look at the “architecture” of *GENDY3*, that is, the structure concerning its partitioning into sequences, tracks and sound patches. In this and the following, we will illustrate the description with screen shots from the New GENDYN Program functioning as a tool of music analysis in the sense of the foregoing discussion. The parameter controls correspond to Xenakis’ original parameter set, except for some additions, which are explicitly mentioned.

## 14.11 The “Architecture” of *GENDY3*

In the user interface of the New GENDYN Program, all entities involved in synthesis and which have parameters to choose, are presented in a number of dedicated dialogs. There are dialogs for the piece entity, the sequence entities, and the sound entities. Let us first have a look at the dialog for the piece entity.

### 14.11.1 Parameters for the *GENDY3* piece

The parameters for the *GENDY3* piece entity are depicted in figure 14.8 on the next page.

**The In-Time-Mapping  $\psi$ .** Close to the left border of the dialog, we see a graphical representation of Xenakis’  $\psi$  mapping function as a sequential list. This function maps the positions of the list (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) onto sequence numbers (10, 1, 2, 9, 3, 8, 4, 5, 7, 11, 6). That is, sequence 10

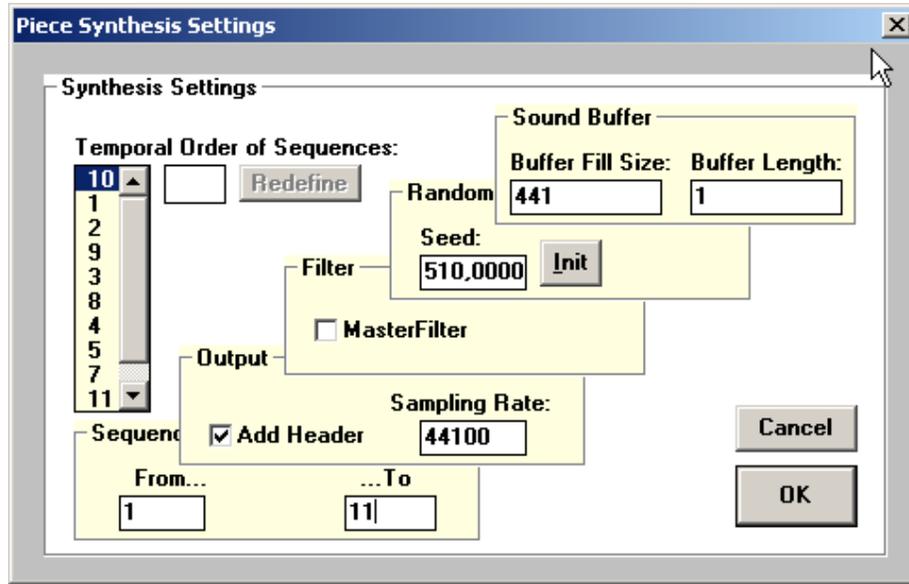


Figure 14.8: The global parameters of *GENDY3*

is synthesized first, then 1, then 2, then 9, and so on. Note that for the premiere of *GENDY3*'s pre-version at ICMC Montréal, the  $\psi$  function was defined differently.

**Sequence Playlist.** At the bottom of the  $\psi$  function, there is a possibility to select parts of the piece for generation. For the whole of *GENDY3*, the first through the 11th sequence are to be synthesized. However, it is possible to synthesize the sequences individually, e.g. for testing purposes.

**Sampling Rate.** Proceeding from bottom to top, we find in the next upper frame a sound output parameter: the sampling rate. Xenakis did probably not set this value other than to 44100 samples per second (CD audio standard) given the fact that higher playback rates were at times not available in a concert situation, nor on CD. However, he may have reserved this setting for future experiments with higher sampling rates which he regarded as favorable for obtaining “sounding fractals” (see [Xen92b], preface). As we now know, the sampling rate directly governs possible GENDYN pitch scales since GENDYN pitches are produced as various subdivisions of the sampling pulse code frequency. Therefore, it could be interesting to alter the GENDYN sampling rate parameter while using the standard sampling rate for playback, in order to obtain different distributions of pitch.

**Add (Wave) Header.** This checkbox has been added on top of Xenakis' original parameters in order to be able to play back GENDYN sound files with standard Windows audio tools. Note that Xenakis did not need this parameter because he sent raw data to a custom hardware D/A converter at CEMAMu.

**Master Filter.** This checkbox activates a lowpass filtering of the mixdown result of the possible 16 GENDYN tracks. Just like the lowpass filters of the individual sounds it has not been used for the synthesis of *GENDY3*. These filters realize a simple 1st order filtering (averaging the values of the current and the last two samples). Since this kind of filtering reduces the digital artifacts (this is one of the reasons for such filters) it would also reduce the specific quality of GENDYN sounds, in the sense of our discussion of idiomatic computer sound. One may wonder why Xenakis introduced these filters into his programming at all.

**Random Seed.** This number is the initialization of the custom random number generator built into GENDYN. This random number generator controls only the random walk of the amplitude values. Recall that for the time random walks Xenakis used the built-in random number function of the Quick Basic runtime system. These numbers are read into the New GENDYN Program from files that were “milked” from Xenakis original program since the Quick Basic random function is not available in C++ nor in Visual Basic.

**Sound Buffer Fill Size and Length.** These settings are dedicated to real time sound output and hence exclusive to the New GENDYN Program. The first is the number of samples written in a chunk to the buffer. The second is the length of the buffer in seconds. With the standard sampling rate of 44100 samples per seconds, a value of 1 means that 44100 samples are written into the buffer before it is flushed to the output device. A fill size of 441 means that there are 100 sound chunks generated per second. In between control is yielded to the user interface for parameter adjustment and / or graphical animation. (Recall that sound synthesis and interface are implemented as two cooperative tasks.)

This concludes the description of the global set of GENDYN parameters. Let us now turn to the set of GENDYN parameters governing the aspects of a GENDYN sequence.

### 14.11.2 Duration structure of *GENDY3* sequences

Let us now look at the parameter interface of a sequence. It is displayed in the sequence dialog (see 14.9 on the following page). For the 11 sequences of *GENDY3*, there are 11 dialogs, to be opened from the menu *Sequence* on top of the application window.

**Fields.** The GENDYN algorithm cuts continuous sound into sound patches by dividing each track of sound into a number of sound “fields”. These are listed in the rightmost frame of the window. Sound generation is suspended as soon as a sound field is entered that has been defined as “mute” (a duration number without selection in the list) during the (foregoing) computation of the sequence architecture. At the next sounding field (a duration number with selection), sound generation is resumed exactly where it had been broken off. This procedure has the same effect as if the sound had been generated in one piece, then cut and laid out “in score”.

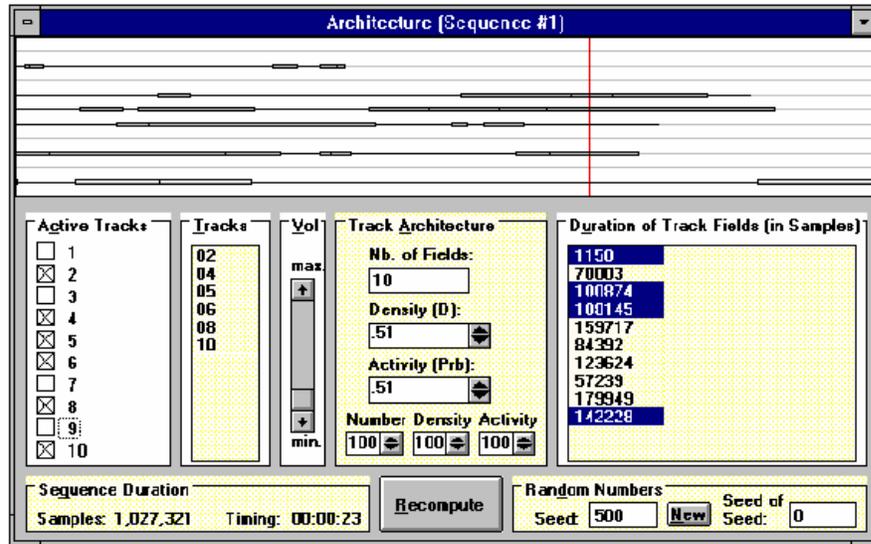


Figure 14.9: The parameters of sequence 10 (the first to be played in *GENDY3*)

**Sequence Architecture Plot.** A picture of the distribution of sound patches in the various active tracks is in the upper part of the window. This picture was also available to Xenakis through a helper program which would plot sequence architectures from PARAG parameter files. (The sound architecture was produced by the various PARAG programs and read into Xenakis' GENDYN program on startup).

**Track Activity.** The choice which tracks participate in synthesis is a parameter to a GENDYN sequence. In the left part of the sequence dialog, all 16 tracks can be activated or muted during synthesis. Xenakis used this feature in GENDYN as a static configuration for whole sequences. The settings for *GENDY3* are: in sequence 2 (in-time 3), tracks 7 to 17 are muted; in sequence 4 (in-time 7), 9 to 16 are muted; in sequence 5 (in-time 8), tracks 1 and 2 are muted; sequence 7 (in-time 9), tracks 11 to 16 are muted; sequence 9 (in-time 4) all but the first 5 tracks are muted; and in sequence 11 (in-time 10) track 1 and track 16 are muted.

The activation and deactivation of tracks, together with real-time parameter control, can be used for the purpose of analysis and demonstration. This permits to e.g. create acoustic demonstrations of the stacking of GENDYN clusters by adding or removing tracks of fixed pitch, on by one during synthesis.

**Field Activity.** As an addition to Xenakis' program, duration numbers can be clicked in the field list to mute / activate them. (This is, however, a digression from the principle of randomness in the creation of a GENDYN architecture.)

**Master Volume.** In order to prevent saturation of GENDYN sound, the am-

plitude sum of GENDYN tracks is divided by the number of active tracks. However, for *GENDY3*, all tracks amplitudes were limited by the amplitude barriers such that their mixdown could never saturate. (Possibly, Xenakis had forgotten about the division factor on mixdown.) The Master volume parameter adjusts this division factor between 1 to 16 and is an addition to Xenakis’ set of sequence parameters.

**Track Architecture.** The stochastic computation of the track’s architecture for a sequence is a basic feature of GENDYN. As for the other parameters, this computation has been made interactive in the New GENDYN Program (by hitting the button *Recompute*). For a given number of fields and a density coefficient, the duration of each of these fields is computed using the exponential probability distribution (see appendix A on page 315). In a second step, for each field there is a yes/no-decision if this field is sounding or muted, using a probability value between 0 and 1.

This concludes the description of the sequence parameters. The three values on top of the *Recompute* button are for convenience only; they adjust the three parameters *Number of Fields*, *Density* and *Activity* by a percentage factor, in order to help finding a suitable architecture. They are by themselves not part of the GENDYN parameters.

**Computation of the Architecture.** On hitting the button *Recompute*, the field durations of each track are computed one by one, with the help of the exponential probability distribution. This distribution assigns a probabilistic value to the duration of each field when given a density value as a parameter. Since it is a distribution function, there exists a one-to-one mapping of a given field duration with the accumulated probability of it appearing in a track. The individual fields vary in length but they all conform to the overall density of the track subdivision into fields expressed by the distribution parameter. The length of the track is simply the sum of its field durations. The activity of each field (i.e. sounding / not sounding) is fixed by a coin-tossing decision (the coin can be biased with the help of the parameter *Activity*). The sound density of a track is therefore a combination of two parametrizations: the density of its subdivision into fields on the one hand and the probability for each field to sound on the other. The sound density of a sequence is the sum of the sound densities of all its active tracks (i.e. those tracks that are not muted).

The computation of GENDYN’s architecture is typical for Xenakis’ stylistic signature in general. One can regard it as the algorithmic realization of his cut-and-paste collage working. A musical analysis of musical works by Iannis Xenakis often reveals musical textures that are strongly coherent within themselves, but spread over the score, as if a contiguous whole of this structure had been cut up and distributed all over the canvas of musical time. (A striking example is the tearing apart of the “paths” of symmetry group operations in *Nomos gamma* [Hof94a].) One can fairly say that Xenakis incorporated his thinking concerning the composition of musical form into the working of his GENDYN program.

Due to probability fluctuations, the exact length of each track is different even if they all share the same field density parameter. Since the total length of a sequence is equal to the length of the longest track, a sequence tends to

”thin” out toward its end as the tracks run out of activity one after the other. The musical effect is a retardation of musical action which very much intensifies the tension before the following sequence surprisingly starts “tutti” with its new sonorities.

### 14.11.3 Sound Synthesis Parameters

Let us now look at the parameter interface of a *GENDY3* sound. It is displayed in the sound dialog (see 14.10). Sound synthesis parameters are specific to tracks. For the up to 16 sound (tracks) of a *GENDY3* sequence, there are 16 dialogs available, to be opened from a menu *Track* on top of the application window.

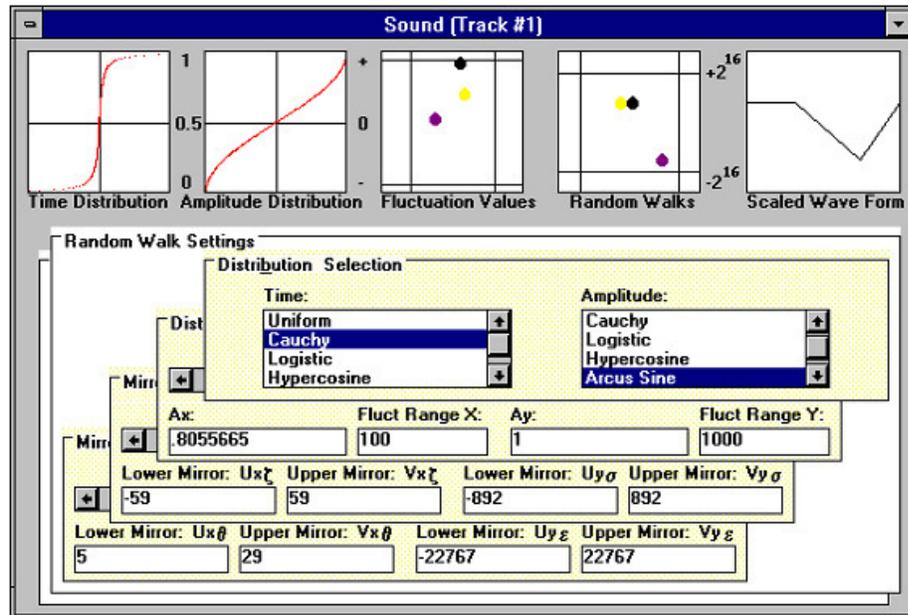


Figure 14.10: The synthesis parameters of a *GENDY3* sound: the distributions, coefficients, and the primary resp. secondary random walk positions for both the time (on the left) and amplitude random walks (on the right). Note that coefficients B are only used for the logistic distribution.

**Number of wave form segments.** This parameters controls the slicing of the wave form to be generated into segments. In order words, it defines the number of breakpoints to compute. In general, decreasing the number of breakpoints results in higher frequency of the sound. Increasing this number decreases the resulting frequency.

However, resulting pitch has also to do with the spacing of the breakpoints. If the segments contain many interpolated samples, a sound may be low even if there are few breakpoints. When comparing a low sound generated with many densely spaced and a low sound with few widely spaced breakpoints, the difference is in the spectrum. Increasing the number of

breakpoints for a sound means increasing the spiking and the roughness of the resulting wave form and therefore a boost of the high partials in the resulting spectrum. In other words, such a sound is low in pitch but sharp in timbre.

Note that there must be at least 2 breakpoints to form a periodic signal, conforming to the Sampling Theorem. One breakpoint results in noise. This can be interesting, too, for these wind-like noises contain a lot of digital artifacts.

**Wave Repetition.** This parameter has not been used by Xenakis for the creation of *GENDY3* but only in the later version of the program for Xenakis’ second GENDYN piece *S709*. This parameter decides how often a wave form is played back unchanged before the next stochastic alteration is applied to it. I included this parameter in the control interface of the New GENDYN Program for future uses by Xenakis, which unfortunately, as already reported, did not happen. However, the wave repetition parameter proved fruitful for the analysis of *GENDY3* because when being played with, it revealed the true nature of the noisy sounds of sequence #9 as (stochastic) frequency modulation noises. When set to a value greater than zero, modulation frequency was decreased, resulting in the disappearance of noise in favor of simple aleatoric melodies. When inverting this process, the melodies grew quicker and quicker, and eventually turned into noise, similar to the impact of increasing the modulation frequency in classical sinusoidal FM.

**Filter.** Both random steps in time (Time Filter) and amplitude (Amplitude Filter) may be subject to a simple Finite Impulse Response filtering, just like the master filter discussed above. Xenakis used only the time random walk filter in one instance: for tracks #1 and #2 of sequence  $\psi = 9$ , what I call “the Apotheosis of Noise”. The effect of this filtering has been discussed above (cf. 14.10 on page 224).

**Real Time Preview.** The control elements in this frame do not represent GENDYN parameters but are additions of the New GENDYN Program in order to generate animations and plots of the synthesis process, as discussed in the preceding sections (see figure 14.11 on the following page). There is a button to start / stop animation, and there is a slider to control the update interval of the graphics. The smaller this update interval, the denser are the calls to the synthesis engine to inquire the current synthesis values, and as a result, the smoother the graphics move resp. the plots draw. As a drawback, computing time is taken off the synthesis and devoted to these calls and the graphical rendering following it. This may lead to sound buffer underrun and discontinuities in the resulting sound signal. This, however, only applies to real-time playback; a non-real-time recording will not contain such disruptions.

All animations can be individually enabled: the plotting of the distribution graphs, of the primary and secondary random walks, and of the resulting wave form (which for obvious reasons has to be fitted into the viewing window, so, unfortunately, change of wave form length is not visible). As an alternative to viewing the wave form, the pitch curve of the sound can

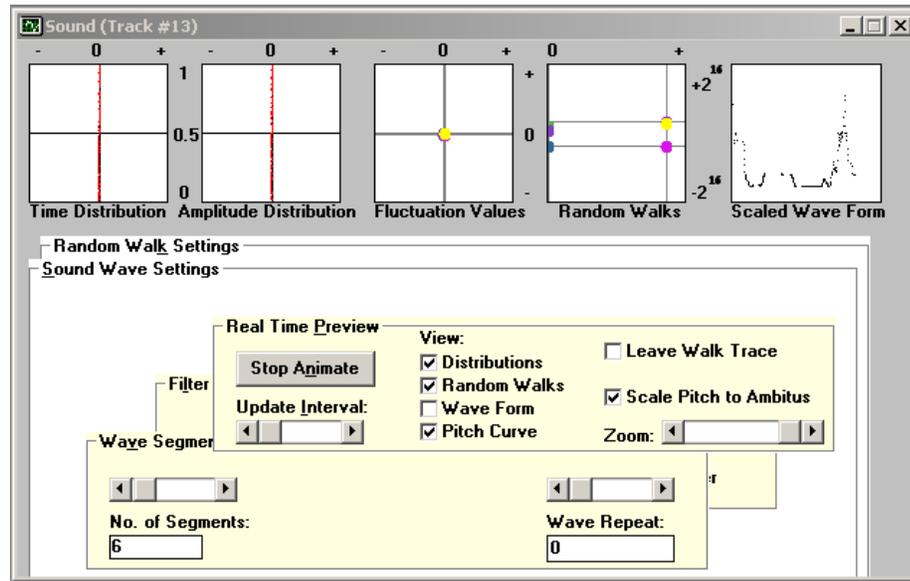


Figure 14.11: The controls for a visual animation of *GENDY3* sound synthesis as described in the text, along with the sound synthesis parameter *no. of wave form segments*. The remaining controls for filters and wave repetition are not used in *GENDY3*.

be plotted on the same canvas, and this is the way the pitch score for the whole of *GENDY3* has been created. (Plots of individual sounds required two more tunings: the adaption of pitch to the vertical height of the drawing canvas, along with a zoom slider.) There is also an alternate checkbox for the random walk animations: instead of moving the billiard balls around, all positions occupied by the billiard balls over the course of synthesis process can be painted in order to mark their trajectories in random walk phase space. This is the way figures 14.13 on page 237 to 14.16 on page 239 have been created, which eventually led to the discovery of the lawful pitch structures of *GENDYN*, the stochastic scales.

**Distribution functions.** In the uppermost frame, there are two lists of distribution functions that Xenakis implemented in *GENDYN*: Uniform, Cauchy, Logistic, Hyperbolic Cosine, Arcus Sinus, Exponential, and Triangular. These distribution functions fulfil the task to map random numbers with equal distribution produced by the random number generators into numbers distributed according to the characteristic probability distribution specific to these functions. These numbers then serve as stepwidths for the amplitude and time random walks of *GENDYN*. For *GENDY3*, the Uniform as well as the Triangular distribution were not used. Cauchy, Logistic, Hyperbolic Cosine, which have the well-known “bell-like” density functions associated to them, are centered around the y-axis and therefore produce (more or less) equilibrated sonic phenomena like (slow or fast) up-and down glissando movement. The same holds for the Logistic distribution, but which distributes over a wider value ambitus, and which has a second coefficient (only used for the Logistic distribution) dislocating the graph along the x-axis, therefore the Logistic function, in general, is not symmetric to the y-axis. In *GENDY3*, the Logistic function mainly produces noise sounds. Arcus Sine and Exponential distribute to positive values only, therefore, they only push the random walks against their upper barrier. (Note, however, that in mathematical theory, Arcus Sine has a symmetric graph and could be adapted to distribute over both positive and negative values.)

It is not known if Xenakis knew the mathematical  $\beta$  distribution at times of implementing *GENDYN*. This function has the property of covering the characteristics of all functions used by Xenakis in *GENDY3*, within a continuous range of coefficient settings. In other words, instead of changing distributions and adapting the coefficients of each of them, using the  $\beta$  function, one could turn only two knobs, the two which control the coefficients of the  $\beta$  function, to create the same effects with smooth transitions in between.

**Distribution coefficients.** One can say that in *GENDYN*, the coefficients determine the steepness of the distribution functions. For example, the nearer the coefficient  $A$  approaches zero, the more functions like Cauchy and Hyperbolic Cosine nestle up against the vertical y-axis. The effect is that the distribution values keep very close to zero, with only small and rare deviations toward negative or positive values. Such a parameter setting is important for smooth and slow-motion glissandi. The other distribution functions are also governed by the  $A$  coefficient, with different

effects. Only the Logistic function is influenced by the second coefficient  $B$ : it moves the graph along the x-axis.

**Positions of the random walk mirrors.** For each random walk, there are 2 barrier positions to be fixed: one for the upper and one for the lower barrier. In sum, there are 8 positions to be set: 2 times four for each of the four random walks of *GENDY3* (i.e. primary time and amplitude random walks and secondary time and amplitude random walks). These are shown in the two respective parameter frames. The captions for these values follow Xenakis' own naming as found in [Xen92b]. Note that the time random walks do not accept a lower barrier position less than zero, since time cannot flow backwards (there can be no negative number of interpolation values between two successive wave breakpoints). In the animation pictures on top of the parameter dialog, the barrier positions are represented as straight lines moving left and right (time barriers) resp. up and down (amplitude barriers). They move as soon as the values are changed, in contrast to the other animations which need synthesis running in order to function.

This concludes the description of GENDYN sound synthesis parameters as of its 1991 version and thereby the description of the whole set of parameters available to the synthesis of *GENDY3*.

Let us now turn to the runtime analysis of *GENDY3* and see what we can find out by looking at the GENDYN algorithm on synthesis.

## 14.12 Analysis of the Dynamic Behavior of the Stochastic Synthesis

In contrast to the general “static” features of Stochastic Sound described above, some prominent aspects of *GENDY3* can only be studied by watching the algorithm in action. We may call these aspects of the Dynamic Stochastic Synthesis “emergent” features of the algorithmic action, because they cannot be recognized by analyzing the logic of the program but only by “looking at” (and listening to) the generation of huge amounts of sound data over longer periods of time.

**Random number distribution graphs.** In the first two animation pictures on top of the dialog, the distribution graphs are plotted. The time distribution graph is in the first picture and the amplitude distribution graph in the second. On synthesis, each random number drawing paints a red dot where the (randomly drawn) probability value and its corresponding step value according to the distribution function intersect, and after some time, these dots form the distribution function's graph.

**Primary random walks.** The random walks are one-dimensional (either increment of sample ticks or increment of amplitude), but it is convenient to display them in a combined way as billiard balls moving in a two-dimensional space of time (x-axis) and amplitude (y-axis). The barriers which confine each of these two random walks, then, are visualized as the

four borders of this two-dimensional plane. Each pair of these  $2 \times n$  parallel random walks is represented by a billiard ball. Each billiard ball is assigned another color.

**Secondary random walks.** The secondary random walks are visualized in the same way. Since each primary random walk “drives” a secondary random walk, there are as many billiard balls in the second plane as there are in the first. They are equally colored, so the impact of the primary on the secondary ones becomes obvious.

**The wave form.** The momentary wave form shape directly results from the positions (both horizontal and vertical) of the secondary random walk “billiard” balls. The vertical positions of the billiard balls and of the wave form’s breakpoints are exactly the same, whereas the spacing of the breakpoints in time corresponds to the horizontal position of the billiard balls on the x-axis. When a billiard ball is situated close to the lower (i.e. left-hand side) barrier, this means a minimal time increment (at least one sample tick) from the preceding breakpoint in the wave form to the one corresponding to the billiard ball. Conversely, when a billiard ball is situated close to the upper (i.e. right-hand-side) barrier, the corresponding breakpoint is set  $n$  sample ticks further in time,  $n$  being the position of the billiard ball on the x-axis of the secondary random walk plane. With the billiard balls moving over time, the breakpoints of the resulting wave form follow the same “dance” in a two-dimensional plane, but spaced out in musical microtime. In the visual representation, the wave form is constantly scaled to fit the picture window, whereas in reality, the wave form constantly contracts and expands due to the change of spacings between its breakpoints.

The visualizations in the user interface of the New GENDYN Program are not an end in itself. Their function is to help developing an understanding for the interdependency of the various synthesis parameters (stochastic distribution parameters, random walk barriers, number of breakpoints) and their influence on the nature and dynamics of the synthesis. They are especially instructive when looked at as an ensemble.

For example, the impact of a change of a distribution coefficient can immediately be seen by a change of the distribution graph, then, after a while, by a change in the movements of “billiard balls” of the primary random walk, then, as a consequence, the change in movement of the “billiard balls” of the secondary random walk, and finally as a change in the transformation process of the wave form while at the same time hearing the change in the timbral quality of the generated sound.

As another example, by widening or narrowing the barriers of, say, the primary time random walk, more or less “space” is allocated to the movement of the “billiard balls” and hence the speed of movement of the “billiard balls” in the secondary random walk is changed, and, as a consequence, the contraction and expansion of the wave form which results at the same time in a clearly audible frequency modulation of the sound.

It becomes also obvious how different the dynamic “behavior” of the primary random walks is in comparison to the secondary random walks: the movement of the billiard balls in the second plane is very different to those in the first

plane. Another observation is that Dynamic Stochastic Synthesis cannot be fully controlled by the settings of its synthesis parameters: for example, when the barriers are set to leave more space for the movement of the billiard balls, it still depends on the “diffusion” speed of the billiard balls how fast (and how) they will fill out the new space.

### 14.13 “Runtime” Analysis of GENDY3

Extensive studies of this kind with the New GENDYN Program have revealed a number of dynamic characteristics of the Xenakis’ implementation of Dynamic Stochastic Synthesis:

**Saturated Amplitude and Saturated Spectrum.** The two random walks in series connection (a primary driving a secondary one) tend to produce extreme (either minimal or maximal) values for the breakpoint coordinates. All settings where the barriers of the primary random walk are set to values comparatively smaller than those of the secondary random walk push the secondary one towards its barriers. A waveform constructed with such extreme time intervals or amplitude values is mainly characterized by three different geometric shapes, depending on how these values follow each other: sharp “spikes” (opposite amplitude signs and two times minimal time increment between three successive breakpoints), triangular shapes (opposite amplitude signs and two times maximal time increment between three successive breakpoints), and rectangular shapes (same sign and maximal increment between two successive breakpoints). All of these forms produce spectra with many high partials that change over time - hence the particular “harshness” of GENDYN sound (see figure 14.12 on the next page for an example).

**“Overtone Music”.** With the Stochastic Synthesis constantly changing the form of the wave, different partials of the spectrum become prominent over time. If this change is slow enough, and if the pitch is fixed, the acoustic effect is like a kind of overtone music, much similar to the acoustic effect of a continually changing resonance body attached to a complex sound which amplifies certain partials and absorbs others (e.g. the oral cavity in the case of “overtone singing” techniques). The relative stability of the partials over a certain time interval has to do with the “attractors” of the dynamic stochastic process which come about for specific combinations of the synthesis parameters (e.g. small variance of distribution functions, small random walk space for the primary and large random walk space for the secondary random walk).

**Emergence of Microtonal, Non-Tempered Scales.** The emergence of microtonal scales in some of the GENDYN “solo voices” is due to a combinatorics of the waveform’s segment lengths which tend, just as the amplitude values, to be extreme (either minimal or maximal). This is due to the ingenious idea of Xenakis’ to switch two random walk in series, where upon each step, the second one takes the position of the first one as the width of the step. When the first random walk is confined to a very small space, the positions of the second one become relatively stable. The combinatorics law governing the resulting pitches is described below.

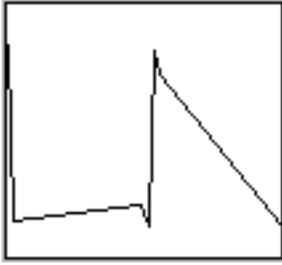


Figure 14.12: A typical GENDYN waveform: extreme values in both amplitude and time offset construct a saturated waveform with spikes.

A look at the trace of a random walk particle in the phase space of amplitude value vs. time offset can help to understand some of its dynamics. In figure 14.13 and 14.14 on the next page, the dynamic behavior of random walks is displayed by plotting GENDYN's primary and secondary random walks during a time interval of some seconds. Both figures show the primary random walks in the left frame, and the secondary ones in the right frame. Different random walks are plotted with different color shade. Note that the random walks are one-dimensional: the amplitude random walk is plotted on the y-axis, and the time random walk on the x-axis. A point in the plot of a secondary random walk, therefore, denotes a current time offset of a breakpoint to its predecessor breakpoint as well as its current amplitude. A point in the plot of a primary random walk denotes the current time and amplitude *delta* taken as the stepwidth input for the associated secondary random walk.

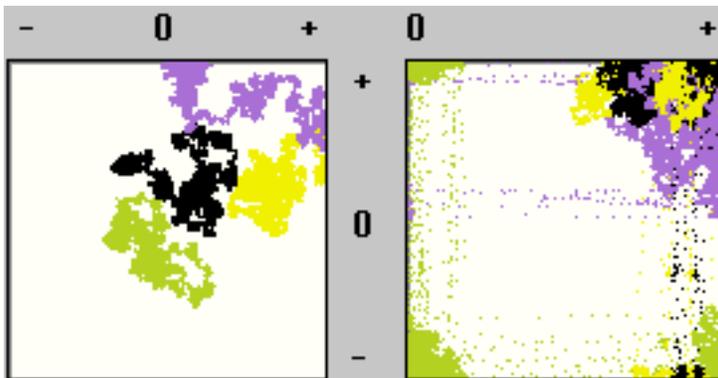


Figure 14.13: Trajectories of 4 parallel primary (left frame) and secondary (right frame) random walks. The primary ones are driven by the hyperbolic cosine distribution.

In figure 14.13, the mirrors are set to maximum position (+100 and -100) and are therefore not visible in the plot. The primary random walks follow fractal curves which are typical attractors of Brownian movement (compare

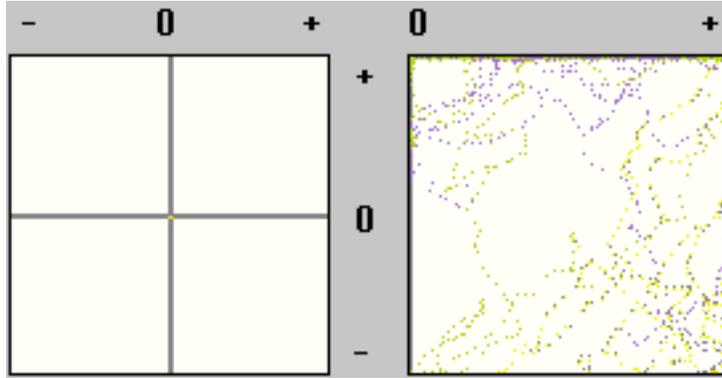


Figure 14.14: The same as figure 14.13 on the previous page, but the primary random walks (left frame) are restricted to a minimal interval of  $[-1 \cdots +1]$  by a pair of barriers in both amplitude (horizontal bar) and time increment (vertical bar).

[Man83]). Their diffusion in space is isotropic and spreads all over phase space (ergodic process). In contrast, the movements of the secondary random walks, taking the current locations of the primary ones as their input, are extremely anisotropic. When the current position of a primary random walk is close to zero (i.e. a small stepwidth value), then the secondary one is driven to one of the four corners of the phase space, depending on the sign of the step input. Locations in between are not stable (see the thin trajectories connecting the corners). They are passed through when the primary random walk crosses the border between positive and negative values. For the time offset, this behavior results in capricious pitch curves, typical for the *GENDY3* “solo voices”, staying fixed for a comparably long time and then rocketing into high or low registers.

When the primary random walks reach out for greater values, the momenta for the movements of the secondary random walks increase. If the momentum becomes significantly greater than the distance from the confining barrier, the secondary random walk jumps between two centers of “attraction” (see the pairs of spots of equal shade in the upper right corner of figure 14.13 on the previous page, right frame). This oscillating behavior (dynamic equilibrium of period 2) contributes to a supplementary richness of the sound if the spacing of the spots is narrow, or even results in a virtual polyphony of one voice split into two moving in parallel (compare figure 14.32 on page 256).

In figure 14.14, in contrast to figure 14.13 on the preceding page, the mirrors of the primary random walks are set to their minimal positions  $-1$  and  $+1$ . They close in the random walks to values within the narrow interval of  $[-1 \cdots +1]$  (represented by the crosshair lines in the left frame of figure 14.14). Therefore, the sign of the stepwidth input for the secondary random walks always changes as the primary ones are constantly thrown across the zero border by the barriers. Consequently, the secondary random walks always change direction - there is no chance for stable attractors. The result is a chaotic fluctuation in pitch and amplitude, a “buzzy” and “noisy” sound very typical for some *GENDY3*

"background voices".

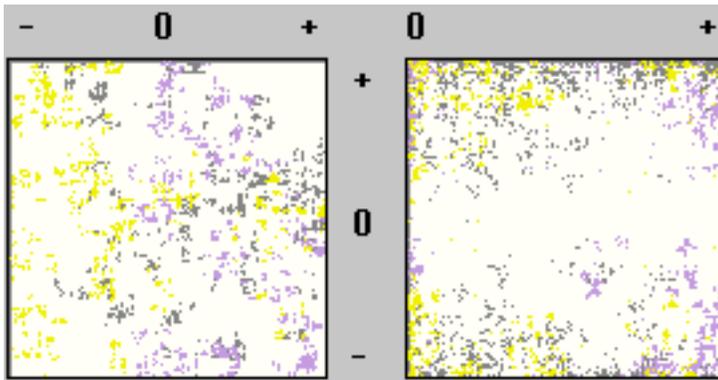


Figure 14.15: Trajectories of 4 parallel primary (left frame) and secondary (right frame) random walks. The primary ones are driven by the Cauchy distribution.

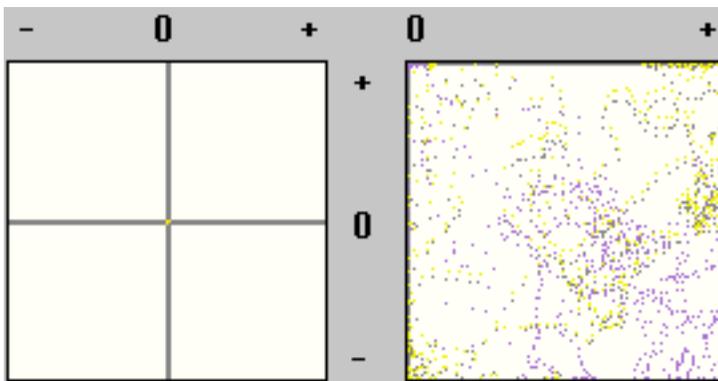


Figure 14.16: The same as figure 14.15, but the primary random walks (left frame) are restricted to a minimal interval of  $[-1 \cdots + 1]$  by a pair of barriers in both amplitude (horizontal bar) and time increment (vertical bar).

The choice of the stochastic distribution functions which govern the primary random walks also have a strong influence on the sound dynamics. Figure 14.15 is the same as figure 14.13 on page 237, except for the fact that the distribution function is changed from Hyperbolic Cosine to Cauchy. In figure 14.15, the primary random walks seem more dispersed in random walk space since the Cauchy distribution yields extreme values. The secondary random walks, in contrast, show a declining concentration toward the center. This is the effect of the barriers reflecting excesses back toward the inside. This picture corresponds to the music: the Hyperbolic Cosine distribution is responsible for the delicate slow and smooth glissandi curves in *GENDY3*. The Cauchy distribution, in

contrast, generates a pitch movement in steps, jumping from one pitch to the other with a little portamento in between.

Let us illustrate this with a graphical example, created with the New GENDYN Program. A GENDYN waveform is shown in figure 14.18 on the next page. In figure 14.17, the waveform's breakpoints are represented as billiard balls in a 2D space whose horizontal positions correspond to the spacing values. (The vertical dimension is amplitude). This is a simple example, where  $i = j$  (corresponding to the value called "rall" in a sketch by Xenakis, see figure 14.25 on page 251). In this case, the random walk space is nil and all breakpoint spacings are equal. This would be a typical situation for the fixed cluster pitches of *GENDY3*. The resulting wavelength is

$$\text{wavelength} = mi \quad (14.2)$$

where  $m$  is the number of breakpoints, and  $i (= j)$  is the spacing for all breakpoints.

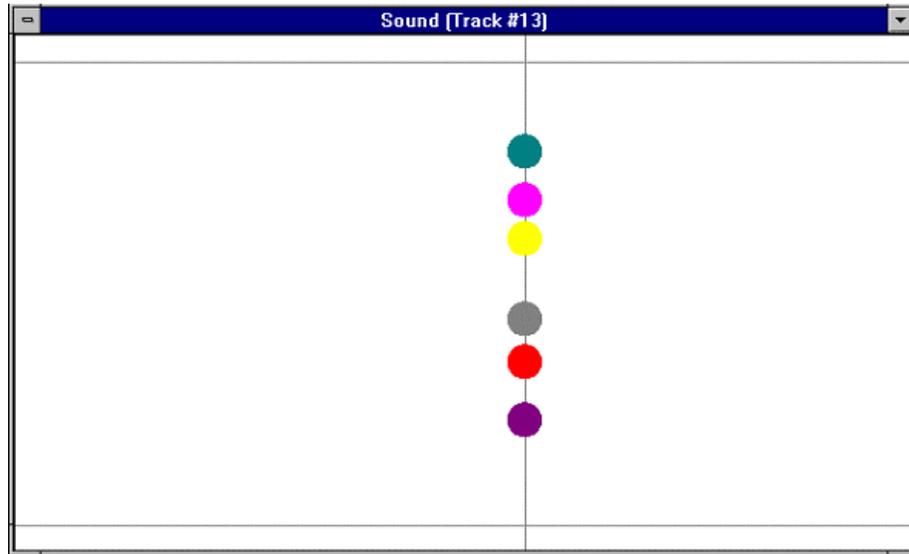


Figure 14.17: A GENDYN sound with fixed pitch, as the result of setting both time random walk barriers to the same value (see the one vertical line on which all billiard balls are aligned). The sum of all breakpoint spacings yields the length of the output waveform period and hence the frequency of the sound.

In general, however,  $j \geq i$  and the billiard balls have space to move. In figure 14.19 on page 242, we see that the billiard balls seem to be glued to either the left (i.e. minimal spacing) and right (i.e. maximal spacing) barrier confining the random walk space. All left/right combinations are possible (their probability follows the Bernoulli distribution), so in principle we could list all possible (i.e. "potential") pitches of a GENDYN scale which correspond to a given number of breakpoints  $m$  and positions  $i, j$  of the random walk barriers:

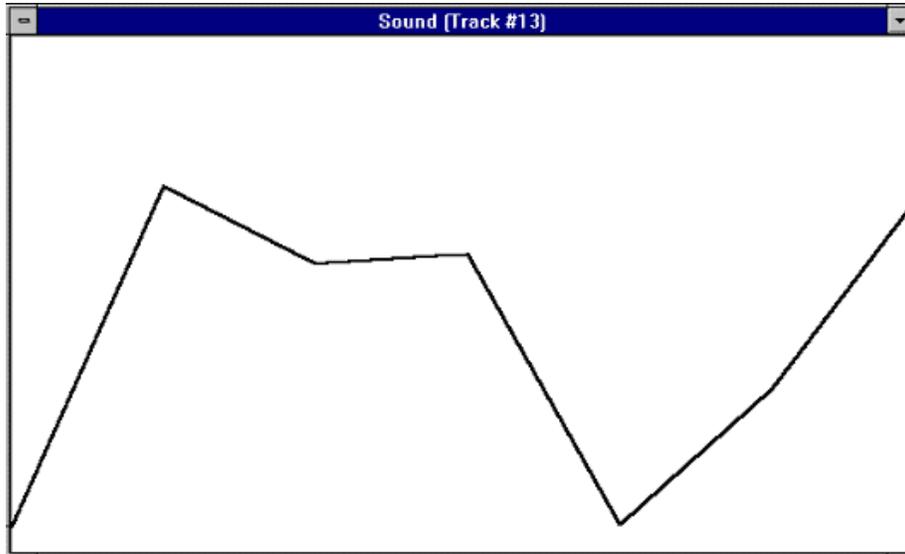


Figure 14.18: A possible waveform corresponding to figure 14.17 on the preceding page. All spacings between the 6 breakpoints are equal. The signal is periodic and the sound has a definite and stable pitch.

$$\text{wavelength } n_m = ki + (m - k)j, k = 0..m \quad (14.3)$$

However, from time to time (when random fluctuations accumulate) a billiard ball breaks free and gets attracted by the other side. The barriers are like two centers of gravitation, and the billiard balls are like planets getting caught by either of them. When billiard balls cross sides, the resulting pitch changes: it gets higher if a billiard ball crosses from right to left and lower if it crosses from left to right. If changes are slow, a glissando is perceived. If changes are sudden, interval steps are heard. The dynamics of pitch change depend on the specific distribution chosen for the steps in the primary random walk. For example, the Cauchy distribution would create discrete jumps in pitch (figure 14.29 on page 255), whereas the hyperbolic cosine distribution would create smooth glissando curves such as the one in the first section of the piece (figure 15.1 on page 277).

The musical pitch corresponding to wavelength  $n$  can be found with equation 14.1 on page 215, where  $f_s$  is the sampling frequency (44100 Hz in our case) and  $n$  the current wavelength.

### 14.13.1 Microtonal, Non-temperated Scales

The possible scales of pitched sound can be described by enumerating all possible combinations of (minimal and maximal) segment lengths within a waveform. There are  $n + 1$  combinations yielding different wave lengths ( $k$  minimal

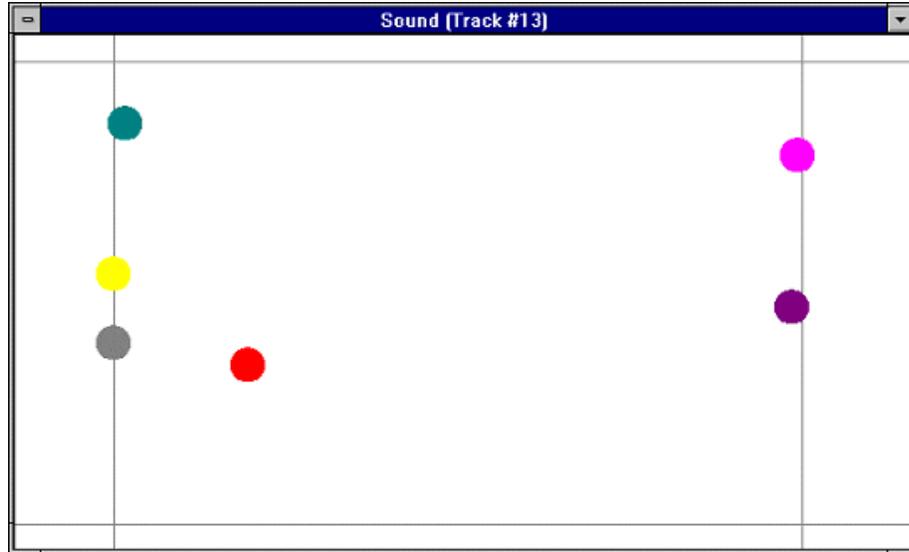


Figure 14.19: The same as figure 14.17 on page 240, except that the pitch random walk barriers are now set to different positions: four sample ticks as a minimum on the left, and 17 sample ticks as a maximum on the right. As the dynamics of 2nd order random walks drive the breakpoint spacings to either the left or right barrier, the output waveform period length is the sum of minimal and maximal breakpoints spacings. There are  $m + 1$  possible pitches,  $m$  being the number of breakpoints. One breakpoint in the figure is on its way from left to right, lowering pitch towards the next lower pitch step of the scale.

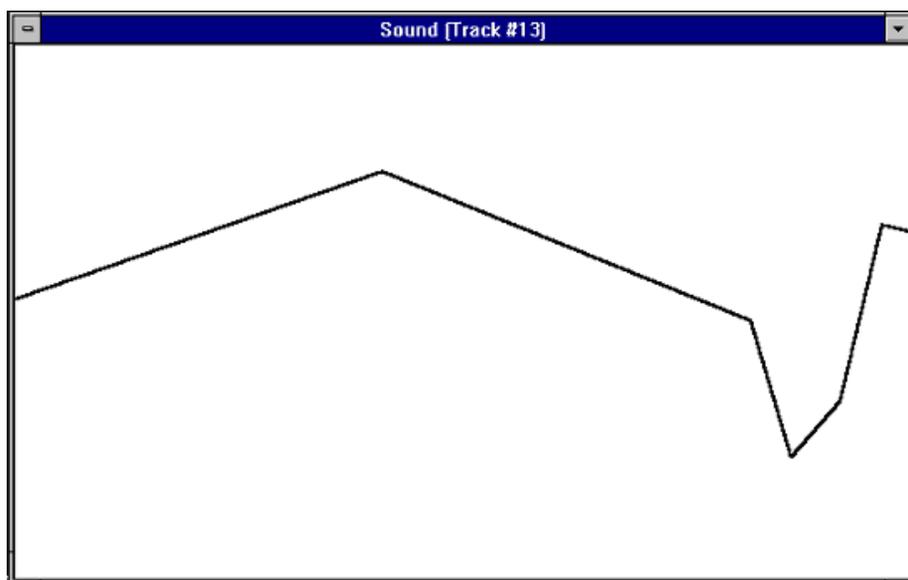


Figure 14.20: A waveform corresponding to figure 14.19 on the facing page. Two spacings are maximal, four others minimal (or close to minimal as for one of them). Note that both waveforms (figure 14.18 on page 241 and 14.20) are graphically scaled to fit into the viewing window.

and  $n - k$  maximal lengths added together, for  $k = 0..n$ ), see equation 14.3 on page 241. The stochastic “melody” thus runs through high, low, and intermediate pitches statistically distributed around a middle pitch according to the Bernoulli distribution:

$$b_n(k) = \binom{n}{k} \left(\frac{1}{2}\right)^n, k = 0, 1, \dots, n \quad (14.4)$$

if we assume a symmetrical distribution graph and a long observation time for  $n$  breakpoints thought to take either a minimal or a maximal time offset to the preceding one.

The characteristic difference of joining the “notes” of the scale by discontinuous jumps, or by portamento curves, depends on the different distribution characteristics of eg. the steep Cauchy or the smooth Hyperbolic Cosine distribution.

The stochastic deformation of a wave form’s length is the most prominent acoustical aspect of the Dynamic Stochastic Synthesis. It can be conceived of as a generalized form of Frequency Modulation (FM), a sort of stochastic FM, where the modulator is not a periodic signal, but a stochastic one.

FM, in its most basic form as sine wave FM, is a powerful sound synthesis technique and widely commercially exploited. However, more sophisticated implementations of FM, like the complex feedback FM realized on the UPIC system with its 64 wavetable oscillators ([Bel97]), or the GENDYN sound synthesis, have proved to be much more powerful than that, for they can be used as techniques for algorithmic composition in the sense of Sonological Emergence (see quotation in section 5.7.5 on page 76).

It is interesting to compare stochastic FM to complex feedback FM, where the interdependencies of multiple feedback loops build a complex nested system with UPIC’s oscillator banks that send the UPIC onto the road to chaotic oscillation, with much the same self-organizing features and timbral characteristics as experienced with the GENDYN algorithm. Both techniques enlarge the scope of classical FM towards the infrasonic time domain: they are able to generate the macroscopic time structure of a composition. In the case of the Stochastic Synthesis, all pitch movement of the sounds is controlled by stochastic FM.

The classification of different sonorities according to the movement of their fundamental frequency over time explains much of the musical potential of the Stochastic Synthesis. There are sounds with smooth and delicate glissando pitch movement that claim specific attention on the side of the listener, much in the sense of a differentiation of “solo voices” before “background voices”; then there are very agitated, “buzzing” sounds whose glissando pitch movement is too rapid to be perceived as glissandi but still perceived as pitch; then there are still more agitated sounds whose frequency fluctuations can only be perceived as a rich timbre, even noise; and finally there are static tones with a fixed, rigid fundamental frequency.

The pitch of the GENDYN sounds is correlated to the length of the stochastically transformed waveform. This wavelength is the sum of the lengths of the wave’s segments which follow independent random walks with independent step widths. The random walks are driven by different probability distributions which insert different tendencies into the random walks. For example, the wonderful smooth glissandi of *GENDY3* are obtained with the Hyperbolic Cosine. The

Logistic distribution favors greater fluctuation values, and so it generates the pitched noise typical for *GENDY3*. The exotic, microtonal scales of *GENDYN* are produced with the help of the Cauchy distribution which generates both very small and very big random walk steps. Consequently, the associated pitch movement is a discrete, stepwise melody based on the combinatorics of the wave segment's lengths, and hence a finite musical scale. The pitch stays stable for a considerable amount of time, and then leaps off to another stable pitch, generating an effect reminiscent of the "sieved" pitch structures to typical of Xenakis' late instrumental music.

The formula for the frequencies of a scale's pitches is as follows:

$$f_i = \frac{f_s}{i \frac{U_{x_\phi} + (n-i)V_{x_\phi}}{i}}, i = 1, 2, \dots, n \quad (14.5)$$

where  $f_s$  is the sampling rate,  $U_{x_\phi}$  (following Xenakis' own notation) is the lower secondary time random walk mirror, and  $V_{x_\phi}$  is the upper secondary time random walk mirror. The musical pitches are derived from these absolute frequencies by the following formula, which gives the interval distance (positive or negative) from the note a4 (concert pitch):

$$I = \log_\alpha \left( \frac{f}{f_{a4}} \right) \quad (14.6)$$

where  $\alpha$  is the geometric progression factor of the 12-tone tempered scale ( $\sqrt[12]{2}$ ) taken as the logarithmic base, and  $f_{a4}$  is the frequency of the concert pitch (typically 440 Hz). The octave  $O$  (in English notation, French and German octave denotations can be assigned accordingly), pitch class  $P$ , and cent deviation  $C$  can be easily computed from this number as its integer division by 12, the modulo-12 residuum of the nearest integer, and its fraction:

$$O = \lfloor \frac{I + \frac{1}{2}}{12} \rfloor + 4 \quad (14.7)$$

$$P = \lfloor I + \frac{1}{2} \rfloor \bmod 12, \text{ if } I \geq 0 \quad (14.8)$$

$$12 + \left( \lfloor I - \frac{1}{2} \rfloor \bmod 12 \right), \text{ if } I \leq 0 \quad (14.9)$$

$$C = \lfloor \left( (I - \lfloor I + \frac{1}{2} \rfloor) 100 \right) + \frac{1}{2} \rfloor \quad (14.10)$$

I implemented the above equations in an auxiliary analysis program that generates MUSIC<sub>T</sub><sub>E</sub><sub>X</sub> note output for all possible pitch collections available to *GENDYN* sound synthesis - see figures 14.27 on page 253 and 14.28 on page 253. The (simple) user interface of the helper program is shown in figure 14.21 on the following page.

Stochastic Frequency modulation, as implemented in *GENDYN*, has nothing in common with known modulation techniques. First, the modulated waveforms are not sinusoidal but arbitrarily complex because their shape is randomly generated. Second, the modulation signal is not sinusoidal, nor does it follow any mathematical function, as its internal variations are random and uncorrelated.

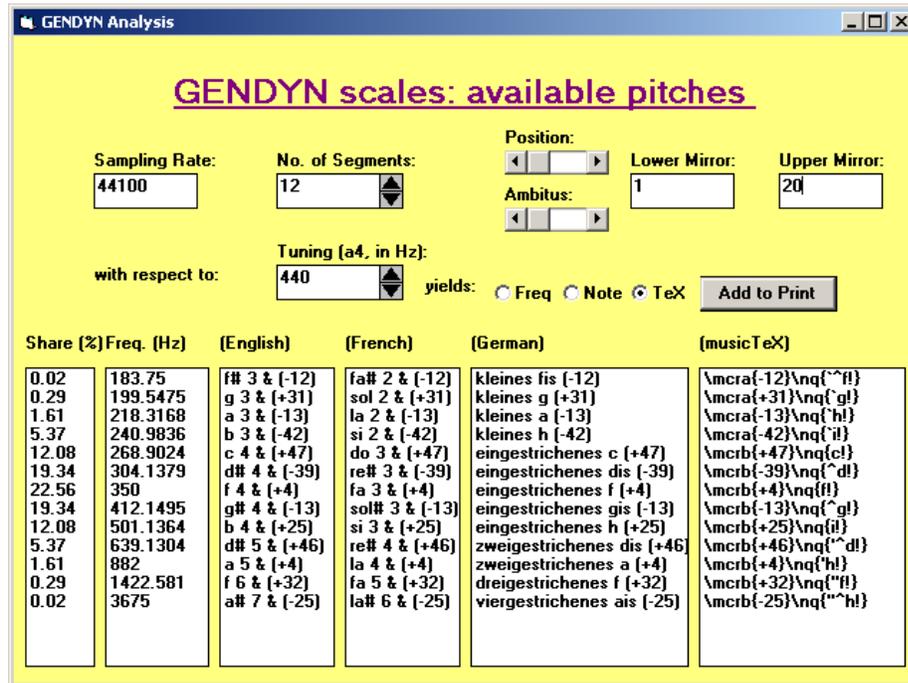


Figure 14.21: The simple graphical user interface of an auxiliary analysis program for computing GENDYN cluster and GENDYN scale pitches from a given sound synthesis parameter setting. The user chooses the parameter values for the number of waveform breakpoints and the upper / lower positions of the secondary time random walk barriers. A list of available pitches is then shown in the list boxes below. The greater the spacing of the barriers, the longer grows the list of available pitches. If the spacing is nil, only one resulting pitch is computed. Therefore, the pitches of a GENDYN cluster have to be computed one by one.

Xenakis' dynamic stochastic synthesis is beyond any standard FM theory because there is no closed mathematical formula for both the carrier and modulator time function. Timbral characteristics and pitch movement come about as an emergent by-product of accumulating random wave form fluctuations. GENDYN is therefore a good example of Sonological Emergence (see section 5.7.5 on page 76): the transformation of waveforms on the timescale of the sampling pulse code yields macroscopic musical evolutions like pitch contours, shifting spectra and sometimes even rhythmic patterns.

It is interesting to see how Xenakis achieved to create one of his most typical stylistic idiosyncrasies in a domain completely alien to macroscopic pitch composition. Even if this was not originally conceived, Stochastic Synthesis, through its self-organizing properties, confronted Xenakis with a feature of his own macroscopic musical language, as an emergent, auto-organizing effect of chaotic micro-sound synthesis.

These musical phenomena immediately stem from the technological conditions of Dynamic Stochastic Synthesis but are only accessible through a procedural, i.e. runtime analysis of the program. They are general features of music synthesized with the GENDYN program. Now, let us see how they help to shape a particular composition like *GENDY3*.

## 14.14 Exploring the Pitch Structure of *GENDY3*

GENDYN stochastic frequency modulation opens up a continuous synthesis space between stable pitch and complex timbre, between steady tones and perfect noise. Both extremes are demonstrated in *GENDY3*: section 4 (beginning at 5'27") is a combination of five noisy tracks (figure 14.24 on page 249), whereas the last section, section 11 (beginning at 17'05"), is made of just one big cluster of 15 fixed pitches, perforated by stochastically distributed pauses.

In general, agitated frequency modulation results in instable, buzzing sounds very characteristic for many tracks of *GENDY3*. The higher the speed of Stochastic Frequency Modulation, the more complex the resulting spectra, reaching up to noise.

In general, the more freedom the composer gives to GENDYN sounds, the richer gets the musical structure, but the less can the shape of these structures be controlled. For example, the composer cannot define the actual shape of a glissando curve or the exact sequence of a collection of pitches. He can define the potential ambitus of the glissando, but cannot make sure that the full ambitus will be exhausted by synthesis. He can even extract a sieve-like collection of pitches from the frequency continuum, but cannot decide when and how long each one of these pitches will appear.

### 14.14.1 Clusters in *GENDY3*

Fixed pitches are created when the random walk used to vary the frequency is reduced to a space of zero, by setting the two reflecting mirrors to equal values. Xenakis' sketch (see figure 14.25 on page 251) reveals that he was perfectly aware of individual pitches and how he could combine them into cluster configurations. Compare his notation of the 7-tone-cluster of the first section of *GENDY3* with my notation gained from the analysis of his synthesis algorithm (see figure 14.26

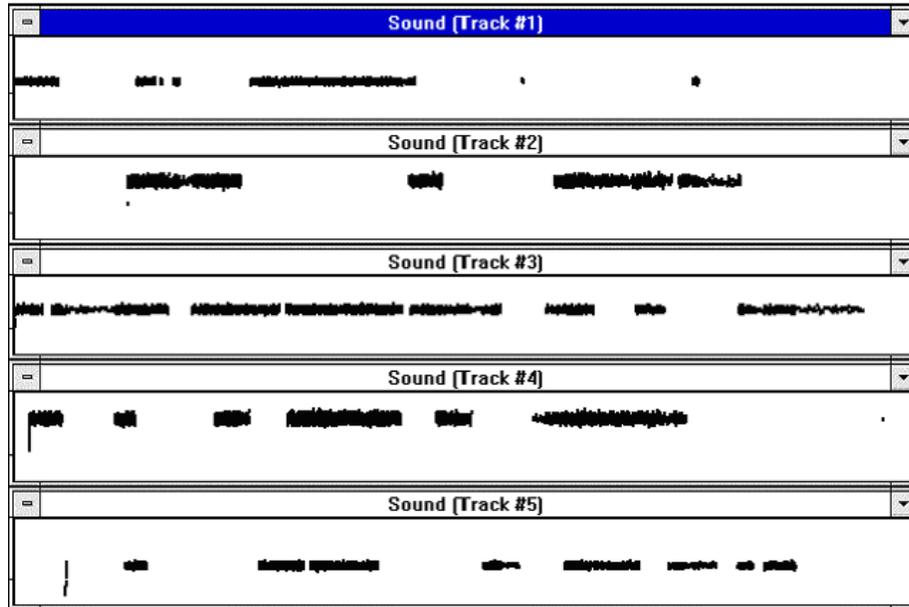


Figure 14.22: The entire section 4 (sequence # 9) of *GENDY3*. The overall sound is built of five noisy tracks, each of which has a characteristic timbre generated by heavy stochastic frequency modulation.

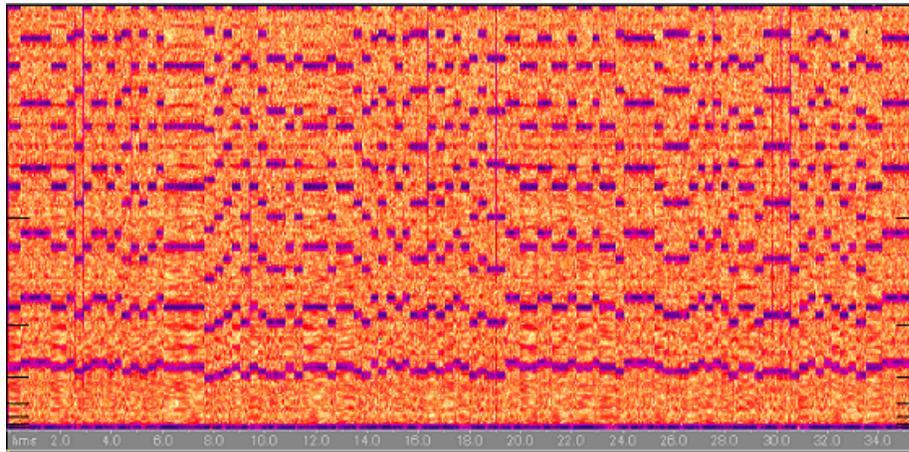


Figure 14.23: The spectral view of the “snake” hiss of seq. #1 (in-time #2), track #5 in slow motion (its stochastic FM decelerated by a factor of 1000 by 1000 times repeating the same wave length). Result: the microstructure of the noise is in fact an aleatoric “melody” in the high register (four pitches: 2594, 2755, 2940, 3151 Hz, i.e. e7, f7, f#7, g7).

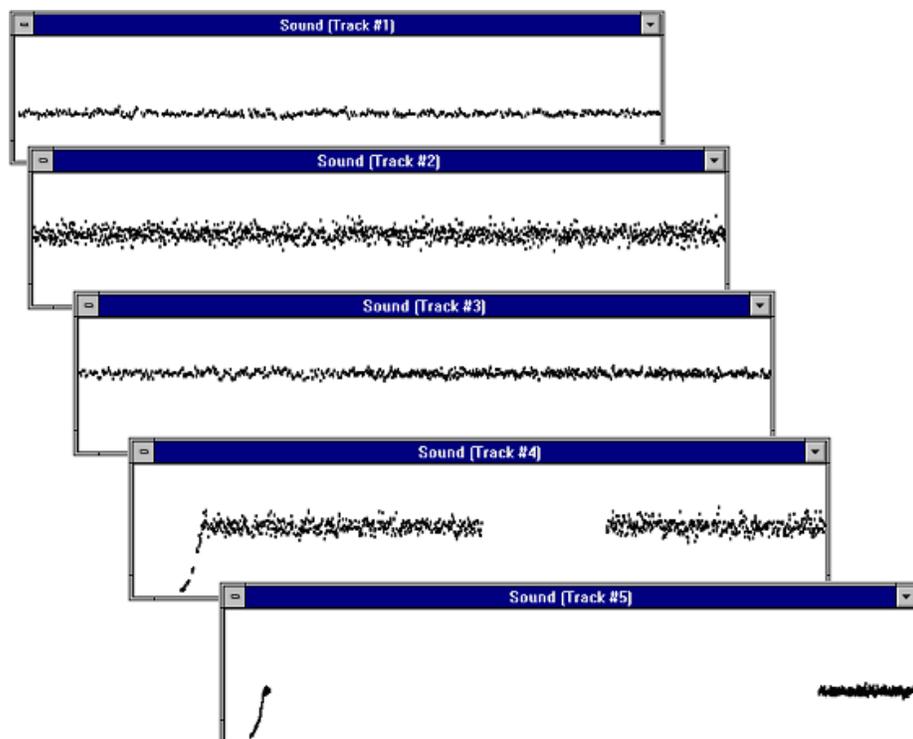


Figure 14.24: The noise under the magnifying glass (Begin of sequence #9).

Length	51 Segments	40 Segments	26 Segments	15 Segments	13 Segments
1	a 5 (-30)	c# 6 (-10) X	g# 6 (+36)	f# 7 (-12)	g# 7 (+36)
2	a 4 (-30)	c# 5 (-10)	g# 5 (+36)	f# 6 (-12)	g# 6 (+36)
3	d 4 (-32)	f# 4 (-12)	c# 5 (+34)	b 5 (-14)	c# 6 (+34) X
4	a 3 (-30)	c# 4 (-10)	g# 4 (+36)	f# 5 (-12)	g# 5 (+36)
5	f 3 (-17)	a 3 (+4)	e 4 (+50) X	d 5 (+2)	e 5 (+50)
6	d 3 (-32)	f# 3 (-12)	c# 4 (+34)	b 4 (-14) X	c# 5 (+34)
7	b 2 (+1)	d# 3 (+21)	b 3 (-33)	g# 4 (+19)	b 4 (-33)
8	a 2 (-30)	c# 3 (-10)	g# 3 (+36)	f# 4 (-12)	g# 4 (+36)
9	g 2 (-34)	b 2 (-14)	f# 3 (+32)	e 4 (-16)	f# 4 (+32)
10	f 2 (-17)	a 2 (+4)	e 3 (+50)	d 4 (+2)	e 4 (+50)
11	d# 2 (+18)	g 2 (+39)	d# 3 (-15)	c 4 (+37)	d# 4 (-15)
12	d 2 (-32)	f# 2 (-12)	c# 3 (+34)	b 3 (-14)	c# 4 (+34)
13	c 2 (+29)	e 2 (+50) X	c 2 (-4)	a 3 (+48)	c 3 (-4)
14	b 1 (+1)	d# 2 (+21)	b 2 (-33)	g# 3 (+19)	b 3 (-33)
15	a# 1 (-19)	d 2 (+2)	a 2 (+48)	g 3 (+0)	a 3 (+48)
16	a 1 (-30)	c# 2 (-10)	g# 2 (+36)	f# 3 (-12)	g# 3 (+36)
17	g# 1 (-35)	c 1 (-15)	g 2 (+31)	f 3 (-17)	g 3 (+31)
18	g 1 (-34)	b 1 (-14)	f# 2 (+32)	e 3 (-16)	f# 3 (+32)
19	f# 1 (-28) X	a# 1 (-7)	f 2 (+39)	d# 3 (-9)	f 3 (+39) X
20	f 1 (-17)	a 1 (+4)	e 2 (+50)	d 3 (+2)	e 3 (+50)

Table 14.1: A small cut of Xenakis’ decision space for cluster aggregates (Xenakis’ choices are marked by a capital ‘X’) of the 7-note-cluster for Sequence #10 (in-time #1) (sampling rate=44.1kHz, a 4=440Hz). Note that there are much less possible pitches for fixed pitches than for pitches in stochastic scales (as seen in this table and table 14.3 on page 257), because pitch is fixed by forcing all segments of the wave form to have the same “length” (i.e. number of interpolated samples between wave breakpoints).

on page 252). In contrast, he was not able to define very low or very high pitches (see “grave” and “aigu” in his sketch) or to define the exact contours of freely moving pitch. His arrow reading “desc.[ending]” denotes the solo voice whose pitch scale and contour can now be perfectly described (see the pitches of the scale in table 15.11 on page 271 and the plot of its pitch contour in figures 15.1 on page 277 and 14.11 on page 232).

Note that with zero time random walk space, there are much less possible pitches than if the different segment lengths can combine freely within a non-zero time interval. For the cluster of the first section of *GENDY3*, Xenakis used pitches with 13 breakpoints à 3 samples for track #1, 26 à 5 samples for track #3, 51 à 19 for track #4, 13 à 19 each for track #5, 15 à 6 for track #9, 40 à 1 each for track #14, and 40 à 13 for track #16, marked with an “X” in table 14.1. (Track #13 has moving pitch; the others are silenced.)

Interestingly enough, when realizing *GENDY3*, Xenakis obtained fixed pitch in a different way, forcing his program to bypass the random walks altogether, directly specifying the length of waveform segments. Seemingly, he was so much interested in fixed pitch that, in order to get it, he sacrificed the stochastic “purity” of his program, not being able to see that in fact Stochastic Frequency Modulation contains fixed pitch as a special case (a random walk whose space

s?	degr%	I max	nall	PARAG310.BAS
398	1	13	3	
382	2	13	8	
399	3	26	5	
385	4	51	19	two groups
386	5	13	19	
? 388	6	40	29	two groups
387	7	13	14	
389	8	6	4	
400	9	15	6	
391	10	16	6	
392	11	16	0	wavy desc.
393	12	6	0	wavy
394	13	40	1	
395	14	40	9	
396	15	40	9	
397	16	40	13	

Figure 14.25: A sheet from Xenakis' sketchbook at CEMAMu, documenting the pitch parameters utilized for the first section of *GENDY3*. These were coded into the BASIC program PARAG310.bas. In *GENDY3*, only tracks 1, 3, 5, 9, 13, 14, and 16 are sounding, all others being muted.

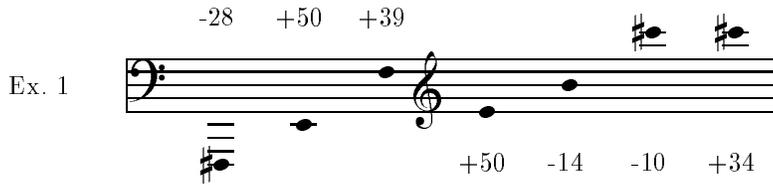


Figure 14.26: The pitches of the “perforated” 7-tone cluster underlying the opening section of *GENDY3*. These as well as other pitches can be computed from the sound synthesis parameter settings of the opening section of *GENDY3* (sequence #10) as shown in table 15.11, track no. 13, with the help of the program shown in figure 14.21 on page 246, complementing and partly correcting Xenakis’ own studies on GENDYN pitch (as seen in his sketch 14.25 on the previous page).

is reduced to nil).

Cluster pitches are characterized by the fact that all breakpoint spacings have same length. As a consequence, all possible GENDYN cluster pitches can be enumerated by summing up multitudes of the number of a waveform’s breakpoints according to formula 14.2 on page 240 (see figure 14.27 on the facing page). Therefore, the clusters which Xenakis actually composed can be pointed out as pitch subsets within the universal set of all GENDYN cluster pitches. In this way, Xenakis’ unique creation is put into the context of the possibilities offered by Stochastic Synthesis as such. The systematization of all available cluster and scale configurations outlines the complete decision space, of which Xenakis only knew a small but interesting part.

As can be seen from the tables, a certain repertoire of non-temperated pitches is “hard coded” into the algorithm, and actually directly depends on the sampling rate and its different subdivision permutations. It is strange to see here how the arbitrary conventional consumer-oriented rate of 44.1 kHz finally decides upon the pitches used in a musical artwork!

Xenakis did provide for a possible change of the sampling rate in his program, an option which was actually open to him due to the custom digital-to-analogue converter at CEMAMu which would have permitted him to experiment with different sampling rates. However, there is no evidence that Xenakis used this option with GENDYN.

#### 14.14.2 Scales in *GENDY3*

The most striking emergent feature of the Stochastic Synthesis in general and *GENDY3* in particular is the emergence of clearly discernible pitch scales. The pitch set of a GENDYN scale is determined by the position of the barriers defining the pitch random walk space. Pitches come about as a result of a combination of breakpoint spacings of different size within a waveform period. Therefore, the pitches included in scales are in general others than those included in clusters. For an example, please compare figures 14.27 on the next page



Figure 14.27: In this example of cluster pitches, the number of breakpoints is 10. From left to right, all spacings of the waveform are set to 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, respectively. Deviation from noted pitch is shown in cent values. The same can systematically be done for all other numbers of breakpoints greater than 1. Please note the repetition of some pitches in different octaves.

and 14.28. Also, it can be shown that there are more scale pitches than cluster pitches ([Hof01a]).

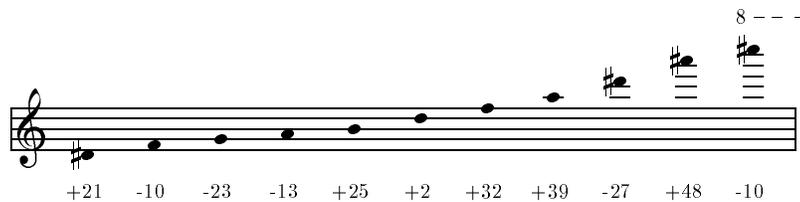


Figure 14.28: In this example of scale pitches, 10 breakpoints yield 11 available pitches between the barrier positions 14 and 1. That is, lowest and highest pitch are equal to figure 14.27, but pitches in between follow a different, notably a non-octavating, plan. Deviation from noted pitch is shown in cent values.

I once presented *GENDY3* to a group of music students at the beginning of the GENDYN Project, before I started the analysis part. I told the students that all sound was stochastically computed, and that all structure they were going to hear were consequently by chance. After listening, the students were convinced that I had told them complete nonsense, because such a concentration of diminished fifths could, in their ears, never be the result of chance! Knowing the functioning of GENDYN now, one can fairly say that both of us, the students and me, were actually right. The algorithm does work with random numbers, but the auto-organizing process of Dynamic Stochastic Synthesis crystallizes pitch movement into a well-defined “atonal” pitch set which is defined by the combinatorics of wave segments (cf. the ‘mirror-chord  $f\sharp-c-f-b$  as a subset of scale no. 7 in table 14.2 on the next page).

# seg.	k = 0	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10
1	c# 7 (-10)	f 11 (-23)									
2	c# 6 (-10)	c 7 (+6)	f 10 (-23)								
3	f# 5 (-12)	c 6 (+47)	b 6 (+25)	a# 9 (-25)							
4	c# 5 (-10)	f# 5 (-40)	c 6 (+6)	a# 6 (+48)	f 9 (-23)						
5	a 4 (+4)	c# 5 (-31)	f 5 (+32)	c 5 (-35)	a# 6 (-25)	c# 9 (-10)					
6	f# 4 (-12)	a 4 (-13)	c 5 (+47)	f 5 (+4)	b 5 (+25)	a 6 (+4)	a# 8 (-25)				
7	d# 4 (+21)	f# 4 (-26)	a 4 (-30)	c 5 (+27)	f 5 (-23)	b 5 (-14)	g# 6 (+36)	g 8 (+8)			
8	c# 4 (-10)	d# 4 (+9)	f# 4 (-40)	a 4 (-47)	c 5 (+6)	e 5 (+50)	a# 5 (+48)	g# 6 (-29)	f 8 (-23)		
9	b 3 (-14)	c# 4 (-21)	d# 4 (-3)	f 4 (+46)	g# 4 (+36)	c 4 (-15)	e 5 (+23)	a# 5 (+11)	g 6 (+8)	d# 8 (-27)	
10	a 3 (+4)	b 3 (-23)	c# 4 (-31)	d# 4 (-15)	f 4 (+32)	g# 4 (+19)	c 4 (-35)	e 5 (-3)	a# 5 (-25)	f# 6 (+47)	c# 8 (-10)

Table 14.2: Available GENDYN scales

A table of available GENDYN scales for time random walks with barriers fixed at 1 and 20 sample ticks, respectively, for 1, 2, ..., 10 wave segments (sampling rate=44.1kHz, a4=440Hz). No. 7 is the scale of Sequence #2 (in-time #3), Track #2.

If the change over time of the fundamental frequency is not too fast, a musical pitch is perceived (e.g. stable or moving glissando), and the plot picture is a more or less smooth line. If the fundamental moves faster, a kind of nervous “buzz” is perceived, and the plot shows a thick line or a comb structure within the pitch variation interval. Even greater modulation of the fundamental frequency generates the complex effects of frequency modulation which extend into the realm of noise. However, if the ambitus of the frequency modulation is not too wide, a kind of “pitched” noise is perceived which actually is nothing else than an “aleatoric melody” on a fixed pitch scale, but accelerated beyond the threshold of pitch perception (see below, section 14.14.3). In all these cases, the plotting of the fundamental frequency conveys a very typical picture of the acoustic phenomenon. It is at the same time more general and more precise than any symbolic musical notation (i.e. notes in staves).

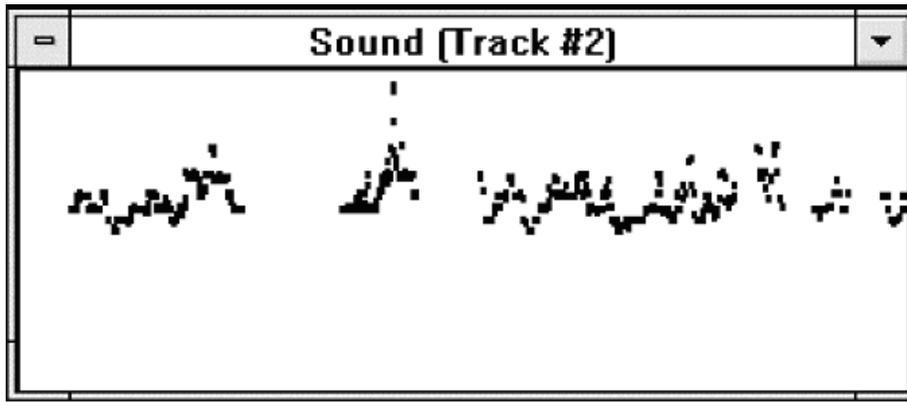


Figure 14.29: Sequence #2 (in-time #3): Stochastic scale (track #2): The intervals become wider from low to high: first a diminished chord (d#3, f#3, a3, c4), then a 4th (f4), a dim. 5th (b4), a 6th (g#5+36 cents), and (almost) a double octave (g7). (Pitch nuances of less than an 8th note ( $\leq 25$  cents) are ignored.)

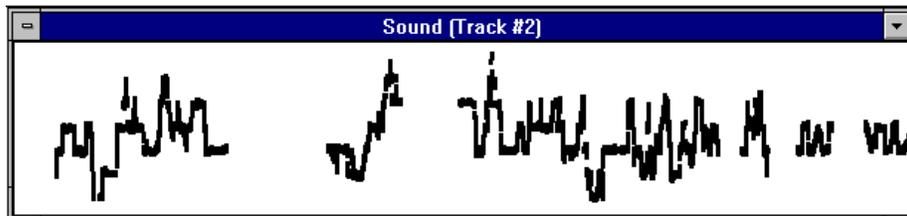


Figure 14.30: Sequence #2 (in-time #3): Stochastic scale (track #2): a clear view of the scale pattern inherent in the pitch movement.

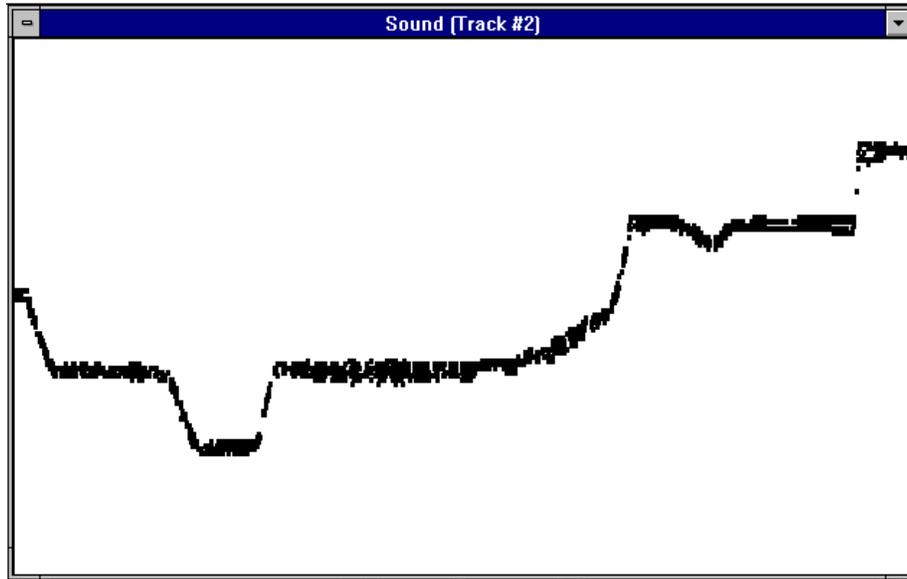


Figure 14.31: Sequence #2 (in-time #3): Stochastic scale (track #2): the magnification shows a virtual heterophony: the pitch jumps between 2 positions creating a parallel 2nd voice.

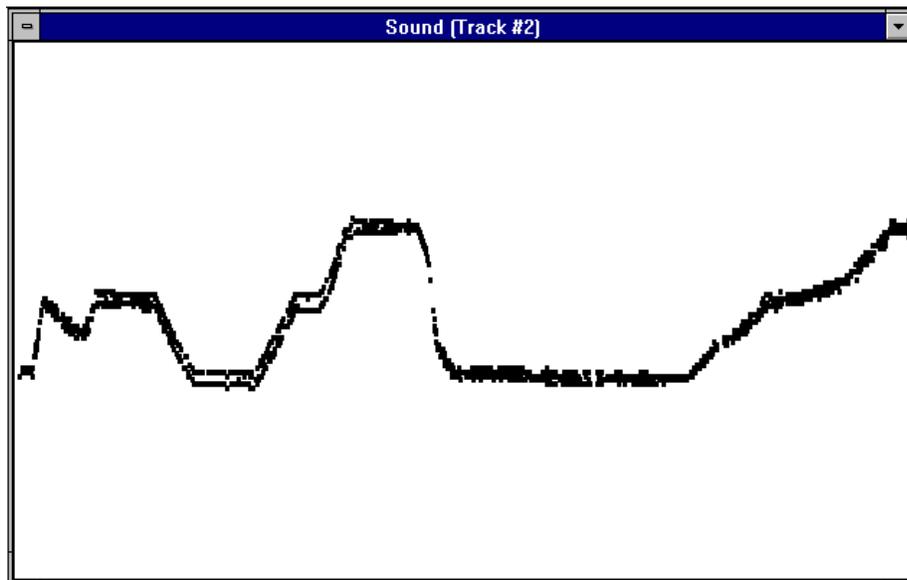


Figure 14.32: Sequence #2 (in-time #3): Stochastic scale (track #2): another magnification of the same track.

Upp. Mirr.	k = 0	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10
1	c# 8 (-10)										
2	c# 7 (-10)	d 7 (-21)	d# 7 (-27)	e 7 (-28)	f 7 (-23)	f# 7 (-12)	g 7 (+8)	g# 7 (+36)	a# 7 (-25)	b 7 (+25)	c# 8 (-10)
3	f# 6 (-12)	g 6 (+8)	g# 6 (+36)	a# 6 (-25)	b 6 (+25)	c# 7 (-10)	d# 7 (-27)	f 7 (-23)	g 7 (+8)	a# 7 (-25)	c# 8 (-10)
4	c# 6 (-10)	d 6 (+25)	e 6 (-28)	f 6 (+32)	g 6 (+8)	a 6 (+4)	b 6 (+25)	d 7 (-21)	f 7 (-23)	g# 7 (+36)	c# 8 (-10)
5	a 5 (+4)	a# 5 (+48)	c 6 (+6)	d 6 (-21)	e 6 (-28)	f# 6 (-12)	g# 6 (+36)	b 6 (+25)	d# 7 (-27)	g 7 (+8)	c# 8 (-10)
6	f# 5 (-12)	g 5 (+39)	a 5 (+4)	b 5 (-14)	c# 6 (-10)	d# 6 (+21)	f# 6 (-12)	a 6 (+4)	c# 7 (-10)	f# 7 (-12)	c# 8 (-10)
7	d# 5 (+21)	f 5 (-23)	f# 5 (+47)	g# 5 (+36)	a# 5 (+48)	c# 6 (-10)	e 6 (-28)	g 6 (+8)	b 6 (+25)	f 7 (-23)	c# 8 (-10)
8	c# 5 (-10)	d 5 (+49)	e 5 (+23)	f# 5 (+17)	g# 5 (+36)	b 5 (-14)	d 6 (-21)	f 6 (+32)	a# 6 (-25)	e 7 (-28)	c# 8 (-10)
9	b 4 (-14)	c 5 (+47)	d 5 (+25)	e 5 (+23)	f# 5 (+47)	a 5 (+4)	c 6 (+6)	e 6 (-28)	g# 6 (+36)	d# 7 (-27)	c# 8 (-10)
10	a 4 (+4)	b 4 (-33)	c 5 (+47)	d 5 (+49)	f 5 (-23)	g 5 (+39)	a# 5 (+48)	d 6 (+25)	g 6 (+8)	d 7 (-21)	c# 8 (-10)

Table 14.3: A table of available GENDYN scales for 10 breakpoints (fixed), the lower time random walk barrier set to 1 (fixed) and the upper one set to 1, 2, 3, ..., 10 sample ticks (sampling rate=44.1kHz, a 4=440Hz).

### 14.14.3 Noise in *GENDY3*

*GENDYN* noise is produced as a result of fast stochastic FM. I discovered this when playing with a slowing modulation parameter that Xenakis had built into a later version of his program. Increasing the value of this parameter transformed noisy sounds into stochastic pitch contours. In other words, *GENDYN* noise is glissandi or stepwise pitch movement “melodies” accelerated beyond the threshold of pitch perception. In section 4 of *GENDY3*, five of such noisy sounds are combined, forming a kind of apotheosis of noise lasting more than two minutes (figure 14.22 on page 248).

The zooming-in of this specific kind of noise reveals its true nature as being a result of stochastic FM on the microsound level. Figure 14.24 on page 249 is a spectral view of the resulting noisy FM, slowed down by a factor of 1000, resulting in a kind of aleatoric chromatic figure of four notes in a high register. I call the specific sonority of this sound “snake hiss” (for want of a better nickname). It is actually due to a program bug in Xenakis’ original program. Of the hundreds of variables in his *GENDYN* implementation, Xenakis forgot to declare one which was to sum up three successive values of the time random walk in order to divide them by three: a simple FIR filter meant to average the successive lengths of 3 breakpoints. This variable, being local in scope, was implicitly re-declared by the BASIC run-time system with value 0 each time. The effect is a division by 3 of all the time increments between breakpoints, rising the frequency band of the stochastic noise into the spectral region of hiss.

We clearly see that *GENDYN* noise is, in fact, nothing else than stochastic pitch movement, accelerated beyond the threshold of perception. It was Xenakis himself who thought of a means to slowing down his stochastic FM by introducing an additional parameter for repeating a wave form unchanged  $n$  times in a later version of *GENDYN* (the one which served to compose *S709* by the end of 1994). This parameter has been retained in the implementation of the New *GENDYN* program in order to be able to generate slow-motion pictures of the synthesis.

## 14.15 Graphical Representation of the Resynthesis

The capricious pitch movement of the individual sound layers of *GENDY3* can be visualized individually or in “full score”. The New *GENDYN* Program, in addition to the duration structure of the superimposed sounds, offers the option to output the instantaneous fundamental frequency of the sounds during synthesis, defined as their momentary wave length. This information is not directly taken from the algorithmic action of the program but derived from it as the sum of all the wave segment’s lengths used to construct the present waveform.

### 14.15.1 The “Chirping Machine”

Just as sequence no. 9 (in-time #4), the “apotheosis of noise”, can be regarded as a kind of acoustic audification of the mathematical formula of logistic distribution, sequence no. 8 (in-time #6) is completely governed by one other single distribution formula: the “Cauchy distribution”. All 16 tracks are driven by this

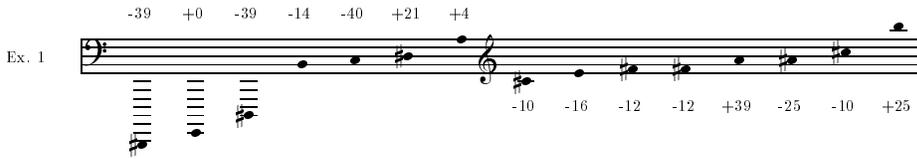


Figure 14.33: The 15-tone cluster of seq. #6 at the end of *GENDY3*.

distribution formula, which is characterized by a very steep distribution graph. Roughly 68% of the values (those within the Gaussian standard deviation interval of probability of ca.  $[0.16 \cdots 0.84]$ ) are contained within the small interval of  $[-0.002 \cdots + 0.002]$ , for a typical coefficient of 0.001 used by Xenakis.

Such small random steps must leave the momentary random walk position unaffected which means that all wave segments keep their momentary width (in sample ticks) and pitch is stable. However, in rare cases (ca. 0.1% or 1:1,000) values exceed the interval  $[-0.5 \cdots + 0.5]$  which makes it more probable that some of these bigger steps in the same direction accumulate to a value that will change the pitch of the sound. A spontaneous increment or decrement of 1 sample tick is statistically created every 1,667 drawings from the Cauchy-distributed random number generator (0.06% or ca. 1:1,667). For  $a4 = 440$  Hz and 7 wave segments, this will statistically happen every 0.5 seconds, according to formula 14.11.

$$\left(\frac{440 \cdot 7}{1667}\right)^{-1} = 0.541233766 \tag{14.11}$$

The values of the Cauchy distribution increase dramatically toward very rare cases. In our example, one of one billion drawings yields a value of  $\pm 31,840$  which represents almost a maximal short integer. Fortunately, this large number is broken down to a reasonable size by the random walk mirrors.

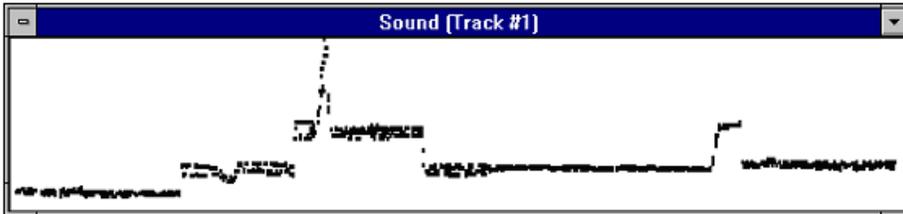


Figure 14.34: A zoom into one of the noisy chirping tracks of sequence #8 (in-time #6), 0'48" - 0'50" reaching to extreme pitches at the border of perception.

If the mean frequency and the rate of change are slow enough, the result

are clearly discernible steps of non-temperated scales. If the “speed” of these stochastic “modes” exceeds a certain time threshold, scale structure is transformed into timbre micro-structure. The typical FM timbre of the Cauchy distribution is a high-frequency chirping, with a strong component of “wind” noise added to it. To convey a very personal impression, the dense chirping of this sequence gives me the feeling of looking into an agitated swarm of insects dancing in glaring sunlight.

## Chapter 15

# Description of *GENDY3*

*GENDY3* is built out of 11 sections (“sequences” in Xenakis’ terms) which are well differentiated by the combination of sonic characters they contain. It is fairly easy to tell one sequence from the other. Yet, it is not always so easy to tell exactly when one sequence ends and the other starts. With the resynthesis of the piece, the delineation of the piece’s sections becomes a matter of generating a sequence at a time and playing them back one after the other. Then, *GENDY3* can be either played in one go or its sections be played individually with random access. In this way, the order of the sequences can be altered and compared to the order Xenakis chose. In other words, the simple re-synthesis of the music is already a first analytic action.

As for Xenakis, it is very probable that he tried different orderings. The ordering  $\phi$  of the sequences’ ID numbers (called  $\psi$  in the GENDYN program) seems to indicate that Xenakis worked this way: 10, 1, 2, 9, 3, 8, 4, 5, 7, 11, 6.

The idea of analysis by resynthesis can be taken one step further, from the horizontal ordering of the sequences into the (vertical) layering of the sonic tracks. After taking the piece apart into a succession of its sequences, the sequences in turn can be taken apart by individually re-generating each of their track layers. Again, the music can be listened to unchanged, if one chooses to play back the collection of tracks over a digital multi track system. But now it becomes possible to attenuate or mute distinct tracks, to amplify others, to explore the sonic depth of the piece or even, say, project the tracks into space by assigning them to 16 different loudspeakers.

Each of the 11 sequences is composed of up to 16 sound tracks, similar to the tracks of a multitrack tape. In each sequence, the number of sound tracks may differ (in *GENDY3*, there are examples for 5, 8, 11, and 16 simultaneous tracks). On synthesis, these tracks are created and simultaneously mixed down to the “master track” of the resulting sound file. It is possible to hear the presence of several sound layers, but it is not easy to really tell how many of them are there exactly, as well as which sonic event comes from which track exactly, especially if some tracks share similar sonic characteristics.

Sometimes, there are sonic events which are composed by several tracks which have similar sonic characteristics and whose sound patches are adjacent or overlapping at this specific point in time. This is e.g. the case at the very beginning of the piece. The start of the glissando which descends from the highest heights is musically an entity, but physically consists of a fortuitous

In-time no. $\phi$	Outside-time no. $\psi$	Nickname	Duration	Timing
1	10	The Exposition	1'27"	0'0"-1'27"
2	1	The Harbor	2'39"	1'27"-4'06"
3	2	Heavy Metal	1'20"	4'06"-5'26"
4	9	Apotheosis of Noise	2'08"	5'26"-7'34"
5	3	Foggy Atmosphere	1'56"	7'34"-9'30"
6	8	Twittering Machine	2'15"	9'30"-11'45"
7	4	Buzzy Insects	1'33"	11'45"-13'18"
8	5	Rough Seascape	1'44"	13'18"-15'20"
9	7	The Sirenes	1'59"	15'20"-17'01"
10	11	Plein jeu	1'16"	17'01"-18'17"
11	6	Stochastic Organ	1'37"	18'17"-19'54"

Table 15.1: Overview of *GENDY3*

The 11 sections of *GENDY3*. The nicknames are explained in section 15.2 on page 274.

constellation of three sound tracks: 1, 13 and 14 (see figure 15.1 on page 277).

Let us have a look at all these different sounds to gain an impression of their sounding “material”. We will see that a number of sonic characters are always there, but never identical. We can therefore group the sounds into sonic “families” whose combinations determine the piece. One has the impression of timbral personalities which play their musical role in *GENDY3* much like actors play their role in a theater play. It is important to note, however, that physically these sound families are not disparate phenomena but just condensation poles in a sonic continuum reaching from perfect noise to poor frozen tones. There is no a priori grouping of sounds inherent in the synthesis procedure — timbral qualities of sound come about as a constellation of parameter settings in a more or less continuous parameter space. This parameter space is not at all exhausted by the sounds of *GENDY3*: there are more possible sounds the GENDYN program can create. The systematization into sound families is a retrospective activity where, upon analysis, classes of sound are extracted from an existing composition.

If one looks at the numeric parameter values Xenakis chose for the tracks of *GENDY3*, one cannot help having the impression that many of these are just initial guesses, instead of empirically proven values. Many of these values are simple decimal fractions and these are repeated across several sounds. This is because Xenakis could not work interactively with his GENDYN program — it was just too slow. Xenakis had to come back the next day into his office to listen to the result of his trials. He could not establish a direct feedback loop which would have helped to fine-tune timbral qualities. Given these circumstances, it is all the more surprising that Xenakis got such a broad variety of sounds out of his program. Most of his guesses turned out to be good guesses. There are very few “accidents” in the parameter settings, e.g. sounds that are too weak or sounds that die out over the course of a track. We can see examples for this in the synopsis tables (see tables 15.2 on the next page to 15.12 on page 272).

Despite of all technical difficulties, it is most probable that Xenakis developed a musical intuition for the sounds of GENDYN, how they met his aesthetic preferences (e.g. glissandi, scales, richness and spectral ruggedness), and how these depended on certain parameter settings. Therefore, studying the sonic

No.	#segs	lower	upper	distr	coef	fluct	freq.(Hz)	pitches	description
2	40	58	58				18.7	d0+31	beyond infra-sound
13	16	13	13				212.019	g#3+36	cluster
16	16	13	13				212.019	g#3+36	cluster
12	16	12	12				229.687	a#3-25	cluster
11	16	11	11				250.568	b3+25	cluster
10	16	10	10				275.625	c#4-10	cluster
15	16	9	9				306.25	d#4-27	cluster
9	16	9	9				306.25	d#4-27	cluster
14	16	8	8				344.531	f4-23	cluster
4	19	0	30	arcsin	1.0	+/-2		d#2-9 ... e2+10	bass drone
1	12	0	20	arcsin	1.0	+/-2		f#3-12 ... g#3-29	modulated pitch
3	11	0	20	hypcos	.1	+/-2		g3+39 ... a3+21	modulated pitch
5	13	0	20	logistic	1 / 6	+/-4		b6+25, d7-21, f7-23, g#7+36	snake sizzling
7	16	0	30	cauchy	.001	+/-1	91.875, 97.78271, 104.5024, 112.2137, 121.1538, 131.6418, 144.1176, 159.2058, 177.8226, 201.3699, 232.1053, 273.9131, 334.0909, 428.1553, 595.9459, 980, 2756.25	f#2-12, g2-4, g#2+11, a2+34, b2-33, c3+11, d3-32, d#3+40, f3+32, g3+47, a#3-7, c#4-21, e4+23, a4-47, d5+25, b5-14, f7-23	"solo" glissando voice
8	15	0	20	cauchy	.1	+/-2	147, 156.9395, 168.3206, 181.4815, 196.875, 215.1219, 237.0968, 264.0719, 297.973, 341.8605, 400.9091, 484.6154, 612.5, 832.0755, 1297.059, 2940	d3+2, d#3+15, e3+36, f#3-33, g3+8, a3-39, a#3+30, c4+16, d4+25, f4-37, g4+39, b4-33, d#5-27, g#5+3, e6-28, f#7-12	buzz (rapid gliss./large ambitus)
6									muted

Table 15.2: *GENDY3* (Seq #1 / in-time #2)

In this as in the following tables, tracks are ordered systematically, not in their numeric ordering, as can be seen in the first column "No.". "#segs" means "number of [wave form] segments", "lower" and "upper" are the positions of the secondary time random walks, "distr" and "coef" apply to the stochastic distributions driving them, "fluct" designates the positions of the primary time random walks. Note the occurrence of relatively stable pitch in spite of open random walk spaces in tracks #1, #3 and #4 through oscillations (by the fluctuation value) against the upper barrier. Comparable situations are also to be found in several tracks of other sequences.

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
1	40	3	3				367.5	f#4-12	cluster, rough ampli- tude
2	7	0	20	cauchy	.001	+/-2	315, 364.4628, 432.3529, 531.3253, 689.0625, 980, 1696.154, 6300	d#4+21, f#4-26, a4-30, c5+27, f5-23, b5-14, g#6+36, g8+8	"solo" porta- mento voice
3	11	0	20	arcsin	3.0	+/-2	200.454 ... 222.727	g3+39 ... a3+21	modulated pitch
4	10	4	4				1102.5	c#6-10	cluster, bril- liant cauchy amplitude
5	49	0	20	logistic	1 / 6	+/-2	45 ... 900	f#1-47 ... g1+35	bass drone, modulated pitch
6	40	29	29				38.017	d#1-39	cluster, slow hycos am- plitude evo- lution
7									muted
8									muted
9									muted
10									muted
11									muted
12									muted
13									muted
14									muted
15									muted
16									muted

Table 15.3: *GENDY3* (Seq #2 / in-time #3)

No.	#segs	low	upp	distr	coef	fluct	freqs	pitches	description
7	40	58	58				17.7		infrasonic (partials audible)
16	23	12	12				159.7826	d#3+46	cluster
14	16	10	10				275.625	c#4-10	cluster
10	19	7	7				331.5789	e4+10	cluster
15	14	9	9				350	f4+4	cluster
13	13	9	9				376.9231	f#4+32	cluster
9	30	3	3				490	b4-14	cluster
12	10	6	6				735	f#5-12	cluster
11	7	7	7				900	a5+39	cluster
8	23	0	20	logistic	1 / 6	+/-2	95.6895 ... 106.5217	g2-38 ... g#2+44	modulated pitch
5	14	0	20	logistic	1 / 6	+/-2	157.5 ... 175	d#3+21 ... f3+4	modulated pitch
3	4	0	20	arcsin	3.0	+/-2	551.25 ... 612.5	c#5-10 ... d#5-27	modulated pitch
2	6	0	20	cauchy	.001	+/-2	367.5, 436.6337, 537.8049, 700, 1002.273, 1764, 7350	f#4-12, a4-13, c5+47, f5+4, b5+25, a6+4, a#8-25	"solo" portamento voice
4	16	0	30	cauchy	.001	+/-1	91.875, 97.78271, 104.5024, 112.2137, 121.1538, 131.6418, 144.1176, 159.2058, 177.8226, 201.3699, 232.1053, 273.9131, 334.0909, 428.1553, 595.9459, 980, 2756.25	f#2-12, g2-4, g#2+11, a2+34, b2-33, c3+11, d3-32, d#3+40, f3+32, g3+47, a#3-7, c#4-21, e4+23, a4-47, d5+25, b5-14, f7-23	"solo" portamento voice (ambitus is restricted)
6	15	0	20	cauchy	.1	+/-2	147, 156.9395, 168.3206, 181.4815, 196.875, 215.1219, 237.0968, 264.0719, 297.973, 341.8605, 400.9091, 484.6154, 612.5, 832.0755, 1297.059, 2940	d3+2, d#3+15, e3+36, f#3-33, g3+8, a3-39, a#3+30, c4+16, d4+25, f4-37, g4+39, b4-33, d#5-27, g#5+3, e6-28, f#7-12	agitated buzz
1	15	0	20	cauchy	1.0	+/-2	147, 156.9395, 168.3206, 181.4815, 196.875, 215.1219, 237.0968, 264.0719, 297.973, 341.8605, 400.9091, 484.6154, 612.5, 832.0755, 1297.059, 2940	d3+2, d#3+15, e3+36, f#3-33, g3+8, a3-39, a#3+30, c4+16, d4+25, f4-37, g4+39, b4-33, d#5-27, g#5+3, e6-28, f#7-12	agitated buzz

Table 15.4: (*GENDY3* (Seq #3 / in-time #5))

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
8	23	0	20	arcsin	3.0	+/-2	95.8696 ... 106.5217	g2-38 ... g#2+44	modulated pitch (breathy sound), overtone effect (hypcos coeff 0.1)
2	19	0	20	arcsin	3.0	+/-2	116.0526 ... 128.9474	a#2-7 ... c3-25	modulated pitch (breathy sound)
4	10	0	30	arcsin	1.0	+/-2	147 ... 157.5	d3+2 ... d#3+21	modulated pitch (breathy sound) overtone effect (octave + decime) hypcos coeff. 0.1
7	14	0	20	arcsin	3.0	+/-2	157.5 ... 175	d#3+21 ... f3+4	modulated pitch (breathy sound), overtone effect (hypcos coeff. 0.1)
5	13	0	20	logistic	1 / 6	+/-2	169.6154, 182.9875, 198.6487, 217.2414, 239.6739, 267.2727, 302.0548, 347.2441, 408.3333, 495.5056, 630, 864.7059, 1378.125, 3392.308	e3+50, f#3-19, g3+23, a3-22, a#3+48, c4+37, d4+49, f4-10, g#4-29, b4+6, d#5+21, a5-30, f6-23, g#7+36	agitated, but within very narrow pitch ambitus: about a semitone/tone at lower barrier (breathy sound)
3	7	0	20	arcsin	3.0	+/-2	315 ... 350	d#4+21 ... f4+4	modulated pitch (breathy sound)
1	30	3	3				490	b4-14	sharp, penetrating, frozen sound (vanishing: hypcos ampl.: coeff 0.01: graph negative only)
6	16	0	20	hypcos	3.0	+/-2	137.8125, 146.5116, 156.383, 167.6806, 180.7377, 196, 214.0777, 235.8289, 262.5, 295.9731, 339.2308, 397.2973, 479.3478, 604.1096, 816.6667, 1260, 2756.25	c#-10, d3-4, d#3+9, e3+30, f#3-40, g3+0, a3-47, a#3+20, c4+6, d4+14, e4+50, g4+23, a#4+48, d5+49, g#5-29, d#6+21, f7-23	agitated buzz (vanishing: hypcos ampl. coef 0.001)
9									muted
10									muted
11									muted
12									muted
13									muted
14									muted
15									muted
16									muted

Table 15.5: GENDY3 (Seq #4 / in-time # 7)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
7	40	58	58				19.0086	d#0-39	extreme low register, breathy sound
14	35	58	58				21.7241	f0-8	cluster (extremely low and noisy amplitude: logistic coef 0.0001)
13	31	29	29				49.0545	g1+2	cluster (very low and noisy amplitude: logistic coef 0.0001)
16	51	13	13				66.5158	c2+29	cluster (noisy amplitude: logistic coef 0.0001, but higher volume and more presence: no. of segs)
15	45	9	9				108.8889	a2-18	cluster (low and noisy amplitude: logistic coef 0.0001)
12	27	11	11				148.4848	d4+19	see 16
10	23	10	10				191.7391	g3-38	see 15
9	11	12	12				334.0909	e4+23	see 15
11	7	14	14				450	a4+39	see 15
4	19	0	30	arcsin	1.0	+/-2	77.3684 ... 82.8947	d#2-9 ... e2+10	modulated pitch (low bass drone, overtone effect)
5	8	0	20	logistic	1 / 6	+/-2	275.625 ... 306.25	c#4-10 ... d#4-27	nervously modulated pitch (semitone to tone), breathy sound
3	3	0	20	arcsin	3.0	+/-2	735 ... 816.6666	f#5-12 ... g#5-29	modulated pitch (high breathy sound - low volume)
6	9	0	15	hypcos	3.0	+/-1	326.6667, 364.4628, 412.1495, 474.1935, 558.2278, 678.4615, 864.7059, 1191.892, 1917.391, 4900	e4-16, f#4-26, g#4-13, a#4+30, c#5+12, e5+50, a5-30, d6+25, a#6+48, d#8-27	high agitated buzz with vanishing amplitude (cauchy 0.001)
8	15	0	20	cauchy	0.1	+/-2	147, 156.9395, 168.3206, 181.4815, 196.875, 215.1219, 237.0968, 264.0719, 297.973, 341.8605, 400.9091, 484.6154, 612.5, 832.0755, 1297.059, 2940	d3+2, d#3+15, e3+36, f#3-33, g3+8, a3-39, a#3+30, c4+16, d4+25, f4-37, g4+39, b4-33, d#5-27, g#5+3, e6-28, f#712	dominating high agitated buzz (hypcos coef 0.1)
1									muted
2									muted

Table 15.6: *GENDY3* (Seq # 5 / in-time # 8)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
6	40	58	58				19.0086	d#0-39	cluster
3	50	36	36				24.5	g0+0	cluster
5	40	29	29				38.0172	d#1-39	cluster
14	40	9	9				122.5	b2-14	cluster
15	23	15	15				127.8261	c3-40	cluster
4	40	7	7				157.5	d#3+21	cluster
2	40	5	5				220.5	a3+4	cluster
1	40	4	4				275.625	c#-10	cluster
9	15	9	9				326.6666	e4-16	cluster
8	40	3	3				367.5	f#4-12	cluster
13	15	8	8				367.5	f#4-12	cluster
12	14	7	7				450	a4+39	cluster
11	16	6	6				459.375	a#4-25	cluster
7	40	2	2				551.25	c#5-10	cluster
10	4	11	11				1002.273	b5+25	cluster
16	4	0	20	logistic	1 / 6	+/-2	551.25, 722.9508, 1050, 1917.391, 11025	c#5-10, f#5-40, c6+6, a#6+48, f9-23	stable pitch and ex- tremely low amplitude (+/-2)

Table 15.7: *GENDY3* (Seq # 6 / in-time # 11)

material of *GENDY3* is a way of emulating part of the compositional process.

## 15.1 Distribution Functions

There is one parameter in the GENDYN algorithm which has a great impact onto the sonic quality of the sound: the specific distribution function chosen for the time random walk which acts on both depth and dynamics of the stochastic frequency modulation. There are six distribution functions available in the GENDYN program (for their mathematical formulae, see appendix A on page 315):

**Cauchy distribution.** This distribution has a steep graph symmetric to the origin, creating rare but important deviations off a mean value in both positive and negative direction. This fact is responsible for the “jumps” in the pitch movement, and therefore for the appearance of (non-tempered) musical scales.

**Hyperbolic cosine distribution.** Having a graph which is much more flattened than the Cauchy distribution, it is responsible for slow and smooth glissandi curves.

**Logistic distribution.** This one is responsible for agitated, “buzzing” pitch movement. It has a second parameter which dislocates the graph along the x-axis, which pushes the random walks against one of the barriers, creating interesting noisy effects.

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
10	16	10	10				275.625	c#4-10	fixed pitch breathy low amplitude
3	8	0	20	cauchy	.001	+/-15	275.625, 312.766, 361.4754, 428.1553, 525, 678.4615, 958.6957, 1633.333, 5512.5	c#4-10, d#4+9, f#4-40, a4-47, c5+6, e5+50, a#5+48, g#6-29, f -23	high scale, rougher ampl.
4	8	0	20	cauchy	.001	+/-10	see 3	see 3	high scale, rougher ampl.
5	15	0	20	cauchy	.001	+/-10	147, 156.9395, 168.3206, 181.4815, 196.875, 215.1219, 237.0968, 264.0719, 297.973, 341.8605, 400.9091, 484.6154, 612.5, 832.0755, 1297.059, 2940	d3+2, d#3+15, e3+36, f#3-33, g3+8, a3-39, a#3+30, c4+16, d4+25, f4-37, g4+39, b4-33, d#5-27, g#5+3, e 6-28, f#7-12	“solo” portamento voice, sharp timbre (lower than track #6)
6	18	0	20	cauchy	.001	+/-2	122.5, 129.3255, 136.9565, 145.5446, 155.2817, 166.4151, 179.2683, 194.2731, 212.0192, 233.3333, 259.4118, 292.053, 334.0909, 390.2655, 469.1489, 588, 787.5, 1191.892, 2450	b2-14, c3-20, c#3-21, d3-15, d#3-3, e3+17, f3+46, g3-15, g#3+36, a#3+2, c4-15, d4-10, e4+23, g4-8, a#4+11, d5+2, g5+8, d6+25, d#7-27	“solo” portamento voice, very sharp timbre, more aggressive and higher than track #5
2	6	0	15	cauchy	.001	+/-3	490, 580.2632, 711.2903, 918.75, 1297.059, 2205, 7350	b4-14, d5-21, f5+32, a#5-25, e-28, c#7-10, a# 8-25	very high scale, smooth amplitude
1	4	0	20	cauchy	.001	+/-5	551.25, 722.9508, 1050, 1917.391, 11025	c#5-10, f#5-40, c6+6, a#6+48, f9-23	chirping sound down to scale, rough amplitude
9	6	0	20	cauchy	.001	+/-2	367.5, 436.6337, 537.8049, 700, 1002.273, 1764, 7350	f#4-12, a4-13, c5+47, f5+4, b5+25, a6+4, a#8-25	high scale, very low breathy amplitude
7	15	0	20	cauchy	0.1	+/-2	see 5	see 5	agitated buzz
8	15	0	20	cauchy	0.1	+/-2	see 5	see 5	see 8
11									muted
12									muted
13									muted
14									muted
15									muted
16									muted

Table 15.8: *GENDY3* (Seq # 7 / in-time # 9)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
1	4	0	20	cauchy	.001	+/-5			noisy chirping
2	6	0	15	cauchy	.001	+/-3			noisy chirping, like 1
3	8	0	20	cauchy	.001	+/-10			chirping, portamenti up to third
4	8	0	20	cauchy	.001	+/-10			treble, ethereal "solo" voice
5	9	0	20	cauchy	.001	+/-5	245, 273.9131, 310.5634, 358.5366, 424.0385, 518.8235, 668.1818, 938.2979, 1575, 4900	b3-14, c#4-21, d#4-3, f4+46, g#4+36, c5-15, e5+23, a#5+11, g6+8, d#8-27	high ethereal portamento "solo" voice
6	8	0	20	cauchy	.001	+/-2			chirping
7	5	0	20	cauchy	0.1	+/-2			chirping, extremely fast, "wet", splashy effect
8	7	0	20	cauchy	0.1	+/-2			chirping, like 7
9	6	0	20	cauchy	.001	+/-2			chirping, like 3
10	9	0	30	cauchy	.001	+/-1			chirping, lower, more portamento, scale leaps only in (rare) high register
11	11	0	21	logistic	.001 / 0.1	+/-1			chirping, like 7, but blurred, not as high
12	4	0	23	cauchy	.001	+/-1			chirping, like 1
13	6	0	22	cauchy	.001	+/-1			chirping, like 2
14	3	0	29	cauchy	.001	+/-1			chirping, like 1, but extremely low amplitude (arcsin)
15	6	0	24	cauchy	.001	+/-1			chirping, like 2
16	4	0	25	cauchy	.001	+/-1			chirping, like 2

Table 15.9: GENDY3 (Seq # 8 / in-time # 6)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
1	51	0	20	logistic	1 / 6	+/-4			noise (with time filter)
2	13	0	20	logistic	1 / 6	+/-4			noise (with time filter)
3	26	0	20	logistic	1 / 6	+/-4			noise
4	13	0	20	logistic	1 / 6	+/-4			noise
5	51	0	20	logistic	1 / 6	+/-4			snake sizzling
6									muted
7									muted
8									muted
9									muted
10									muted
11									muted
12									muted
13									muted
14									muted
15									muted
16									muted

Table 15.10: *GENDY3* (Seq # 9 / in-time # 4)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
4	51	19	19					f#1-25	cluster
16	40	13	13					e2+50	cluster
5	13	19	19					f3+37	cluster
3	26	5	5					e4+50	cluster
9	15	6	6					b4-14	cluster
14	40	1	1					c#6-10	cluster
1	13	3	3					c#6+34	cluster
13	6	0	32	hypcos	.001	+/-1	229.6875, 273.9131, 339.2308, 445.4546, 648.5294, 1191.892, 7350	a#3-25, c#4-21, e4+50, a4+21, e5-28, d6+25, a#8-25	"solo glissando voice"
2									muted
6									muted
7									muted
8									muted
10									muted
11									muted
12									muted
15									muted

Table 15.11: *GENDY3* (Seq # 10 / in-time # 1)

No.	#segs	low	upp	distr	coef	fluct	freqs	itches	description
1									muted
2	13	8	8				424.0385	g#4+36	cluster
3	26	5	5				339.2308	e4+50	cluster
4	51	19	19				45.5108	f#1-28	cluster
5	13	19	19				178.5425	f3+39	cluster
6	40	29	29				38.0172	d#1-39	cluster
7	13	17	17				199.5475	g3+31	cluster
8	6	4	4				1837.5	a#6-25	cluster
9	15	6	6				490	b4-14	cluster
10	16	6	6				459.375	a#4-25	cluster
11	16	0	31	logistic	.001 / 0.1	+/-1	88.91129, 94.63519, 101.1468, 108.6207, 117.2872, 127.4566, 139.557, 154.1958, 172.2656, 195.1327, 225, 265.6627, 324.2647, 416.0378, 580.2632, 958.6957, 2756.25	f2+32, f#2+40, g#2-45, a2-22, a#2+11, c3-45, c#3+12, d#3-15, f 3-23, g3-8, a3+39, c4+27, e4-28, g#4+3, d5-21, a#5+48, f7-23	glissando curves, no scale; logis- tic biased toward posi- tive values: therefore only lower part of itches avail- able, "solo" voice
12	16	1	1				2756.25	f7-23	cluster
13	6	0	32	hypcos	.001	+/-1	229.6875, 273.9131, 339.2308, 445.4546, 648.5294, 1191.892, 7350	a#3-25, c#4-21, e4+50, a4+21, e5-28 d6+25, a#8-25	same setting as in seq. #10, (in time #1), only short patches
14	40	1	1				1102.5	c#6-10	cluster
15	40	9	9				122.5	b2-14	cluster
16									muted

Table 15.12: *GENDY3* (Seq # 11 / in-time # 10)

**Arcus sinus distribution.** In the specific way as implemented by Xenakis, it pushes pitch against the upper mirror (i.e. it immobilizes pitch movement at its lowest possible position). One could make better use of this distribution, since it is the only one that does not concentrate its values around a mean one but disperses them toward their extremes.

**Triangular distribution.** In Xenakis' implementation, has the same effect as the Arcus sinus distribution, but pushes against the lower mirror (i.e. immobilizes pitch at its highest position).

**Exponential distribution.** This one is not symmetric around the origin, and therefore has the same effect as the Arcus sinus distribution. This distribution is also used for the computation of all the tracks' sound patch ("field") durations within a sequence.

The listing of the sounds in tables 15.2 on page 263 to 15.12 on the preceding page does not follow the numbering of the corresponding tracks in the GENDYN program but is the result of an analysis of their qualities (noisy/pitched, foreground/background, continuous/discrete pitch movement) and within those categories, the perception of their pitch register. The order goes from pitched/background/low to noisy/foreground/high. In other words, the order chosen is musically motivated. Even for the passionate avantgarde music listener (like myself), pedal bass sounds are conceived as a sort of "fundament" in a composition, and to my opinion, *GENDY3* does not make an exception in this respect. The same holds for the perception of "foreground" and "background" tracks: in *GENDY3*, in every sequence, there is at least one track which stands out in the way of a kind of "solo" action – either by the elegance of its pitch curves or the impression of chanting a mode on the steps of a weird musical scale, or by a dominating nervous modulation activity, or just by superior loudness.

On the other hand, there is always a majority of tracks whose pitch is either frozen or whose pitch movement is somehow uniform otherwise, i.e. tracks which are less capable of playing a musically prominent role and have therefore the function of establishing a background to the "solo" tracks, an atmosphere, a sound texture or just the impression of a kind of acoustic "depth". It is surprising how these artificial sounds, which all stem from the same algorithmic generation procedure, differentiate in a manner that they create a listening experience which is not flat but having a perspective in the sense of experiencing an artificial sound "world".

This impression is certainly due to our "classically" trained hearing where a profiled pitch movement is attributed to a "solo" voice while more static or uniform pitch movements are attributed to "tutti" voices. Moreover, since our ears are more sensitive to small microtonal pitch modulation, they inevitably single them out and tend to follow their course even against a jungle of other sounds having a pitch modulation either too small or too big in order to be perceived as a melodic line. In this way, a "classical" differentiation between principal and supporting voices builds up in the listener's ear, be it "composed" or not.

I have generated, with the New GENDYN Program, a hypothetical music example which covers the entire continuum of depth of GENDYN frequency modulation, reaching from perfect noise to frozen sound spectrum. In between,

there are intermediary stages of pitch modulation like glissando or stepwise portamento movement. This example was produced by changing the parameters of a *GENDYN* sound from a maximum degree of freedom for the stochastic processes of modulation (maximal variance of the distribution functions, maximal random walk space) down to zero modulation (zero variance and/or zero random walk space).

This example is hypothetical because in *GENDY3*, where parameters do not change within a sequence, we have no such continuum of modulation depth. However, this hypothetical example is very instructive because it shows that those sonic phenomena which strike the ear the most are not the extremes of noise on the one hand and frozen pitch on the other but the stages of modulation in between, where pitch and spectrum change in a perceptible but unforeseeable way. The richness of *GENDY3* consists in the simultaneous and/or successive combination of many different degrees of such “intermediary” modulation phenomena. Obviously, our ears, acculturated over centuries to harmonic music, are most sensitive to combinations of modulated spectra with pitch information content.

## 15.2 A Description of the Sections of *GENDY3*

My description of *GENDY3* does not proceed in chronological order but in a systematic order. It follows the dissection of the piece into 11 sections, each governed by the durational architecture and the sound synthesis parameter set of a *GENDYN* “sequence” entity.

We will start with the most radical sequence which exclusively features noises as its sonic content. We will see that this is an extreme case of (stochastic) frequency modulation. We contrast this one with its extreme opposite, where frequency modulation is nil, a sequence which contains only tracks with stable pitch, forming a sort of broken-up 16-part cluster.

Then we will have a look at the intermediary forms where pitch is modulated in various degrees, producing different flavors of glissando and portamento movement between the steps of a scale: elegant pitch arcs, stochastic “maqam”-like improvisation (this is only a metaphor!), or nervous humming and buzzing sounds.

### 15.2.1 The “Apotheosis of Noise”

Let us start with section no. 4 (sequence # 9), the “Apotheosis of Noise”. On a first listening, one hears nothing but noise, extending over more than 2 minutes. But if we dissect the layers by resynthesis, we obtain 5 separate tracks which exhibit quite different qualities of noise: the sizzling of a snake, the squeaks of the wheels of a car taking a U-turn, and another like a starting jumbo jet (see 14.24 on page 249). And if we turn the knob of time-loop built into the *GENDYN* program and suspend stochastic deformations of the wave form for more and more periods, we start to hear delicate random “melodies” in the high register (see figure 14.23 on page 248). In other words, the noises of section no. 4 are nothing else than melodies accelerated beyond the threshold of pitch perception. And this is exactly what frequency modulation is all about.

### 15.2.2 The “Stochastic Organ”

At the extreme opposite end, let us take section no. 11 (sequence # 6, see 15.12 on page 272). It only contains sound layers with fixed pitch. There is no musical action except switching these pitches on and off, in a seemingly unintentional, Cageian manner, yielding a cluster broken up in time and producing random combinations of harmony. As we know, this randomness is calculated by rigid formulae, as are all harmonic constellations of the piece. One could speak of a kind of random counterpoint.

### 15.2.3 The “Exposition”

Section 1 of *GENDY3* is an example for a beautiful and elegant “solo” glissando before the background of a 7-tone cluster (see table 15.11 on page 271). I call the glissando “beautiful” because it gives the (false) impression of being “composed”, as if its arc of climax had been deliberately designed. The glissando starts with the highest pitch possible because here, at the start of the piece, the time random walks have position zero which means least possible wave length. After this initial descent, pitch rises in three gentle curves, each reaching a step higher, and then shoots up again toward the high end of the scale. As we know, the steps of the scale get larger as pitch rises, so toward the summit, pitch movement turns into somersaulting jumps. After this culmination, pitch falls softly back down to its starting position.

This “solo” glissando of section 1 is a happy coincidence indeed, for in general, pitch movement settles around a middle value of highest probability. However, given that the stochastic processes are governed by randomness, the least probable is also possible, and that is the nice surprise of stochastics in the opening section of *GENDY3*. The scale of Ex. 1 in figure 15.12 on page 287 is the underlying scale of this exceptional opening glissando.

### 15.2.4 The “Twittering Machine”

Another section, section no. 6 (sequence # 8) is a good example of a complex sonic atmosphere somewhere in between tone and noise (see table 15.9 on page 270). On the one hand, one can perceive “melodic” fragments, on the other hand, one has to do with a sonic aggregate state escaping all tonal perception. I call this section, because of its bird-like twittering, “Twittering Machine (hommage à Paul Klee)”. This section is a combination of 16 similar sounds, all in agitated movement between the upper steps of a scale, an example of a homogeneous section of *GENDY3*, in contrast to most of the other sections which are combinations of different sonic characters. It is as if one was looking into a swarm of insects dancing in glaring sunlight. Speed and depth of the stochastic frequency modulations is such that it dwells on the transition between pitch and noise.

### 15.2.5 The “Foggy Atmosphere”

Section 5 (see table 15.4 on page 265) contains several tracks (track #5 etc.) with stable, but modulated pitch in the baritone range. This kind of modulation produces a kind of stochastic “vibrato”. In addition, their spectrum is

“roughened” due to a certain degree of amplitude modulation. In these cases, the stochastic “vibrato” is produced by a random walk pushing the segment lengths against the upper barrier (Logistic with a beta parameter set to a high value or arcus sine distribution, both are unsymmetric distributions). The amplitude distribution in these cases is the Hyperbolic Cosine distribution which accounts for the relatively smooth transitions in the amplitude modulation of the wave form’s breakpoints, creating a sort of “overtone music”. The result are pedal tones which convey an impression of containing “air” as if they were produced by hoarse organ pipes. They create a fuzzy impression, sounds with blurred contours as if perceived through a fog. These pedal tones are set against a brilliant “solo voice” with its stochastic improvisation over the “maqam” of the underlying scale.

### 15.2.6 “Heavy Metal”

Section 3 (see table 15.3 on page 264) contains brilliant, metal-like cluster sounds and features a magnificently exalted solo voice.

### 15.2.7 “The Sirenes”

Section no. 9 (see table 15.8 on page 269) is again one of the homogeneous sonic landscapes. It is built mainly of glissando sirenes.

### 15.2.8 “Plein jeu”

Section no. 10 (see table 15.12 on page 272) is a combination of solo and background voices which produce an impression of “plein jeu”: saturated, brilliant sounds, which give this section before-the-last a finalizing character.

### 15.2.9 “The Buzzy Insects”

Section no. 7 (see table 15.5 on page 266) exhibits tracks with specific activities concerning pitch modulation: a fast, nervous, insect-like humming and buzzing (compare track #6 in section no. 7 to tracks #6 and #8 in section no. 8). These pitch modulations are too fast to be perceived as a melodic line but also too slow to be perceived as noise.

### 15.2.10 “The Harbour”

One of the most prominent sonorities of section no. 2 (see table 15.2 on page 263) is a snake sizzling which is present almost throughout the section. This sonority resurges later as one of the five noise tracks in section 5, the “Apotheosis of Noise”. Section no. 2 is grounded, like section no. 8, in a very low bass drone which is, here, actually formed by three tracks with very low pitch. In the lower-middle register, another sonority recalls the impression of a foghorn or klaxon. The musical foreground is represented by a sluggishly undulating glissando in low register.

### 15.2.11 “Rough Seascape”

Section no. 8 (see table 15.6 on page 267) exposes rough sounds. Aside from the few humming and buzzing voices, even the fixed pitches show either rough contours, as can be seen in the pitch/duration plot of this section (see figure 15.8 on page 283) or are strongly modulated in amplitude which creates a ragged sonority.

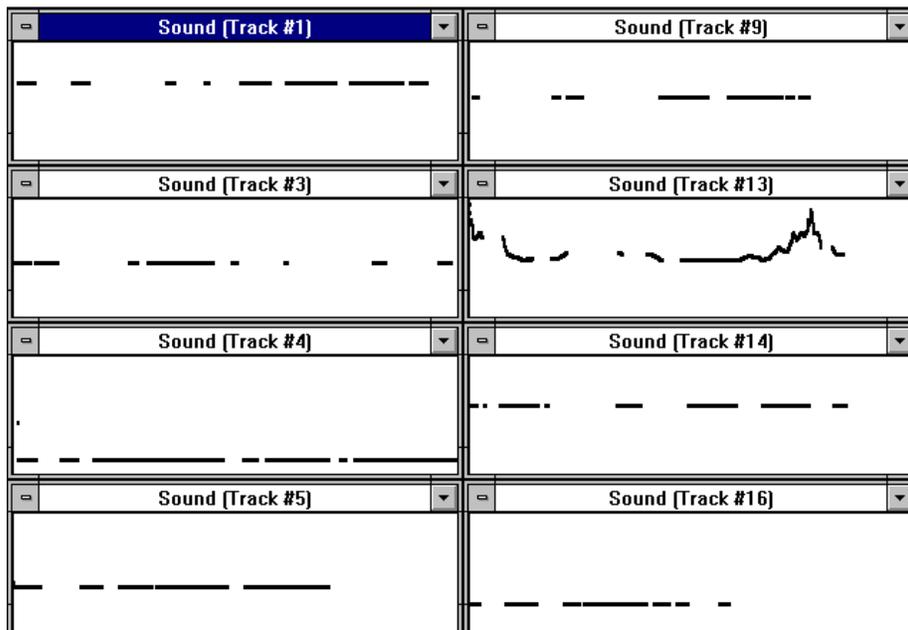


Figure 15.1: First section of *GENDY3* (sequence  $\psi = 10$ ) “The Exposition”. This and the other duration/pitch scores have been generated with the New GENDYN Program. The nick-names in quotes are added for convenience and better orientation, they are invented by me and not in any way hinted at by Xenakis.

## 15.3 Scales

I have tried to extract pitch information of all sections which exhibit a prominent glissando or portamento pitch movement. It is obvious that those sonic tracks which strike out as “solo” voices in the music are based upon a fixed collection of pitches which correspond to a specific scale system. In figure 15.12 on page 287, I have assembled the most prominent of them. They are, of course, of a theoretic kind insofar as they enumerate all “physically” or “algorithmically” possible pitches — not all of them are to be heard in the music.

We have already seen, by observation of the procedural behavior of the *GENDY3* program, that glissando or portamento pitch movement which is slow enough to be perceived as “musical” (in contrast to “insect-like” humming and

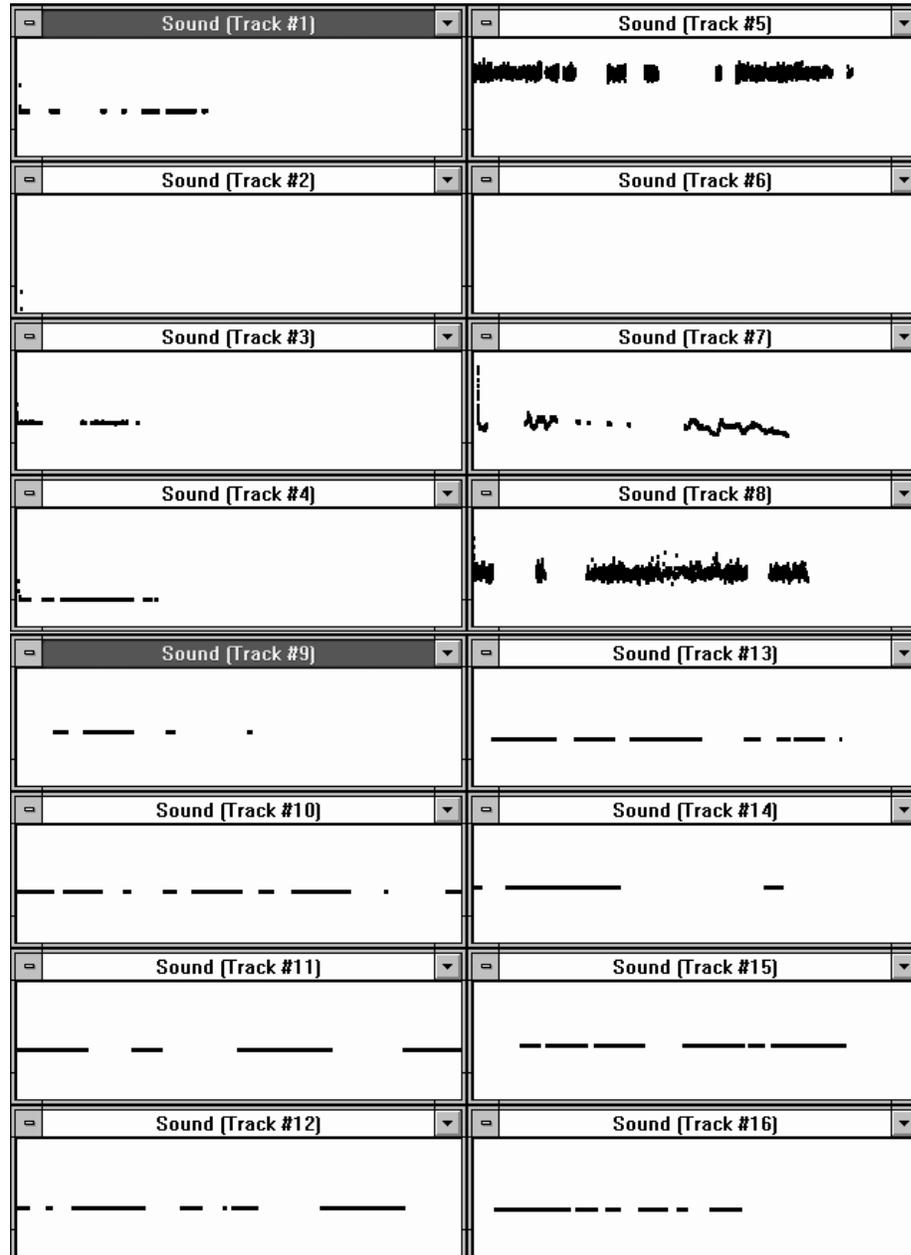


Figure 15.2: Second section of *GENDY3* (sequence  $\psi = 1$ ): “The Harbour”

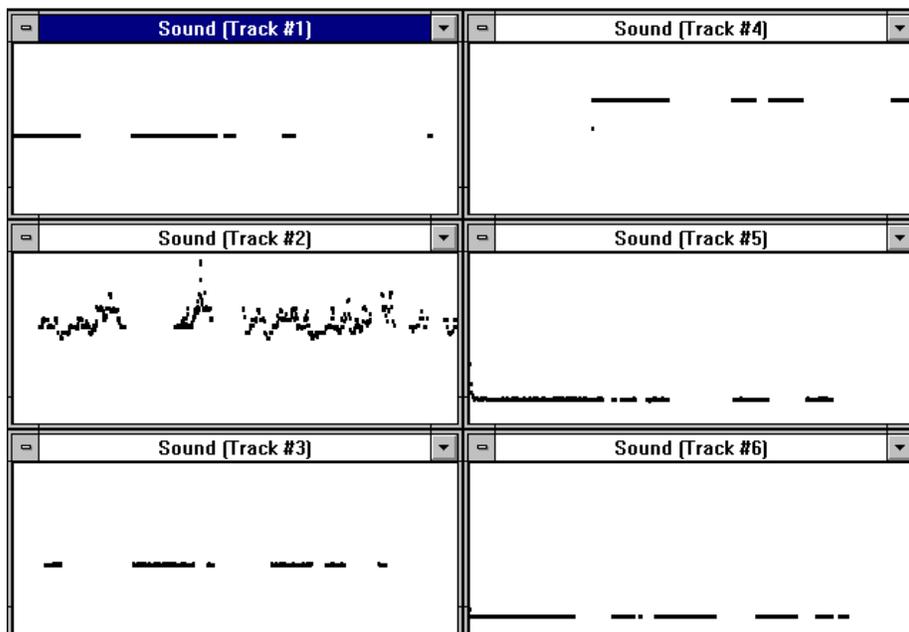


Figure 15.3: Third section of *GENDY3* (sequence  $\psi = 2$ ) “Heavy Metal”

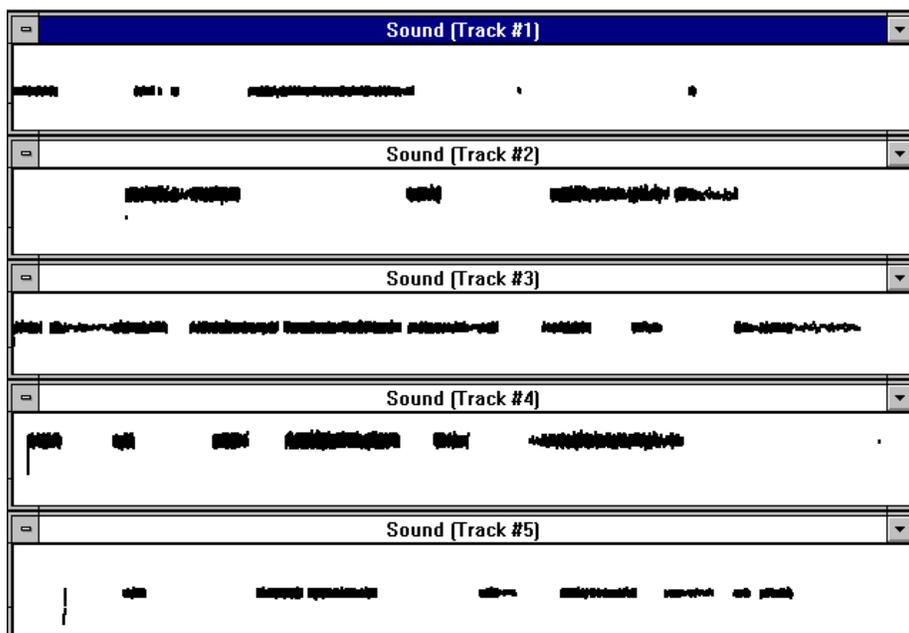


Figure 15.4: Fourth section of *GENDY3* (sequence  $\psi = 9$ ): “The Apotheosis of Noise”

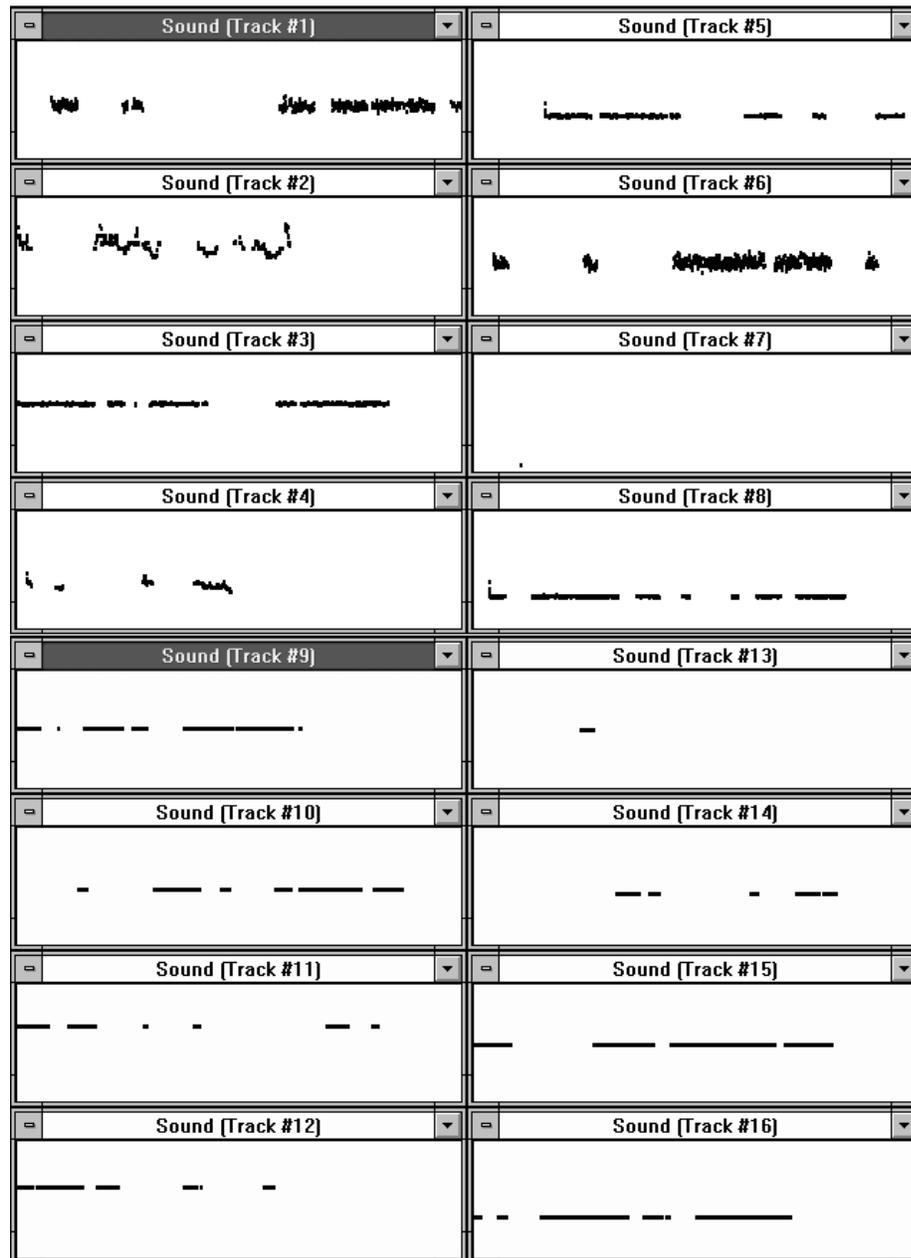


Figure 15.5: Fifth section of *GENDY3* (sequence  $\psi = 3$ ) “The Foggy Atmosphere”

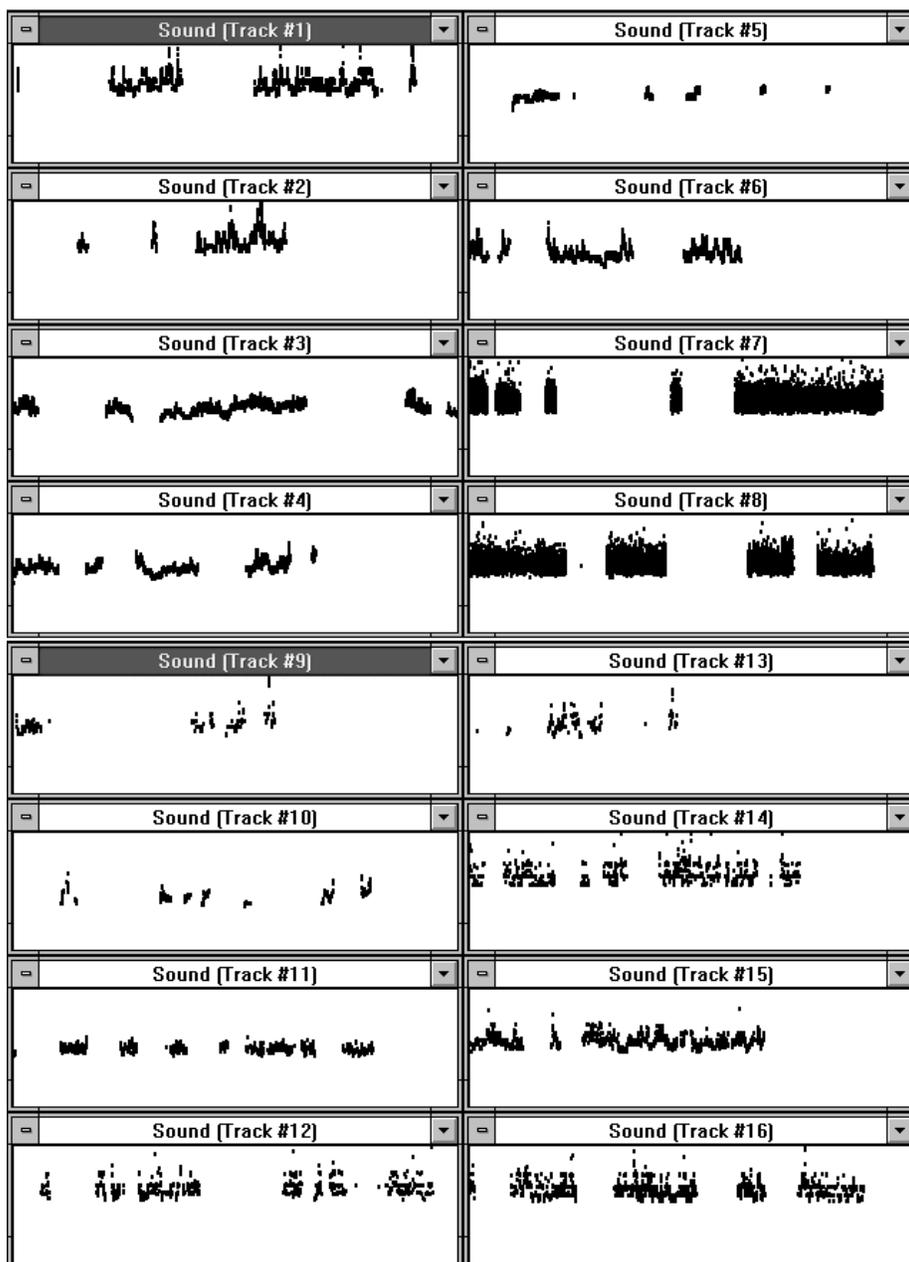


Figure 15.6: Sixth section of *GENDY3* (sequence  $\psi = 8$ ): “The Chirping Machine”

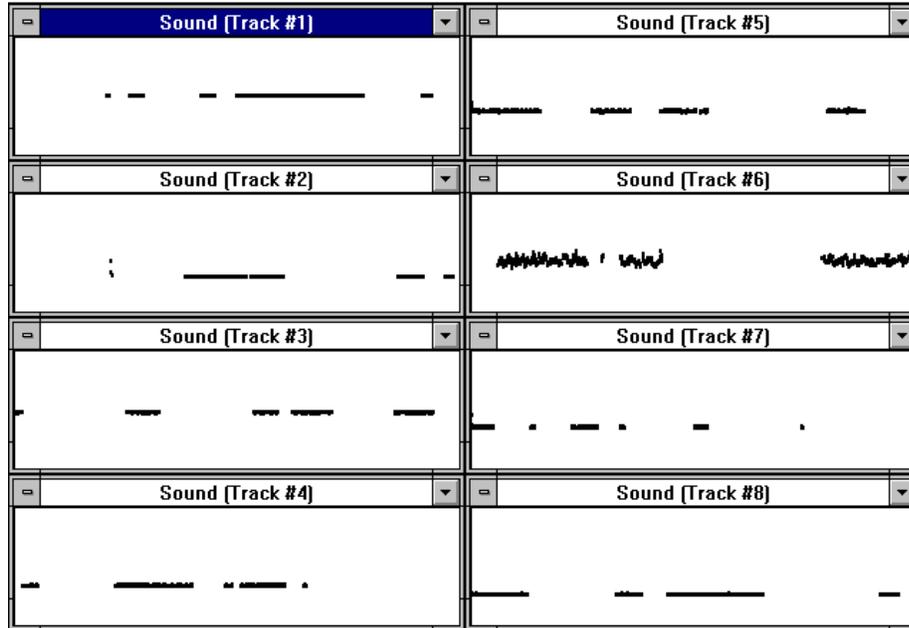


Figure 15.7: Seventh section of *GENDY3* (sequence  $\psi = 4$ ) “The Buzzy Insects”

buzzing or noise sounds) is produced by an effect of auto-stabilization: the secondary random walks tend to “freeze” at both the lower and the upper random walk barriers. Stable pitch thus comes about as a combination of both minimal and maximal wave form segments which add up to a certain wave length (which is the inverse of frequency). When a pitch random walk of a wave segments breaks free from its stable position and moves to the opposite side of the random walk space, a change in pitch of the resulting wave form is perceived. This change is audible because one wave segment changes length by a maximum degree (either from minimal to maximal length or the other way round) but it is still gradual because in general, it is only one wave segment at a time whose length is changed. This is the reason that pitch movement of such sounds is structured by a pitch scale, even when these steps are big, which is true toward the high pitches, as can be seen in the notation examples.

Moreover, extremely high or low pitches within the scales are rarely to be heard since they correspond to rare configurations of wave segment lengths, i.e. either all minimal or all maximal. That is, statistically speaking, they represent only two out of a much greater number of possible micro states. Recall that a middle pitch (“macro state”) can be formed by a number of combinations (“micro states”) of minimal and maximal wave segments, regardless of their internal order within the wave form.

Therefore, the “solo” glissando or portamento voices, from their initial pitch left by the preceding section, soon reach a kind of equilibrium middle pitch, from where they reach out to both ends of the scale. The activity of leaving the equilibrium position and the impact of the moves toward the low or the

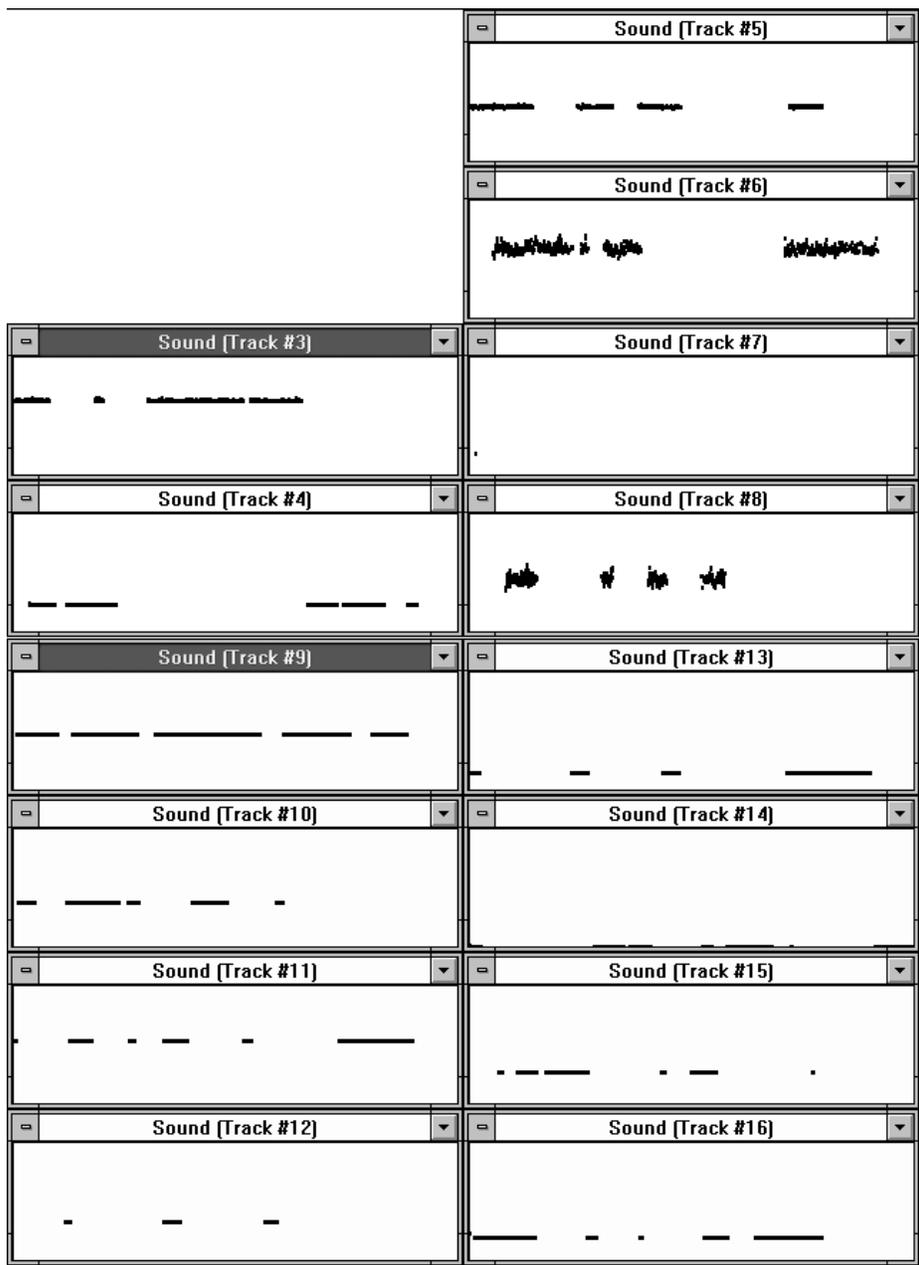


Figure 15.8: Eighth section of *GENDY3* (sequence  $\psi = 5$ ): “Rough Seascape”

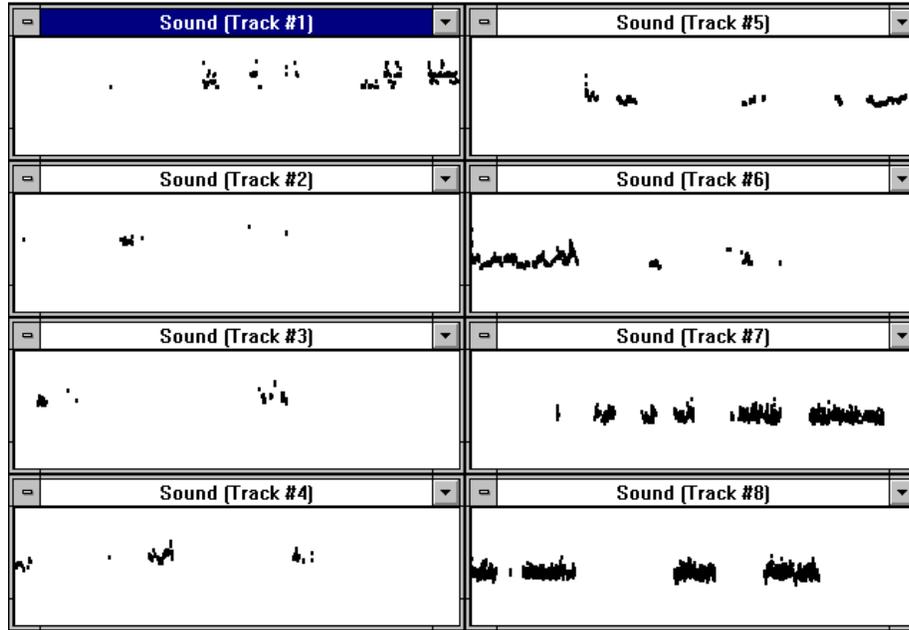


Figure 15.9: Ninth section of *GENDY3* (sequence  $\psi = 7$ ) “The Sirenes”

high register depend on the dynamics of the primary random walks which are governed by their distribution functions and their coefficients, and the barrier positions. For example, a minimal primary random walk space (say,  $[-1 \dots +1]$ ) with a very steep and symmetrical distribution function is responsible for smooth glissando movement whereas a greater primary random walk space, together with a more scattering distribution is responsible for those exalted, jump-like pitch movements so characteristic for some *GENDY3* sections.

The human ear has a tendency to “adapt” perceived pitch to frames of reference such as known pitch scales and intervals. So, in spite of the fact that GENDYN does not offer “clean” musical intervals, there are many possibilities to perceive e.g. a tritone or a seventh note within the GENDYN scales (see 15.12 on page 287). By the same token, an impression of “exotic” scales is often conveyed in the portamento “solo” tracks probably from the fact that there is no “western” scale pattern that would fit; so one is ready to adapt the perceived pitches to a finer quarter-tone grid of, say, a purported arab maqam system. This is not atypical for Xenakis’ music, since he himself used to carefully design scales using quarter-tones (and less!) as a building element, and he created a whole theory for generating complex pitch scales out of simple elements, his famous “sieve theory”. Similar to “sieved” scales which (not by the theory itself, but through the usage by Xenakis) provided him with non-octaviating scales, GENDYN scale pitches also tend not to be congruent across the octaves. In addition, they reach over the whole audible spectrum, very similar to Xenakis’ formalism of sieved pitch.

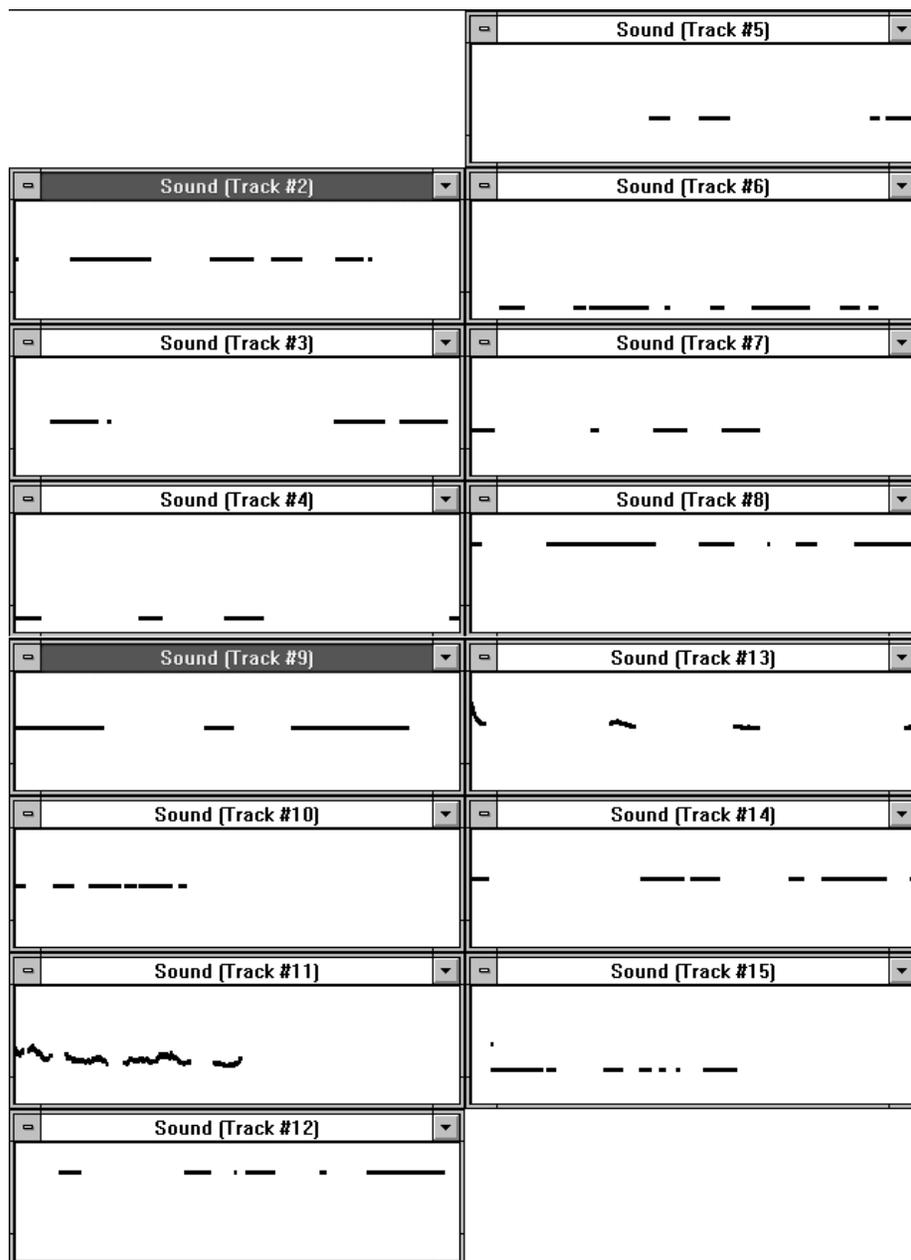


Figure 15.10: Tenth section of *GENDY3* (sequence  $\psi = 11$ ) “Plein jeu”

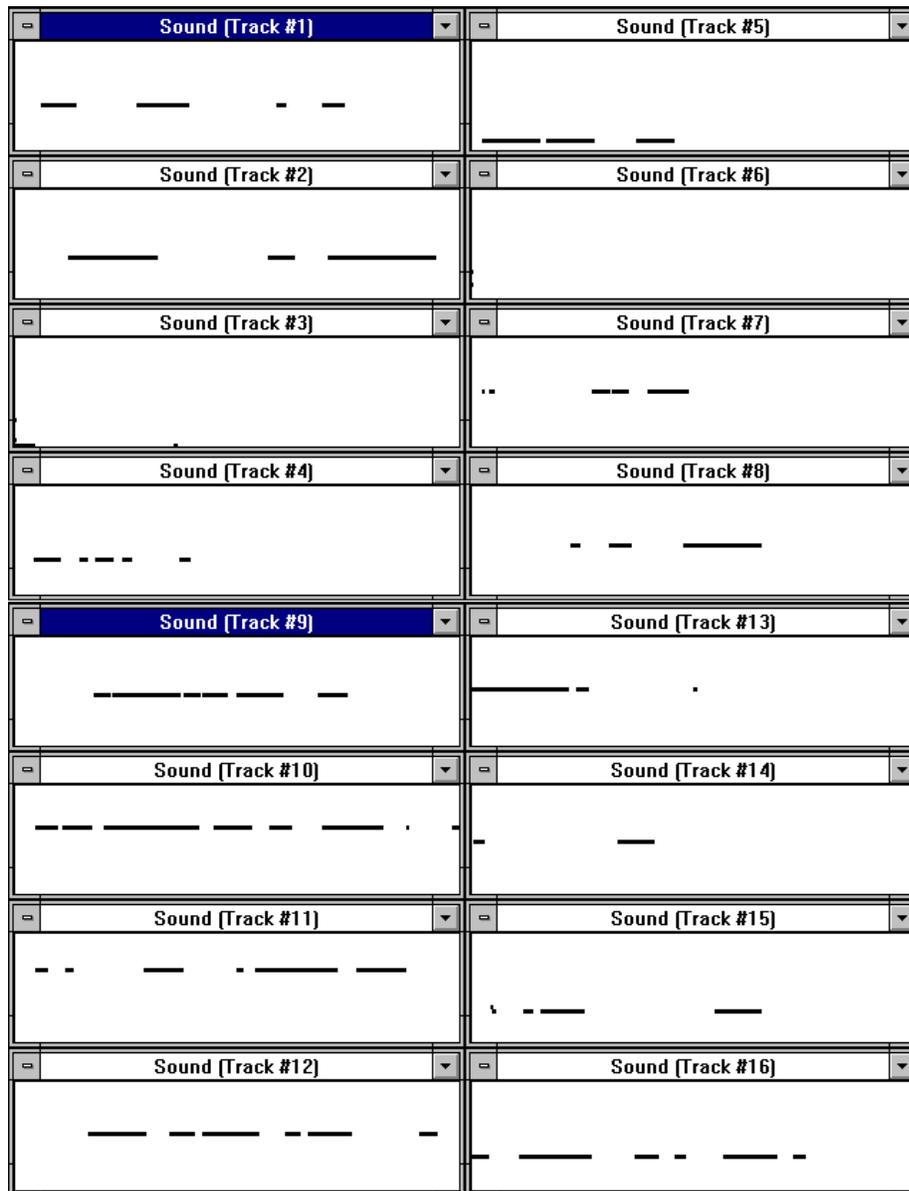


Figure 15.11: Eleventh section of *GENDY3* (sequence  $\psi = 6$ ): “The Stochastic Organ”

Ex. 1:  $-25$   
 $-21 +50+21 -28 +25 -25$

Ex. 2:  $-12 -4 +11+34-33 +11 -32 +40+32+47 -7$   
 $-21 +23-47 +25-14 -23$

Ex. 3:  $+21 -26 -30 +27-23 -14 +36 +8$

Ex. 4:  $-12 -13 +47 +4 +25 +4 -25$

Ex. 5:  $-14$   
 $-21 -3 +46+36-15 +23+11 +8 -27$

Ex. 6:  $+2 +15+36-33 +8 -39 +30$   
 $+16+25-37 +39-33 -27 +3 -28 -12$

Ex. 7:  $+32+40-45 -22 +11 -45 +12-15 -23 -8 +39$   
 $+27-28 +3 -21 +48-23$

Figure 15.12: The most prominent *GENDY3* scales forming the pitch structure of the “solistic” voices of sequence #10 (in-time #1), track #13 (Ex. 1); sequence #1 (in-time #2), track #7 (Ex. 2); sequence #2 (in-time #3), track #2 (Ex. 3), very similar but not identical to sequence #3 (in-time #5), track #2 (Ex. 4); sequence #8 (in-time #6), track #5 (Ex. 5); sequence #5 (in-time #8), track #8 as well as Sequence #7 (in-time #9), track #5 (same parameter setting!) (Ex. 6) and sequence #11 (in-time #10), track #11 (Ex. 7). See figures 15.2 on page 263 to 15.12 on page 272 for a systematic listing.

## 15.4 A Generalized Notion of FM As a Key Into *GENDY3*

Through the procedural analysis of *GENDY3* we have been able to dive into the minutest details of this composition. We have explored noises, stable tones, slow “singing” and fast “buzzing” glissando arcs. We were able to witness the transformation of sound from noise to glissando movement to stable tone and vice versa through “playing with the knobs” of the New GENDYN Program, within a continuum of frequency modulation phenomena.

All this helped to develop an intuition that the most prominent sound qualities of *GENDY3* can be linked, though not reduced, to the quantitative movement of fundamental frequency in time of the individual GENDYN sounds, that means, to the concept of (the stochastic flavor of) Frequency Modulation. In a past text of mine ([Hof94a]), I spoke of a Grand Unified Theory of Composition (in analogy to the GUT which the theoretical physicist are after), namely the consequent implementation of Stochastics in music synthesis. I still regard this theory as a prerequisite to the composing of *GENDY3*, but would like to complete it with a theory which builds on top of, namely the composition of complex musical pitch structures (including noise) with the help of the concept of Stochastic Frequency Modulation (cf. [Hof04]).

Stochastic Frequency Modulation results from the incessant, randomly driven spreading and shortening of a wave period which stays intact as such but undergoes transformations to an extent that it becomes unrecognizable. The dynamics of this transformation, from gradual to abrupt, essentially defines the sonic character of the sounds and their evolution in *GENDY3*.

Within the ample and general sonic concept of Frequency Modulation lies the narrow and specific phenomenon of musical pitch and its evolution in time. We have seen that there is no exact border between pitched and complex sound, and that both can be transformed into one another using Stochastic Synthesis.

Xenakis’ sketches tell us that he was compositionally conscious about the scope of GENDYN synthesis concerning complex pitch structures. However, he did not have the means for a deeper analysis of the impact of parameter choice onto complex pitch structure in *GENDY3*. In this chapter, I tried to reconstruct the composer’s situation between chances and constraints of his GENDYN algorithm concerning pitch synthesis. A systematization of available cluster and scale configurations outlines the complete decision space of which Xenakis only knew a small but interesting part.

The various dynamics of pitch modulation are a dominant feature of *GENDY3* and express themselves as such varying structures as buzzes, glissando curves, fixed clusters and asymmetric, non-octaviating scales (reminiscent of the “sieves” in his instrumental music). With the exception of the buzzes, all of these pitch structures are also present in Xenakis’ instrumental music. It is surprising how Xenakis seemed to have succeeded in making *GENDY3* sound “typically Xenakis”, in spite of its abstract concepts and the less-than-optimal technical conditions.

## 15.5 Music Out of Nothing?

In his 1991 ICMC paper (reprinted in [Xen92b]), Xenakis pointed out that his GENDYN compositions are created “out of the void”, by probabilities only, comparable to a “big bang” situation.

If the procedure of stochastic sound synthesis is called “Polygonal Variation”, one may ask: what original polygonal wave form does the algorithm start with? In fact, it is silence: a polygon with all its vertex values set to zero. In this sense, the stochastic synthesis is “Music Out of Nothing” in a verbal sense: the creation of the sound results of probability fluctuations only, from nothing, as was probably the case in the Big Bang, the creation of our universe.

Our closer look at the dynamics of the sound creation process, however, revealed that the specific constraints by which these blindly raging probabilities are “tamed” - both by the specific design of the computing algorithm and the choice of its parameters - are very decisive for the acoustic quality of the sounds to emerge. To dwell on the metaphor: if the boundary conditions and parameters of the Big Bang had been different, we would probably not have come into existence as human beings in the first place! It seems to me that the “few premises”, as Xenakis humbly called them in his paper, i.e. the algorithmic “boundary conditions” and parameters he chose, are more important for the specific character of a composition like *GENDY3* than Xenakis himself might have believed at the time of composing.



## Chapter 16

# Future Work

As has been stated while describing the GENDYN project (see section 14.2 on page 212), this dissertation is but the beginning of a second life of Xenakis' GENDYN. We have stressed that GENDYN is more than just a sound synthesis method, and it is more than just a way of composing music. It is a landmark both in the legacy of Xenakis as it is in the history of Algorithmic Composition. As such, there is still a lot to be done in researching more thoroughly the history of the GENDYN concept during the lifetime of the composer. The ideas behind GENDYN emerge as early as in the late 1950s and have been present in his philosophical and compositional thinking ever since, be it in an obvious or in a subcutaneous manner. Many aspects remain yet to be explored:

- What other approaches to Stochastic Synthesis did Xenakis investigate in either Bloomington or CEMAMu given the exhaustive agenda given by himself in the chapter “New Proposals in Microsound” (see [Xen92a]? We have stated in section 3.8 on page 56 that Dynamic Stochastic Synthesis might be one of those but what about the others?
- What has Dynamic Stochastic Synthesis in common with the routines appended around 1971 to his ST program from 1958-62, extending it to also generate microsound, in addition to composing a musical macrostructure? Did he ever use this facility for experimental compositions, for nothing seems to be preserved from this except a few stochastic sounds in “La Lgende d'Eer”? And even, have those stochastic sounds been generated with these subroutines at all? Did Xenakis integrate the subroutines into the main program in order to produce stochastic music in one go as with GENDYN, and if so, would it be possible to reconstruct such early attempts at creating a “Music Out of Nothing”? A history of the Bloomington and CEMAMu institutes is still to be written, but doubts are if there remains enough material that survived their closing.

As for the productive side of GENDYN, much of the possibilities of the GENDYN project are far from being exhausted.

**Refactoring the New GENDYN Program.** The implementation of the New GENDYN Program made in the course of this dissertation is not state-of-the art anymore. It should be renewed by devising a new user

interface and possibly some more extensions to make it more usable to musicologists and composers alike. Cooperative multitasking should be replaced by multi-threading to combine sound synthesis and interactive control in one and the same user process.

**GENDYN synthesis as a building brick to CAC systems.** It would be interesting to wrap a Dynamic Stochastic Synthesis engine as a component (a kind of “stochastic oscillator”) to be controlled by CAC systems such as C-Music, Max-MSP, OpenMusic, Common (Lisp) Music and others to enrich the synthesis palette for a greater circle of electronic musicians.<sup>1</sup>

**A pedagogical Web demo.** A GENDYN applet would be to demonstrate GENDYN sound synthesis on the Web. A picture box could show the animation of one GENDYN sound, along with real-time sound output and the control of the main GENDYN parameters: barriers (mouse position: movement to the right opens the right time barrier toward bass sounds; movement up and down controls the amplitude barrier pair and thus the volume). In order to generate real-time sound, access to the client’s native sound system via JAVA media APIs would be needed. Nick Collins has done a similar work for Apple’s iPhone and iPod ([Col05]).

**Controlling synthesis parameters.** It could be interesting, given the fact that Xenakis tried to make GENDYN sounds less static in *S709* by changing the parameter values over time, to change them in a more controlled way by either

- using the interface of the New GENDYN Program and record those changes in a script file to edit and / or re-execute at will
- using hand-drawn or computed envelopes
- using multi-dimensional input devices such as a data glove to alter a multitude of them in parallel and in real time.
- It could be interesting, also, to open the random walk barriers widely at the attack of a sound and then restore their user-set position to obtain more plosive sounds.

**Harnessing Human-machine interaction.** Controlled by a human during live performance or in the process of composing, the gap between instrumental and computer sound would be bridged by adopting a radically new approach, where neither the computer is made to “imitate” instrumental sound, nor instrumental sound used to “embellish” or “humanize” genuine computer sound. Mutual penetration of human and computational rendering of sound would be based on equal grounds, with the computer taking benefit of the specific complexity of human creative activity on the one hand and the human taking benefit of the specific complexity of computational algorithmic action on the other.

---

<sup>1</sup>Several flavors of GENDYN oscillators have already been implemented by Nick Collins as unit oscillators in SuperCollider [McC96], a very powerful language for sound synthesis and composition (see [Col]). There are also extensions or ameliorations of Stochastic Synthesis, like Bökesoy’s “Stochos” ([BP03]) and Brown’s “IDSS” ([Bro05]), who also cites work by Chang ([Cha99]). Another scholar currently (2009) working on a GENDYN software synthesizer is Felix Pfeifer from Potsdam University.

**A sound transformation tool.** The algorithm of Dynamic Stochastic Synthesis has as its core the incessant stochastic transformation of the physical sound signal sent to the loudspeaker. In the “classical”, purely generative approach, the initial sound signal to be fed into the algorithm is perfect silence, i.e. sonic design starts from scratch and sound comes into being “from nothing” by probability fluctuations only. It would be interesting to explore the application of stochastic transformation to existing sound sampled into the program. The Dynamic Stochastic Synthesis would thus become a novel sound *transformation* tool. Depending on the boundary conditions imposed onto the Dynamic Stochastic Synthesis process by the user, transformation could vary from slight stochastic enrichment of the sound spectrum to complete independent complex sonic evolution. The degree of faithfulness of the transformation would also depend on the rate and the frequency (regular or irregular) with which the original sound is sampled. In one extreme, the stochastic signal could be defined to faithfully follow in the (harmonic and pitch) tracks of the original sound; in the other extreme, it would just be triggered by the input sound and take on its own route.

**Scripting the synthesis.** Interpolations between pre-established parameter configurations could be automated by scripting. This could be achieved either by programming or by recording window and control events and feeding these parameter changes into the synthesis engine.

**Enhancing the efficiency of sound synthesis.** Multi-threaded or even parallel versions would simply realize the multi-layered program logic (the DSS algorithm is inherently parallel in nature) but may be difficult to coordinate and schedule.

**Sound synthesis using Sieves.** Another proposition of Xenakis’ has neither been realized by himself nor by any other person: doing sound synthesis using sieves and / or cellular automata. It could be interesting to use the experience gained by the GENDYN project to attempt a realization based on this idea.

Much still needs to be done to shed some more light onto Xenakis’ lifetime activities in electroacoustics and computer composition which have played such an important role in his life as an artist and composer, even when his output in this area has been much less than his “classical” scores for instruments [Hof01b].



# Conclusion

In this part, I attempted an analysis and systematization of the sonic landscape of *GENDY3* (1991), and how it can be perceived as a composed musical entity standing out as a masterwork of its creator Iannis Xenakis. After our discussion of the underlying theory and after a close analysis of its structure, we have come to a situation where we now know that parts of this perception of *GENDY3* as a “composed” masterwork is true only in a metaphoric way. Some of the most striking sonic patterns like e.g. the various pitch contours, supporting the sensation of delicately designed arcs of musical development, could only be created through setting favorable conditions for such musical phenomena to happen, within the framework of Xenakis’ ingenious idea of Dynamic Stochastic Synthesis.

*GENDY3*, due to the richness of its sounds, covering the whole gamut between perfect noise and frozen pitch, evokes a wealth of musical sensations and associations on the part of the listener, analogies to “solo” and “background”, to “harmonies” and “modes”, to “opening” and “finale”. However, we can say that these sensations and associations are constructed during the (repeated) listening process on the side of the listener. They are not really “there” in a sense of agreed conventions of musical composition, as would be the case with the canon of affects and musical figures within traditional music.

In order to explain these effects and the possible essence of a GENDYN composition, we had to take resort to notions like Radical Constructivism as applied to the theory of Art, to the notion of Interactive Composition, and to the theory of Stochastics in a wider sense, as a means to create auto-organizing processes within a chaotic environment. We discussed the aesthetics of the purely artificial sound world of GENDYN before the background of notions like idiomatic computer music and stated that such approaches to the creation of art pertain to a sociological, if not political statement within a technicized society. From musical analysis proper, we went the way towards the signification of a composition as an example of an attitude to composing, and in Xenakis’ case one can say just as much: as an example of his attitude to living.

It is hoped that by discussions like this, one of the most provoking parts of Xenakis’ legacy, the fulfillment of his lifelong dream of an Automated Art in composition, his GENDYN program and music, will keep to be accessible for further study by musicologists and composers alike, and delivering its fruits in both domains of science and art, for the next generations to come.



# Bibliography

- [AB86] William Aspray and Arthur Burks, editors. *Papers of John von Neumann on Computing and Computer Theory*. Number 12 in Charles Babbage Institute reprint series for the history of computing. MIT Press, Cambridge (MA), 1986.
- [Ago98] Carlos Agon. *OpenMusic: un langage visuel pour la composition musicale assistée par ordinateur*. PhD thesis, Université de Paris VI, 1998.
- [Ame87] Charles Ames. Automated composition in retrospect: 1956–1986. *Leonardo Music Journal*, 20(2):169–185, 1987.
- [Ame91] Charles Ames. A catalog of statistical distributions: Techniques for transforming random, determinate, and chaotic sequences. *Leonardo Music Journal*, 1(1):55–70, 1991.
- [Ass93] Gérard Assayag. CAO: vers la partition potentielle. *Les cahiers de l'Ircam: La composition assistée par ordinateur*, 3(1), 1993.
- [Ass98] Gérard Assayag. Computer assisted composition today. In *Proceedings of the First Symposium on Computer and Music*, pages 9–17, Corfu, Greece, 1998. The Department of Music, Ionian University.
- [Ass99] Gérard Assayag. Du calcul secret au calcul visuel. In Hugues Vinet and François Delalande, editors, *Interfaces homme-machine et création musicale*, pages 37–65, Paris, 1999. Hermes Science Publications.
- [Bai] Kevin Baird. Applications and effects of physical modeling synthesis. <http://music.calarts.edu/~kbaird/Thesis/4thPaper.html>.
- [Bar66] Pierre Barbaud. *Initiation à la composition musicale automatique*. Dunod, Paris, 1966.
- [Bas68] M. Bastiaans. Gabor's expansion of a signal to gaussian elementary signals. In *Proceedings of the IEEE 68*, pages 538–539, 1968.
- [BDG88] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*. Number 11 in EATCS Monographs on Theoretical Computer Science. Springer, Berlin etc., 1988.

- [BE94] B. Becker and G. Eckel. The use of technology in contemporary music. In *Proceedings of the 5th International Symposium on Electronic Arts*, 1994. [http://www.uiah.fi/bookshop/isea\\_proc/nextgen/j/23.html](http://www.uiah.fi/bookshop/isea_proc/nextgen/j/23.html).
- [Bee88] Michael J. Beeson. Computerizing mathematics: Logic and computation. In Rolf Herken, editor, *The Universal Turing Machine. A Half-Century Survey*, pages 191–225, Berlin, Hamburg, 1988. Kammerer & Unverzagt.
- [Bel97] Angelo Bello. Timbral transformation and spatialization using the UPIC system (an interesting side-effect of implementing FM on the UPIC system). preprint, personal communication, 1997.
- [Bel98] Angelo Bello. An application of interactive computation and the concrete situated approach to real-time composition and performance. In *Proceedings of the International Computer Music Conference*, pages 93–97. International Computer Music Association, 1998. Conference at Ann Arbor, Michigan.
- [Bel99] Angelo Bello. Simultaneous spatialization and synthesis of sound with nonlinear functions. preprint, personal communication, 1999.
- [Bel01] Angelo Bello. Notes on composing with the UPIC system: The equipment of Iannis Xenakis. In Makis Solomos, editor, *Présences de Iannis Xenakis*, pages 93–97, Paris, 2001. CDMC.
- [Ber79] Paul Berg. PILE — a language for sound synthesis. *Computer Music Journal*, 3(1):30–41, 1979. reprinted in Curtis Roads and John Strawn (eds.), *Foundations of Computer Music*, MIT Press: Cambridge, MA, 1985, pp. 160–187.
- [Ber92] Pierre Bernard. The UPIC as a performance system. *Contemporary Music Review*, 6:47–57, 1992. part 1.
- [Bey91] Peter Beyls. Chaos and creativity: The dynamic systems approach to musical composition. *Leonardo Music Journal*, 1(1):31–36, 1991.
- [Bey97] Peter Beyls. Aesthetic navigation: Musical complexity engineering using genetic algorithms. In *Journées d’Informatique Musicale JIM97*, Lyon, France, 1997. <http://www.grame.fr/jim97/proceedings.html#Beyls>.
- [Bis91] John Bischoff. Software as sculpture: Creating music from the ground up. *Leonardo Music Journal*, 1(1):37–40, 1991.
- [Boe90] Rainer Boesch. L’électricité en musique. *Contrechamps*, 11:134–145, 1990. special edition ‘La musique électronique’.
- [Bor95] Georgina Born. *Rationalizing Culture: IRCAM, Boulez, and the Institutionalization of the Musical Avant-Garde*. The University of California Press, Berkeley, Los Angeles, London, 1995.
- [Bou94] Pierre Boulez. *Penser la musique aujourd’hui*. Number 124 in Collection tel. Éditions Gonthier / Gallimard, Paris, 1963 / 1994.

- [Bou99] Richard Boulanger, editor. *The Csound Book: Perspectives in Software Synthesis, Sound Design, and Programming*. MIT Press, Cambridge, MA, 1999.
- [BP03] Sinan Bökesoy and Gérard Pape. Stochos: Software for real-time synthesis of stochastic music. *Computer Music Journal*, 27(3):33–43, 2003.
- [Bra89] Ronald N. Bracewell. Die Fourier-Transformation. *Spektrum der Wissenschaft*, pages 90–92, August 1989. German edition of Scientific American.
- [Bre90] Albert S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, Cambridge (MA), London, 1990.
- [Bre94] Martha Brech. *Analyse elektroakustischer Musik mit Hilfe von Sonagrammen*, volume 118 of *European University Studies 36*. Peter Lang, Frankfurt/Main, etc., 1994.
- [Bro05] Andrew R. Brown. Extending dynamic stochastic synthesis. In *Proceedings of the International Computer Music Conference*, pages 111–114, Barcelona, 2005. ICMA.
- [BRT80] Paul Berg, Robert Rowe, and David Theriault. SSP and sound description. *Computer Music Journal*, 4(1):25–35, 1980.
- [Brü69] Herbert Brüin. Infraudibles. In Heinz von Foerster and James W. Beauchamp, editors, *Music by Computers*, pages 117–121. John Wiley and Sons, New York, London, Sydney, Toronto, 1969.
- [Brü71] Herbert Brüin. *Über Musik und zum Computer*. Braun, Karlsruhe, 1971.
- [Brü83] Herbert Brüin. Notes to “Soniferous Loops” and “Non Sequitur VI”, 1983. cassette with 3 LPs.
- [Bur94a] Kristine Helen Burns. *The History and Development of Algorithms in Music Composition, 1957- 1993*. PhD thesis, Ball State University, Muncie, Indiana, Ann Arbor, 1994.
- [Bur94b] Kristine Helen Burns. History of electronic and computer music including automatic instruments and composition machines. <http://music.dartmouth.edu/wowem/electromedia/music/frames.html>, 1994.
- [Cam97] Antonio Camurri, editor. *Proceedings of KANSEI—The Technology of Emotion Workshop*. AIMI [Associazione di Informatica Musicale Italiana], 1997.
- [Cas92] Michael Casey. HS: A symbolic programming language for computer assisted composition. Abstract: <http://music.dartmouth.edu/Theses.html>, 1992.
- [Cas94] John L. Casti. *Complexification. Explaining a Paradoxical World Through the Science of Surprise*. Abacus, London, 1994.

- [Cha75] Gregory J. Chaitin. Randomness and mathematical proof. *Scientific American*, 232(5):47–52, May 1975.
- [Cha84] Joel Chadabe. Interactive composing: An overview. *Computer Music Journal*, 8(1):22–27, 1984.
- [Cha90] Jacques Chareyron. Digital synthesis of self-modifying waveforms by means of linear automata. *Computer Music Journal*, 14(4):25–41, 1990.
- [Cha94] Arun Chandra. The linear change of waveform segments causing non-linear changes of timbral presence. *Contemporary Music Review*, 10(2):157–169, 1994.
- [Cha96] Joel Chadabe. The history of electronic music as a reflection of structural paradigms. *Leonardo Music Journal*, 6:41–44, 1996.
- [Cha97a] Joel Chadabe. *Electric Sound. The Past and Promise of Electronic Music*. Prentice Hall, New Jersey, NY, etc., 1997.
- [Cha97b] Gregory J. Chaitin. The limits of mathematics, 1997. <http://www.cs.auckland.ac.nz/CDMTCS/chaitin/>.
- [Cha99] J. H. Chang. Composing noise. Technical report, Royal Conservatory Den Haag, Institute of Sonology, 1999.
- [Che95] Marc Chemillier. Analysis and computer reconstruction of a musical fragment of György Ligeti's melodien. *Muzika*, 6(2):34–48, 1995. extended version of an article presented at the International Symposium on Music and Mathematics, Bucarest 1994, <http://www.info.unicaen.fr/marc/Bucarestwww/>.
- [Chi93] Michel Chion. The state of *musique concrète*. *Contemporary Music Review*, 8:51–55, 1993. part 1.
- [Cho85] John Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7):526–534, 1985. Reprinted in C. Roads and J. Strawn (eds.): *Foundations of Computer Music*, Cambridge (MA): MIT Press, 1985, pp. 6–29.
- [Cho97] Insook Choi. A chaotic oscillator as a musical signal generator in an interactive performance system. *Interface: Journal of New Music Research*, 26:17–47, 1997.
- [Chu41] Alonzo Church. *The calculus of lambda conversion*. Number 6 in *Annals of Mathematical Studies*. Princeton University Press, Princeton, 1941.
- [Col] Nick Collins. GENDY1, GENDY2, GENDY3. <http://supercollider.svn.sourceforge.net/viewvc/supercollider/trunk/build/Help/UGens/Oscillators/Gendy3.html>.
- [Col05] Nick Collins. iGENDYN. <http://www.cogs.susx.ac.uk/users/nc81/iphone.html#iGendyn>, 2005.

- [Dac94] John Dack. Pierre Schaeffer and the significance of radiophonic art. *Contemporary Music Review*, 10:3–11, 1994. part 2.
- [Dah74] Carl Dahlhaus. *Zur Problemgeschichte des Komponierens*, pages 40–73. Number 7 in *Berliner Musikwissenschaftliche Arbeiten*. Musikverlag Emil Katzibichler, München, 1974.
- [Dav93] Paul Davies. *The Mind of God. Science and the Search for Ultimate Meaning*. Penguin, London etc., 1993.
- [Dav96] Hugh Davies. A history of sampling. *Organised Sound*, 1(1):3–11, 1996.
- [DBD<sup>+</sup>98] F. Dechelle, R. Borghesi, M. DeCecco, E. Maggi, B. Rován, and N Schnell. jMax: a new, JAVA-based editing and control system for real-time music applications. In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1998.
- [Del96] François Delalande. La musique électroacoustique, coupure et continuité. *Musurgia*, 3(3), 1996.
- [Del97] François Delalande. “Il faut être constamment un immigré”. *Entretiens avec Xenakis*. INA/GRM: Bibliothèque de Recherche Musicale. Buchet/Chastel, Paris, 1997. (Edited broadcasting interview from 1981).
- [Del02] François Delalande. Musik ohne Noten: Einführung in das Hören und Analysieren elektroakustischer Musik. In Rudolf Frisius, editor, *Konzert - Klangkunst - Computer. Wandel der musikalischen Wirklichkeit*, number 42 in *Veröffentlichungen des Instituts für Neue Musik und Musikerziehung Darmstadt*, pages 225–240, Mainz etc., 2002. Schott.
- [DiS94] Agostino DiScipio. Formal processes of timbre composition challenging the dualistic paradigm of computer music. In *Proc. of the International Symposium on Electronic Arts (ISEA)*, Helsinki, 1994. <http://www.uiah.fi/bookshop/isea.proc/nextgen/j/19.html>.
- [DiS95a] Agostino DiScipio. Centrality of techné for an aesthetic approach on electroacoustic music. *Journal of New Music Research*, 24(1):369–383, 1995.
- [DiS95b] Agostino DiScipio. Inseparable models of materials and of musical design in electroacoustic and computer music. *Journal of New Music Research*, 24(1):34–50, 1995.
- [DiS97] Agostino DiScipio. Questions concerning music technology. From Heidegger’s view to Feenberg’s Subversive Rationalization. *Sonus*, 18(1), 1997. Special Issue “Music and Technology”.
- [DiS98] Agostino DiScipio. From Concret PH to Gendy301. Compositional models in Xenakis’ electroacoustic music. *Perspectives of New Music*, 36(2):201–243, 1998.

- [Dob97] C. Dobrian. *MSP: the documentation*. Cycling '74, Santa Cruz, CA, 1997.
- [Doo00] Paul Doornbusch. *The Music of CSIRAC: Australia's First Computer Music*. The Humanities, Melbourne, 2000. ISBN: 1863355693.
- [Doo02] Paul Doornbusch. Corrosion. Music for instruments, computers and electronics. EMF CD 043, 2002. Available from [www.emf.org](http://www.emf.org).
- [Eck98] Gerhard Eckel. Technological musical artifacts. In *Proceedings of the First Symposium on Computer and Music*, pages 45–51, Corfu, Greece, 1998.
- [Eco96] Umberto Eco. *Das offene Kunstwerk*. Number 222 in Suhrkamp Taschenbuch Wissenschaft. Suhrkamp, Frankfurt/Main, 7 edition, 1996. Opera aperta, Milano: Casa Ed. Valentino Bompiani, 1962.
- [Egg91] Hans Heinrich Eggebrecht. *Musik im Abendland, Prozesse und Stationen vom Mittelalter bis zur Gegenwart*. Piper, München, Zürich, 1991.
- [Eic94] Randolph Eichert. *Xenakis und die mathematische Grundlagenforschung*. Number 5 in fragmen. Pfau, Saarbrücken, 1994.
- [Eig76] Manfred Eigen. Wie entsteht Information? – Prinzipien der Selbstorganisation. *Berichte der Bunsen-Gesellschaft für Physikalische Chemie*, 80:1059–1074, 1976.
- [Ess91] Karlheinz Essl. Computer aided composition. *Distel*, 46/47, 1991. <http://www.ping.at/users/essl/bibliogr/cac.html>.
- [Ess92] Karlheinz Essl. Kompositorische Konsequenzen des Radikalen Konstruktivismus. *positionen*, 11:2–4, 1992. reprinted in: Lothar Knessl (ed.), *zwischen-ton*, Wien, 1993. <http://www.ping.at/users/essl/bibliogr/rad-konstr.html>.
- [FBHL84] Abraham A Fraenkel, Yehoshua Bar-Hillel, and Azriel Levy. *Foundations of Set Theory*. Number 67 in Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers, Amsterdam, New York, 1984.
- [Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, New York, 3rd edition, 1968.
- [Fey99] Richard P. Feynman. *Feynman Lectures on Computation*. Penguin, London, 1999. Lecture notes edited by Hey, Anthony J. G. and Allen, Robin W.
- [FGG96] Golo Föllmer, Julia Gerlach, and Frank Gertich. *Musik..., verwandelt. Das Elektronische Studio der TU Berlin 1953-1995*. wolke verlag, Hofheim, Germany, 1996.
- [FH88] Anthony J. Field and Peter G. Harrison. *Functional Programming*. Addison-Wesley, Wokingham etc., 1988.

- [Föl02] Golo Föllmer. Towards a new Gesamtkunstwerk? Total Sampling, Re-Bricolage und Media- Hopping im Netz. In Rudolf Frisius, editor, *Konzert - Klangkunst - Computer. Wandel der musikalischen Wirklichkeit*, number 42 in Veröffentlichungen des Instituts für Neue Musik und Musikerziehung Darmstadt, pages 19–35, Mainz etc., 2002. Schott.
- [Fow02] Michael Fowler. The one-dimensional random walk. <http://galileo.phys.virginia.edu/classes/152.mfi.spring02/RandomWalk.htm>, 2002.
- [Fri02] Rudolf Frisius. Musikhören und Musikanalyse in einer gewandelten Wirklichkeit. In Rudolf Frisius, editor, *Konzert - Klangkunst - Computer. Wandel der musikalischen Wirklichkeit*, number 42 in Veröffentlichungen des Instituts für Neue Musik und Musikerziehung Darmstadt, pages 172–224, Mainz etc., 2002. Schott.
- [Gab58] Denis Gabor. Acoustical quanta and the theory of hearing. *Nature*, 159(1044):591–594, 1958.
- [Gan88] Robin Gandy. The confluence of ideas in 1936. In Rolf Herken, editor, *The Universal Turing Machine. A Half-Century Survey*, pages 55–111, Berlin, Hamburg, 1988. Kammerer & Unverzagt.
- [Gar85] H. Gardner. *The Mind's New Science: A History of the Cognitive Revolution*. Basic Books, New York, 1985.
- [Ges98] Yann Geslin. Sound and music transformation environments: A twenty-year experiment at the “Groupe de Recherches Musicales”. In Xavier Serra, editor, *Proceedings of the First COST-G6 Workshop on Digital Audio Effects (DAFX98)*, pages 241–248, Barcelona, 1998. Audiovisual Institute (IUA) of the Pompeu Fabra University.
- [Gib01] Benoît Gibson. Théorie des cribles. In Makis Solomos, editor, *Présences de Iannis Xenakis*, Paris, 2001. CDMC.
- [GL88] Les Goldschlager and Andrew Lister. *Computer Science. A Modern Introduction*. Prentice Hall International, London, 1988.
- [Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.
- [Goe98] Jeanpaul Goergen. Das Wunder des gezeichneten Tons. Die “tnende Handschrift” von Rudolf Pfenninger. *Filmblatt*, 4(9):15–17, 1998. <http://filmblatt.de/index.php?id=63,85,0,0,1,0>.
- [Gor96] Vande Gorne. Une histoire de la musique électroacoustique. *Revue Ars Sonora*, 1996. [http://www.imagnet.fr/manca/invite/asr/asr3\\_07.html](http://www.imagnet.fr/manca/invite/asr/asr3_07.html).
- [Gre00] Brian Greene. *The Elegant Universe. Superstrings, Hidden Dimensions, and the Quest for the Ultimate Theory*. Vintage, London etc., 2000.

- [Han54] Eduard Hanslick. *Vom musikalisch Schönen. Ein Beitrag zur Revision der Ästhetik der Tonkunst*. Rudolph Weigel, Leipzig, 1854.
- [Har99] James Harley. *The Music of Iannis Xenakis*. Contemporary Music Studies. Harwood Academic Press, 1999.
- [HBLB98] Peter Hoffmann, Guillaume Ballet, Fabien Lévy, and Riccardo Borghesi. Studio online: a “Killer Application” for the internet. In *Journées d’Informatique Musicale JIM99*, CNET, Issy-Les-Moulineaux, France, 1998.
- [Hei79] Werner Heisenberg. *Quantentheorie und Philosophie: Vorlesungen und Aufsätze*. Reclam, Stuttgart, 1979.
- [HI58] Lejaren Hiller and Leonard Isaacson. Musical composition with a high-speed digital computer. *Journal of the Audio Engineering Society*, 1958. reprinted in Schwanauer, Stephan M., Levitt, David A. (eds), *Machine Models of Music*, Cambridge (MA), London: MIT Press, 1993, pp. 9-21.
- [Hil70] Lejaren Hiller. Music composed with computers - a historical survey. In Harry B. Lincoln, editor, *The Computer and Music*, pages 42–96. Cornell University, Ithaca, N.Y, 1970.
- [Hil81] Lejaren Hiller. Composing with computers: A progress report. *Computer Music Journal*, 5(4):7–21, 1981. Reprinted in *The Music Machine*, ed. Curtis Roads, 1989, Cambridge MA, MIT Press, pp. 75–89.
- [HKO91] Andy Hunt, Ross Kirk, and Richard Orton. Musical applications of a cellular automata workstation. In *Proceedings of the International Computer Music Conference*, pages 165–168, Montreal, 1991. International Computer Music Association.
- [Hoc96] Jürgen Hocker. Das Pianola, 1996. Konzert vom 21. September, Musikalisches Nachtstudio, Staatbibliothek, Otto-Braun-Saal, 22.30, program notes.
- [Hof94a] Peter Hoffmann. *Amalgam aus Kunst und Wissenschaft: Naturwissenschaftliches Denken im Werk von Iannis Xenakis*, volume 110 of *European University Studies 36*. Peter Lang, Frankfurt/Main, etc., 1994.
- [Hof94b] Peter Hoffmann. Implementierung funktionaler Programmiersprachen durch Graphersetzungssysteme. Technical report, Freie Universität Berlin, computer science department, 1994.
- [Hof95] Peter Hoffmann. Zelluläre Automaten als Grenzphänomene zwischen Physik und Informatik. unpublished, 1995.
- [Hof96] Peter Hoffmann. Implementing the Dynamic Stochastic Synthesis. In *Troisièmes journées d’informatique musicale (JIM 96)*, volume 4 of *Les cahiers du GREYC [Groupe de Recherches en Informatique, Image, Instrumentation de Caen]*, pages 341–347, Ile de Tatihou, France, 1996. <http://www.ircam.fr/repmus/jim96/>.

- [Hof98] Peter Hoffmann. Design and implementation of the new GENDYN program. In *XIIIth Colloquium on Musical Informatics (CIM) 98*, Gorizia, Italy, 1998.
- [Hof00] Peter Hoffmann. The new GENDYN program. *Computer Music Journal*, 24(2):31–38, 2000.
- [Hof01a] Peter Hoffmann. Analysis through resynthesis: Gendy3 by Iannis Xenakis. In Makis Solomos, editor, *Présences de Iannis Xenakis*, pages 185–194, 16, place de la Fontaine aux Lions, 75019 Paris, 2001. Centre de documentation de la musique contemporaine (CDMC).
- [Hof01b] Peter Hoffmann. La musique électroacoustique de Iannis Xenakis. In François-Bernard Mâche, editor, *Portrait(s) de Iannis Xenakis*, Portrait(s), pages 171–182. Bibliothèque Nationale de France, Paris, 2001.
- [Hof02] Peter Hoffmann. Towards an 'Automated Art': Algorithmic Processes in Xenakis' Compositions. *Contemporary Music Review*, 21(2/3):121–131, 2002.
- [Hof04] Peter Hoffmann. “something rich and strange” exploring the pitch structure of gendy3. *Journal of New Music Research*, 33(2):137–144, 2004. publisher: Routledge, part of the Taylor & Francis Group.
- [Hof07] Boris Hofmann. *Mitten im Klang. Die Raumkompositionen von Iannis Xenakis aus dem 1960er Jahren*. PhD thesis, Technische Universität Berlin, 2007.
- [Hol79] Steven R. Holtzman. An automated digital sound instrument. *Computer Music Journal*, 3(2):53–61, 1979.
- [Hol94] Steven R. Holtzman. *Digital Mantras: The Languages of Abstract and Digital Worlds*. MIT Press, Cambridge, MA, 1994.
- [Jac95] Bruce L. Jacob. Composing with genetic algorithms. In *Proceedings of the International Computer Music Conference*, Banff Alberta, 1995. International Computer Music Association. [http://www.eecs.umich.edu/blj/papers/Proceedings of the International Computer Music Conference95.html](http://www.eecs.umich.edu/blj/papers/Proceedings%20of%20the%20International%20Computer%20Music%20Conference95.html).
- [Jaf95] David A. Jaffe. 10 criteria for synthesis techniques. *Computer Music Journal*, 19(1):76–87, 1995.
- [JL83] Ray Jackendoff and Fred Lerdahl. *A Generative Theory of Tonal Music*. MIT Press, Cambridge (MA), London, 1983.
- [Joh97] Tom Johnson. Lab reports. In *Journées d'Informatique Musicale JIM97*, Lyon, France, 1997. <http://www.grame.fr/jim97/proceedings.html>.
- [Joh98] Tom Johnson. Automatic music. In *Journées d'Informatique Musicale JIM98*, number 148 in Publications du Laboratoire de mécanique et d'acoustique (LMA), pages F1–1–F1–5, Marseille, France, 1998. Centre National de la recherche scientifique (CNRS) Marseille. 05.-07.05.1998, La Londe-les-Maures, France.

- [Jon81] Kevin Jones. Compositional applications of stochastic processes. *Computer Music Journal*, 5(2):45–61, 1981.
- [Kae86] Werner Kaegi. Controlling the VOSIM sound synthesis system. *Interface*, 15:71–82, 1986.
- [KB] Rebecca Kim and Liubo Borissov. Iannis Xenakis’s Bohor (1962). <http://www.music.columbia.edu/liubo/bohor/present/>.
- [Kin91] Hartmut Kinzler. György Ligeti: Decision and automatism in ‘désordre’, 1ère étude, premier livre. *Interface*, 20:89–124, 1991.
- [Koe83] Gottfried Michael Koenig. Aesthetic integration of computer-composed scores. *Computer Music Journal*, 7(4):27, 1983. Reprinted in *The Music Machine*, ed. Curtis Roads, 1989, Cambridge MA, MIT Press, p. 620.
- [Kra94] Gregory Kramer, editor. *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume 18 of *SFI Studies in the Sciences of Complexity, Proc.*, Reading MA etc., 1994. The Santa Fe Institute (SFI), Addison-Wesley. Proceedings of the First International Conference of Auditory Display (ICAD), Santa Fe, New Mexico, 1992.
- [Lam86] Leslie Lamport. *LaTeX. A Document Preparation System*. Addison Wesley, Reading, MA etc., 1986.
- [Lar97] Régis Renouard Larivière. A propos du concept ‘objet sonore’. *Revue Ars Sonora*, 1997. [http://www.imagnet.fr/manca/invite/asr/005asr/asr5\\_04.html](http://www.imagnet.fr/manca/invite/asr/005asr/asr5_04.html).
- [Las88] Otto Laske. Introduction to cognitive musicology. *Computer Music Journal*, 12(1):43–57, 1988.
- [Las92] Otto E. Laske. “furies and voices”: Composition-theoretical observations. In Denis Baggi, editor, *Readings in Computer-Generated Music*, pages 181–197. IEEE Computer Society Press, Los Alamos, CA, etc., 1992.
- [Lau96] M. Laurson. *PatchWork: A Visual Programming Language and Some Musical Applications*. PhD thesis, Sibelius Academy, Helsinki, 1996.
- [LDSS91] Eric Lindemann, Francois Dechelle, Bennett Smith, and Michel Starkier. The architecture of the IRCAM musical workstation. *Computer Music Journal*, 15(3):41–9, 1991.
- [Lév03] Fabien Lévy. *Complexité grammatologique / complexité perceptive en musique. Étude esthétique et scientifique du décalage entre la production et la réception des processus musicaux sous l’angle des théories de l’information et de la complexité*. PhD thesis, École des Hautes Études en Sciences Sociales (EHESS), Paris, 2003.
- [Lig70] György Ligeti. *Articulation*. Schott, Mainz, 1970. Transcript of the 1958 tape composition by Rainer Wehinger.

- [Lit93] David Little. Composing with chaos; applications of a new science for music. *Interface*, 22(1):23–51, 1993.
- [Lor80] Denis Lorrain. A panoply of stochastic 'canons'. *Computer Music Journal*, 4(1):53–81, 1980.
- [Loy89] Gareth Loy. Composing with computers - a survey of some compositional formalisms and music programming languages. In Max V. Mathews and John R. Pierce, editors, *Current Directions in Computer Music Research*, pages 291–396. MIT Press, Cambridge, MA, 1989.
- [LR81] Louis LePrince-Ringuet. Sur Iannis Xenakis. In Maurice Fleuret, editor, *Regards sur Iannis Xenakis*, page 53. Stock, Paris, 1981.
- [Mac84] Tod Machover. A view of music at IRCAM. *Contemporary Music Review*, 1:1–10, 1984. part I (Special Issue: “Musical Thought at IRCAM”).
- [Man83] Benoît Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman and Company, New York, 1983.
- [McA87] Stephen McAdams. Music, a science of the mind? *Contemporary Music Review*, 2:1–61, 1987.
- [McC96] James McCarthy. Supercollider: a new real-time synthesis language. In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1996.
- [Mil90] Dale Millen. Cellular automata music. In *Proceedings of the International Computer Music Conference Glasgow*, pages 314–316. International Computer Music Association, 1990.
- [Mir95a] Eduardo Reck Miranda. An artificial intelligence approach to sound design. *Computer Music Journal*, 19(2):59–96, 1995.
- [Mir95b] Eduardo Reck Miranda. Granular synthesis of sounds by means of a cellular automaton. *Leonardo Music Journal*, 28(4):297–300, 1995.
- [Mir98] Eduardo Reck Miranda. *Computer Sound Synthesis for the Electronic Musician*. Focal Press, Oxford, 1998.
- [Mir99] Eduardo Reck Miranda. Evolving complex sounds with cellular automata: an approach to granular synthesis. In *Journées d'Informatique Musicale (JIM'99)*, pages 109–115, CNET, Issy-Les-Moulineaux, France, 1999.
- [Mor93] William Moritz. Optische Poesie: Oskar Fischinger — Leben und Werk. In Hilmar Hoffmann and Walter Schobert, editors, *Schriftenreihe des Deutschen Filmmuseums*, number 9 in Kinematograph, pages 7–90. Deutsches Filmmuseums Frankfurt/Main, Frankfurt/Main, 1993.
- [Mor94] Hans Moravec. *The Age of Mind. Transcending the Human Condition Through Robots*. Bantam Books, 1994.

- [Mos91] Ulrich Mosch. Informationstheorie und Komponieren. *Positionen. Beiträge zur Neuen Musik*, pages 22–26, 1991.
- [MRS93] Gérard Marino, Jean-Michel Racziński, and Marie-Hélène Serra. The new UPIC system: Origins and innovations. *Perspectives of New Music*, 31(1):258–269, 1993.
- [MS94] Stephanie Mason and Michael Saffle. L-systems, melodies and musical structure. *Leonardo Music Journal*, 4(4):31–38, 1994.
- [MWSH93] Eduardo Reck Miranda, Geraint Wiggins, Alan Smaill, and Mitch Harris. Surveying musical representation systems: A framework for evaluation. *Computer Music Journal*, 17(3):31–42, 1993.
- [Myh52] John Myhill. Some philosophical implications of mathematical logic: Three classes of ideas. *Review of Metaphysics*, 6(2):165–198, 1952.
- [Nez00] Netochka Nezvanova. The internet. A musical instrument in perpetual flux. *Computer Music Journal*, 24(3):38–41, 2000.
- [Nyq28] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 1928.
- [OHK91] Richard Orton, A. Hunt, and Ross Kirk. Graphical control of granular synthesis using a cellular automaton and the freehand program. In *Proceedings of the International Computer Music Conference Montreal*, pages 416–418. International Computer Music Association, 1991.
- [Ort96] Richard Orton. Design strategies for algorithmic composition. *Contemporary Music Review*, 15:39–48, 1996. parts 3-4.
- [Pap95] Gérard Pape. Composition and the structure of sound. Manuscript in the library of Les Ateliers UPIC (now Centre de création musicale Iannis Xenakis, Paris), undated [1995].
- [Paped] Gérard Pape. Sound sculpting with UPIC. Manuscript in the library of Les Ateliers UPIC (now Centre de création musicale Iannis Xenakis, Paris), undated.
- [Pen88] Roger Penrose. On the physics and mathematics of thought. In Rolf Herken, editor, *The Universal Turing Machine. A Half-Century Survey*, pages 491–522, Berlin, Hamburg, 1988. Kammerer & Unverzagt.
- [Pen89] Roger Penrose. *The Emperor's New Mind. Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press / Vintage, London etc., 1989.
- [Pen94] Roger Penrose. *Shadows of the Mind. A Search for the Missing Science of Consciousness*. Oxford University Press / Vintage, London etc., 1994.

- [pHH96] [pseudonym:] Huge Harry. A computational perspective on twenty-first century music. *Contemporary Music Review*, 15(3–4):151–157, 1996.
- [Pie92] John Pierce. Recollections by John Pierce. In *The Historical CD of Digital Sound Synthesis*, number 13 in Computer Music Currents, pages 9–29. Schott Wergo Music Media, Mainz, 1992. CD WER 2033-2, 1995.
- [Pop91] Steven Travis Pope, editor. *The Well-tempered Object — Musical Applications of the Object-Oriented Software Design*. MIT Press, Cambridge, MA, 1991.
- [Puc88] Miller Puckette. The patcher. In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1988.
- [Puc91] Miller Puckette. FTS: A real-time monitor for multiprocessor music synthesis. *Computer Music Journal*, 15(3):58–67, 1991.
- [Puc96] Miller Puckette. Pure data. In *Proceedings of the International Computer Music Conference*, pages 58–67. International Computer Music Association, 1996.
- [PZ90] Miller Puckette and David Zicarelli. *MAX — An Interactive Graphical Programming Environment*. Opcode Systems, Menlo Park, 1990.
- [RDF93] Xavier Rodet, Philippe Depalle, and Adrian Freed. Performance, synthesis and control of additive synthesis on a desktop computer using  $FFT^{-1}$ . In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1993.
- [Rei93] Dirk Reith. Elektronische Musik und Algorithmische Komposition. In Bernd Enders, editor, *Neue Musiktechnologie*, pages 124–137. B. Schott’s Söhne, Mainz etc., 1993.
- [RM88] André Riotte and Marcel Mesnage. Analyse musicale et systèmes formels: un modèle informatique de la 1<sup>ère</sup> pièce pour quatuor à cordes de Stravinsky. *Analyse musicale*, 4(10):51–67, 1988.
- [RM89] André Riotte and Marcel Mesnage. Les variation pour piano opus 27 d’Anton Webern. Approche cellulaire baraquénne et analyse assistée par ordinateur. *Analyse musicale*, 5(14):41–67, 1989.
- [Roa78] Curtis Roads. Automated granular synthesis of sound. *Computer Music Journal*, 2(2):61–62, 1978. revised and updated version in Curtis Roads and John Strawn, editors, *Foundations of Computer Music*, pages 145–159. MIT Press, Cambridge, MA, London, 1985.
- [Roa85a] Curtis Roads. Grammars as representations for music. In Curtis Roads and John Strawn, editors, *Foundations of Computer Music*, pages 402–442. MIT Press, Cambridge, MA, 1985.

- [Roa85b] Curtis Roads. Interview with Gottfried Michael Koenig. In Curtis Roads and John Strawn, editors, *Foundations of Computer Music*, pages 568–580. MIT Press, Cambridge, MA, London, 1985.
- [Roa92] Curtis Roads. Composition with machines. In John Paynter, Tim Howell, Richard Orton, and Peter Seymour, editors, *Companion to Contemporary Musical Thought*, volume 1, pages 399–419. Routledge, London, 1992.
- [Roa94] Curtis Roads. Physical modeling: The history of digital simulations of acoustic instruments. *Keyboard*, 20(9):89,91–4,100–3, 1994.
- [Roa96] Curtis Roads. *The Computer Music Tutorial*. MIT Press, Cambridge, MA etc., 1996.
- [Roa02] Curtis Roads. *Microsound*. MIT Press, Cambridge, MA, 2002.
- [Roj95] Raol Rojas. On basic concepts of early computers in relation to contemporary computer architectures. In *Proceedings of the 13th World Computer Congress*, volume 2, pages 324–331, Hamburg, 1995. <http://www.inf.fu-berlin.de/rojas>.
- [Ros88] Robert Rosen. Effective processes and natural law. In Rolf Herken, editor, *The Universal Turing Machine. A Half-Century Survey*, pages 523–537, Berlin, Hamburg, 1988. Kammerer & Unverzagt.
- [Rot87] Gerhard Roth. Autopoiesis und Kognition: Die Theorie H. R. Maturanas und die Notwendigkeit ihrer Weiterentwicklung. In Siegfried J. Schmidt, editor, *Der Diskurs des Radikalen Konstruktivismus*, number 636 in Suhrkamp Taschenbuch Wissenschaft, pages 256–286. Suhrkamp, Frankfurt/Main, 1987.
- [Row92] Robert Rowe. *Interactive Music Systems*. MIT Press, Cambridge, MA, 1992.
- [RP98] Francis Rousseau and François Pachet. La question des espaces dans le projet “la partition intérieure interactive”. In J.-M. Chouvel and Makis Solomos, editors, *L’espace: Musique/Philosophie*, Collection Musique et Musicologie: les Dialogues, pages 41–52, Paris, 1998. L’Harmattan.
- [SC91] Carla Scaletti and Alan B. Craig. Using sound to extract meaning from complex data. Technical report, University of Illinois, National Center for Supercomputing Applications, 1991. [http://www.ncsa.uiuc.edu/VRIVR/vr\\_dis\\_papers.html](http://www.ncsa.uiuc.edu/VRIVR/vr_dis_papers.html).
- [Sca87] Carla Scaletti. Kyma: An object-oriented language for music composition. In *Proceedings of the International Computer Music Conference*, 1987.
- [Sch] Schottstead. Common lisp music. <http://ccrma.stanford.edu/software/clm/>.

- [Sch44] Erwin Schrödinger. *What is Life? The Physical Aspects of the Living Cell*. Cambridge University Press, Cambridge, 1944. German edition: Was ist Leben? Die lebende Zelle mit den Augen des Physikers betrachtet, München: Piper 1999.
- [Sch52] Pierre Schaeffer. *À la recherche d'une musique concrète*. Éditions du Seuil, 1952.
- [Sch66] Pierre Schaeffer. *Traité des objets musicaux. Essais interdisciplines*. Éditions du Seuil, 1966. reprinted 1998.
- [Sch93] David A. Schwanauer, Stephan M. and Levitt, editor. *Machine Models of Music*. MIT Press, Cambridge (MA), London, 1993.
- [Sch98] Eric D. Scheirer. The mpeg-4 structured audio orchestra language. In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1998.
- [SE98] Joachim Stange-Elbe. "Cooking a Canon with RUBATO©". Performance Aspects of J. S. Bach's "Kunst der Fuge". In *Proceedings of the International Computer Music Conference 1998*, pages 179–186, 1998.
- [Sei92] Uwe Seifert. Cognitive science: A new research program for musicology. *Interface*, 21:219–238, 1992.
- [Ser93] Marie-Hélène Serra. Stochastic composition and stochastic timbre: GENDY3 by Iannis Xenakis. *Perspectives of New Music*, 31:236–257, 1993.
- [Sin99] Jenny Sinclair. "This 'old' hand hid a note or two". *Fairfax I.T. News From the World of Information Technology*, 1999. <http://www.it.fairfax.com.au/hardware/19991123/A57309-1999Nov22.html>.
- [Smi91] Julius O. Smith. Viewpoints on the history of digital synthesis. In *Proceedings of the International Computer Music Conference*, pages 1–10, Montreal, 1991. McGill University, International Computer Music Association.
- [Sol96] Makis Solomos. *Iannis Xenakis*. Compositeurs. Echos du XXe siècle. P.O.Éditions, Mercuès, 1996.
- [Sol97] Makis Solomos. Schaeffer phénoménologue. In Denis Dufour, editor, *D'après Schaeffer*. INA/GRM, Paris, 1997.
- [Spr57] A. M. Springer. Acoustical time regulator. *Journal of the Acoustic Society of America*, 29(3):341–343, 1957.
- [Sto90] Tom Stonier. *Information And the Internal Structure of the Universe*. Springer, London, 1990. German edition: Berlin, Heidelberg: Springer, 1991.
- [Str84] Marco Stroppa. The analysis of electronic music. *Contemporary Music Review*, 1:175–180, 1984.

- [Sup97] Martin Supper. *Elektroakustische Musik und Computermusik. Geschichte – Ästhetik – Methoden – Systeme*. Wissenschaftliche Buchgesellschaft Darmstadt, 1997. Lizenzausgabe wolke verlag, Hofheim-Taunus, 1997.
- [SW49] Claude E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, 1949.
- [Szi29] Leo Szilard. Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen. *Zeitschrift für Physik*, 53:840–856, 1929.
- [Tau91] Heinrich Taube. Common music: A music composition language in Common LISP and CLOS. *Computer Music Journal*, 15(2):21–32, 1991.
- [Ter98] Daniel Teruggi. *Le système Syter. Son histoire, ses développements, sa production musicale, des implication dans le langage électroacoustique d'aujourd'hui*. PhD thesis, Université Paris VIII, Paris, 1998.
- [Tur36] Allan Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 45(2):230–265, 1936.
- [Tur50] Allan Turing. Computing machinery and intelligence. *Mind*, 59:33–60, 1950.
- [Var96] Bálint Andràs Varga. *Conversations with Iannis Xenakis*. Faber and Faber, London, 1996.
- [VTR91] J. Varela, Francisco, Evan Thompson, and Eleanor Rosch. *The Embodied Mind. Cognitive Science and Human Experience*. MIT Press, Cambridge (MA), London, 1991.
- [Weg97] Peter Wegner. Why interaction is more powerful than algorithms. *Communications of the Association for Computing (CACM)*, 40(5):80–91, May 1997. see also <http://www.cs.brown.edu/people/pw/>.
- [Wol84] Steven Wolfram. Software für Mathematik und Naturwissenschaften. *Spektrum der Wissenschaft*, 7(11):151–157, November 1984. (German edition of The Scientific American).
- [Wol94] Steven Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, Reading, MA, 1994.
- [Xen55] Iannis Xenakis. La crise de la musique sérielle. *Gravesaner Blätter*, 1:2–4, 1955. reprinted in Kéleütha, Paris: L'Arche, 1994, pp. 39–43.
- [Xen57] Iannis Xenakis. Les événements rares. unpublished typescript, copy in the library of the Internationales Musikinstitut, Darmstadt, partly reprinted in Xenakis, *Formalized Music*, 1992, 1957.

- [Xen60] Iannis Xenakis. Elements of stochastic music. *Gravesaner Blätter*, 18:84–105, 1960. later incorporated into Xenakis: Musique formelles / Formalized Music.
- [Xen63] Iannis Xenakis. Musiques formelles, nouveaux principes formels de composition musicale. *Revue Musicale no 253–254*, 1963. Re-edition, with appendix of complete listing of the ST-program: Paris:Stock, 1981.
- [Xen78] Iannis Xenakis. *La Légende d'Eer*, pages 8–12. Centre Georges Pompidou, Paris, 1978.
- [Xen79] Iannis Xenakis. *Art/Sciences. Alliages*. Casterman, Tournai, 1979. Transcript of Xenakis' thesis defence held at La Sorbonne University, Paris, in 1976.
- [Xen85] Iannis Xenakis. Music composition treks. In Curtis Roads, editor, *Composers and the Computer*, pages 172–191. William Kaufmann, Los Altos, CA, 1985.
- [Xen91] Iannis Xenakis. Dynamic stochastic synthesis. In *Proceedings of the International Computer Music Conference*, Montreal, 1991. International Computer Music Association.
- [Xen92a] Iannis Xenakis. *Formalized Music. Thought and Mathematics in Composition*, chapter New Proposals in Microsound Structure, pages 242–253. Pendragon Press, Stuyvesant, NY, 1992.
- [Xen92b] Iannis Xenakis. *Formalized Music. Thought and Mathematics in Composition*. Pendragon Press, Stuyvesant NY, 1992. revised edition, with additional material compiled by Sharon Kanach.
- [Xen92c] Iannis Xenakis. *Formalized Music. Thought and Mathematics in Composition*, chapter Dynamic Stochastic Synthesis, pages 289–293. Pendragon Press, Stuyvesant, NY, 1992.
- [Xen92d] Iannis Xenakis. *Formalized Music. Thought and Mathematics in Composition*, chapter More Thorough Stochastic Music, pages 295–299. Pendragon Press, Stuyvesant, NY, 1992.
- [Xen92e] Iannis Xenakis. *Formalized Music. Thought and Mathematics in Composition*, chapter Concerning Time, Space and Music, pages 255–267. Pendragon Press, Stuyvesant, NY, 1992. Similar to “Sur le temps” in *Redécouvrir le temps*, Bruxelles: Éditions de l'Université, tome II, 1988, pp. 193–200.
- [Xen94a] Iannis Xenakis. Gendy3 for computer-generated tape. Neuma CD 450-862, 1994. available from [www.emf.org](http://www.emf.org).
- [Xen94b] Iannis Xenakis. *Musique et originalité*, pages 39–43. L'Arche, Paris, 1994.
- [Xen96] Iannis Xenakis. Determinacy and indeterminacy. *Organised Sound*, 1(3):143–155, 1996.

- [Xen97] Catalogue général des œuvres. Paris, 1997.
- [Zei03] Anton Zeilinger. *Einsteins Schleier*. C.H. Beck, München, 2003.

# Appendix A

## Stochastic Distributions Used in *GENDY3*

Here is the collection of “inverted” probability distribution formulae used in GENDYN (see chapter 15.1 on page 268 for their description). They are to transform a uniformly distributed number sequence between 0 and 1 into a number sequence distributed in a way characteristic to its corresponding “non-inverted” distribution formula (Exponential, Cauchy, Arcus sinus, etc.). In this way, they function as a kind of “external force” driving the random walks of time and amplitude variation (see chapter 3.8 on page 56). For an in-depth discussion of that matter, see the seminal article by [Lor80].

In the collection of equations below,  $f(x)$  denotes the probability density function.  $F(x)$  denotes the corresponding distribution function (the integral of the density function).  $F(y)$  denotes the “inverted” distribution function. What does it mean, “inverted”? Think of it the following way: In the case of  $F(x)$ , you put in some number  $x$  and you get out a probability number  $y$  between 0 and 1 out. In the case of  $F(y)$ , you put in a number  $y$  between 0 and 1 in and you get out some number  $x$ . Thinking of the graph of the distribution, in the first case you are looking for the  $y$  coordinate (scaled between 0 and 1) of a point on that graph, given its  $x$  coordinate. In the second case, you are looking for the  $x$  coordinate of a point on the same graph, given its  $y$  coordinate. You can see examples of these graphs in figures 14.10 on page 230.

Cauchy distribution:

$$f(x) = \frac{1}{\pi} \left( \frac{\alpha}{(\alpha^2 + x^2)} \right) \quad (\text{A.1})$$

$$F(x) = \frac{1}{2} + \left( \frac{1}{\pi} \arctan \left( \frac{x}{\alpha} \right) \right) \quad (\text{A.2})$$

$$F(y) = \alpha \tan \left( \pi \left( y - \frac{1}{2} \right) \right) \quad (\text{A.3})$$

“Logistic” distribution:

$$f(x) = \frac{-\alpha e^{-\alpha x - b}}{(1 + e^{-\alpha x - b})^2} \quad (\text{A.4})$$

$$F(x) = \frac{1}{1 + e^{-\alpha x - b}} \quad (\text{A.5})$$

$$F(y) = -\frac{\ln\left(\frac{1-y}{y}\right) + \beta}{\alpha} \quad (\text{A.6})$$

Cosinus Hyperbolicus:

$$f(x) = \frac{1}{\pi \cosh x} \quad (\text{A.7})$$

$$F(x) = 1 - \frac{2}{\pi} \arctan(e^{-x}) \quad (\text{A.8})$$

$$F(y) = \alpha \ln\left(\tan\left(\frac{y\pi}{2}\right)\right) \quad (\text{A.9})$$

Arcus Sinus:

$$f(x) = \frac{1}{\pi\sqrt{x(1-x)}} \quad (\text{A.10})$$

$$F(x) = \frac{2}{\pi} \arcsin(\sqrt{x}) \quad (\text{A.11})$$

$$F(y) = \alpha \left( \frac{1}{2} - \frac{1}{2} \sin\left(\left(\frac{1}{2} - y\right)\pi\right) \right) \quad (\text{A.12})$$

Exponential distribution:

$$f(x) = \alpha^2 e^{-\alpha^2 x} \quad (\text{A.13})$$

$$F(x) = 1 - e^{-\alpha^2 x} \quad (\text{A.14})$$

$$F(y) = -\frac{\ln(1-y)}{\alpha} \quad (\text{A.15})$$

## Appendix B

# Core C++ Code and User Manual of the New GENDYN Program

```

// G E N D Y N by Iannis Xenakis
//=====
//24-9-93/4-11-1992/9-1992 QBasic Version by Xenakis
//11-1995/01-96/03-96 C++ Version by Peter Hoffmann

// =====
// "This programme controls several stochastic-dynamic sound-fields.
// A stochastic dynamic sound-field is made out of a wave-length
// divided in a certain number of segments. Each one of these segments
// is stochastically varied by a cumulated probability-distribution.
// At the ends of each segment are computed the amplitudes that will
// form the waveform polygone placed on that wave-length.
// For the abscissa: a probability distribution and 2 times 2 elastic
// mirrors. For the amplitudes: a probability distribution and 2 times
// 2 elastic mirrors. In between the vertices a linear interpolation
// that completes the waveform polygone." (I.X.)
// =====
#include <math.h> // for trigonometric functions
#include "stdafx.h" // for memory allocation
#include "gendyn07.h"
//===== Constants =====
const attackTime = 500; // 10000 = ca. 226 ms if sampling rate = 44.1 kHz
const decayTime = 1000; // 10000 = ca. 226 ms if sampling rate = 44.1 kHz
const float PI = 3.14159265359;
//===== Global Access Variables =====
Control *control;
fstream dataFile; // global for objects to share (static member doesn't work)
Output output; // writing sound data to sound device or disk
Random rnd; // random number generator
Lehmer lehmer; // random generator using Lehmer's formula
//===== Default control settings =====
// NOTE: these variables have to be accessible from "static GenControl::generate"
// set initial button position for play control:
short GenControl::suspended = TRUE; // boolean switch to idle loop
short GenControl::reset = FALSE; // boolean switch to reset computation
// set initial source and target of synthesis:
short GenControl::record = FALSE;
short GenControl::preview = FALSE;
short GenControl::seqID = 1;
short GenControl::nTrack = 1;
// reasonable initial sound device settings:
short GenControl::bufferTime = 1; // buffering in seconds
long GenControl::seconds = 10; // sound length to be produced
short GenControl::writeHeader = TRUE; // prepend sound file format?
short GenControl::soundFileOpen = FALSE; // boolean state flag
long GenControl::noOfSamples = 4410; // to be produced at each call
// set initial synthesis object reference to "invalid"
Sound *GenControl::sound = 0; // a particular sound (i.e. track)
Piece *GenControl::piece = 0; // the whole piece to be generated
// output flags:
short GenControl::blocked = 0; // retry writing same sample
//===== Class Member Functions =====

```

```

//===== Filters =====
inline void Filter::pass (short & value) {
//*****
/* shuffle sample over delay line and add up weighted contributions */
//*****
    short i; // delay time in samples
    // shift delay line one sample right:
    for (i = 1; i <= order; i++) {
        delay[i] = delay[i - 1];
    }
    delay[0] = value; // prepend current sample to delay line
    value /= float(length); // init accumulator
    for (i = 1; i <= order; i++) {
        value += (delay[i] / float(length)); // add up uniformly weighted delay
    }
}

//===== Elastic Barriers =====
inline void Mirrors::pass (float & value) {
//*****
/* elastic mirrors or barriers */
/* NOTE: this is only integer arithmetic. */
//*****
    short swap; // swap intermediate
    long excess; // to measure excess of argument value
    long reflection; // amount of reflection towards the inside

    if (upper == lower) { // no space between barriers
        value = upper; // return result independent of value and exit
        return;
    }
    else if (lower > upper) {
        swap = lower, lower = upper, upper = swap; // swap barriers and proceed
    }
    if (value > upper) { // value exceeds upper mirror?
        //NOTE: for compatibility with GENDY301, simulate Basic float to int conversion!
        excess = long(value - upper + 0.4999999); // by how much does it exceed?
        if (excess) {
            reflection = long(excess % diameter); // compute last reflection
            if ((excess / diameter) % 2) { //reflected 2, 4, 6, ... times
                value = lower + reflection; // value bounced off lower mirror
            }
            else {
                value = upper - reflection; // value bounced off upper mirror
            }
        }
        else {
            value = upper;
        }
    }
    else if (value < lower) { // value smaller than lower mirror?
        //NOTE: for compatibility with GENDY301, imulate Basic float to int conversion!
        excess = long(lower - value + 0.4999999); // by how much does it fall short?
        if (excess) {

```

```

        reflection = long(excess % diameter); // compute last reflection
        if ((excess / diameter) % 2) { // reflected 2, 4, 6, ... times
            value = upper - reflection; // value bounced off upper mirror
        }
        else {
            value = lower + reflection; // value bounced off lower mirror
        }
    }
    else {
        value = lower;
    }
}
}

```

```

inline void RandomWalk::step (float & currPosition, short seg) {
//*****
/* Random Walk of polygon vertex coordinates by 1) drawing a stochastic *
/* variation value and 2) adding it to value of the corresponding *
/* vertex coordinate in the predecessor polygon. *
/* Both are controlled by a pair of reflecting (elastic) mirrors *
//*****
    if (!(mirrorFluct && mirrorWalk && distr)) return; // make sure objects exist
    if (mirrorWalk->isOpen()) {
        if (mirrorFluct->isOpen()) {
            flucts[seg] = flucts[seg] + distr->lookup ();
        }
        mirrorFluct->pass (flucts[seg]); // limit fluctuation
        currPosition += flucts[seg]; // execute random step
    }
    mirrorWalk->pass (currPosition); // limit cumulation
}

```

```

//===== Probability Distribution Functions =====
inline float Distr301::lookup (void) {
    // NOTE: simulate GENDYN301 distinction between time and amplitude random numbers!
    if (kind) { // amplitude distribution: use LEHMER random numbers!
        while (!(z = lehmer.draw()));
    }
    else {
        while (!(z = rnd.draw())); // avoid zero value!
    }
    return float(distribute(z)); // compute one function value
}

```

```

inline double Distr301::expon (float z, float a) {
    return (-(log(1 - z)) / a);
}

```

```

inline double Distr301::trngl (float z, float a) {
    return (a * (1 - sqrt(1 - z)));
}

```

```

inline double Distr301::arcsin (float z, float a) { // arcus sine distribution

```

```

//NOTE: this function is not symmetric:
return (a * (.5 - .5 * sin((.5 - z) * PI)));
}

inline double Distr301::cauchy (float z, float a) { // cauchy distribution
return (a * tan((z - .5) * PI));
}

inline double Distr301::hypcos (float z, float a) { // hypercosine distribution
return (a * log(tan(z * PI / 2.0)));
}

inline double Distr301::logist (float z, float a, float b) { // logistic
if (!a) a = 0.0000001; // prevent division by zero!
return (-(log((1 - z) / z) + b) / a);
}

inline double Distr301::sinus (float a, float b) {
// NOTE: this function does not depend on a random number but on currTime variable:
// NOTE: call of this function only makes sense when synthesizing pieces:
return (a * sin(GenControl::getCurrTime() * (2 * PI / 44100.0) * b));
}

inline float Distr301::distribute (float prb) {
//*****
/* switch between distribution functions, exclude excessive values      *
//*****
switch (selected) {
case CAUCHY:
value = float(cauchy(prb, coef)); break;
case LOGIST:
//NOTE: cocoeff is parameter "b" (translation on abscissa)
value = float(logist(prb, coef, coCoef)); break;
case HYPCOS:
value = float(hypcos(prb, coef)); break;
case ARCSIN:
value = float(arcsin(prb, coef)); break;
case EXPON:
value = float(expon(prb, coef)); break;
case SINUS:
value = float(sinus(coef, coCoef)); break;
case UNIFORM:
value = 0;
error ("Gendyn.Distr301::distribute: no uniform distribution (enum 0) in Gendy301."); break;
case TRNGL:
value = 0;
error ("Gendyn.Distr301::distribute: no triangular distribution (enum 7) in Gendy301."); break;
default:
value = 0;
error ("\nGendyn.Distr301::distribute: no such distribution function\n");
}
return value;
}

```

```

}

//===== Random Number Generator(s) =====

inline float Lehmer::draw (void) {
//*****
/* draw a random number from interval [0..1[ with equal probability */
//*****
    double fraction; // for common subexpression elimination

    fraction = (x * a + c) / m;
    x = float((fraction - floor(fraction)) * m);
    return (x / m); // [0..1[
}

inline short Lehmer::draw (short lower, short upper) {
//*****
/* draw a random number from interval [lower..upper] with equal probability */
//*****
    x = x * a + c;
    return short((short(x) % (upper - lower + 1)) + lower);
}

void Random::init (unsigned short seed) {
    srand (seed);
    // NOTE: emulate QB random generator by reading its random numbers from file:
    if (randFile){
        fclose(randFile); // reset to start of file
        randFile = 0;
    }
    char *dataDir = 0;
    char path[_MAX_PATH] = "";
    char dir[_MAX_DIR] = "";
    char fname[13] = "";
    char buf[5];
    if (!GenControl::piece) return;
    dataDir = getenv("GENDYNDATA");
    if (dataDir) strcat(strcpy(dir, dataDir), "/"); // else assume cwd
    // build sequence specific random number input file name:
    strcpy(fname, "rand");
    if (GenControl::piece->getCurrSeq() < 10) strcat(fname, "0");
    strcat(fname, _itoa(GenControl::piece->getCurrSeq(), buf, 10));
    if (seed < 10) strcat(fname, "0");
    strcat(fname, _itoa(seed, buf, 10)); // abuse seed argument as track number!
    strcat(fname, ".dat");
    randFile = fopen (strcat(strcpy(path, dir), fname), "rb");
}

inline short Random::draw (short lower, short upper) {
//*****
/* draw a random number from interval [lower..upper] with equal probability */
//*****
//NOTE: def random: [0..MAXINT] -> [0..MAXINT-1]

```

```

    return (lower + short((rand() / float(RAND_MAX)) * (upper - lower) + 0.5)); // scale
}

inline float Random::draw (void) {
//*****
/* draw a random number from interval [0..1[ with equal probability      *
//*****
    float randomNumber;

    if (randFile) {
        if (fread (&randomNumber, 4, 1, randFile)) {
            return randomNumber;
        }
        else {
            init(510);
            return 0.5; // arbitrarily
        }
    }
    // NOTE: force result to be less than 1:
    return (rand() / float(RAND_MAX + 1.0));
}

inline void Sound::generate (short & firstWave, short & sample) {
//*****
/* calculation of current track's sample point which is either a      *
/* vertex of the wave polygon or an interpolation point in between      *
//*****
    if (!(poly && walkX && walkY)) return; // to be sure objects exist
    //===== first wave form of this field =====
    if (firstWave) { // flag set by caller
        firstWave = 0; // reset
        seg = 0; // reset
        timeLeft = 0; // reset
        poly->getYRef(0) = 0; // force zero start
        wavesLeft = 0; // no wave repetitions pending
    }
    //===== linear interpolation =====
    if (timeLeft > 1) { // still interpolation points pending?
        timeLeft--; // timeLeft = [0..poly.vertices[seg].x]
        prec = prec + incr; // interpolate incrementally
        sample = short(prec); // update for next interpolation point
    }
    //===== polygon vertex =====
    else { // lookup end point of interpolation:
        sample = (short) poly->getYRef(seg); // look up vertex ordinate
        if (seg < noOfSegs) seg++; // update segment counter
        //===== beginning of next wave form =====
        else { // end of current wave form
            seg = 1; // reset segment counter
            //NOTE: simulate GENDY301 destruction of 1st time vertex random walk:
            //NOTE: this destruction is always "destroyed" again with the good value!
            //poly->getXRef(1) = poly->getXRef(noOfSegs);
            if (wavesLeft > 0) wavesLeft--; // wave repetition = [0..waveRepeat]
        }
    }
}

```

```

}
//===== variation of the polygon =====
walkY->step (poly->getYRef(seg), seg); // new amplitude
if (!wavesLeft) { // calculate new polygonal wave form
    if (waveRepeat && (seg == noOfSegs)) // new wave form completed
        wavesLeft = waveRepeat; // reset for repetitions to come
    walkX->step (poly->getXRef(seg), seg); // new time interval
}
timeValue = poly->getXRef(seg);
//===== filter time value =====
if (filterTime && walkX->mirrorWalk->isOpen()) timeValue /= float(3);
// => snake-like fluctuation of high partials and drop-outs
//===== set up interpolation counter =====
//NOTE: for compatibility with GENDY301, simulate Basic float to int conversion!
timeLeft = short (timeValue + 0.4999999); // look up vertex abcissa
//===== compute new sample increment =====
if (timeLeft > 1) { // compute gradient:
    incr = (poly->getYRef(seg) - float(sample)) / float(timeLeft);
    prec = sample; // no update if nothing to interpolate
}
}
//===== Filter Sample Value =====
if (filterTrack) trackFilter.pass (sample);
sample *= 327; // inflate to 16 bit resolution => sieve effect (leave ~ 6 bit space)
totalTime++; // increment time counter
}

inline void Track::generateSample (long currTime, short & sample) {
    // cut out sound fields:
    // update counters, if necessary:
    //===== new sound field =====
    // NOTE: for compatibility with GENDY301, end active fields later than inactive ones:
    if (field > noOfFields) { // track over?
        sample = 0; // no contribution anymore
        return;
    }
    if (getField(field)->active?
        currTime > getField(field)->endPoint: // end of current field reached?
        currTime == getField(field)->endPoint) {
        field += 1; // next field to sound
        if (field > noOfFields) { // track over?
            sample = 0; // no contribution anymore
            return;
        }
        if (getField(field)->active) { // init lap / rem counters
            //NOTE: do not reset wave if previous field was active, too:
            if (!(getField(field-short(1))->active)) {
                firstWave = TRUE; // set flag to pass to Sound
                //NOTE: for compatibility with GENDY301, restore even random walk positions:
                if (even) { // last active field ended synthesizing even wave?
                    for (short i = 1; i <= sound->seg; i++) { // restore 1..seg random walk positions
                        sound->getWalkX()->flucts[i] = fluctX[i];
                        sound->getPoly()->getXRef(i) = polyX[i];
                    }
                }
            }
        }
    }
}

```

```

    }
    else {
        for (short i = sound->seg+short(1); i <= sound->noOfSegs; i++) {
            // restore 1..seg random walk positions
            sound->getWalkX()->flucts[i] = fluctX[i];
            sound->getPoly()->getXRef(i) = polyX[i];
        }
    }
    even = TRUE; // new active GENDY301 field always starts even wave
} // NOTE: end of GENDY301 compatibility
timeElapsed = 1; // [1..attackTime]
timeRemain = decayTime; // [decayTime..1]
fadeOutTime = getField(field)->endPoint - decayTime;
}
}
//===== within a sound field =====
if (getField(field)->active) { // field sounding?
    //NOTE: for compatibility with GENDY301, identify even waves:
    if (sound->timeLeft <= 1) { // random walk positions will now be updated
        if (sound->seg == sound->noOfSegs) { // next wave will start now
            even = short(!even); // toggle odd wave number
            if (even) { // save random Walk positions before they are overwritten
                fluctX[1] = sound->getWalkX()->flucts[1];
                polyX[1] = sound->getPoly()->getXRef(1);
            }
        }
        else {
            if (even) { // save random Walk positions before they are overwritten
                fluctX[sound->seg+short(1)] = sound->getWalkX()->flucts[sound->seg+short(1)];
                polyX[sound->seg+short(1)] = sound->getPoly()->getXRef(sound->seg+short(1));
            }
        }
    } // NOTE: end of GENDY301 compatibility
    getSound()->generate (firstWave, sample); // firstWave is cleared by Sound
    //===== attenuation of attack and decay =====
    if (timeElapsed < attackTime) { // avoid clicking of attack
        sample = (sample / attackTime) * timeElapsed++; // integer arithmetics!
    }
    if (currTime > fadeOutTime) { // prevent underflow!
        sample = (sample / decayTime) * --timeRemain; // integer arithmetics!
    }
}
else {
    sample = 0; // no contribution
}
}

```

```

inline void Sequence::generateSample (short & master, short & endOfSeq) {
//*****
/* according to the architecture of the pre-calculated score, the *
/* calculation of each successive sound sample is triggered. *
/* For each sequence, the ordinate value of each current sample is given by *

```

```

/* summing up the contribution of all active tracks. The tracks are a      *
/* succession of sound fields. The fields start at sample number          *
/* "Fields(Seq, Track, Field(Track)-1)" and end at sample number        *
/* "Fields(Seq, Track, Field(Track))".                                   *
/******
static short track;                // iteration over tracks
static short sample;              // current amplitude value of current track

master = 0;                        // reset accumulator
//===== round trip through tracks =====
for (track = 1; track <= noOfTracks; track++) {
    if (activeTracks[track]) {
        getTrack(track)->generateSample (currTime, sample);
        if (sample) master += (sample / volReductionFactor);
    }
}
// NOTE: for compatibility with GENDY301, incr counter AFTER lookup of track fields:
currTime++;                        // increment local time counter
if (currTime >= duration) {
    endOfSeq = TRUE;                // tell Piece to increment in-time sequence counter
    reset(); // for next time to come
}
}

//NOTE: for compatibility with GENDY301, read GENDY3 initial random walk positions directly from disk
void Piece::readWalkPositions(void) {
    // build file names
    char path[_MAX_PATH] = "";
    char seqname[3] = "";
    char trackname[3] = "";
    char buf[5];

    if (seq > 1) { // do not for 1st sequence
        if (seq < 10) strcpy(seqname, "0");
        strcat(seqname, _itoa(seq, buf, 10));
        // connect only least common numbers of tracks:
        for (short i = 1; i <= getSeq(seqID)->getNoOfTracks(); i++)
        {
            // read GENDY3 initial random walk positions for this sequence directly from disk:
            if (i < 10) strcpy(trackname, "0");
            else strcpy(trackname, "");
            strcat(trackname, _itoa(i, buf, 10));
            flucFile = fopen (strcat(strcat(strcat(strcat(strcpy(path, dir), flucname), seqname), trackname), ".dat"), "rb");
            walkFile = fopen (strcat(strcat(strcat(strcat(strcpy(path, dir), walkname), seqname), trackname), ".dat"), "rb");
            if (flucFile) {
                if (walkFile) {
                    for (short j = 1; j <= getSeq(seqID)->getTrack(i)->getSound()->noOfSegs; j++) {
                        // NOTE: read random walk positions into secondary tables
                        // (these are copied into main tables on first active field!)
                        // (see Track::generateSample add-on emulation of GENDY301 algorithmic behavior)
                        if (!fread (&(getSeq(seqID)->getTrack(i)->*getSound()->getPoly()->getXRef(j)*polyX[j]), 4, 1, walkFile))
                            error ("Gendyn.Piece::generateSample: cannot read initial random walk positions");
                        if (!fread(&(getSeq(seqID)->getTrack(i)->*getSound()->getWalkX()->flucts[j]*fluctsX[j]), 4, 1, flucFile))

```

```

        error ("Gendyn.Piece::generateSample: cannot read initial random walk positions");
    }
    fclose(walkFile); // reset
}
fclose(flucFile); // reset
}
getSeq(seqID)->getTrack(i)->even = FALSE;
}
}
}
//NOTE: end of compatibility with GENDY301

inline void Piece::generateSample (short & master) {
    if (totalTime < duration) {
        if (seq <= lastSeq) { // iteration through sequences
            if (endOfSeq) { // new sequence starts now
                if (inTime) seqID = inTime->map(seq); // get sequence identifier
                else {
                    error("Gendyn.Piece::generateSample: cannot map sequence numbers to sequence IDs");
                    exit(1);
                }
                endOfSeq = FALSE;
                //NOTE: for compatibility with GENDY301, read GENDY3 initial random walk positions directly from disk
                readWalkPositions();
                //NOTE: end of compatibility with GENDY301
                initRandom();
            }
            // endOfSeq is set by sequence object:
            getSeq(seqID)->generateSample (master, endOfSeq);
            if (endOfSeq) {
                seq++; // next sequence
            }
            if (filterMaster) masterFilter.pass(master);
            totalTime++; // increment time counter
        }
    }
}

void Piece::initRandom (void) {
    rnd.init(seed);
    // NOTE: for compatibility with GENDY301, init independent random generators for each track:
    for (short i = 1; i <= getSeq(seqID)->getNoOfTracks(); i++) {
        // have track specific random generator open appropriate random number sequence file:
        ((Distr301 *) (getSeq(seqID)->getTrack(i)->getSound()->getWalkX()->distr))->rnd.init(i);
    }
    //NOTE: end of compatibility with GENDY301
}

void Piece::reset (void) {
    short i;

    totalTime = 0; // reset time count
    for (i = 1; i <= noOfSeqs; i++) {

```

```

    getSeq(i)->reset(); // reset all sequences' counters
}
seq = firstSeq;
if (inTime) {
    duration = 0; // reset to recount
    for (i = firstSeq; i <= lastSeq; i++) {
        duration += getSeq(inTime->map(i))->getDuration();
    }
    seqID = inTime->map(seq);
}
endOfSeq = FALSE;
//NOTE: for compatibility with GENDY301, read GENDY3 initial random walk positions directly from disk
readWalkPositions();
//NOTE: end of compatibility with GENDY301
initRandom();
}

// NOTE: GenControl::"generate" is idle loop function, declared as "static" and defined here:
// NOTE: this function can not be declared as "inline"!
void GenControl::generate (void) {

    // BUGFIX 10.09.06: sound generation may be one buffer fill ahead, but not more, otherwise it would overwrite buffer content
    long position = (suspended || reset)?0:output.getPosition();
    long generated = preview ? sound->getTotalTime():piece->getTotalTime();
    if ((position != 0)&&((generated-44100+noOfSamples)>position)) return;
    // NOTE 44100 = buffer length in samples
    if (preview) generateSound();
    else generatePiece();
}

inline void GenControl::generateSound (void) {
    static short sample;
    static short firstWave = FALSE;

    if (!sound) return;
    for (long i = 1; i <= noOfSamples; i++) { // count idle loop
        if (suspended) { // pause button pressed
            if (reset) { //restart button pressed
                reset = 0; // reset restart flag
                // add restart code here (e.g. reinitializing of polygon)
                sound->initSound();
            }
            break;
        }
        // repeat trying to write same value, if last write didn't succeed:
        sound->generate (firstWave, sample); // create new sample value
        blocked = output.write (sample);
    }
}

inline void GenControl::generatePiece (void) {
    static short master;

```

```

if (!piece) return;
for (long i = 1; i <= noOfSamples; i++) {
    if (suspended) { // pause button pressed
        if (reset) { //restart button pressed
            reset = 0; // reset restart flag
            // add restart code here (e.g. resetting of time count)
            piece->reset();
        }
        break;
    }
    // repeat trying to write same value, if last write didn't succeed:
    piece->generateSample (master); // create new sample value
    blocked = output.write (master);
}
}

void GenControl::quit(void) {
    exit (0); // quit
}

short Control::initOutput (void) {
    short check1 = FALSE; // still OK
    short check2 = FALSE; // still OK

    // try to open sound device in any case:
    if (record && (!soundFileOpen)) { // don't reopen if not told explicitly by "reset"
        check1 = output.initFile(soundFileName, samplingRate);
        if (check1) {
            error ("Gendyn.Control::initOutput: record to sound file not ready");
            soundFileOpen = FALSE;
        }
        else {
            soundFileOpen = TRUE;
        }
    }
    check2 = output.init(samplingRate, bufferTime, seconds);
    if (check2) { // OK?
        error ("Gendyn.Control::initOutput: real time sound output not ready");
    }
    return short(check1 && check2); // no output at all
}

void Control::close (void) {
    output.close();
}

void Control::closeSoundFile (void) {
    if (soundFileOpen) {
        output.closeFile(samplingRate);
        soundFileOpen = FALSE;
    }
}

```

```

void Control::pause (void) {
    output.pause();
}

void Control::stop (void) {
    output.stop();
}

void Control::restart (void) {
    if (preview) sound->initSound();
    output.restart();
}

////////////////////////////////////
// NOTE: the following functions are thought to be overridden in derived control
// classes to create control objects as specialized heirs of synthesis objects.
////////////////////////////////////

void Control::init (void) {
    if(!(piece = new Piece)) {
        error (strerror(errno));
        exit (1);
    }
    piece->init();
}

void Piece::init (void) {
    if (!(inTime = new Mapping)) {
        error (strerror(errno));
        exit (1);
    }
    initSeq(0);
}

void Piece::initSeq(short seqID) {
    if ((seqID >= 0) && (seqID <= noOfSeqs)) {
        if (!(seqs[seqID] = new Sequence)) {
            error (strerror(errno));
            exit (1);
        }
        seqs[seqID]->initTrack(0);
    }
    else {
        error ("Gendyn.Piece::initSeq: illegal sequence ID");
        exit (1);
    }
}

void Sequence::initTrack(short track) {
    if ((track >= 0) && (track <= noOfTracks)) {
        if (!(tracks[track] = new Track)) {
            error (strerror(errno));
            exit (1);
        }
    }
}

```

```

    }
    tracks[track]->initField(0);
}
else {
    error ("Gendyn.Sequence::initTrack: illegal track number");
    exit (1);
}
}

void Track::init (void) {
    if (!(sound = new Sound)) {
        error (strerror(errno));
        exit(1);
    }
    sound->init();
}

void Track::initField(short field) {
    if ((field >= 0) && (field <= noOfFields)) {
        if (!(fields[field] = new Field(field))) {
            error (strerror(errno));
            exit (1);
        }
    }
    else {
        error ("Gendyn.Track::initField: illegal field number");
        exit (1);
    }
}

void Sound::init (void) {
    walkX = new RandomWalk; // create new object
    if (!walkX) {
        error (strerror(errno));
        exit(1);
    }
    walkX->init();
    walkY = new RandomWalk; // create new object
    if (!walkY) {
        error (strerror(errno));
        exit(1);
    }
    walkY->init();
    poly = new Poly; // create new object
    if (!poly) {
        error (strerror(errno));
        exit(1);
    }
}

void RandomWalk::init(void) {
    mirrorFluct = new Mirrors; // create mirror object
    if (!mirrorFluct) {

```

```

        error (strerror(errno));
        exit(1);
    }
    mirrorWalk = new Mirrors; // create new mirror object
    if (!mirrorWalk) {
        error (strerror(errno));
        exit(1);
    }
    if (!(distr = new Distr301)) { // create new distribution object
        error (strerror(errno));
        exit(1);
    }
}

//***** end of reading parameterized synthesis objects *****

short Output::init (unsigned long samplingRate, unsigned short bufferTime, unsigned long seconds) {
    short answer;

    // pass no of channels, quantization, buffersize and loop estimation:
    if (bufferTime == 0) bufferTime = 1;
    answer = waveOut.init (samplingRate, 1, 16,
        bufferTime * samplingRate,
        seconds / (unsigned long) bufferTime);
    if (answer) { // device not ready
        soundDeviceOpen = FALSE;
    }
    else {
        soundDeviceOpen = TRUE;
    }
    return answer;
}

short Output::initFile (char *soundFileName, long samplingRate) {
    char path[_MAX_PATH] = "";
    char *soundDir;

    //===== open sound file =====
    soundDir = getenv("GENDYNSOUND");
    if (soundDir) strcat(strcpy(path, soundDir), "/"); // else assume cwd
    if (Control::writeHeader) {
        strcat(strcat(path, soundFileName), ".WAV");
    }
    else {
        strcat(path, soundFileName);
    }
    soundFile = fopen (path, "wb");

    if (!(soundFile)) {
        error (strerror(errno));
        soundFileOpen = FALSE;
        return -1;
    }
}

```

```

//===== write initial header =====
recorded = 0; //reset
writeHeader(recorded, samplingRate);
soundFileOpen = TRUE;
return 0;
}

void Output::closeFile (long samplingRate) {
    if (soundFileOpen && soundFile) {
        // write final header
        rewind(soundFile);
        writeHeader(recorded, samplingRate);
        fclose (soundFile);
        soundFile = 0; // reset
        soundFileOpen = FALSE;
    }
}

void Output::writeHeader(unsigned long duration, long samplingRate) {
    WaveHeader header (duration, samplingRate);
    if (GenControl::writeHeader) header.write (soundFile);
}

inline short Output::write (short sample)
{
    short repeat; // repeat writing same sample value, if output device blocked

    if (soundDeviceOpen) repeat = waveOut.write (sample);
    else repeat = 0;

    if (GenControl::record && soundFileOpen && !repeat) // write to sound file, too
    {
        fwrite (&sample, 2, 1, soundFile); //(buffer, 2, buffersize, soundFile);
        recorded++;
    }
    return repeat; // report to caller
}

long Output::getPosition ()
{
    return waveOut.getPosition();
}

WaveHeader::WaveHeader (unsigned long totalTime, long samplingRate)
{ // construct sound file format header

    strcpy(RIFF, "RIFF");
    strcpy(WAVEfmt, "WAVEfmt ");
    formatTag = 16; // MS PCM file
    number3 = 0; // ...
    noOfChannels = 1; // mono
    number5 = 1; // ...
    samplesPerSec = samplingRate;
}

```

```
bitsPerSample = 16; // maximal integer quantization
bytesPerSec = samplesPerSec * (bitsPerSample / 8);
blockAlign = 2; // i.e. 2 bytes per sample (?)
strcpy(DATA, "data");
dataBlockLength = totalTime * noOfChannels * (bitsPerSample / 8);
fileLength = dataBlockLength + 36; // add wave header length
}

void WaveHeader::write (FILE *soundFile) {
    fwrite(this, sizeof(WaveHeader), 1, soundFile);
}

//=====
//===== End of Class Member Functions =====
//=====
```

# The New GenDyn Program: A Tutorial

Peter Hoffmann, CEMAMu, CNET E 655, 3, av.de la République, 92131 Issy-les-Moulineaux.

## 1. Introduction

The new GenDyn program implements the Dynamic Stochastic Synthesis algorithm by Iannis Xenakis as documented in Formalized Music [Xenakis 1991]. Sound synthesis is controlled by a graphical user interface. It helps editing the synthesis parameters while listening to the sound output. Sound may also be recorded to a sound data file for later playback and sound edit procedures with standard soft-or hardware. The new GenDyn program is a prototype implementation ready to be adapted to the musical needs of a composer. The current version runs on a PC under Windows 3.1. It needs a fast processor and a sound card for real time sound output.

## 2. Getting Started

Start Windows. With the aid of the Windows program manager, start the GenDyn program. The easiest way is to double-click the GenDyn program icon (the Greek letter "Xi"). After a little while, you will see an empty frame with a title and a row of menu commands as shown in Figure 1.



Figure 1: The startup menu of the new GenDyn program

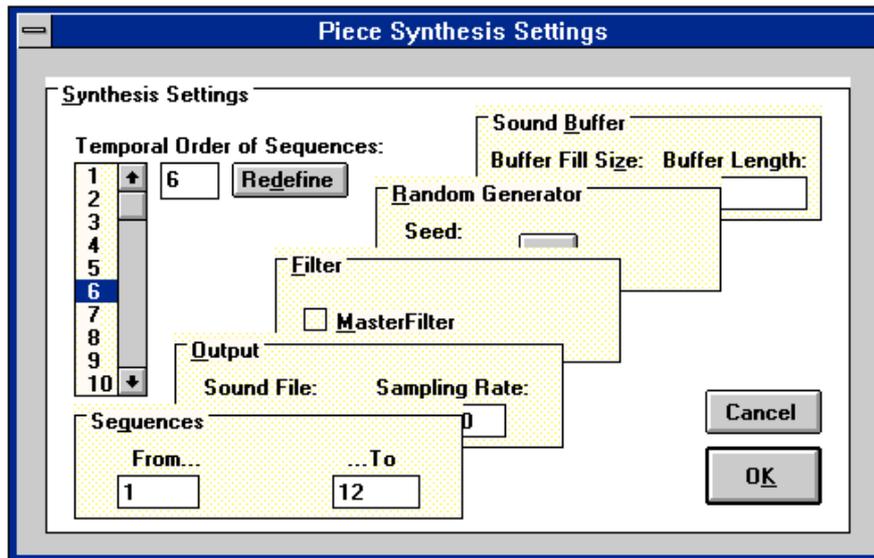
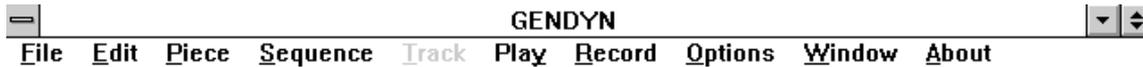
With the help of the menu commands you can define, play and record a GenDyn musical piece. Let's start top down and play a sample piece called "SAMPLE" to see how it works.

- Click with the mouse on the command "Play". (Press and release the left mouse button without moving the mouse.) A menu will pull down. The sub-command "Play Piece" is already selected, so press "Enter" (the return key). A window will open and show the current synthesis parameter settings concerning the piece as a whole.

### 2.1. Viewing and Editing the Piece Settings

The settings concerning the piece as a whole are (see Figure 2):

1. The first and the last sequence of the piece to be synthesized.
2. The list of the temporal order for these sequences (the order in which they follow each other).
3. The name of the sound file and its sampling rate.
4. The activation of a global low-pass filter (optional).
5. The initialization of the random generator (optional).
6. The size and the filling-up rate of the real time sound output buffer.



**Figure 2: GenDyn parameter settings for the generation of a musical piece**

- Click with the mouse on the numbers in the frame named "Sequences" and change them by typing numbers on your keyboard. The temporal list to the top of it will adapt to the new settings.
- Click with the mouse on a number in the list named "Temporal order of sequences". It will be highlighted. Click with the mouse in the little window to the right and type a new number. Click on the button named "Redefine". The number in the list will change.
- You will rarely need to change the other settings except in some special situations. Close the window by clicking on the button reading "OK". If you have entered "illegal" numbers, you will be prompted to correct them first.

If all goes well, a window will open and show you the first sequence to be synthesized (see Figure 3). If you have a sound card properly installed, you will hear the sound of that sequence as soon as the sound buffer is filled up. Note that the playback of the sound is split into chunks of the sound buffer's size (default: one second). These chunks only fit to a continuous sound if your computer is fast enough to compute them ahead in time. Otherwise you will hear it loop over the buffered sound while adding bits of new sound at its end.

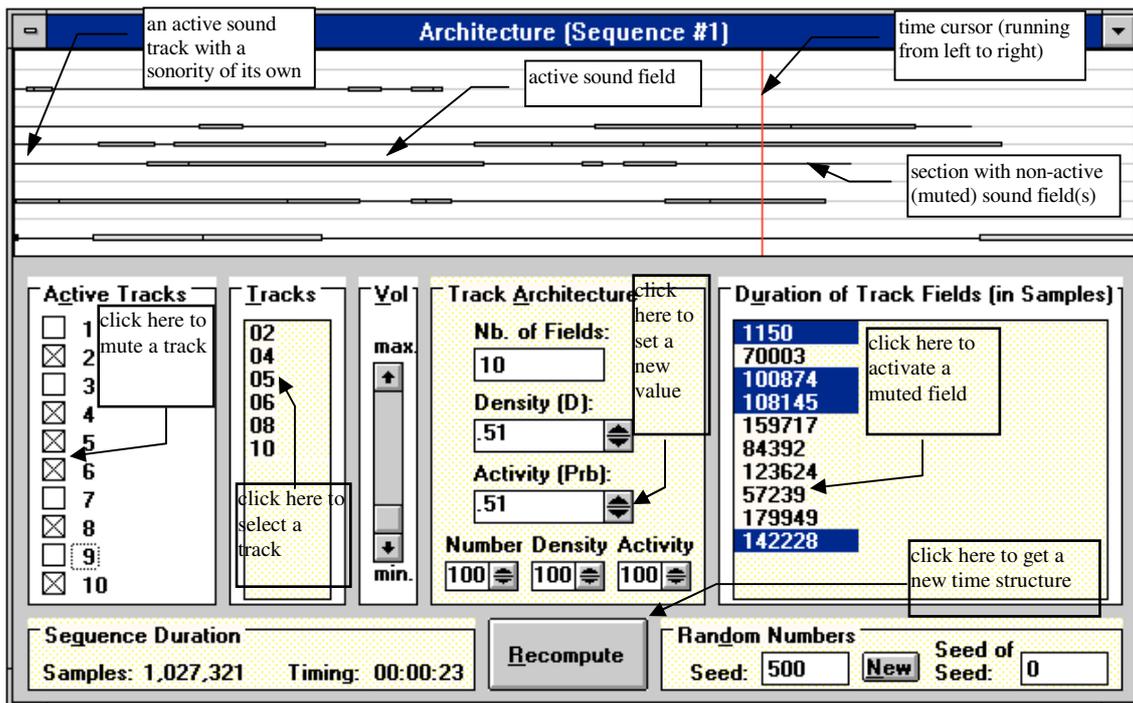
## 2.2. Viewing and Editing Sequence Settings

A GenDyn piece consists of a succession of sequences. They control a multitude of sounds playing simultaneously and in succession according to a precomputed stochastic sound architecture. You currently have at least one of these graphically displayed on your screen. You can change its time structure by some mouse-clicks and/or key strokes.

- Click with the mouse on the command button called "Recompute" at the bottom of the sequence window. After a while, the new architecture will be displayed. Repeat until you like its structure. You will note that the probability fluctuations make it look quite different each time.
- You can activate and de-activate individual sound tracks of the sequence architecture. Click on the check boxes to the extreme left of the window. If, for example, you un-check the box numbered "1",

the time structure of track number one disappears and does not sound during synthesis. If you check it it will come into play again.

- To the right of the check boxes, the numbers of the active sound tracks are listed. (For the sake of numerical ordering, numbers less than 10 are given a leading zero.) If you click on one of these numbers, the parameter settings for that specific track are displayed on the right to it: the number of sound fields along with their density and activity. There are little "up" and "down" arrows to the right of the numbers with a well-defined value range. Keep the mouse key pressed on them until they reach the right value. The number of fields is to be chosen freely, click with the mouse on it and use the keyboard to edit that number. Then click on "Recompute". If you want to change the parameters of all sequence tracks in the same way, you better set the global percentage factors below these values. When you then click on "Recompute", the parameter values of all tracks will be multiplied with these factors.
- On the extreme right of the window, all field durations of a track are listed. Highlighted durations are sounding, the others not. If you keep the "Control" key of your keyboard pressed down, you can change the sound activity of the durations by clicking on them. (This key is on the extreme lower left on most PC keyboards.)
- You can set new seeds for the random generator before recomputation either by typing them to the right of the "Recompute" button or by clicking on the "New" button to the right of the seed number to have it chosen at random. You can set a seed for that, too.



**Figure 3: Setting the parameters for the stochastic computation of a sequence architecture**

To resume, the settings for the stochastic generation of a sequence architecture are (see Figure 3):

1. Activation or de-activation of individual tracks.
2. The number of fields for each track.
3. The density of the fields along the track (high density results in short field durations).
4. The percentage of sounding fields per track.

You can always choose "Play Piece" in the "Play" menu to listen to the edited time structure. You will note that playback is not very loud. You can boost the volume with the aid of the "Vol" slider.

- Click with the mouse on the upper arrow of that slider. Every click increases the master volume of that sequence. But beware: too much volume will result in sharp sound "clickings". The minimal default setting of the volume slider is low, but shure. In general, you have to listen to a whole sequence in order to find out the maximal setting without sound clicks.

You can always view other sequences by clicking the menu command "Sequences". It will show you a list of sequence numbers. Click on the number of the sequence you wish to be displayed.

### 2.3. Viewing and Editing Sound Track Settings

Each sequence in a GenDyn piece is a superposition of sound tracks with a sound evolution of their own. This evolution is controlled by a set of parameters that can be edited in real time during synthesis with the aid of a track window (see Figure 4).

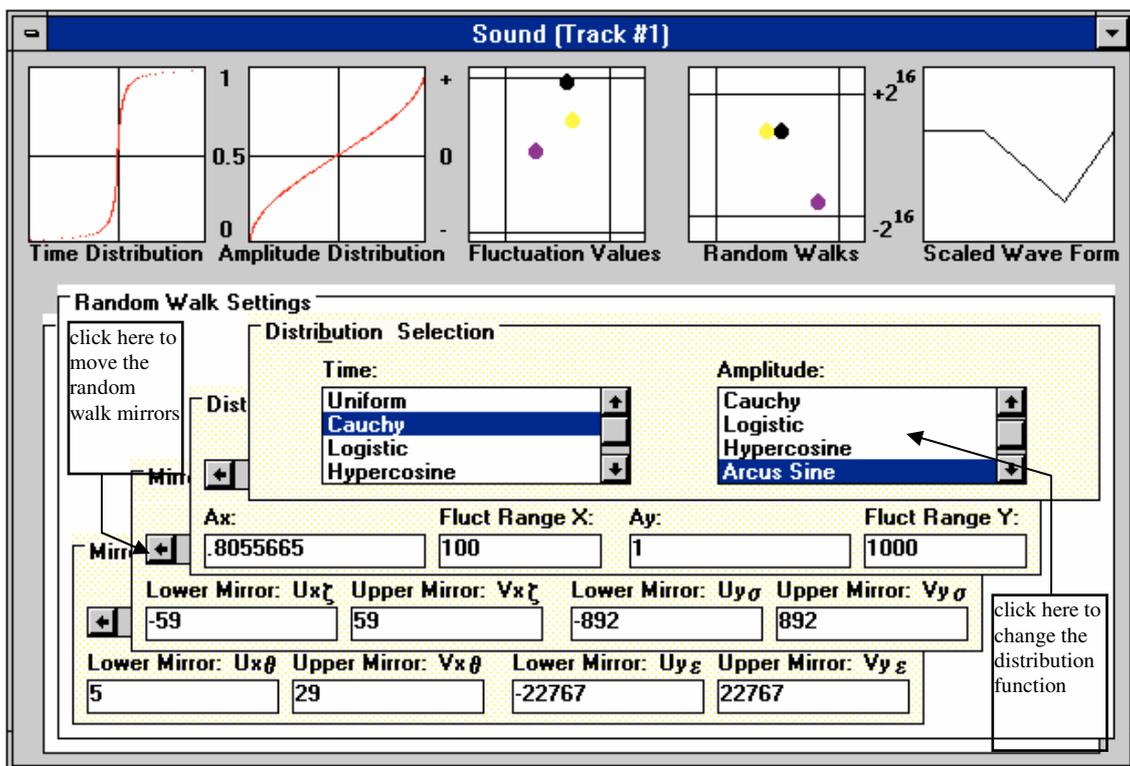
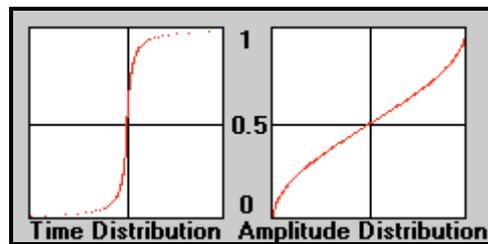


Figure 4: Setting the parameters for the stochastic computation of a GenDyn sound

- Click with the mouse on the menu "Tracks" and choose a number in its track number list. A track window will open. It takes a while for the probability distributions to be graphed in the two little boxes at the top of the window. The three other boxes are empty for the moment.
- Start synthesis of this particular sound by clicking on the menu "Play" and then "Play Sound". If all goes well, you will hear the sound controlled by the current sound parameter settings.
- There are two sets of parameters that you can bring to the foreground alternatively. One concerns the sound waves as such (e.g. the number of wave segments), the other the random walks acting on it (e.g. mirror settings). Click with the mouse on the frame called "Random Walk Settings". It will come to the foreground (as in Figure 4).

- In the frame at the bottom called "Mirrors Random Walk", there are four boxes with numbers. The two numbers to the left give the lower and upper mirror settings for the time random walk. The two numbers to the right give the lower and upper mirror settings for the amplitude random walk. You have two sliders for the time mirrors but only one slider for the amplitude mirrors, because the amplitude space is symmetric to its origin (i.e. amplitude value zero). Note that the least value for the lower time mirror is zero as there is no negative time (at least not in the GenDyn context).
- Click with the mouse on the arrows of the sliders. The mirror positions will change, as can be seen in the forth box on top of the window. Note that the display is scaled to fit to all possible settings, so the movement of the mirror lines is not absolute. The amplitude mirrors change the volume of the sound while the lower time mirror controls the high partials and the upper time mirror controls the minimum frequency of the sound. If both time mirror have the same value, there is no time fluctuation and a stable "tone" will be heard.
- You are now ready to set the other random walk settings with the aid of the sliders in the other sub-frames by clicking on them. Time settings are always on the left and amplitude settings on the right of the frames. Changing the fluctuation mirror settings will move the mirror lines in the third box on top of the window. Changing the distribution settings will recompute and display the distributions in the first two boxes (see Figure 5).



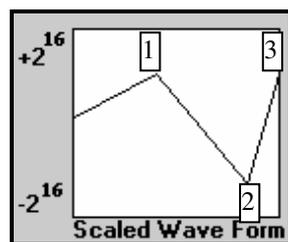
**Figure 5: Weak (left) and strong (right) variance of fluctuation values**

Note that on a slow computer, manipulating parameter settings takes computation time away from the synthesis, resulting in sound discontinuities. This is particularly critical in the case of the distribution settings. You might want to enlarge the sound buffer to give the synthesis more time to compute ahead by choosing "Play Piece" in the "Play" menu and increasing the "Buffer Size" (see Figure 2).

#### 2.4. Animating the Synthesis

You can visualize the sound synthesis in the three rightmost boxes on top of the track window.

- Click with the mouse on the frame "Sound Wave Settings". Then click on the topmost sub-frame called "Real Time Preview". Then click on the command button called "Animate". The pictures will only move if sound synthesis is "on".
- The wave form of the current sound is displayed in the rightmost box on top of the track window (see Figure 6). By default, it is updated once per second. You can increase the update rate with the help of the slider to the right of the "Animate" button.

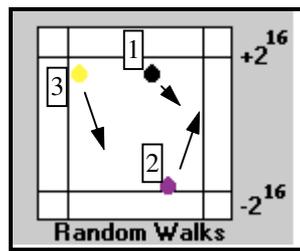


**Figure 6: A simple wave polygon with 3 vertices, forming 3 segments**

Note that fast update rates slow down synthesis. The effect is visually convincing (film-like) but acoustically boring (loopings). In general, a synthesis you can follow wave by wave with your eyes is too slow to be conceived by your ears while real-time synthesis is too fast for the eyes to follow.

Note also that the wave form displayed is scaled, i.e. compressed or dilated to fit into the box. In reality the absolute wave length "breathes" according to the time fluctuations of its segments, but you can only see the change of its segments' length in relation to each other, i.e. its shape. If you want to see the wave forms resulting of the superposition of several tracks in a sequence (the "master" sound), you better record that sequence (see below) and view the sound "outside time" with the help of a standard sound editor.

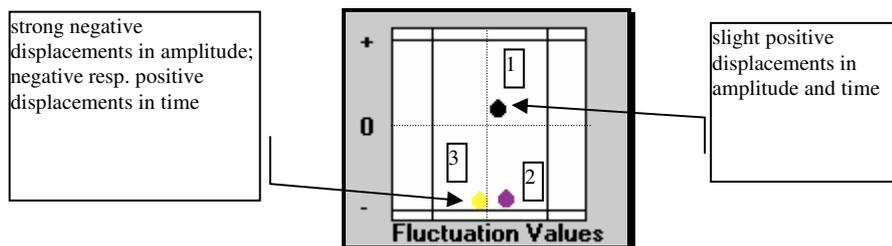
- The random walks of each wave segment are displayed in the box left to the sound wave box (see Figure 7). The segments are visualized as "billiard balls" moving left and right (time random walk) and up and down (amplitude random walk).



**Figure 7: The movement of the polygone vertices (balls) in a random walk space**

While the height of each "billiard ball" can be directly matched to the vertical position of a wave form vertex, its horizontal position does not directly correspond to the horizontal position of the vertex in the polygone. For example, "ball" number one is right of "ball" number three in time random walk space, but its corresponding vertex is left of "ball" one's vertex in the wave form (compare Figure 7 to Figure 6). Both share the same amplitude but are discerned by their time increment to the preceeding vertex (i.e. segment width) in the time (horizontal) dimension.

- The steps in the random walks of each segment are visualized in the box left to the random walk box (the third of the boxes). They correspond to the fluctuation values drawn from the time and amplitude distribution functions. Changing the steepness (the coefficient) of these distributions directly influences the velocity of the fluctuation "balls" in fluctuation space as well as the "diffusion" of the random walk balls in random walk space. The correspondence of the "balls" is again shown by numbering in Figure 8 (compare to Figure 7).



**Figure 8: Random fluctuations of the polygone vertices in time and amplitude**

Note that the correspondence between fluctuation, random walk and wave form display is as better as update is more frequent and sound computation is slowed down. If a sound wave has many segments, the animation becomes very time-consuming and synthesis very slow, resulting in nearly identical loops over the sound buffer. This makes itself less and less heard, however, since the independent movement of many segments make for a very complex sound with fluctuations averaging over a longer time scale, and discontinuities between the buffer sound chunks are hard to perceive.

### 3. Creating your Own Pieces

#### 3.1. The "File" Menu

Now you are ready to define your own GenDyn pieces. You might want to start from scratch or use existing sound tracks or sequences you have already tailored to your needs.

##### 3.1.1. The "Save As..." Command

- Click on the "File" menu command. A menu pulls down containing "Load" and "Save" commands. If you want to continue with your current settings, choose "Save As...". You will be prompted to save them under a new name (just alter the highlighted old name in "XYZ" and click on "OK"). From now on, you work out your own piece called "XYZ".

##### 3.1.2. The "Save" Command

The GenDyn program saves your settings automatically as soon as you close a sequence or track window (see below). If during work you have found interesting settings and want to make sure that they are saved before you continue working, choose "Save" in the "File" Menu.

##### 3.1.3. The "Open" Command

Choose "Open" in the "File" menu to open saved settings. You will be prompted for the name of an existing parameter file. The easiest way is to click on a greyed file name in the list and then click on "OK".

##### 3.1.4. The "New" Command

This clears your workspace if you want to start from scratch. After this command, you are left with one sequence containing one track. You then can add new tracks and new sequences (see below) and set their parameters.

The last command in the "File" menu is to quit the program. You will be prompted to confirm.

#### 3.2. Adding New Sequences and Tracks

If you want to increase the number of sequences in your piece, choose the "New" command in the "Sequences" menu. If you want to increase the number of sound tracks in a given sequence, choose the "New" command in the "Tracks" menu. Note that the new track has zero length and will not sound in the sequence before you set its "number of fields", "field density" and "field activity" parameters and recompute the sequence architecture to include the new track (see section 2.2. Viewing and Editing Sequence Settings).

#### 3.3. Deleting Sequences and Tracks

You will rarely need to delete a sequence or a track, if you do not like them, since you can change all of their settings at will. If you do not want them to be included in your piece, you can mute tracks in the sequence window on the one hand and exclude sequences from the "Temporal order" list on the other hand. (This list is displayed when you choose "Play Piece" in the "Play" menu, see 2.1. Viewing and Editing the Piece Settings). You might want to get rid of unused sequences and tracks in order to minimize the parameter file and its loading and saving time. Choose "Delete" in the "Sequences" respective "Tracks" menu. Note that "Delete" deletes the currently selected sequence or track (the sequence or track window you last used). In the status bar on the bottom of the screen, to the right edge, the current sequence number and track number are displayed. If there is no current sequence or track selected, these numbers are cleared and a message will prompt you to select a sequence or track number in the "Sequences" or "Track" menu.

## 4. Recording the Music

### 4.1. The "Record" Menu

You can record everything you hear into a sound file for later playback with a standard sound editor. Note that unlike the real time sound, recorded sound is continuous, regardless of how slow it is computed by your computer.

- Click with the mouse on the menu command "Record" and then choose "On". This sequentially writes anything you synthesize from now on into a sound file named after your current parameter file.

You can work incrementally on a piece by playing sequence after sequence. If you want to listen to sequences before adding them to the sound file, you have of course to switch off recording by choosing "Off" in the "Record" menu. If you want to continue recording, choose "On" again. Your recording will resume where it had been interrupted.

- If you want to close a record session and start a new one, choose "Off" and then "Rewind" in the "Record" menu. After confirmation, this deletes what you have recorded so far. (It is a bit like rewinding a magnetic tape, except that it takes virtually no time.)
- If you want to keep what you have recorded, you have to change the parameter file name with the aid of the "SaveAs..." command in the "File" menu before opening a new session. This is a limitation that guarantees a one-to-one correspondence between parameter file and sound file.

Note that you cannot listen to and edit (e.g. undo) a piece of recording from within the GenDyn program. You can of course do it with a standard sound editor. Depending on your system, you might be forced to close your record session before by choosing "Rewind".

## 5. Managing Sequence and Track Windows

With the help of the "Sequences" and "Tracks" Menus, you can always close the current sequence or track window with the "Close" command. This will not delete the sequence or track; you can always redisplay it by selecting its number in the "Sequences" or "Tracks" menu. If you have lost overview, a good idea could be to minimize (iconify) all sequence and/or track windows by choosing the command "Minimize All". "Restore All" will undo that.

### 5.1. The "Windows" Menu

You can select windows by clicking on the "Windows" menu command. This pulls down a list with all windows currently displayed. Click on the entry of the window you wish to bring to the foreground. There are also three convenient commands to re-organize your workspace:

#### 5.1.1. The "Tile" Command

The effect of this command is shown in Figure 9: The workspace is "tiled" with windows in order to give maximal overview. The ordering of the windows depends on what window was selected before choosing that command. The last selected window will appear first when you choose the "Tile" command.

#### 5.1.2. The "Cascade" Command

This shows you the windows like playing cards in a way that you can read their titles.

#### 5.1.3. The "Arrange Icons" Command

This puts your icons neatly back in rows, if you have them moved around (e.g. to reorder them on your screen).

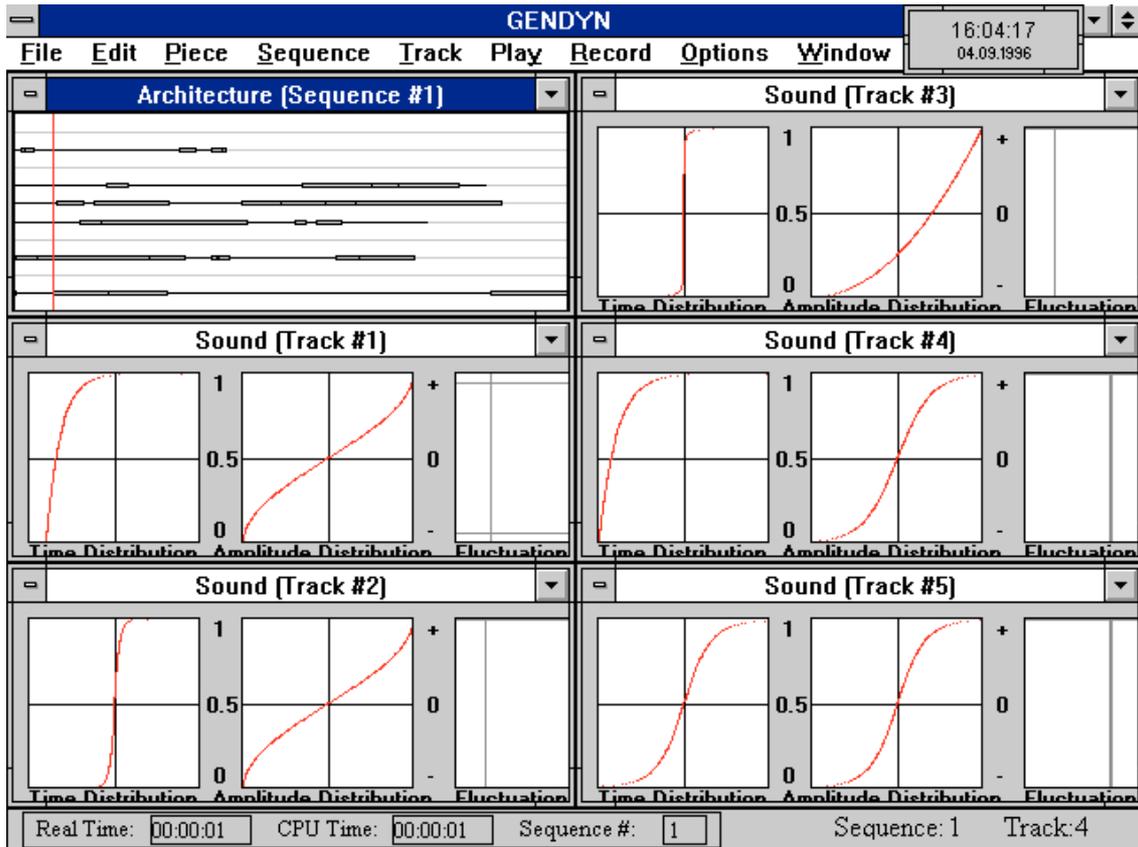


Figure 9: The GenDyn interface with the first sequence and five of its tracks loaded

## 6. Facilitating Your Work

### 6.1. The "Edit" Menu

There is a way to copy parameter values between editable text boxes. (Editable text boxes are those that display a vertical cursor when you click on them.) Highlight the value to be copied by double-clicking on it. Then choose "Copy" in the "Edit" menu. Now you can click on the text box you want to copy to. Choose "Paste" in the "Edit" menu.

### 6.2. Keyboard Shortcuts

Many things you can do with the mouse you can do with your keyboard, too. While working with the mouse is more intuitive, pressing the keyboard can accelerate and facilitate actions that you have to do very often. It can also be a good idea to trigger an action with the keyboard keys in order to leave the mouse pointer in its current position to return to it later.

#### 6.2.1. The "Tab Order"

Text and graphical controls (like check boxes, command buttons, etc.) inside a window are enumerated in a "Tab Order". That means that you can cycle through these controls by repeatedly pressing the "Tab" (or "Tabulation") key on your keyboard. (The Tab key is on most PC keyboards the key on the left with the two arrows on it.) It is as if you click your mouse on each of these controls. The Tab key can save you time and effort grabbing and positioning the mouse any time you want to move to another control.

### 6.2.2. Access Keys

Any time you see text with an underscored letter, you can type that letter while keeping the "Alternate" key pressed. (It is right of the "Control" key and left of the space bar on most PC keyboards.) This will have the same effect as clicking the mouse on it. If clicking the mouse on such a text had no immediate effect, the "click" it is propagated to the next clickable control in the "Tab Order". There are two special access keys for confirmation and cancelling.

### 6.2.3. Special Keys

If a command button in a window reads "OK", an equivalent to clicking on it is pressing the "Enter" key (the return key). If it reads "Cancel", an equivalent to clicking on it is pressing the "Esc" ("Escape") key. This key is on the extreme upper left on most PC keyboards. With the arrow keys you can move inside lists and menu entries. The spacebar checks and unchecks on check boxes.

### 6.2.4. Maximizing Pictures

The graphical representations on top of the sequence and track windows can be scaled to fill the whole window. Just double-click your mouse on them. To undo that action, single-click.

### 6.2.5. Viewing Sound Parameters

Clicking and double-clicking on the white background of the "Random Walk Settings" and "Sound Wave Settings" frames will cascade the parameter sub-frames in two different ways: a single-click puts them stored away in a row like cardfiles, a double-click presents them flinged up to show all current settings at a glance.

### 6.2.6. Improving User Response

On a slow computer, where synthesis is below real-time, it will take most of the processor's resources. The response to your action (mouse clicks etc.) is therefore delayed. You might prefer a good user response to intense sound computation. Just claim more system attention by reducing the number of samples to be computed at a time. Choose "Play Piece" in the "Play" menu and change the number in the box called "Buffer Fill" before starting (see Figure 2).

## 7. Epilogue

The new GenDyn program is a prototype implementation of the Dynamic Stochastic Synthesis conceived by Iannis Xenakis. It is made to explore the acoustic and musical power of the pure Xenakis algorithm in a systematic way by means of an interactive, real-time control of the synthesis parameter set. While the implementation of the pure algorithm gives already good results, it might have to be elaborated to fully meet the creative needs of a composer. The design of the implementation has been chosen to be as flexible as possible to allow for an evolutionary process of refinement. It is up to the composer-user to foster the further development of the new GenDyn program by his criticism.

### References

[Xenakis 1991] Iannis Xenakis, *Dynamic Stochastic Synthesis and More Thorough Stochastic Music*, in: *Formalized Music*, Pendragon Press, Stuyvesant, NY, 1991, pp. 289- 321.  
 Peter Hoffmann, *Implementing the Dynamic Stochastic Synthesis*, Troisièmes journées d'informatique musicale JIM 96, 16.-18.05.1996, Ile de Tatihou, France (= Les cahiers du GREYC [Groupe de Recherches en Informatique, Image, Instrumentation de Caen], 4, 1996), pp. 341-347.