Computer Music Instruments

Victor Lazzarini

Computer Music Instruments

Foundations, Design and Development

Springer

Victor Lazzarini
Department of Music
Maynooth University
Maynooth
Ireland

ISBN 978-3-319-63503-3 ISBN 978-3-319-63504-0 (eBook)
https://doi.org/10.1007/978-3-319-63504-0

Library of Congress Control Number: 2017953821

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

This book is dedicated to John ffitch, Mathematician, Computer Scientist and Musician.

Foreword

I met Victor Lazzarini for the first time in May 2004 in Karlsruhe, during the second International Linux Audio Conference. I was there to present the Faust audio programming language and Victor was giving a talk on developing spectral signal processing applications. What struck me at the outset was Victor's many talents. Victor is not only a highly experienced composer and educator, and Professor of Music and Dean of Arts, Celtic Studies and Philosophy at Maynooth University. He is also a leading researcher in the field of sound synthesis and audio programming languages, and a core developer of the Csound programming language.

It took us some time to transform our highly interesting informal discussions during international conferences into a real collaboration. But in June 2013 Victor suggested we could embed the Faust compiler into the future Csound 6 release in order to be able to interleave Faust and Csound programs. We therefore decided to collaborate to develop a set of opcodes to compile, run and control Faust code directly from Csound (a detailed description of these opcodes can be found in Chapter 2). The resulting Csound 6 opcodes were released in 2014.

Interestingly, it turns out that interleaving programming languages are at the heart of *Computer Music Instruments*. Indeed, the first part of the book introduces the foundations, from audio signals to programming environments, of the three particular programming languages (Python, Csound and Faust) that are used throughout the book.

The second part of the book is dedicated to sound synthesis and signal processing methods. The techniques presented range from source-filter models to frequency-domain techniques, via granular methods, as well as feedback and adaptive systems. Many practical examples with code are provided in Python, Csound and Faust.

The last part of the book focuses on communication protocols, interactions, and computer music platforms. Mobile phones, embedded systems, and Web platforms are covered. The text is completed by two appendices. The first provides support to understand the mathematical notions used in the book. The second provides additional code examples.

I can strongly recommend this comprehensive book. The style is clear and easy to follow. The examples are excellent. Readers of different levels of expertise will gain

benefit from it. The great talent of Victor Lazzarini is to provide a book that is interesting from multiple perspectives: computer science, signal processing, software development and, of course, sound art and music!

Lyon, December 2016 *Yann Orlarey*

Preface

If we consider the appearance of Max Mathews' first direct digital synthesis program for the IBM 704 mainframe, MUSIC I, as marking the year zero of computer music, then the field is about to reach its sixtieth anniversary in 2018. Throughout these years, amazing developments in software and hardware technology made the production of (digital) electronic music a common feature of today's music. For the first few decades, access to computers for sound creation was limited to only a few lucky individuals. With the popularisation of microcomputers, a revolution was initiated, allowing a much a wider host of practitioners to avail themselves of these wonderful devices. This has continued today and I expect it will extend well into the future, as new possibilities emerge for music, sound art, sound design, and related activities. The personal computing tools we have at hand today, which have various forms, sizes, and capabilities, are incredibly powerful platforms for the development and manipulation of new sound-making objects. This book is dedicated to the study of these computer instruments from the ground up.

The text is organised into three parts, covering three aspects of the topic: foundations; instrument development from the perspective of signal processing; and the design of applications on existing computer music platforms. The first chapter, in Part I, explores basic concepts relating to audio signals in and out of the computer. It provides a gentle introduction to some key principles that will be used throughout the book, with a touch of mathematics and plenty of illustrative examples. This is followed by an introduction to the programming tools that will be used in the following two parts of the book, Python, Csound, and Faust. These three languages were judiciously chosen to cover a wide variety of applications, which range from demonstrating signal-processing algorithms to implementing full applications. Another important feature of these three environments is that they can be nicely interleaved, from a higher to a lower level, and from an application-development to a signal-processing function. This is demonstrated by a full example at the end of the chapter.

The second part of the book is fully devoted to exploring ways of making sound with a computer, the signal-processing design for instruments. It includes some classic approaches, such as source-filter models, clever ways of manipulating mathe-

matical formulae, newer approaches to feedback and adaptive techniques, and the methods of granular synthesis. Part II is complemented by a thorough examination of the principles of frequency-domain analysis-synthesis principles. Although these chapters tend to make continued use of mathematical expressions, the discussion also employs graphical plots and programming examples to complete the exploration of each technique. This three-pronged approach is aimed at providing a full perspective and a thorough examination of these sound-generation algorithms.

Finally, in the third part, we look at some complementary aspects of instrument design: interaction tools and development platforms. Chapter 8 explores various means of communication with sound synthesis programs, with examples presented in the three target programming environments. We also examine graphical user interfaces and the principles of custom hardware for sound control. The final chapter in the book then discusses the various platforms for the development and performance of computer instruments.

This book is designed for readers of different levels of expertise, from musicians to sound artists, researchers, software developers, and computer scientists. More experienced users may want to skip some of the introductory points in Part I, depending on their background. Readers who are unsure about the mathematical language used can avail themselves of Appendix A where all concepts applied in the book are thoroughly explained (assuming just a basic understanding of arithmetics). All important code that is not completely provided in the main text of the book appears fully in Appendix B and is referenced in the relevant chapters.

I hope that this book will prove to be a useful reference for computer music practitioners on the topic of instrument development. It aims to provide a solid foundation for further research, development, music-making, and sound design. Readers are encouraged to experiment with the examples and programs presented here as they develop new perspectives in their practice of sound and music computing.

Maynooth, December 2016 *Victor Lazzarini*

Acknowledgements

I would like to acknowledge the help and encouragement of many of my colleagues at the university and in the computer music community. In particular, I would like to thank some of my close allies in the Csound brotherhood, Steven Yi, Øyvind Brandtsegg, Joachim Heintz, Tarmo Johannes, Rory Walsh, Iain McCurdy, Richard Boulanger and Michael Gogins, from whom I have learned a lot throughout these many years of interaction. In addition, I wish to acknowledge the work of François Pinot in designing such a great Python interface to the Csound API, ctcsound, a much better fit to the system than the automatically-generated wrappers we used to have.

I would also like to mention my colleagues Joe Timoney and Tom Lysaght, at the Computer Science Department, and Rudi Villing, at the Electronic Engineering Department, in Maynooth, to whom I am very thankful for their collaboration and support. Likewise, it is important to mention my co-researchers in the Ubiquitous Music (ubimus) group, Damián Keller, Nuno Otero and others, with whom I have had numerous exchanges that cemented many ideas underpinning my work in this field.

Some of the original research that led to parts of this book was also conducted in collaboration with colleagues from Aalto University in Finland, and I would like to express my gratitude to Jari Kleimola, Jussi Pekkonen, and Vesa Välimäki for their part in this work. Thanks should also go to the Faust development team at GRAME, led by Stéphane Letz and Yann Orlarey (who very kindly contributed a foreword to this book). Theirs is a fantastic piece of work, providing such a wonderful software system for computer music.

This book would not exist without the support, help, and understanding of my family, who many times had to endure my (mental, if not physical) disappearance for hours at weekends and evenings, as I battled with typesetting, programming, and explaining my ideas more clearly. To them, Alice, my wife, and our children Danny, Ellie, and Chris, I would like to make a special dedication.

I am very thankful to my editor at Springer, Ronan Nugent, for facilitating the development of this and other projects, and for the very efficient manner in which

these have been brought to fruition. The peace of mind provided by a well-structured process allows for a much more enjoyable writing experience.

Finally, I would like to pay tribute to John ffitch, to whom this book is dedicated. John has been a very influential figure in the computer music community and has given us a huge contribution in terms of free software, expertise, and guidance. This is just a small token of our gratitude for his generosity and support.

Contents

Part III Application Development

Appendices

Acronyms

0dbfs	Zero decibel full scale
12TET	12-Tone Equal Temperament
ADC	Analogue-to-Digital Converter
ADSR	Attack-Decay-Sustain-Release
AM	Amplitude Modulation
AP	All Pass
API	Application Programming Interface
BP	Band Pass
bpm	beats per minute
BR	Band Reject
CDP	Composer's Desktop Project
CLI	Command-Line Interface
cps	cycles per second
CPU	Central Processing Units
DAC	Digital-to-Analogue Converter
DAW	Digital Audio Workstation
dB	Decibel
DFT	Discrete Fourier Transform
DMI	Digital Music Instrument
DSL	Domain Specific Language
DSP	Digital Signal Processing
FBAM	Feedback Amplitude Modulation
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FIFO	First In First Out
FM	Frequency Modulation
FOF	Fonction d'Onde Formantique, formant wave function
FT	Fourier Transform
GPGPU	General Purpose Graphic Programming Unit
GPIO	General-Purpose Input/Output
GPPL	General-Purpose Programming Language

GPU	Graphic Programming Unit
GUI	Graphical User Interface
HDMI	High Definition Monitor Interface
HP	High Pass
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz
IDE	Integrated Development Environment
IDFT	Inverse Discrete Fourier Transform
IF	Instantaneous Frequency
IIR	Infinite Impulse Response
IO	Input-Output
IP	Internet Protocol
IR	Impulse Response
ISTFT	Inverse Short-Time Fourier Transform
JIT	Just In Time
JS	Javascript
LADSPA	Linux Audio Simple Plugin Architecture
LP	Low Pass
LTI	Linear Time-Invariant
MIDI	Musical Instrument Digital Interface
ModFM	Modified Frequency Modulation
MTU	Maximum Transmission Unit
OS	Operating Systeml
OSC	Open Sound Control
PAF	Phase-Aligned Formant
PD	Phase Distortion
PLTV	Periodic Linear Time-Varying
PM	Phase Modulation
PNaCl	Portable Native Client
PV	Phase Vocoder
PWM	Pulse Wave Modulation
RMS	Root Mean Square
RTP	Realtime Transport Protocol
SpSB	Split-Sideband
STFT	Short-Time Fourier Transform
TCP	Transmission Control Protocol
UDO	User-Defined Opcode
UDP	User Datagram Protocol
UG	Unit Generator
UI	User Interface
VDU	Video Display Unit
VGA	Video Graphics Array
VST	Virtual Studio Technology
WIMP	Windows, Icons, Menus, Pointer