Computer Music Instruments II

Victor Lazzarini

Computer Music Instruments II

Realtime and Object-Oriented Audio

Springer

Victor Lazzarini
Department of Music
Maynooth University
Maynooth, Kildare, Ireland

ISBN 978-3-030-13711-3 ISBN 978-3-030-13712-0 (eBook)
https://doi.org/10.1007/978-3-030-13712-0

Library of Congress Control Number: 2017953821

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

Foreword

Today's tools for music production have become increasingly democratised. Since the advent of the personal computer in the 1980s, means of audio synthesis, recording, editing and processing have become available to the general public. Before that time, a composer or other creative individual would need to go to a big studio or a computer centre to be able to work professionally with sonic creations. Likewise means of content distribution and tools for reaching audiences have become generally available, both for passive and for interactive listening media. Seen together, these technological changes have deeply affected the conditions for creative audio work. With wider and more affordable access, many more individuals from diverse backgrounds can work in this manner, and also the possible outcomes have multiplied. In tandem with this evolution, we have seen that the tools have become easier and easier to use. Many aspects of the expert knowledge of audio practitioners of earlier decades have been coded into the tools. Any piece of technology will affect the possible outcomes of a production process utilising it. This is also the case with audio production tools, by means of the affordances given to the creative individuals working with them. With ease of use comes also a delimitation of possible outcomes: some of the tools offered to the broad mass of creative consumers can be said to offer 'off-the-shelf creativity'. The individual using these tools is not so much creating but instead recombining the elements offered in pleasing ways. With some of the creative decisions being aided by the properties of the tools used, it becomes increasingly important to be able to make our own tools. This book provides a solid basis for doing so by introducing computational concepts and audio programming paradigms together with a firm foundation in programming.

As the book starts with the basics of the operating system, we are never lost for context. We then deal with compiling and running programs, getting to know C and C++ from the ground up and then proceed directly into realtime audio programming. There's as much DSP as we need to get to work and make things. Then, by the time the need for more occurs, the reader's general acquaintance with the field through practical work means that they should be well equipped to understand the literature needed to solve specialised problems outside the scope of this book. The interleaving of programming languages, by means of interfacing them with each

other, allows freedom to choose the best tool for the job. This ability to create freely also allows freedom from the imperatives of commercial actors, as well as freedom to create commercial products should one wish to do so.

I got to know Victor through international communities for open source audio programming, first and foremost though the Csound community. I deeply respect Victor's skills as a programmer, composer, musician, researcher and writer. His productivity seems to know no limit. I had the good fortune of contributing to the book 'Csound: A Sound and Music Computing System' together with Victor, John ffitch, Steven Yi, Iain McCurdy and Joachim Heintz in 2016. I also count myself lucky to be working with Victor in a current research project on crossadaptive processing, where we have also developed new methods of live convolution together with Sigurd Saue.

With all of the creative freedom afforded by the knowledge presented in this book, one could easily forget an additional benefit of this manner of working: transparency. For any future research on the creation process, to be able to trace the steps taken in the production, and to be able to study the intentions and incentives invested in the process of a work's creation could be of great value. Many of today's tools for the creative industry are closed source commercial products that are not compatible across versions of the same tool. This makes archiving for one's own purposes a hard task, and archiving for longer-term purposes nearly impossible. This is not to say that all our current creations deserve to be studied in the future, but it might just happen that someone sometime may be interested in knowing what we did and how we worked. Working with open source software does not in any way guarantee that our projects can be run on future versions of the same software. It merely allows the possibility for someone interested to be able to decode how the software was supposed to work, and then by careful reconstruction to be able to create the environment to open those saved projects. Reconstruction will always be time consuming, but by using open source, at least we offer the opportunity to do so.

Trondheim, March 2018 *Øyvind Brandtsegg*

Preface

This book can be read in a number of different ways. First and foremost, it is a companion volume to *Computer Music Instruments: Foundations, Design and Development*. Here, many ideas and concepts introduced in that book are broken down and explored at a lower level. Another way to read this book is to take it as a fairly complete course on C11 programming, with a slant towards sound and music computing, and an added introduction to key concepts of C++ and object-oriented programming (OOP). It is also possible to take this as an applied Digital Signal Processing text, which uses programming to discuss mathematical concepts. I would also think that a number of other readings can be attempted.

In any case, this book is complementary to its companion, but can also be taken on its own, as an independent text. It is true that many ideas explored here at an implementation level work out the elements of what was described there in more formal ways. There is however a conscious choice (in both books actually) to develop everything from first principles. In this text, we will also pay some attention to the discipline involved in writing code, and for this reason, programming problems are suggested in each chapter. It is my belief that we can only achieve fluency with plenty of practice, and readers who want to achieve a good level of C/C++ programming skills should attempt to solve every exercise proposed.

The book is divided into two parts, the first of which, as I have outlined above, is a comprehensive exploration of the C programming language and fundamental programming concepts, from the ground up. The fact that this language can be discussed fully in this space is one of the great attributes of C: being small. Part I traces a journey from zero to complete realtime audio programming. It equips readers with all the tools necessary to create realtime audio instruments at a reasonably low level. From early on, it prioritises examples and applications that have direct relevance to making sound with computers.

Chapter 1 introduces the reader to the desktop programming environment. In some ways, it picks up where we left off in the first *Computer Music Instruments* book, where a description of modern computing platforms for music making was offered. In the following chapters, we introduce all the components of C programming in a stepwise manner: data types, variables, arithmetics, input and output, control of

flow, arrays, pointers, functions, and data structures. By the time we reach Chapter 8, all of the language has been dealt with, and we start looking at key elements of the C standard library, such as memory allocation, and file input and output.

From Chapter 10 onwards, the focus is completely turned on to sound computing. In fact, we had introduced principles of audio signals as early as Chapter 4. As soon as we find some means of iterating operations, we are off producing sound waveforms. We discuss realtime audio synthesis and processing in Chapter 11 and complement it with MIDI control in the last chapter of Part I. At this stage, many key concepts of audio programming have been explored and we are ready to dive into DSP components, which is one of the main themes of Part II.

The other theme, of course, is OOP. Throughout the chapters in Part II, we continuously demonstrate how this paradigm is extremely useful for the modelling of computer music instruments. In Chapter 13, we introduce it gently by applying its principles to the development of a cornerstone of sound synthesis: the oscillator. Each chapter in Part II is devoted to a set of instrument components that are paired with key C++ programming concepts. Midway through, we are able to discuss the development of a fully-fledged object-oriented library, AuLib, which is used to illustrate the discussion of DSP algorithms, as well as OOP.

The following two chapters are devoted to specific audio processing concepts: delay lines and spectral manipulation. The latter connects very firmly with its companion text, Chapter 7 of *Computer Music Instruments*, and provides a complementary perspective to it. It covers similar ground, but uses programming as the main means to explore frequency-domain processing in a mostly non-mathematical way. The book closes with a look at the concept of plugins, also from an object-oriented perspective. At this point, we return, full circle, to Csound and study the means of developing the building blocks of instruments, opcodes, using C++. This final chapter connects very closely with the topics in the companion text, as it provides the means to implement in a native form many of the principles outlined in that earlier book.

The target audience of this book is aligned with that of its predecessor. While some understanding of acoustics and electronic music would be helpful in assisting the reader to understand some applications, it is not strictly necessary to have prior knowledge of audio DSP or even programming. Familiarity with other languages is also not a requirement, but may allow a faster progression through the first part of the book. C/C++ programmers with no experience with audio may be able to jump into the specific sections dealing with sound and music computing. Together with its companion volume, the present book aims to provide a comprehensive discussion of computational instruments for sound and music.

Maynooth, March 2018 *Victor Lazzarini*

Acknowledgements

Much of this book has been the result of over fifteen years of audio programming teaching at postgraduate level to music technology students. The flow and balance of topics has been tested in a large number of classes and seminars over the years. So I am deeply indebted to all of the students who have worked with me over the years, some of whom have gone on to become researchers, lecturers, and developers, and have made great contributions to the field themselves. In particular, I would like to thank Rory Walsh for taking the time to read some of the trickier sections of this book, helping me to pitch them at the right level, and providing useful comments.

I would also like to acknowledge the help and encouragement of the computer music community, as well as the various contributions to software development, ideas, and concepts that have arisen from them. Special thanks should go to colleagues in the Csound development team John ffitch, Steven Yi, Tarmo Johannes, Joachim Heintz, Stephen Kyne, François Pinot, Alex Hoffmann, and Bernt Isak Waerstad, for their input into this open-source project and also for the enlightening discussions on all matters to do with audio programming and beyond.

I am very grateful for the endorsement given by Øyvind Brandtsegg, who very kindly wrote the foreword for this book. Our collaboration stretches back many years, and recently I have had the chance to work closely with him and Sigurd Saue on some very interesting musical signal processing bits and pieces, which have indirectly contributed to elements in this book.

It is important to note the continued support of Ronan Nugent at Springer, who has been very helpful in facilitating the editorial process for this book.

As ever, the work for this book has been thoroughly supported by the patience and help I get from my wife Alice, and our children Danny, Ellie, and Chris. They are an integral part of any achievements I might be in a position to claim.

Contents

Part I Towards Realtime Audio in C

1 Introduction to the Programming Environment 3
1.1 The Operating System 3
1.1.1 The File System 4
1.1.2 The Terminal 5
1.1.3 Processes 8
1.1.4 The Manual 9
1.1.5 The POSIX Standard 9
1.2 The C/C++ Toolchain 9
1.2.1 Compilers and Interpreters 9
1.2.2 Compiling 10
1.2.3 Running Programs from the Terminal 11
1.3 Introduction to C Programming 11
1.3.1 Character and Keyword Sets 12
1.3.2 Entry Point 12
1.3.3 The `shin` Program 14
1.3.4 Summary 15
1.4 Conclusions 16
Problems 17

2 Data Types and Operators 19
2.1 Variables and Types 19
2.1.1 Encoding 20
2.1.2 Integers 22
2.1.3 Real Numbers 22
2.1.4 Characters 23
2.2 Initialisation, Assignment and Arithmetic Operations 24
2.2.1 Variable Scope 24
2.2.2 Constants 25
2.2.3 Operations 26

2.2.4 Conversion 27
2.2.5 Arithmetic Order 28
2.2.6 The **`sizeof`** Operator 28
2.3 Conclusions 28
Problems 29

3 Standard Input and Output 31
3.1 Printing to the Terminal 31
3.1.1 The Format String 32
3.2 Getting Input from the Terminal 34
3.2.1 Pattern Matching 34
3.3 Character Input and Output 35
3.4 The `calc` Program 36
3.5 Conclusions 37
Problems 37

4 Control of Flow 39
4.1 Conditional and Logical Expressions 39
4.2 Conditional Execution 40
4.2.1 Conditional Operator 42
4.3 Switch 43
4.4 Iteration 45
4.4.1 The **`while`** and **`do`** – **`while`** Loops 45
4.4.2 The **`for`** Loop 47
4.4.3 The **`break`** and **`continue`** Statements 48
4.5 A First Synthesis Program 48
4.5.1 Plotting the Waveform 49
4.5.2 Playing the Sound 52
4.5.3 Other Waveforms 52
4.6 Conclusions 53
Problems 53

5 Arrays and Pointers 55
5.1 Arrays 55
5.1.1 Two-Dimensional Arrays 57
5.2 Strings 57
5.3 Pointers 58
5.4 Pointers and Arrays 60
5.4.1 Pointer Arithmetic 60
5.4.2 Pointers and Strings 63
5.5 Conclusions 65
Problems 65

6 Functions 67
6.1 Function Definition 67
6.1.1 Arguments 68
6.1.2 Variable Lifetime 69
6.1.3 Call Semantics 69
6.1.4 Function Prototypes 70
6.1.5 Parametrised Macros and Inline Functions 70
6.1.6 Variable Argument Lists 72
6.1.7 Recursive Calls 73
6.2 Modular Programming 73
6.3 Pointers to Functions 75
6.4 The C Standard Library 77
6.5 Another Synthesis Program 77
6.5.1 Plotting 79
6.5.2 Realtime 80
6.6 Arguments to `main()` 81
6.6.1 Translating Arguments 82
6.7 Conclusions 83
Problems 83

7 Structures 85
7.1 Defining a New Type 85
7.1.1 Member Access 86
7.1.2 Pointers to Structures 87
7.2 Functions in Structures 88
7.3 Unions 89
7.4 Enumerations 89
7.5 Bitwise Operations 90
7.5.1 Bitwise Logic 90
7.5.2 Bitshift Operators 92
7.6 Conclusions 93
Problems 93

8 Memory Management 95
8.1 Allocating Memory 95
8.1.1 Reallocation 96
8.1.2 Freeing Memory 97
8.1.3 Setting and Copying Memory Blocks 97
8.2 Dynamic Arrays 97
8.3 Linked Lists 99
8.4 Conclusions 102
Problems 103

9 File Input and Output 105
9.1 Standard C Library File IO 105
9.2 Text File Functions 107
9.3 Direct File IO Functions 108
9.3.1 Reading/Writing Position 109
9.3.2 Error Reporting 109
9.4 File System Functions 110
9.5 Programming Examples 110
9.5.1 The `tobin` Program 110
9.5.2 External Score Generation for Csound 111
9.6 Conclusions 112
Problems 113

10 Soundfiles 115
10.1 Digital Audio 115
10.1.1 Sampling Frequency 116
10.1.2 Sample Precision 117
10.1.3 Audio Channels 118
10.2 Basic Operations on Signals 119
10.2.1 A Synthesis Example 120
10.2.2 Byte Order 121
10.2.3 Self-Describing Soundfile Formats 121
10.3 The libsndfile Library 122
10.3.1 Opening Files 122
10.3.2 Reading and Writing 124
10.3.3 Seeking 125
10.3.4 An Example Program 126
10.4 Conclusions 128
Problems 128

11 Realtime Audio 131
11.1 Portaudio 132
11.1.1 Listing Devices 133
11.1.2 Stream Parameters 134
11.1.3 Opening Devices 135
11.1.4 Synchronous Mode 136
11.1.5 Asynchronous Mode 137
11.1.6 Closing Up 139
11.1.7 The `todac` Program 139
11.1.8 An Audio Effect 141
11.2 The Jack Connection Kit 145
11.2.1 Opening a Client 146
11.2.2 Registering Ports 147
11.2.3 The Processing Callback 148
11.2.4 Connecting Ports 148

11.2.5 Closing a Client . . . 149
11.2.6 Application Example . . . 149
11.3 Conclusions . . . 154
Problems . . . 154

12 Realtime MIDI . . . 155
12.1 The Protocol . . . 155
12.1.1 Hexadecimal Notation Revisited . . . 156
12.1.2 MIDI Messages . . . 156
12.1.3 Packing and Unpacking the Status Byte . . . 158
12.2 MIDI Programming Basics . . . 158
12.2.1 MIDI on MacOS . . . 159
12.3 MIDI Programming with Portmidi . . . 163
12.3.1 Timers . . . 164
12.3.2 Opening Devices . . . 165
12.3.3 Output . . . 166
12.3.4 Input . . . 168
12.3.5 A MIDI Synthesiser . . . 169
12.4 MIDI on Jack . . . 174
12.4.1 Example . . . 176
12.5 Conclusions . . . 182
Problems . . . 182

Part II Object-Oriented Audio in C++

13 Oscillators . . . 185
13.1 Moving to C++ . . . 187
13.1.1 C++ Structures . . . 188
13.1.2 Overloading and Optional Parameters . . . 190
13.1.3 Memory Management . . . 191
13.2 The Table Lookup Oscillator . . . 192
13.3 Conclusions . . . 196
Problems . . . 196

14 Interpolation . . . 199
14.1 Linear Interpolation . . . 200
14.2 Cubic Interpolation . . . 201
14.3 Inheritance . . . 202
14.3.1 Polymorphism . . . 205
14.3.2 Oscillator Inheritance Tree . . . 205
14.4 Function Table Objects . . . 209
14.5 Reference Types . . . 210
14.5.1 Copy Constructors . . . 212
14.5.2 Object Reference Arguments . . . 212
14.5.3 Self References . . . 214
14.6 Phase Generators and Table Readers . . . 214

14.6.1 The Phasor . . . 215
14.6.2 Table Reader . . . 215
14.7 Conclusions . . . 216
Problems . . . 217

15 Envelopes . . . 219
15.1 Envelope Generators . . . 219
15.1.1 Linear Envelopes . . . 220
15.1.2 Exponential Envelopes . . . 221
15.2 Access Control and Classes . . . 222
15.2.1 Namespaces . . . 224
15.2.2 A Line Class . . . 225
15.3 Operator Overloading . . . 228
15.3.1 Standard IO Revisited . . . 229
15.4 An Audio Output Class . . . 230
15.5 Conclusions . . . 232
Problems . . . 232

16 Filters . . . 235
16.1 Feedback Filters . . . 236
16.1.1 First-Order Tone Filters . . . 236
16.1.2 Second-Order Filters . . . 238
16.1.3 Fourth-Order Filters . . . 241
16.1.4 Balancing . . . 242
16.2 Templates . . . 243
16.2.1 Templates in the Standard C++ Library . . . 244
16.2.2 Range-Based Loops . . . 247
16.3 Conclusions . . . 247
Problems . . . 248

17 AuLib . . . 249
17.1 Object-Oriented Audio Systems . . . 250
17.2 Library Design . . . 251
17.2.1 Stateful versus Stateless Representations . . . 251
17.2.2 Abstraction and Encapsulation . . . 253
17.2.3 Code Reuse . . . 253
17.2.4 Connectivity . . . 255
17.3 A Tour of the Library . . . 256
17.3.1 Signal Generators . . . 257
17.3.2 Signal Processors . . . 259
17.3.3 Audio Input and Output . . . 259
17.4 Synthesis and Processing Control . . . 260
17.5 An AuLib Instrument . . . 260
17.6 Conclusions . . . 264
Problems . . . 264

18 Delay Line Processing 265
18.1 Circular Buffers 266
18.2 Fixed-Delay Effects 267
18.2.1 Comb Filters 272
18.2.2 All-Pass Filters 273
18.3 Variable Delay Lines 275
18.4 Multiple Taps 278
18.4.1 Convolution 280
18.5 Lambda Functions 282
18.5.1 Auto Types 283
18.6 Conclusions 284
Problems 285

19 Frequency-Domain Processing 287
19.1 Fundamental Principles 288
19.1.1 Complex Numbers 289
19.1.2 Spectral Analysis 290
19.2 The Fast Fourier Transform 292
19.2.1 Real-to-Complex and Complex-to-Real Transforms 298
19.3 Fast Convolution 302
19.3.1 Overlap Add 305
19.3.2 Overlap Save 305
19.3.3 Multiple Partitions 306
19.3.4 Convolution Reverb 310
19.4 Streaming Spectral Processing 312
19.4.1 Spectral Analysis 315
19.4.2 Resynthesis 317
19.4.3 Spectral Manipulation 320
19.5 Conclusions 323
Problems 324

20 Plugins 325
20.1 Plugins in Csound 325
20.2 Framework Design 326
20.2.1 The Base Classes 327
20.2.2 Deriving Opcode Classes 328
20.2.3 Registering Opcodes with Csound 331
20.3 The Csound Engine Object 332
20.4 Opcode Programming 333
20.4.1 Delay Line 333
20.4.2 Table-Lookup Oscillator 335
20.4.3 Text Processing 336
20.4.4 Spectral Processing 337
20.4.5 Array Processing 339
20.4.6 External Resources 341

20.4.7 Multithreading Opcodes 343
20.5 Conclusions 343
Problems 343

Appendix

A AuLib Reference 347
A.1 Library-Wide Definitions 347
A.2 `AudioBase` 348
A.3 Deriving New Classes 350
A.4 Audio DSP Classes 352
A.5 Control Classes 354
A.5.1 MIDI Synth Example 356
A.6 Other Classes 360
A.7 Building AuLib 360

References 363

Index 367

Acronyms

0dbfs	Zero decibel full scale
ADC	Analogue-to-Digital Converter
ADSR	Attack-Decay-Sustain-Release
AP	All Pass
API	Application Programming Interface
BP	Band Pass
BR	Band Reject
cps	cycles per second
DAC	Digital-to-Analogue Converter
dB	Decibel
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FS	File System
GUI	Graphical User Interface
HAL	Hardware Audio Layer
HP	High Pass
Hz	Hertz
IDFT	Inverse Discrete Fourier Transform
IF	Instantaneous Frequency
IIR	Infinite Impulse Response
IO	Input-Output
IR	Impulse Response
ISTFT	Inverse Short-Time Fourier Transform
LFO	Low Frequency Oscillator
LP	Low Pass
LSB	Least Significant Byte
MIDI	Musical Instrument Digital Interface
MSB	Most Significant Byte

OLA	Overlap-Add
OLS	Overlap-Save
OOP	Object-Oriented Programming
OS	Operating System
PCM	Pulse Code Modulation
PID	Process Identifier
PV	Phase Vocoder
RMS	Root Mean Square
STFT	Short-Time Fourier Transform