

Loadbang

Programming Electronic Music in Pd

Johannes Kreidler

Second Edition 2013

© Johannes Kreidler

All rights reserved by the publisher Wolke Verlag, Hofheim

Translation: Mark Barden

Cover design: Friedwalt Donner and pd-graz

Typesetting: michon, Hofheim

Printing: Fuldaer Verlagsanstalt

ISBN 978-3-95593-055-4

Table of Contents

Preface	13
Introduction to this book's methodology	14
1. Introduction to Pd	15
1.1 General remarks	15
1.2 Installing and setting up Pd	18
2. Programming with Pd for the first time	19
2.1 Introduction	19
2.1.1 A simple example	19
2.1.2 Surface elements in Pd	24
2.1.3 Summary	26
2.1.4 Appendix	27
2.1.4.1 List of all objects	27
2.1.4.2 Help file	27
2.1.4.3 Duplication	28
2.1.4.4 Short cuts	28
2.1.4.5 Comments	28
2.1.5 For those especially interested: Atoms	28
2.2 The control level	30
2.2.1 Mathematical operations and order	30
2.2.1.1 Theory	30
2.2.1.1.1 Basic mathematical functions	30
2.2.1.1.2 Order	31
2.2.1.1.3 Expression	33
2.2.1.1.4 Other mathematical operations	33
2.2.1.1.5 Float and counter	35
2.2.1.1.6 Summary	36
2.2.1.2 Applications	36
2.2.1.2.1 Two frequencies—two volume levels	36
2.2.1.2.2 An interval	37
2.2.1.2.3 Random melody	37
2.2.1.2.4 Rounding	37
2.2.1.2.5 How long is this score?	37
2.2.1.2.6 Counting in series	38
2.2.1.2.7 Random without repetitions	38
2.2.1.2.8 More exercises	38
2.2.1.3 Appendix	39
2.2.1.3.1 Input for bang	39
2.2.1.3.2 How numbers are represented	39
2.2.1.3.3 More on trigger	39

2.2.1.4	For those especially interested	40
2.2.1.4.1	About series	40
2.2.1.4.2	Regarding float	40
2.2.2	Different types of data	40
2.2.2.1	Theory	40
2.2.2.1.1	Bang—a GUI object	40
2.2.2.1.2	Messages	40
2.2.2.1.3	Lists	41
2.2.2.1.4	Messages with variables	42
2.2.2.1.5	Messages: Set	43
2.2.2.1.6	Makefilename	43
2.2.2.1.7	Openpanel	44
2.2.2.1.8	Simple data storage	44
2.2.2.1.9	Route	44
2.2.2.1.10	Demultiplex	45
2.2.2.1.11	Spigot	46
2.2.2.1.12	Toggle	46
2.2.2.2	Applications	47
2.2.2.2.1	Using lists with pitches and dynamics	47
2.2.2.2.2	On/off switch	47
2.2.2.2.3	Pitches with names	47
2.2.2.2.4	A simple sequence	47
2.2.2.2.5	A limited counter	48
2.2.2.2.6	More exercises	49
2.2.2.3	Appendix	49
2.2.2.3.1	Symbol boxes	49
2.2.2.3.2	Slider	49
2.2.2.3.3	Radio	50
2.2.2.3.4	Using Slider and Radio	50
2.2.2.4	For those especially interested:	
Other type specifications and more about boxes		50
2.2.3	Time operations	51
2.2.3.1	Theory	51
2.2.3.1.1	Metro	51
2.2.3.1.2	Delay	51
2.2.3.1.3	Pipe	51
2.2.3.1.4	Line	52
2.2.3.1.5	Timer	53
2.2.3.2	Applications	54
2.2.3.2.1	Automatic random melody	54
2.2.3.2.2	Glissando	54
2.2.3.2.3	Glissando melody	55
2.2.3.2.4	Irregular random rhythms	55
2.2.3.2.5	Canons	56
2.2.3.2.6	Rests	57
2.2.3.2.7	Crescendo/Decrescendo	58

2.2.3.2.8	Metronome	58
2.2.3.2.9	More exercises	59
2.2.3.3	Appendix	60
2.2.3.3.1	Distributing lists	60
2.2.3.3.2	Time resolution for control data	60
2.2.4	Miscellaneous	61
2.2.4.1	Sending and receiving	61
2.2.4.1.1	Send/Receive	61
2.2.4.1.2	Sending with lists	63
2.2.4.1.3	A series of lists	63
2.2.4.1.4	Value	64
2.2.4.2	Loadbang	64
2.2.4.3	GUI options	64
2.2.4.3.1	Number and symbol box	65
2.2.4.3.2	Bang	65
2.2.4.3.3	Toggle	66
2.2.4.3.4	Slider	66
2.2.4.3.5	Radio	67
2.2.4.3.6	Canvas	67
2.2.4.3.7	Examples of altered GUI objects	67
2.2.4.3.8	Change font size	68
2.2.4.3.9	Tidy up	68
2.2.4.4	Subpatches	69
2.2.4.4.1	Space	69
2.2.4.4.2	Modularization	71
3.	Audio	73
3.1	Basics	73
3.1.1	Pitch	73
3.1.1.1	Theory	73
3.1.1.1.1	Controlling speakers digitally	73
3.1.1.1.2	Waves	75
3.1.1.1.3	Measurement	76
3.1.1.1.4	Sample rate	79
3.1.1.1.5	Samples—milliseconds	80
3.1.1.2	Applications	80
3.1.1.2.1	Tempered—random	80
3.1.1.2.2	More exercises	81
3.1.1.3	Appendix	81
3.1.1.3.1	Nyquist Theorem	81
3.1.1.3.2	DSP	82
3.1.1.4	For especially interested	82
3.1.1.4.1	da- / ad- conversion	82
3.1.1.4.2	Sound waves	83
3.1.1.4.3	Converting MIDI numbers into frequencies	83
3.1.1.4.4	Noise periodicity	84

3.1.2 Volume	85
3.1.2.1 Theory	85
3.1.2.1.1 Measurement	85
3.1.2.1.2 Problems	89
3.1.2.1.3 Phase	90
3.1.2.1.4 Sound waves are additive	91
3.1.2.2 Applications	93
3.1.2.2.1 Chord	93
3.1.2.2.2 Glissandi	93
3.1.2.2.3 Processing adc-input	94
3.1.2.2.4 Oscillator concert	94
3.1.2.2.5 More exercises	96
3.1.2.3 Appendix	96
3.1.2.3.1 Other tilde objects	96
3.1.2.3.2 Bit depth	97
3.1.2.4 For those especially interested	97
3.1.2.4.1 Sound pressure vs. sound intensity	97
3.1.2.4.2 Control data vs. signals	97
3.2 Additive Synthesis	99
3.2.1 Theory	99
3.2.1.1 The harmonic series	99
3.2.2 Applications	102
3.2.2.1 A random klangfarbe (German: sound color)	102
3.2.2.2 Changing one klangfarbe into another	102
3.2.2.3 Natural vs. equal-tempered	103
3.2.2.4 More exercises	103
3.2.3 Appendix	104
3.2.3.1 Pd's limitations	104
3.2.4 For those especially interested	104
3.2.4.1 Studie II	104
3.2.4.2 Composing with spectra	104
3.3 Subtractive synthesis	105
3.3.1 Theory	105
3.3.1.1 White noise	105
3.3.1.2 Filters	105
3.3.2 Applications	107
3.3.2.1 Filter colors	107
3.3.2.2 Telephone filters	108
3.3.2.3 More exercises	108
3.3.3 Appendix	108
3.3.3.1 White noise and clicks	108
3.3.3.2 Pink noise	109
3.3.3.3 DC offset	109
3.3.4 For those especially interested	110
3.3.4.1 How digital filters work	110
3.4 Sampling	112

3.4.1 Theory.....	112
3.4.1.1 Storing sound.....	112
3.4.1.1.1 Sound files.....	112
3.4.1.1.2 Buffers.....	112
3.4.1.2 Playback of saved sound.....	120
3.4.1.3 Audio delay.....	123
3.4.2 Applications.....	125
3.4.2.1 A simple sampler.....	125
3.4.2.2 With variable speed.....	125
3.4.2.3 Any position.....	126
3.4.2.4 Sampler-player.....	127
3.4.2.5 Loop generator.....	130
3.4.2.6 Reverb.....	132
3.4.2.7 Texture.....	133
3.4.2.8 Comb filter.....	133
3.4.2.9 Octave doubler.....	134
3.4.2.10 Karplus-Strong algorithm.....	136
3.4.2.11 More exercises.....	138
3.4.3 Appendix.....	138
3.4.3.1 Array oscillator.....	138
3.4.3.2 Array playback.....	138
3.4.3.3 Playing back an array in a block.....	139
3.4.3.4 Glissandi of samples.....	139
3.4.3.5 Additive synthesis with array.....	142
3.4.3.6 Latency.....	143
3.4.4 For especially interested.....	144
3.4.4.1 4-point-interpolation.....	144
3.4.4.2 Sample-wise delay.....	145
3.5 Wave shaping.....	146
3.5.1 Theory.....	146
3.5.1.1 Waveforms.....	146
3.5.1.2 Transfer functions.....	151
3.5.1.3 (Controlled) Random waveforms.....	152
3.5.1.4 Wave stealing.....	157
3.5.2 Applications.....	158
3.5.2.1 Singing waveforms.....	158
3.5.2.2 Transfers.....	159
3.5.2.3 Even / odd partials.....	159
3.5.2.4 More exercises.....	160
3.5.3 Appendix.....	160
3.5.3.1 Foldover.....	160
3.5.4 For those especially interested.....	162
3.5.4.1 GENDY.....	162
3.6 Modulation synthesis.....	163
3.6.1 Theory.....	163
3.6.1.1 Ring modulation.....	163

3.6.1.2	Frequency modulation	165
3.6.2	Applications	167
3.6.2.1	More sonically complex ring modulation.	167
3.6.2.2	Live ring modulation	167
3.6.2.3	Live frequency modulation	167
3.6.2.4	More exercises	168
3.6.3	Appendix	168
3.6.3.1	Phase modulation	168
3.7	Granular synthesis.	170
3.7.1	Theory.	170
3.7.1.1	Theory of granular synthesis	170
3.7.2	Applications.	175
3.7.2.1	Live granular synthesis.	175
3.7.2.2	Live with feedback	177
3.7.2.3	More exercises	177
3.7.3	Appendix	178
3.7.3.1	Granular technique as a synthesizer	178
3.8	Fourier analysis	179
3.8.1	Theory.	179
3.8.1.1	Analyzing partials.	179
3.8.1.2	Analyze whatever signal you want.	182
3.8.2	Applications	184
3.8.2.1	Filters	184
3.8.2.2	Folding.	185
3.8.2.3	Compressor	186
3.8.2.4	Spectral delay	187
3.8.3	Appendix	188
3.8.3.1	Fiddle~	188
3.8.3.2	Tuner.	190
3.8.3.3	Octave doubler #2.	191
3.8.3.4	Pitch follower.	191
3.8.3.5	More exercises.	191
3.9	Amplitude corrections	192
3.9.1	Theory.	192
3.9.1.1	Limiter.	192
3.9.1.2	Compressor	194
3.9.2	Applications.	194
3.9.2.1	Larsen tones.	194
3.9.2.2	More exercises.	195
3.9.3	Appendix	195
3.9.3.1	Movements in space	195
3.9.4	For those especially interested	197
3.9.4.1	Other windows	197

4. Controlling sound	200
4.1 Algorithms	200
4.1.1 Theory	200
4.1.1.1 What are algorithms?	200
4.1.2 Applications	201
4.1.2.1 Stochastics	201
4.1.2.2 Recursive systems	204
4.1.2.3 More exercises	205
4.1.3 Appendix	205
4.1.3.1 DSP loop	205
4.1.4 For those especially interested	206
4.1.4.1 Algorithmic composition	206
4.2 Sequencer	207
4.2.1 Theory	207
4.2.1.1 Text file	207
4.2.1.2 Qlist	209
4.2.2 Applications	210
4.2.2.1 Score for a patch	210
4.2.2.2 More exercises	212
4.2.3 Appendix	212
4.2.3.1 Modifying qlist	212
4.2.4 For those especially interested	214
4.2.4.1 Creating lists externally: Lisp	214
4.3 HIDs	215
4.3.1 Theory	215
4.3.1.1 Keyboard and mouse	215
4.3.1.2 MIDI	216
4.3.1.3 Using signals to control sound	218
4.3.2 Applications	218
4.3.2.1 Playing patches live	218
4.3.2.2 More exercises	219
4.3.3 Appendix	220
4.3.3.1 Other HIDs	220
4.3.3.2 Video input	220
4.3.4 For those especially interested	220
4.3.4.1 Instrument design	220
4.4 Network	221
4.4.1 Netsend / Netreceive	221
4.4.2 OSC	221
5. Miscellaneous	223
5.1 Streamlining	223
5.1.1 Theory	223
5.1.1.1 Subpatches	223
5.1.1.2 Abstractions	226
5.1.1.3 Expanding Pd	230

5.1.2 Applications	231
5.1.2.1 Customize your Pd	231
5.1.2.2 More exercises	232
5.1.3 Appendix	232
5.1.3.1 Creating a patch automatically	232
5.1.4 For those especially interested	235
5.1.4.1 Writing your own objects	235
5.2 Visuals	236
5.2.1 Theory	236
5.2.1.1 Pd is visual and this can be programmed	236
5.2.2 Applications	237
5.2.2.1 Main patch window and subpatches	237
5.2.2.2 Canvasses as display	237
5.2.2.3 Canvasses as expanded GUI	239
5.2.2.4 More exercises	240
5.2.3 Appendix	240
5.2.3.1 Data structures	240
5.2.4 For those especially interested	247
5.2.4.1 GEM	247
Afterword	248
Appendix A. Solutions	249
2.2.1.2.8	249
2.2.2.2.6	250
2.2.3.2.9	251
3.1.1.2.2	253
3.1.2.2.5	254
3.3.2.3	255
3.4.2.11	256
3.5.2.4	261
3.7.2.3	262
3.8.3.5	263
3.9.2.2	264
4.1.2.3	265
4.2.2.2	267
5.1.2.2	268
5.2.2.4	269
Index	273

Preface

This book is the result of my experience of teaching electronic music. Through the teaching process, I became familiar with the most common stumbling blocks students encounter—especially when the student’s native language is not the language in which lessons are conducted.

Pd (Pure Data) is a professional, high-performance programming language for electronic sound processing. It is *open source*, i.e. available for free on the Internet. One disadvantage of this is that Pd is only discussed in certain institutions or Internet forums. The complicated technical terminology usually found there is enormously difficult for beginners to understand. This book will help first-time users to clear those first few hurdles when learning Pd.

Pd’s main designer, Miller Puckette, is also writing a book about the theory and technology of electronic music processing with Pd. Surely there is no better teacher of a programming language than the person who designed it; his primarily scientific approach certainly does cover all the material in a thorough, systematic fashion. However, his method of teaching can be difficult to comprehend. My pedagogical experience has been that Puckette’s text demands a large amount of mathematical, computer science, and terminological knowledge from its readers.

This book is designed for self-study, principally for composers. It begins with explanations of basic programming and acoustic principles before gradually building up to the most advanced electronic music processing techniques. Some knowledge of physics is assumed and explanations of basic physics concepts have been intentionally omitted. My book’s teaching approach is focused primarily on hearing, which I regard as a faster and more enjoyable way to absorb new concepts than through abstract formulas. In terms of mathematics, I explain only what is absolutely necessary to comprehend a given processing concept. I explain the various techniques from a compositional perspective, rather than attempting a computer science-, math-, or physics-based discussion of processing phenomena or structures. Therefore, the decisions and comments I have made are purely subjective and are open to debate.

This book would not have been possible without the support of Prof. Mathias Spahlinger, the expert supervision of Prof. Orm Finnendahl, suggestions and patches from the Pd community, the manuscript editing and DocBook-XML coding efforts of Esther Kochte. I would also like to thank Mark Barden for the English translation and the Musikhochschule Freiburg and the state of Baden-Württemberg for financing the project. This book will hopefully increase interest in electronic music, thereby indirectly enriching the aesthetic discourse of New Music.

Johannes Kreidler, January 2008

Introduction to this book's methodology

The following material begins with basic computer knowledge. The first steps are therefore described in meticulous detail.

Pd can run on different platforms (like Linux, OS X, or Windows) and this book is not platform-specific. Problems relating to the operating system will not be discussed, as they are simply beyond the scope of this tutorial (and it is also quite likely that changes—updates, bug fixes, etc.—will occur in the near future). It is therefore assumed that Pd has been correctly installed and has been integrated with the hardware environment (consult an Internet forum to resolve any of these sorts of problems, e.g. “Pd-list”).

How to use this book: Each lesson is comprised of a theory part, a practice part and an appendix, as well as individual aspects that are explained in greater detail at the end of each section. This in-depth information is aimed at advanced users and is not essential to acquire a basic working knowledge of Pd. I recommend working through the whole book without consulting these additional details first, then going back to learn them later.

Now and again, some fundamental concepts of acoustics are discussed. The exercises contain not only specific compositional questions, but also applications that are useful for musicians' everyday needs—e.g. tools like the metronome or tuning device. In this respect, the tutorial could be used by interpreters as well as composers.

Chapter 1

Introduction to Pd

1.1 General remarks

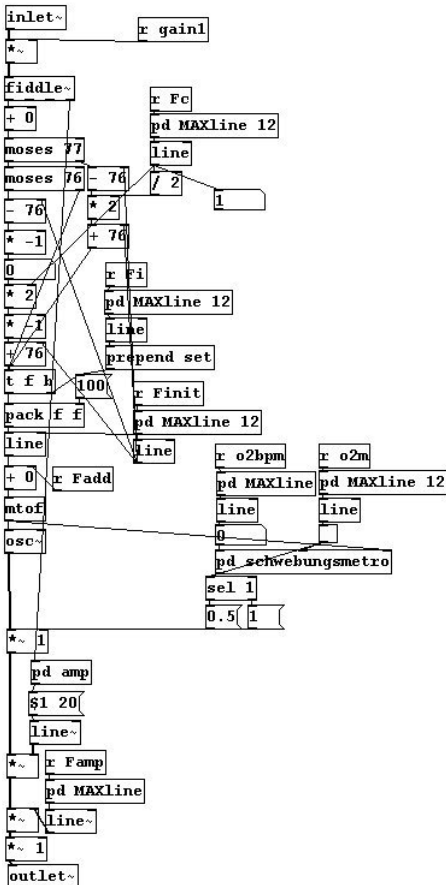
Pd (Pure Data) is a programming language for electronic music. Creating music on a computer is technically referred to as DSP (digital signal processing). “Digital” means that information is represented by digits—computers, as you may know, work only with numbers. “Signal” is the technical term for a special mode of computer operation that deals with sound. “Processing” refers to functions executed by the computer.

Pd was initiated by American software engineer Miller Puckette, who previously co-developed the well known and similarly structured software Max/Msp. Pd is not commercial software; i.e., it was not developed by a corporation and is not for sale. Instead, it is “open source”: its source code can be viewed by anyone. This source code is also not the (patented) property of a corporation, but is rather freely available to all. This also means that, provided sufficient knowledge, anyone can change the program. Today, many other programmers, musicians, acoustic engineers, and composers have joined Miller Puckette to continue Pd’s development. As a result of this, there is no final, definitive version of Pd; the program is under constant development. In addition to the huge advantage of free availability on the Internet, it is also “democratically” expanded and optimized on a professional level. One drawback to this is that a detailed operating manual for users who lack programming experience has not existed until now. In contrast to a corporation, which has a monetary interest in ensuring that first-time users can easily operate new software, the open source movement lacks such a driving force to make itself accessible. This book is an attempt to fill that gap.

In precise terms, Pd is a “real-time graphical programming environment for audio processing”. Traditionally, programmers work with text-based programming languages. They create what is called “code”, which is processed by a computer to produce a result. To carry out its programming functions, Pd uses visual objects that the user places and alters on the screen. These visual objects—small boxes that can be connected to each other—are a throwback to analogue studios that were used to produce electronic music before the advent of computers: various devices—today symbolized by our little boxes—are connected to each other using lines that—like cables—symbolize physical connections between the boxes. (Due to this type of connection, Pd is referred to as a datastream-oriented programming language.)



An analog studio—devices are connected with cables.



Pd boxes are connected to each other.

One major advantage of Pd is the aspect of „real-time“. This means that, in contrast to most programming environments where a text is first entered that must be separately processed by the computer before obtaining a result, changes in Pd can be made during performance. Like on a classical instrument, the user hears the result instantaneously and can change it immediately. This makes Pd especially well suited for use in live performance.

Pd has become much more than a programming language for electronic music. Since users across the globe can participate in the project, there are user-programmed modules for what are called „externals“: video, Internet connection, joystick integration, etc. Whole libraries of these modules even exist („external libraries“). Some of these externals have been integrated into the regular version of Pd.

1.2 Installing and setting up Pd

Readers of this book should have Pd installed on their computer so they are able to try out the processes described. Without this simultaneous practical experience, this tutorial will be difficult to understand.

First you need a computer with at least 128 MB main memory, a 500 MHz processor and ca. 500 MB hard disk space (these are the absolute minimum requirements!). Pd works with the following operating systems: Linux, OS X, and Windows.

Then you need to download the newest version of Pd-extended from the Internet. Enter “Pd-extended” into an Internet search engine. Since the address for the download portal may change in the future, no link to the site will be provided here. Pd-extended is a version of the original software (also called “Pd vanilla”) that has been expanded with numerous libraries. Most of the exercises described here work with the original version of Pd, but not all of them. The extra objects in Pd-extended make the program much more practical in general. This tutorial assumes Pd-extended version 0.39 or higher.

Once Pd has been installed, we open it from the directory Pd/bin/. A window appears. This is the main control center, so to speak. Here you can test whether Pd is functioning properly: In the main menu, click on **Media** → **Test Audio and MIDI**. Under “TEST SIGNAL”, click first on the box next to “-40”, then on the box next to “-20”. You should hear a sine tone coming out of the computer’s loudspeaker (A4). If you do not, then you need to adjust your hardware settings (under **Media** → **Audio settings**). More information regarding problems that arise at this stage cannot be given here. For help resolving any problems, please consult the “Pd-list”, a forum of Pd users on the Internet. If a microphone is connected, the digits in at least the leftmost two boxes under “AUDIO INPUT” should change in response to sound picked up by the microphone. As long as the test tone is working, you can work with the program without a microphone. (By Chapter 3 at the latest, however, you will sometimes need a microphone.)