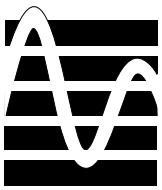
Volume 25 • THE COMPUTER MUSIC AND DIGITAL AUDIO SERIES



DESIGNING AUDIO OBJECTS FOR MAX/MSP AND PD

Eric Lyon

■

A-R Editions, Inc.

Middleton, Wisconsin

Library of Congress Cataloging-in-Publication Data

Lyon, Eric, 1962–
Designing audio objects for Max/MSP and Pd / Eric Lyon. – 1st ed.
p. cm. — (The computer music and digital audio series)
Includes bibliographical references and index.
ISBN 978-0-89579-715-5 (alk. paper)
1. Max (Computer file : Cycling '74) 2. Pure Data (Computer file) 3. Computer sound processing. I. Title.
ML74.4.M39L96 2011
006.4'5–dc23

2011018589

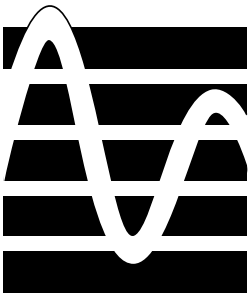
ISBN 978-0-89579-715-5

A-R Editions, Inc., Middleton, Wisconsin 53562

Printed in the United States of America

10 8 6 4 2 1 3 5 7 9

Contents



Preface

This is a book about designing and implementing audio objects for Max/MSP and Pd. Audio objects are modular pieces of audio processing code which are loaded on demand into a program such as Max/MSP or Pd. The great advantage of writing audio objects, compared with writing monolithic standalone audio programs, is that you only need to implement the essential functionality of your object. Once completed, your audio object can interact in potentially unlimited ways with other objects, and it can make use of all audio capabilities provided by the host program.

IS THIS BOOK FOR YOU?

This book is primarily for readers who want to learn how to design audio objects for Max/MSP or Pd. You may be an advanced user of Max/MSP or Pd, or a DSP professional wishing to build a performance-oriented implementation of your DSP algorithm, or an Ableton Live user who has gotten involved with Max/MSP through Max for Live. Or perhaps you are a computer musician working primarily with other audio software. Since this book focuses on the *process* of developing modular audio digital signal processing (DSP) code, it could be useful for readers who wish to develop plug-ins for other platforms, such as VST, Audio Units (AU), SuperCollider, or Csound. It could even be useful for readers who wish to write standalone audio applications for computers or mobile devices like the iPhone.

This book can be used for classes on computer music focused on audio programming or in advanced courses on Max/MSP or Pd, but it is specifically designed for self-study. I hope it will prove useful to independent computer musicians who would like to achieve a deeper understanding of and facility with their instrument. More than twenty years ago, Gareth Loy, an extraordinarily talented audio programmer (and one of my teachers at UCSD), observed that audio programming was an oral tradition, an observation that remains valid today. This book is intended to put some of that oral tradition on paper where it can be of use to contemporary electronic musicians who might not have easy access to lessons with an expert audio programmer.

WHY AUDIO OBJECTS?

There are already so many audio objects available for Max/MSP and Pd that you could spend a creative lifetime just exploring the capabilities of these existing objects. So, why would you want to design more audio objects? Then again, why work with Max/MSP or Pd when there are so many creative options available in modern digital audio workstations (DAWs) like Logic and Nuendo? If you are at all like me, the answer is that you are deeply involved in the process of working with sound and desire extremely detailed control over your sonic workflow. DAWs, wonderful though they are, present a fixed notion of the flow and graphic representation of sound in time. Max/MSP and Pd allow you to organize the sonic, temporal, and graphic elements of electronic music with much greater flexibility.

But Max/MSP and Pd have some significant limitations, most notably that you can only patch together objects that already exist. Sooner or later, you may want to work with an audio object that does not yet exist. This book was written to help you acquire the programming and design skills needed to build that object. For example you might want to write a Max object that implements an algorithm that you found in a DSP book. Or you might want to port a Pd object to Max/MSP. You might want to implement DSP algorithms of your own invention. Once you acquire the capability to design audio objects, the creative possibilities expand rapidly.

WHAT YOU NEED TO KNOW

I assume that you are conversant with Max/MSP or Pd and that you have a basic acquaintance with the C programming language. If you are familiar with JavaScript programming inside of Max/MSP, then you probably have enough programming experience to pick up the C you need as you go along. But if procedural programming is completely new to you, then you will need a good resource to get up to speed. There are many good online C programming tutorials, such as *The GNU C Programming Tutorial* by Mark Burgess. *The Audio Programming Book*, edited by Richard Boulanger and Victor Lazzarini, teaches both C and C++ with a focus on audio applications. You might also consider obtaining a classic: Kernighan and Ritchie's *The C Programming Language*. It's what I used to teach myself C in 1985, and the fact that this slim volume is still in print is a testament to its efficacy.

A NOTE ABOUT TERMINOLOGY

In the parlance of both Max/MSP and Pd, audio objects are often referred to as externals. I will adopt that terminology in this book (for the origin of the term *external*, please see Miller Puckette's afterword in this book). When I refer individually

to either Max/MSP or Pd, I will use their full names. However, given the similarities between the programs, when I refer to both Max/MSP and Pd, I will simply call them Max.

OVERVIEW

Chapter 1 presents the mechanics of building Max/MSP externals on Mac OS X and Windows and building Pd externals on Linux, Mac OS X, and Windows. Chapter 2 develops key concepts for writing audio signal processing in C with a focus on signal vectors. Chapter 3 works through the complete design and coding process for a simple audio object that multiplies two input signals together.

Chapter 4 develops a variable delay object with feedback and, in the process, shows how to overcome the lower limit on delay with feedback imposed by the Max signal vector size. Chapter 5 develops a wavetable lookup oscillator that can generate different waveforms internally and transition smoothly between one waveform and the next. Chapter 6 develops a sequencer with sample-accurate timing, obtaining greater temporal accuracy than is possible with the Max event scheduler. Chapter 7 presents a non-real-time editor that operates on the contents of Max/MSP buffers and Pd arrays.

Chapter 8 focuses on spectral processing, first showing how to develop objects for use with Max/MSP's *pfft~* system and then showing how to replicate the functionality of *pfft~* in the Pd environment. Chapter 9 improves on the variable delay object developed in chapter 4, with a focus on pointer arithmetic and various optimization techniques. Chapter 10 covers attributes in Max/MSP and shows how to build attributes into the oscillator developed in chapter 5 for a more flexible user interface.

Chapter 11 discusses practical aspects of debugging, working through the process of fixing the code for a buggy Pd object. Chapter 12 discusses the process of porting a Csound unit generator to Max/MSP. Chapter 13 develops the design of an external that implements dynamic stochastic synthesis, a DSP algorithm invented by Iannis Xenakis. Chapter 14 discusses Max for Live and shows how to integrate some of the objects developed earlier in the book into a Live device. Chapter 15 puts the skills developed in this book into context and suggests directions for further exploration.

ACKNOWLEDGMENTS

I would like to thank Joshua Kit Clayton, Emmanuel Jourdan, Timothy Place, Miller Puckette, and David Zicarelli for answering innumerable questions about C coding for Max/MSP and Pd. Dr. Richard Boulanger provided the initial impetus for this project, and enthusiastically reviewed the writing during its early stages. Mark

Dolson, Jean Laroche, Gary Kendall, and Miller Puckette all lent a sympathetic ear as I tuned up the spectral processing chapter. Robert Henke provided helpful comments on the Max for Live chapter. The first readers of the full text, Peter Castine, Christopher Haworth, and Elizabeth Hoffman, all provided constant constructive feedback on the code, concepts, clarity, and correctness of the writing. James L. Zychowicz provided indispensible editorial guidance. I am deeply grateful for their assistance.

I would like to thank David Zicarelli for providing the foreword and Miller Puckette for providing the afterword to this book. It is fitting that the voices of these master coders feature prominently in a text which builds upon their foundational work as the creators of Max/MSP and Pd.

I would like to thank the communities of Max/MSP and Pd users for their interest in my own externals over the years. These communities convinced me many times over of the value of sharing software and thereby provided the strongest motivation for writing this book. I would also like to thank all of my colleagues in the School of Creative Arts at Queen's University Belfast for contributing to the remarkably supportive and creative environment in which this project was brought to completion.

Eric Lyon
Belfast, 2012