Extracted from:

Programming Sound with Pure Data

Make Your Apps Come Alive with Dynamic Audio

This PDF file contains pages extracted from *Programming Sound with Pure Data*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit http://www.pragprog.com.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2014 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Programming Sound with Pure Data

Make Your Apps Come Alive with Dynamic Audio

Tony Hillerson

Edited by Jacquelyn Carter

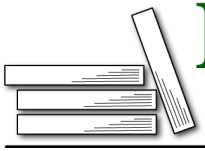
Programming Sound with Pure Data

Make Your Apps Come Alive with Dynamic Audio

Tony Hillerson

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Jacquelyn Carter (editor)
Potomac Indexing, LLC (indexer)
Candace Cunningham (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Copyright © 2014 The Pragmatic Programmers, LLC.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.
ISBN-13: 978-1-93778-566-6
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—January, 2014

CHAPTER 1

Introduction

This is a book about programming sound. Just as in any other programming book, in this book we'll cover the technical skills and tools that will enable you to tell a computer how to do something—namely, make sound. But just as with any other programming language, good design is important. I don't mean visual design; I mean careful thought and practice concerning "how things work." So I don't want you to think about yourself as a sound *programmer*; I want you to think of yourself as a sound *designer*.

Sound design is a practical art. Sound designers draw on an understanding of physical phenomena, technical knowledge, and intuition to create sound experiences. This book is about giving you the technical knowledge and providing some practical examples that will help you grow your understanding of how sound works, which will enable you to design and program the sound you want to hear in the digital experiences you create.

I have a theory: since humans rely primarily on their visual sense—way more than the other senses, in fact—experiences that involve the other senses have a greater chance of creating an unexpectedly good experience. Since the visual part of most digital experiences is so prevalent, it's surprisingly more realistic when the other senses are involved.

Digital experiences aren't heavily tactile yet, besides the mundane input methods such as typing or mousing, or maybe playing on a game controller shaped like a guitar. There are interesting new input methods like the Nintendo Wii Remote, Microsoft Kinect, Leap Motion device, and so on, but there's nothing mainstream that provides tactile feedback. And since Smell-O-Vision has yet to take off,[1] the audible experience is the best field within which we

1. http://en.wikipedia.org/wiki/Smell-O-Vision

• Click HERE to purchase this book now. discuss

can create that extra, sensory magic in addition to the visual to immerse the user in the experience.

Before we talk about Pure Data, the language and tools we'll be working with throughout the book, let's discuss a little about sound design and what it means. Then we'll take a look at Pure Data from a high level to get oriented before we dive in and start using it to make sound.

Getting Started with Sound Design

Let's talk a little about sound design. This is a general overview to give you an idea of the scope of the field, and to provide a context for the more technical chapters to follow.

The Sound Designer's Goals

The digital sound designer may have any of a number of goals when creating a sound. Sometimes these goals overlap, but in general this is what we want to achieve in an experience using sound.

Adding Audible Feedback

Adding audible feedback is a common use of sound in digital experiences. Such feedback could be a clicking sound when the user presses a button, a beep or bell tone when a task is complete, or a notification when a message arrives.

Fulfilling Expectation

When some event takes place onscreen, it can carry with it the expectation that if this event happened in the *real world*, a sound would be part of the event. In a game, a weapon is fired, a rock falls to the ground, or something is stretched or struck or thrown. The user expects things to sound a certain way depending on a lot of factors, such as the realism of the experience and conventions for similar experiences. Gauging these expectations and designing to meet them are goals for a sound designer.

Communicating a Mood

Background music in a game is a clear example of sound for communicating a mood, but it needn't be only in a game. Think of the startup sound of your computer system of choice. The Windows and Mac startup sounds are designed to signal an event, but also set the mood for stepping into a fresh, clean session.

Creating Immersion

Immersion may be related to communicating a mood or fulfilling expectations, such as when getting ambient environment sounds right in a game, but the

• Click HERE to purchase this book now. discuss

goal of immersing the user goes beyond those to creating a believable, emotionally involving experience. The goal is to make the user forget about the outside world.

Prompting an Emotional Response

Creating an emotional response is related to immersion, but I call it out separately because there may be a goal to quickly get a response from the user, maybe with a musical stab, or the scream of a monster from an unexpected direction, or a sound cue signaling the successful completion of a task.

Types of Sound

Now that you have some possible goals in mind, let's discuss the kinds of sounds you may use to reach these goals.

Music

Music is an extremely powerful type of sound. The human response to music is a mystical one, and even if your interest lies more in creating other types of sound effects, gaining an understanding of music will help. This book will not cover music directly, in terms of musical theory or creating musical instruments, but a lot of the skills we discuss can be applied to musical applications.

Ambience

Ambient sounds are those that occur in the background. These sounds don't have to be continuous, but they often are. The effects are there to help with creating an immersive environment or to fulfill the expectation of a certain situation. If in a game there's a scene where the wind is blowing, the user will expect to hear wind, for instance.

Effects

Non-environmental effects, sometimes called *hard effects*, are those that occur in conjunction with an event in the foreground, or somewhere the user's attention is expected to be. Such effects include a button click, a door slam, and a laser blast.

The Sound Designer's Methods

Given those categories for effects, let's look at how sound can be created.

Sampling

The easiest way to get a realistic sound is to record it. This is called *sampling*. Using a digital recorder out in the field to record the sound of a stream or a jet flying overhead is a reliable way to capture a sound. Sampling can be done

• Click HERE to purchase this book now. discuss

in a sound studio, too, which offers more control over the environment the sound is recorded in, but places limits on what kind of sound can be captured. You can see the tradeoffs between the two.

Sample Library

Recording your own samples can require a fairly extensive investment in space, time, and equipment. If you've done any work with sound in apps in the past, you've probably cut out a lot of work and bought a prerecorded sample library. There's a number of these professional sound libraries out there for different budgets, and a number of online services offer various samples a la carte.

Synthesis

Sound synthesis is the process of constructing sound from fundamental sound components—*sound waves*. It really is surprising how realistic sounds can be when built up from generated sound waves. The complex part isn't finding the right sound or environment in the real world, or creating a plausible situation in a studio, but rather understanding how sound is made well enough to be able to re-create the sound from scratch using a synthesis environment. This technique is this book's focus.

Building Sounds

The techniques described in the preceding sections are all time-tested and approved. I'm not making any claims that one is the right one—a mixed approach could be the best, and the needs and budget of each project will make one or more of them make sense and exclude others. This book is primarily about the exciting possibilities digital synthesis opens to sound designers, with the ability to both build sounds from scratch and make use of a sound engine inside the digital experience.

The biggest gain is that of extreme flexibility. If you need the sound of many different types of explosions, you could try to record many different things blowing up. If you instead create a sufficiently complex model of an explosion in a sound-generation application, you can create an infinite number of variations. It's even better if that model can be embedded in your application and react to different parameters controlling the explosion sound—much more useful than a directory full of explosion samples.

This approach mixes well with the other techniques I've described because the samples can be manipulated and controlled in the synthesis environment. That gives new life to your sample library and gives you the option to start

• Click HERE to purchase this book now. discuss

with a sample that may be hard to synthesize. A few synthesis applications are already out there, with a varying amount of complexity, power, and price. This book focuses on a popular application called Pure Data, which is open source, free, stable, and very powerful.

In preparation for jumping in and making sound, let's take a high-level look at what kind of software Pure Data is and how you interact with it.

Introducing Pure Data

Pure Data, or Pd, as its users call it, is an open source, visual programming environment for building audio and visual experiences. It was created in the 1990s by Miller Puckette, and grew out of the ideas behind a previous creation of Miller's called Max, which is still available as a commercial product. The Pure Data website is http://puredata.info.

Pd is part of a class of programming languages and environments that deal with *procedural audio*, or creating sound from routines that generate or modify streams of numbers that will eventually be sent to hardware to produce audio that we can hear.

Pd Is Visual

Pd is a visual programming environment, which means while using it you don't write code as such, but instead manipulate visual objects on the screen, connecting them into a system that produces some desired effect. The visual objects are technically called *atoms*, but we'll use more specific names for them as we talk about them. Each atom has a particular job, whether to send a message to other atoms, hold a number, or configure and control an object from a core library of objects or third-party extensions and abstractions.

A Pd file is called a *patch* and has the extension `.pd`. A patch is a textual data file containing information about how atoms are connected, how they're configured, and sometimes data that they can read from.

When we open a patch in Pd a graphical representation of the patch is displayed in its own window. Atoms can be moved around the screen, connections can be made, values can be adjusted, and so forth. The work area inside the window is called a *canvas*. All Pd programming happens in this visual environment. We'll spend a lot of time going through patches, often looking at images directly taken from Pd patches. Early in the book we'll walk through building patches from scratch, and later we'll switch to a more descriptive, explanatory style, but remember that all the patches described in the book are in the code download in the `pd` directory.

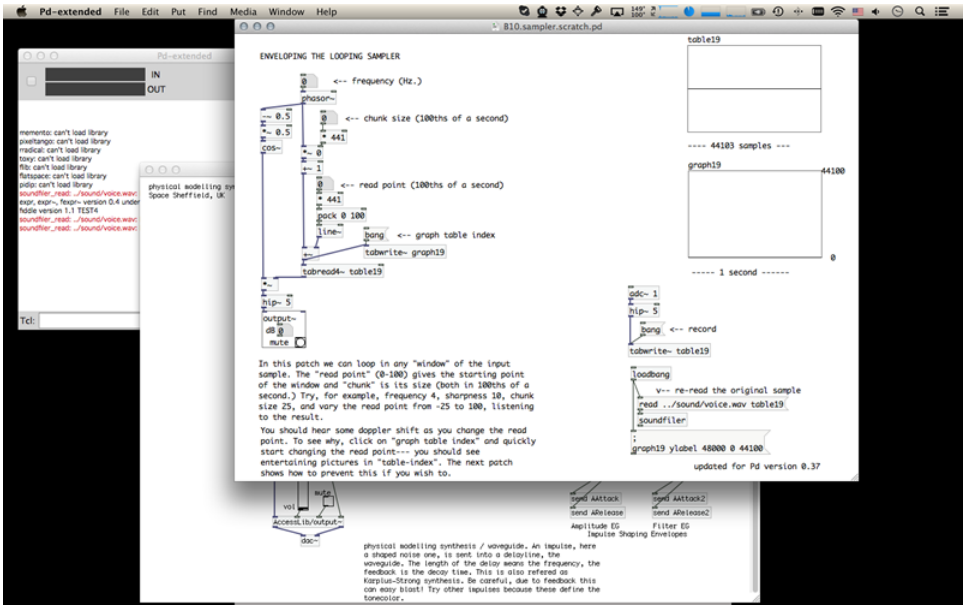
• Click HERE to purchase this book now. discuss

Joe asks:
Can I Edit Patches or Put Them in Source Control?

Although patches are simply text files, It would be a bit of a stretch to call them human-editable. They should be viewed as data files.

Merging changes with those of a colleague is most likely impossible, so to work with Pd in a team environment, either have Git treat Pd files as binary files or deal with conflicts in another way. Making good use of abstractions will help here. Refer to Git's documentation or that of your favorite source-control program for more information.

One of the great things about Pd's visual design is that the graph shown in the patch (as in the figure here) is everything you need to know about how to reproduce the patch and understand the flow of the audio through the system. The information density of a picture of a patch is very high, and just looking at it makes explaining the patch's inner workings much easier than if we had only code to look at.



Pd Is Modular

Pd is great for learning how to build sounds because the visual programming environment makes it easy to communicate what's going on. It continues to be a powerful tool long after you start being productive, too, thanks to its

modular design. It's easy to create abstractions at whatever level of complexity you need. These abstractions can themselves contain abstractions and so on, allowing the Pd programmer to gain a lot of control over and reusability of component pieces of a patch. The abstractions can be stored in a patch or in a separate file.

If Pd doesn't do something you need it to, you can write extensions in C. There are sound-processing extensions, visual libraries for art installations, interfaces between popular hardware controllers, and awesome stuff like Arduino and Raspberry Pi integration.

Pd Is Embeddable

Since in this book we're focused mostly on audio for web and native applications, one of the coolest things we'll do with Pd is embed Pd patches in native apps using the excellent libpd.[2] This will allow us to put dynamic, procedural audio directly into our apps and build a domain-specific language around the audio experience. When I first heard that was possible I nearly jumped up and danced. Not a pretty sight. That means Pd is a ready-made embeddable synthesizer engine for your native apps!

So, although Pure Data is only one of many ways to create and produce sounds that you design, it's a very powerful and useful tool to have in your sound-design tool belt.

Installing Pd

You can find installation instructions at the Pure Data website,[3] but which version should you install? There are two: the original version maintained by Miller, lovingly called Pd Vanilla, and Pd Extended. Pd Extended is easier to install, and comes with a number of third-party extensions, including sound-processing tools and effects. Patches created with extensions from Pd Extended will not run in Pd Vanilla unless the extensions are added to Pd's Vanilla's path, but otherwise they are almost completely compatible. As of this writing Pd Vanilla was at version 0.44, with 0.45 right around the corner. Pd Extended was using Pd 0.43 under the hood.

All of the patches in this book were created with Pd Extended, but I've kept away from any extensions that don't ship with Pd Vanilla—with one notable exception, the `output~` object. I'll make note of it again when we first use it.

2. http://libpd.cc/
3. http://puredata.info

• Click HERE to purchase this book now. discuss

If you choose to use Pd Vanilla, there should be no issue with the patches in this book. If you use any of the third-party extensions that ship with Pd Extended, keep in mind that libpd, which we use to embed Pd in native apps, is a wrapper around Pd Vanilla. You'll need to track down and include any extensions you use alongside your patch.

Other Software

It's not necessary to have any other software for this book, but it might be useful. In a few cases I've included screenshots showing some audio-file analysis in my favorite audio editor, Adobe Audition.[4] A free, open source alternative is Audacity.[5] You're definitely going to want a full-featured audio editor, and either of these is a good choice.

In the final chapters of the book we'll go through two projects: a web game and a task-management app built for Android and iOS. Although you could learn a lot from the discussion in these chapters covering the Pure Data part alone, to get the most from the code you'll need a good text editor for the web code, Eclipse for the Android project,[6] or Xcode for the iOS project.[7]

Let's Get Started

You're beginning (or perhaps continuing) an exciting journey toward being able to design the sound you want to hear, which will give realism, emotional depth, and that extra dimension to make your apps and games come alive.

Joe asks:

What's That Again?

Sound is an immense field. Unbelievably immense. This book is meant to be a practical, hands-on primer. I try to explain enough theoretical background to help make sense of what's going on when we come across a new concept, especially in the next chapter. If the theoretical parts leave you confused or asking more questions, use them as jumping-off points for further study—Wikipedia is a great place to start.

Now that you have a general idea of what we want to accomplish in this book, what the sound designer's goals are, and how we can use Pure Data to accomplish them, let's get started with Pd and make some noise.

4. http://www.adobe.com/products/audition.html
5. http://audacity.sourceforge.net/
6. http://eclipse.org
7. http://developer.apple.com

• Click HERE to purchase this book now. discuss