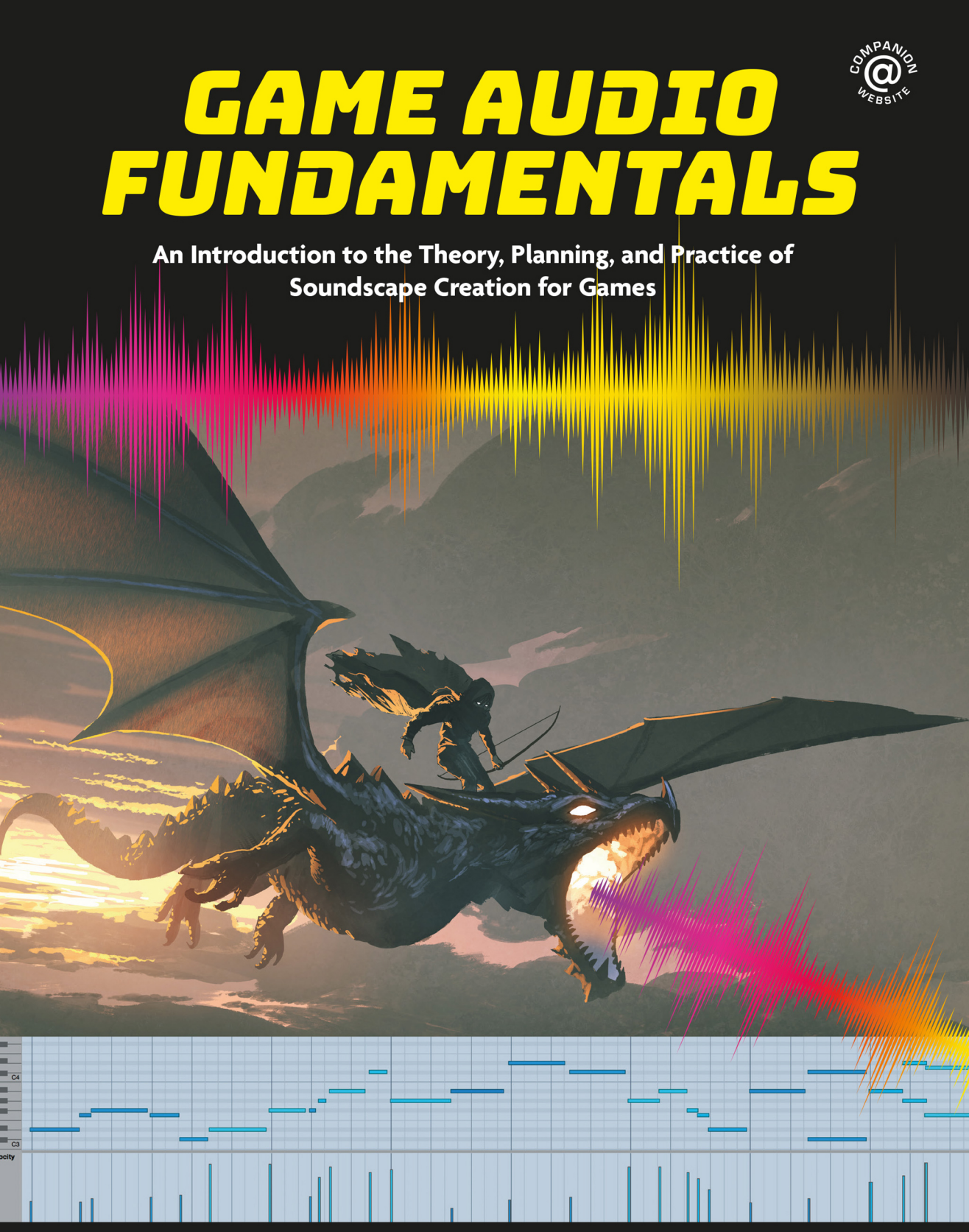
GAME AUDIO FUNDAMENTALS

COMPANION @ WEBSITE

An Introduction to the Theory, Planning, and Practice of Soundscape Creation for Games



A Focal Press Book

KEITH ZIZZA

ROUTLEDGE

Game Audio Fundamentals

"With his *Game Audio Fundamentals*, Keith Zizza has delivered a solid, hands-on book that touches on all the important aspects of creating sound and music for video games!"

— Brian Schmidt, *Game Composer/Sound Designer, Executive Director of GameSoundCon*

Game Audio Fundamentals takes the reader on a journey through game audio design: from analog and digital audio basics to the art and execution of sound effects, soundtracks, and voice production, as well as learning how to make sense of a truly effective soundscape.

Presuming no pre-existing knowledge, this accessible guide is accompanied by online resources – including practical examples and incremental DAW exercises – and presents the theory and practice of game audio in detail, and in a format anyone can understand.

This is essential reading for any aspiring game audio designer, as well as students and professionals from a range of backgrounds, including music, audio engineering, and game design.

Keith Zizza is a game audio instructor and lecturer at Worcester Polytechnic Institute, with over 25 years of professional experience as a game audio designer for studios and publishers such as SEGA, Marvel, EA, and Ubisoft.

Game Audio Fundamentals

An Introduction to the Theory, Planning, and Practice of Soundscape Creation for Games

Keith Zizza

Routledge
Taylor & Francis Group
LONDON AND NEW YORK

Designed cover image: Tithi Luadthong/Shutterstock.com

First published 2024
by Routledge
4 Park Square, Milton Park, Abingdon, Oxon OX14 4RN

and by Routledge
605 Third Avenue, New York, NY 10158

Routledge is an imprint of the Taylor & Francis Group, an informa business

© 2024 Keith Zizza

The right of Keith Zizza to be identified as author of this work has been asserted in accordance with sections 77 and 78 of the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this book may be reprinted or reproduced or utilised in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage or retrieval system, without permission in writing from the publishers.

Trademark notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data
Names: Zizza, Keith, author.
Title: Game audio fundamentals: an introduction to the theory, planning, and practice of soundscape creation for games / Keith Zizza.
Description: Abingdon, Oxon; New York, NY: Routledge, 2023. | Includes bibliographical references and index.
Identifiers: LCCN 2022061195 (print) | LCCN 2022061196 (ebook) | ISBN 9781032111964 (hardback) | ISBN 9781032111957 (paperback) | ISBN 9781003218821 (ebook)
Subjects: LCSH: Video games—Sound effects. | Computer sound processing. | Computer game music.
Classification: LCC GV1469.34.S68 Z59 2023 (print) | LCC GV1469.34.S68 (ebook) | DDC 781.5/4—dc23/eng/20230407
LC record available at https://lccn.loc.gov/2022061195
LC ebook record available at https://lccn.loc.gov/2022061196

ISBN: 978-1-032-11196-4 (hbk)
ISBN: 978-1-032-11195-7 (pbk)
ISBN: 978-1-003-21882-1 (ebk)

DOI: 10.4324/9781003218821

Typeset in Sabon
by codeMantra

Access the companion website: www.keithzizza.net

For Corinne, Talia, and Edward


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

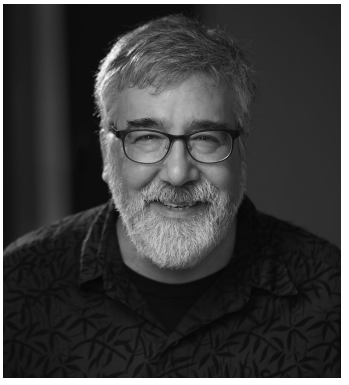
Acknowledgments

There are so many details in putting together a book such as this one, and there are a number of people to thank for helping to make it a reality.

Audrey Tang and Gabriel Whyel at Blue; Ed Tsang at BSI Group; McGraw-Hill and Ken Pohlmann; David Via at ZOOM North America; Wesley DeVore at PreSonus; Martha Vanin at Novation; Bryan Borcherds at Arturia; William Branch at TASCAM; and Nick Fournel at Tsugi; Justin Frankel at Cockos Ltd., makers of REAPER software; Geoffrey Francis, author of *Up and Running: A REAPER User Guide*, for his kind permission in the usage of screenshots, tables, and icons; Jamie Campbell and Shainiel Deo at Halfbrick; Wilbert Roget II, Joanna Fang, Trevor Dikes, and D.B. Cooper for their generous interviewing time; Patrick Brigham for his artistic wizardry; Sharona Jacobs for her top-notch photography; Joel Schwelling, my "sonic mentor" who got me started on this crazy adventure; all of my excellent IMGD colleagues and students at Worcester Polytechnic Institute; the folks at Routledge, namely my editorial assistants Emily Tagg and Adam Woods; production editor Aimée Crickmore; Assunta Petrone and the team at Codemantra; and to my editor Hannah Rowe for believing in this project!

And once again Corinne, Talia, and Edward – for believing in me, and for putting up with my late-night writing sessions for the past two years.

Preface



Keith Zizza

Welcome to the Exciting World of Game Audio!

You have chosen a fantastic time to get acquainted with game audio design. The videogame industry is seeing record growth, audio opportunities are increasing, and exciting new audio software and hardware tools are available to create the soundscape you've always dreamed of. The good news is, we have finally arrived at the point where you don't need to have a million-dollar, or even a thousand-dollar setup, to make great audio. Where we used to have to collect racks of expensive gear, we can now do quite a bit with just a laptop and headphones.

All games deserve a great soundscape, regardless of their complexity level. What matters is that the audio serves the game well and enhances the player's experience. If we can suspend their disbelief and transport them further into the game's world, then job done!

I hope you share the passion that I do for game audio, and that you're eager to see (and hear!) what's possible for audio in the context of games and interactive media. I love the entire process of audio design: everything from sound effects creation to music composition, dialogue production, and implementation. There is nothing more satisfying than hearing all of your hard work come to life in the game you've worked on for months, or in some cases, years. When it's done right, it's worth every second of the wait.

About This Book

If you're a student of interactive media or game development (in any discipline), an aspiring game audio professional, or a developer curious about the subject, then look no further – this book is for you! Here you will find a comprehensive but very practical look at game audio development. In these chapters, I will share my 25+ years of game audio experience with you: everything from audio theory to setting up your studio to critical listening, soundscape planning, and implementation. There are even some DAW exercises for you on the **Companion Website** (what's a DAW, you ask? The answer is in Chapter 5!), including other supplemental material.

Now more than ever, our time is a precious resource. Discussing topics in game audio can trail off on all kinds of wild and exciting tangents. So here, I attempt to cover only the most relevant material with no "fluff" in between, distilling the information down to its essential elements for you.

How This Book Came to Be

There are some excellent books on game audio available, but over the years I have not yet come across one that covers the disciplines of sound design, music, and dialogue equitably, and also contains vital (but easily understandable) introductions to acoustics, digital audio, and music theory for audio designers. After more than a decade of teaching game audio, I thought I'd just go ahead and write it myself!

I also wanted to create a book that contained the audio production process, soundscape listening and troubleshooting, and some resources on finding work. I've been asked so many times by students and professionals alike, "how can I get a job in game audio?" to which there is no short answer. Hopefully, though, I've covered all the basics in the final chapter.

Reader Audience

I wrote this book for a few people.

- **The interactive media or game development student**: Perhaps you are a student aspiring to be an audio designer. But you may also be on an artist or animation track, or learning to be a programmer, writer, designer, or producer. You'll be working with audio folks in one capacity or another as part of a development team, so I've tried to include everything you'd ever need to know here.
- **The rising audio designer**: You are just starting out or have a few audio gig feathers in your cap already. Here you can take a deeper look at the profession, in theory, production, and practice.
- **The game developer**: You are someone who works with one or more audio designers and wishes to know more about the process in general. This book is a great choice, as the knowledge here should help facilitate better communication between you and your sonically minded teammates.

Overview of Material

The Introduction begins with a brief history of game audio, followed by game audio in education and the rising importance of game audio within the game industry itself.

Part I is all about theory.

- Chapters 1–3 are primers on acoustics, digital audio, and music theory for audio designers.
- Chapter 4 discusses game audio tools of the trade, both hardware and software.
- Chapters 5 and 6 are all about DAW essentials: editing, mixing, mastering, rendering, and plug-ins, among other topics.

Part II is comprised of soundscapes and disciplines.

- Chapter 7 introduces soundscapes for games, what their integral components are (sound, music, and dialogue), and some critical listening exercises to better help assess them.
- Chapters 8–10 take a comprehensive look at the three soundscape disciplines: sound design, music, and dialogue, and their subdisciplines and production methods.
- Chapter 11 focuses on the administrative and production sides of game audio: working with a game development studio, how the audio designer interfaces with various departments, audio roles, working with the team, and how kindness and humor are mandatory!

Part III puts this knowledge into practice.

- Chapter 12 discusses implementation: in achieving your sonic vision, all elements must be integrated with great care in order to have a compelling soundscape. Middleware, real-time mixing, and optimization are covered in detail here.
- Chapter 13 concerns the art of listening: soundscapes from several game genres are reviewed, along with the player's audio expectations. As seen in Chapter 5, the soundscape is deconstructed but this time for testing, where aesthetic and technical problems are examined.
- Last but certainly not least, Chapter 14 is about finding audio work in the industry. The competition is fierce, but having a solid portfolio, networking skills, perseverance, and a positive attitude will help pave the road to your ultimate success.

Learning Outcomes

My hope is that after reading this book, you will become enlightened to the world of game audio design and production, and gain a heightened awareness of the audio designer's high level of commitment, skill, and enthusiasm. I also hope that reading this book gives you a deeper understanding of how musicians write compelling interactive scores, how sound designers craft such amazing effects, and how dialogue editors work their magic making people, creatures, and other nonhuman imaginings sound spot-on.

So, find a cozy spot to read, grab your favorite beverage (just not over the mixing console, please!), and we'll be on our way!

Introduction

Highlights in Game Audio History

The Arcade Explosion

Our journey begins among the games in coin-operated arcades. Up until 1971, the only sounds heard from arcade machines were a cacophony of bells, mechanical whirrs and clicks, and the sounds of wood and iron plates being struck by metal solenoids (Figure I.1).[1] No one could have ever imagined the electronic tones and noises a game could emit, never mind that it was from a *videogame*!

Enter *Computer Space*, which was the first commercially released videogame in 1971. It generated tones and noises for the sounds of your rocket battling against a pair of flying saucers.[2] The electronic audio was primitive by today's standards, but being the first game of its kind, *Computer Space* sounded like nothing else (Figure I.2).

Then, in 1972 came Atari's hit game *Pong*, an electronic version of ping pong. It utilized three sounds: a lower tone when the ball bounced off the top or bottom of the screen, a higher tone when a player successfully hit the ball, and a buzzlike tone when a player scored a point.[3] Other videogame manufacturers soon emerged with entries such as *Speedway*, *Gun Fight*, and other *Pong*-like clones, replete with simple tones and noises.

By the mid-1970s, the technology behind arcade games advanced more rapidly with the changeover from transistor- to microprocessor-based hardware. In 1978, Taito released *Space Invaders*, a blockbuster hit which surpassed *Pong*'s popularity (and its total revenue). Rather than the three tones heard in *Pong*, it had a continuous four-note soundtrack along with an impressive set of sound effects (Figure I.3). It ushered in the Golden Age of videogames, and an explosion of innovative titles soon followed.[4]

Popular games such as *Galaxian* (1979), *Pac-Man* (1980), and *Donkey Kong* (1981) added multichannel sound processors ("multichannel" meaning more than one tone or noise that is generated at once), and onboard music synthesizers to enhance the gameplay and provide feedback to the player.[5] Rounding out the auditory experience, games such as *Space Fury* (1981), *Q*Bert* (1982), and *Gauntlet* (1985) made use of voice synthesis. The 1990s and beyond yielded hits such as *Mortal Kombat* (1992), *NBA Jam* (1993), and *Crazy Taxi* (1999), all which incorporated digital audio playback and increased audio quality.

DOI: 10.4324/9781003218821-1

Figure I.1 Computer Space, the first commercially released videogame in 1971 (Flippers/Public domain).

Home Consoles

Atari's *Pong* made it to the home market in 1975, as did *Pong*-like clones such as Coleco's Telstar (1976). Soon after, the Atari 2600 (1977) became the leading home console. Its audio processor controlled two audio channels, one for tone and one for noise-based sounds such as gunfire, explosions, and percussion. During these early years (and even into the 1980s), the programmer was also required to be the game designer, artist, and audio designer, and they would have to create a game with as little as *4 kilobytes* to work with (!), which included any music and sound playback.[6]

In the mid-1980s, the Nintendo Entertainment System (NES, 1985) took the home console industry by storm with its expertly crafted games, exciting visuals, and five-channel audio capability. For the first time, fully arranged music and a wide array of sound effects could exist together for a rich audio experience (Figure I.4).

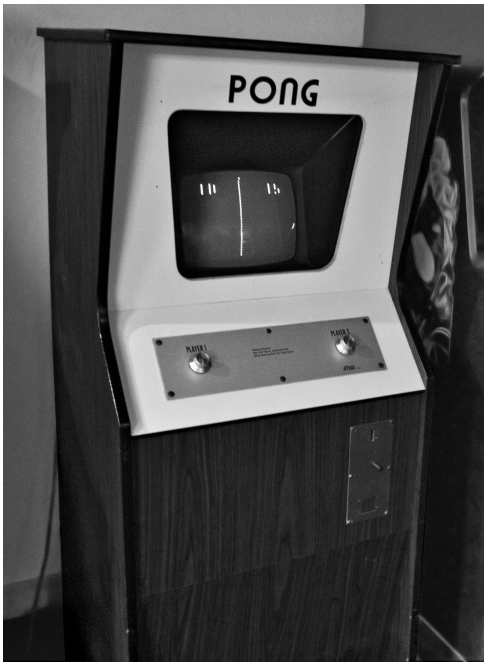


Figure 1.2 Atari's Pong, released in 1972 (Rob Boudon/CC BY-SA 4.0).

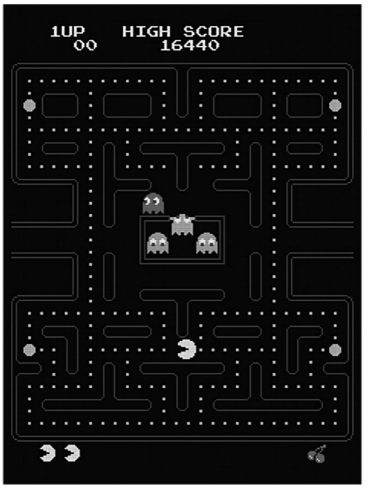


Figure 1.3 Arcade hits such as Pac-Man utilized multichannel audio, which included sound effects and music playing simultaneously (CountingPine/Fair Use).

Figure I.4 Home consoles through the years: The Atari 2600 (1977, left), the Nintendo Entertainment System (1985, center), and the SEGA Genesis (1989, right), each with increasing levels of sonic sophistication: two channels, five channels, and ten channels of audio, respectively (Evan-Amos/Public domain).

Throughout the late 1980s and early 1990s, other home systems cemented their place in history starting with the SEGA Genesis (1989) and the Super Nintendo Entertainment System (SNES, 1991). With each new generation of gaming console, the processing power increased – which for audio meant more channels and better sound processors. By the mid-1990s, the leading competitors in the home market were Nintendo, SEGA, and Sony with its debut console, the PlayStation (PS1, 1995). The PS1 offered a groundbreaking 512K of dedicated audio RAM and 24 channels of digital audio.

In 2001, Microsoft entered the console arena as a legitimate contender with the Xbox (2001), delivering a whopping 256 channels of audio and a dedicated audio processing chip (the Nintendo GameCube, also released in 2001, had 64 audio channels).

As of this writing, the "big three" today are the Sony PlayStation 5 (2020), Xbox Series X/S (2020), and the Nintendo Switch (2016). All offer high-fidelity audio including surround-sound capability.

Home Computers

From the late 1970s onward, personal computers began to rise in popularity due to mass market campaigns and reduced manufacturing costs (Figure I.5). Many companies such as Apple, IBM, Atari, and Commodore were all competing for market share, and innovations were made with every new product release. Though their primary function was productivity, it didn't take long for computer games to become a favorite pastime. Sound generation first consisted of speaker clicks, simple tones and noise, but by 1979, several computers such as the Commodore 64 and Atari 800 sported their own proprietary sound processors, with three and four channels of audio, respectively. But across all brands, the consumer could not modify or upgrade the audio hardware.

It wasn't until 1988 when the first sound hardware upgrades emerged (from two companies, AdLib and Creative Labs): a "sound card" expansion that could be added to IBM PC-compatible machines, both which added stereo digital audio and multichannel music synthesis. Game publishers such as Sierra On-Line and Lucasfilm Games were among the first to support the use of sound cards, and widespread industry support soon followed (Figure I.6).

Today, PC-based sound cards are still available for stereo and surround-sound experiences, though the built-in audio capabilities on a PC without any card is more

Figure 1.5 Two rival home computers: The Commodore 64 (left, 1982) and the Atari 800 (1979, right), both with their own proprietary sound processors (Evan-Amos/Public domain).

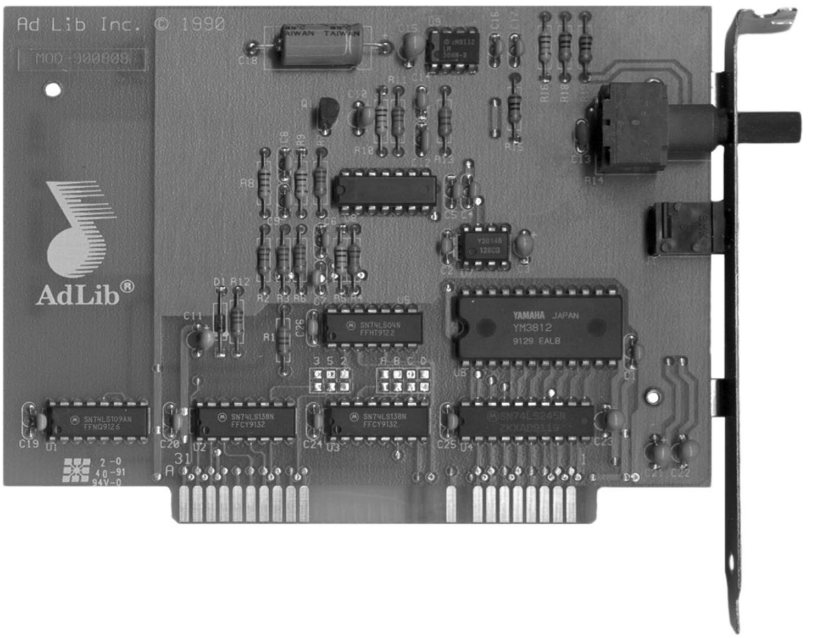


Figure 1.6 The AdLib Sound Card for IBM PC-compatibles (1990). Note the rear volume knob on the upper right (Wikimedia/Ryanicus Girraficus/CC-BY-SA 2.5).

than adequate for consumer-end use. There are now also external hardware audio interfaces for PC and Apple computers for professional use, which we'll see more of in Chapter 4.

The Arrival of MIDI

In 1983, the Musical Instrument Digital Interface protocol, aka MIDI, was standardized. This is a communications protocol that allows computers, electronic instruments, and other device to share data with one another. Rather than send digital audio back

Figure I.7 The Nintendo Game Boy (1989, left), and the Sony PlayStation Portable (2005, right) (Evan-Amos/Public domain).

and forth, which would take a lot of bandwidth, only the information about an instrument or function is sent (such as playing a note, its volume, and length). Sound cards can take advantage of this by reading MIDI-based music data and playing the instruments on their onboard synthesizer chip, for example. Now, soundtracks could be heard on PCs, and sound reasonably similar between the various models of available sound cards.

Handhelds

Loved for their lack of cables and small size, the universe of handheld gaming has always had a special place in the gamer's heart (Figure I.7). In general, their audio, video, and CPU capabilities are inferior to consoles and PCs, due to their smaller footprint, lower-cost components, and convenience of portability.[7]

Mattel is recognized as the first retailer of handhelds, debuting in 1976 with *Auto Race*, which sported a single beep for successfully passing cars (LEDs) and a variable noise generator for engine sounds. Monophonic (1-channel) capability was the norm until Nintendo's Game Boy arrived in 1989, having four audio channels (two tones, one noise, and one programmable), and stereo capability through external speakers or headphones.

Other notable entries in handheld audio history include the Nintendo DS (2004) with 16 channels and built-in stereo speakers; the Sony PlayStation Portable (2004) with 32 channels and special effects; the Sony PS Vita (2011); and the previously introduced Nintendo Switch (2016), both with generous audio capabilities. The Switch can be used as a console *or* a handheld, simply by plugging it into a TV docking station, or by taking it with you! The Switch Lite was introduced in 2019, which is handheld-only.

Mobile and Tablet Games

Games for mobile devices have existed ever since the inclusion of LCD screens to view them on. But these games were often designed to work only with specific brands

Figure I.8 Angry Birds, as played on an iPhone (Rovio/Screenshot by Keith Zizza).

(Nokia, Verizon, etc.), or in limited geographical markets, making it difficult to attain widespread success of any one title or product. That would change in 2005 with the release of Apple's first iPhone, which became a smash success thanks to its ease of use and accessibility. It changed the way people function in society due to its easy access to the Internet, and of course the ability to play games wherever one goes.[8] With competing mobile devices running on the Android platform, there are now more than one million games available between the two.[9] The best phones and tablets can support stereo audio, and the quality of the audio experience (e.g., number of available audio channels) is scalable based on the device's processing power, memory, and operating system (Figure I.8).

One key point to mention is that throughout the history of game audio and even today, sound designers and composers have always been constrained to the limitations of technology. Digital audio capability may exist, but the real estate on a Switch cartridge, or memory storage limits on a mobile phone, force them to get creative with audio implementation.

Audio's Ascent in the Video-Game Industry

Now more than ever, game audio is a thriving and growing discipline within the industry. There are many reasons why this is happening.

More High-Quality Playback Experiences

The popularity of videogames is at an all-time high today, and so too is the demand for high-quality audio. This is in part due to the surge in gamers worldwide, across all platforms, but also, thanks to COVID-19 keeping everyone locked down in 2020, gaming has gone through the proverbial roof, with sales more than doubling within

just eight years from $105 to $206 billion.[10] There are more audio options available to gaming consumers than ever before: games are enjoyed using traditional stereo or surround-sound speakers and also headphone-based spatial technologies such as Dolby Atmos™[11] and Sony's 360 Reality Audio.[12] Some of the biggest game releases include a full complement of orchestral soundtracks, custom sound effects, and many thousands of lines of dialogue.

More Internal Support from Development Teams

Where game audio was once created by a single individual in the 1970s and 1980s, audio production and implementation is now often delegated to several people. The blockbuster ("AAA") game titles employ multiple audio personnel including sound designers, composers, dialogue editors, and audio programmers, all under the supervision of an audio lead or director.

Rise in Career Opportunities

With the maturation of game audio as a whole, the number of jobs to create, code, and produce it have increased rapidly in just the past few years as well. Today, there is an especially high demand for audio designers and programmers, but also very specialized personnel such as technical audio designers (more on this in Chapter 11).

Emergence in Videogame and Interactive Media Programs

There are now hundreds of college-level and professional programs around the world for interactive media and game development study. Several colleges also offer undergraduate and graduate programs in game and/or interactive audio, allowing students to gain a competitive edge in the ever-competing market of game audio production.

Which Brings Us to This Book!

There's a little of everything in *Game Audio Fundamentals*, but that's because a game audio designer requires knowledge in many areas: the science of sound, digital audio, a bit of music theory, hardware and software tools, game audio disciplines, production, and implementation. I've put it all together here, and I hope you enjoy the journey!

Notes

1 Collins, Karen. "Game Sound in the Mechanical Arcades: An Audio Archaeology." Game Studies: The International Journal of Computer Game Research, October 2016. http://gamestudies.org/1601/articles/collins.
2 "Computer Space Arcade Game (1971, Nutting Associates)." YouTube. Old Classic Retro Gaming, April 8, 2010. https://www.youtube.com/watch?v=b3BQsCCwo8w.
3 "Original Atari Pong (1972) Arcade Machine Gameplay Video." YouTube. andys-arcade, December 11, 2014. https://www.youtube.com/watch?v=fiShX2pTz9A.
4 Gallagher, Jason M. "How Space Invaders Became a Gaming Phenomenon." Den of Geek, August 12, 2018. https://www.denofgeek.com/games/how-space-invaders-became-a-gaming-phenomenon/.

5 Pacman, in particular, had an innovative soundscape, with a multitude of auditory information provided to the player: https://gamerant.com/pac-man-bandai-namco-iconic-sound-design-arcade-flow-legacy/.
6 Demetrakopoulos, Petros. "Creating a Game for Atari 2600 in 2022." Medium. Geek Culture, July 12, 2022. https://medium.com/geekculture/creating-a-game-for-atari-2600-in-2022-a425cc1f4873.
7 With one exception being the Nintendo Switch, which functions as either console or handheld.
8 Karcz, Anthony. "10 Years with the IPhone: How Apple Changed Modern Society." Forbes. Forbes Magazine, January 10, 2017. https://www.forbes.com/sites/anthonykarcz/2017/01/09/apple-iphone-10-year-anniversary/?sh=34454c1f77d0.
9 Stojanovic, Milica. "Top 20 Key Mobile Game Statistics for 2022." PlayToday.co, July 16, 2022. https://playtoday.co/blog/stats/mobile-game-statistics/.
10 Williams, Lara. "A Pandemic Is a Dream Come True for Gamers." Bloomberg.com. Bloomberg, January 16, 2022. https://www.bloomberg.com/opinion/articles/2022-01-16/pandemics-boost-for-video-game-industry-is-a-dream-come-true-kyh9nekz.
11 "Gaming in Dolby Atmos." Dolby. Accessed July 29, 2022. https://www.dolby.com/experience/games/.
12 Sam, Raymond. "PS5 3D Audio-Everything You Need to Know." TheGamingSetup, April 1, 2022. https://thegamingsetup.com/ps5/buying-guides/3d-audio.

Bibliography

Collins, Karen. *Game Sound: An Introduction to the History, and Practice of Videogame Music and Sound Design*. Cambridge, MA: MIT Press, 2008.

Donovan, Tristan, and Richard Garriott. *Replay: The History of Videogames*. Lewes, East Sussex: Yellow Ant, 2010.

Marks, Aaron, and Jeannie Novak. *Game Audio Development*. Clifton Park, NY: Delmar Learning, 2009.

Stanton, Richard. *A Brief History of Videogames: From Atari to Virtual Reality*. London: Robinson, 2019.

Williams, Andrew. *History of Digital Games: Developments in Art, Design, and Interaction*. Boca Raton, FL; London; New York, NY: CRC Press, 2017.


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Part I

Theory and Tools


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 1

Acoustics and Analog Audio

So ... What Is Sound?

There's nothing quite like being outside on a crisp autumn morning. With the birds singing, the cool breeze making slight whispers, and the rustling of fallen leaves every now and again. Can you hear it? Imagine yourself outside on that pleasant day, listening to the sounds of nature all around you as you are sitting on a bench, next to a beautiful lake. Now that I've set the scene for you, let's find a nearby pebble and toss it in the water. You will see the pebble disrupt the surface, causing vibrations that create cascading ripples in the water (which is also quite satisfying to watch!).

The same basic principle applies to sound. When we hear the *splash* of the pebble, the tremors from that event create vibrations in the air – pressure waves that expand outward in all directions. That is **sound** in the simplest sense, a vibration in the air that reaches our ears (Figure 1.1).

Let's look at sound another way. If you play a drum with a mallet, it strikes the drum head causing it to vibrate. As the head moves inward, tiny air molecules in front of the head (inside the drum) become crowded together, creating a region of higher air pressure known as **compression**.[1] The head then bounces back in the opposite direction, and **rarefaction** occurs in front of the head, where the molecules have become spread out and the air pressure is slightly lower. The compression and rarefaction make up a single vibration, and we may hear several vibrations that become increasingly smaller in volume as the energy fades, until it is gone. We perceive these vibrations – alternating compressions and rarefactions that reach our ears – as **sound**. Between each compression and rarefaction, there is a state of **equilibrium**, which is a normal air pressure level with air molecules spread out more uniformly. Figure 1.2 shows what happens with a single, complete vibration of the drumhead.

The vibrating air molecules are bouncing off each other, but they do not actually travel with the sound. They are temporarily disrupted in the air, merely transporting the energy to adjacent molecules, then returning to their original location. Those adjacent molecules transfer the energy to their neighbors, and so on, in a chain reaction through the air. In this way, the energy is transported as a **wave** – just like the "wave" at your favorite stadium! In fact, the fundamental way energy travels in our world, and everywhere in the cosmos for that matter, is all through waves.

DOI: 10.4324/9781003218821-3

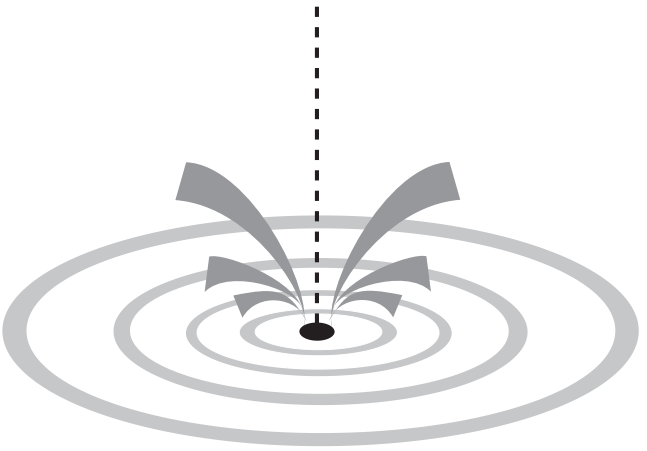


Figure 1.1 A pebble dropped into water, causing ripples to expand outward across the surface.

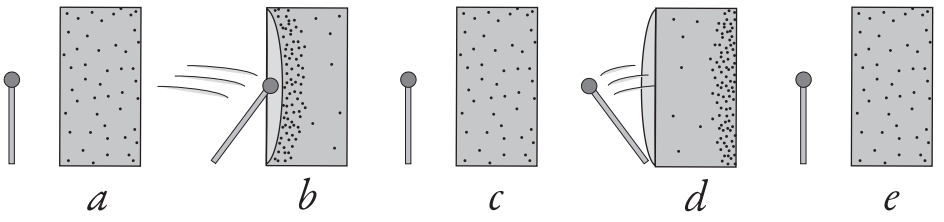


Figure 1.2 A single vibration of a drumhead. (a) Air pressure inside the drum is at **equilibrium** with the air pressure outside. (b) The initial strike of the drumhead causes **compression** (increased air pressure) in front of the head, where molecules are pressed together more closely. (c) The head bounces back at the point where air pressure is at equilibrium again. (d) The head continues past the point of equilibrium in the opposite direction, causing **rarefaction** (decreased air pressure) in front of the head, where molecules spread out from one another. (e) The head and air pressure return once again to equilibrium and the vibrations continue, diminishing until the energy runs out.

How Do We Hear Sound?

Sound is first received by the outer ear at the **auricle** (or pinna), a series of folds that help collect and amplify sound as it approaches the ear canal (Figure 1.3). Sound then travels to the **eardrum**, where it vibrates and sends that acoustical energy to the **ossicles** in the middle ear. There, three tiny bones called the **malleus** (also known as the "hammer"), the **incus** ("anvil"), and **stapes** ("stirrup") all work together, carrying and further amplifying the vibrations to the **cochlea** in the inner ear. Inside this coiled structure, thousands of specialized, tiny hairs called **cilia** move when receiving vibrations which they are specifically attuned to and send that information to the brain as electrical impulses which we perceive as sound.

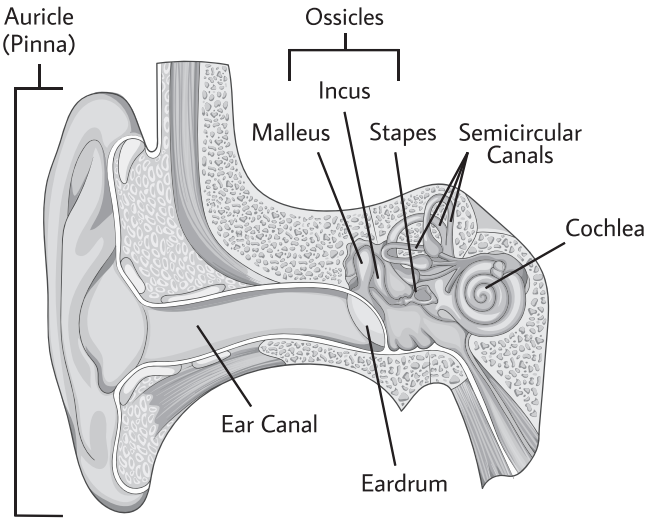


Figure 1.3 Basic anatomy of the ear.

Pitch and Loudness

Psychoacoustics, or the study of sonic perception, can be broadly categorized into two groups: **pitch**, which defines how high or low a vibration sounds to the listener, musically speaking, and **loudness**, which is a measurement of how loud or soft the vibration feels to the listener. We interpret all of the vibrations in our surroundings through detecting pitch and loudness, in a relatively broad spectrum which we will cover later in this chapter.

Measuring Sound

There are several ways to measure the characteristics of sound waves. It is worth noting that while this book is intentionally not math-intensive (insert the applause sound effect!), a few concepts will be covered in this section that introduce some basic mathematical formulas you can choose to examine.

Let's revisit sound waves once again, this time using a tuning fork. When the tuning fork is struck, it begins to vibrate. As the tines move back and forth, the surrounding air molecules are also set into vibrational motion. Compressions and rarefactions are created, with vibrations moving outward in all directions, gradually diminishing in strength until the energy is lost.

These compressions and rarefactions can be measured over time, on a graph. If the horizontal axis of the graph is *time*, and the vertical axis is *air pressure level*, a single vibration of the tuning fork might look like this (Figure 1.4):

Cycle and Period

This wave is an example of a sinusoidal, or **sine wave**, representing simple harmonic motion under perfect conditions (a vibrating tuning fork, in this case). The moment

Figure 1.4 Air pressure level over time. Compression is represented above normal pressure (positive values) and rarefaction below normal pressure (negative values).

before the vibration starts, the surrounding air particles are at rest. Looking at the vertical axis on the far left, the wave rests between + and –, which represents normal air pressure (zero change). As compression builds over time, the wave rises to its maximum level (or **crest**) and then descends, until it drops below normal air pressure and enters rarefaction. Upon reaching maximum rarefaction level (or **trough**), the wave air pressure rises up again until it reaches a state of rest once more. This "round trip" of a wave is called a **cycle**. For our purposes going forward, we will say a cycle is specifically measured on a graph as above, from ***rest → full compression → rest → full rarefaction → rest***. The measurement of the time in seconds it takes for a cycle to complete is called the **period**.

Transverse versus Longitudinal Views

It is very important to note that **air particles are not traveling in an S-shape!** We are simply graphing vibrational motion over time in a way that is easier to visualize. Sound waves are considered **longitudinal waves**, in that air particles move parallel with the wave motion. If you play a guitar and point it toward the far wall of a room,

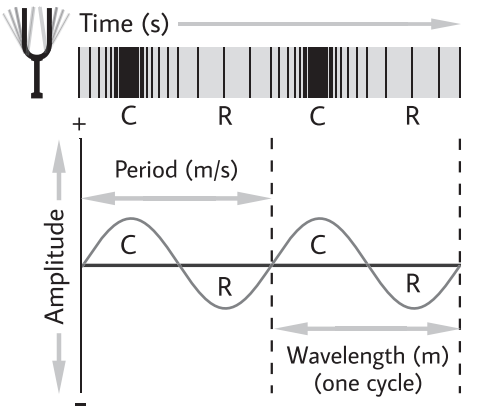


Figure 1.5 Playing a guitar and generating sound waves. The back-and-forth movement of the string causes vibrational energy in the air, creating an alternating pattern of compressions (the concentric gray circles) and rarefactions. Though the waves are shown here as a 2-D circular pattern, they expand outward spherically from the sound hole of the guitar.

the vibrational energy will head toward that far wall, with air particles vibrating in the same direction. The energy will be passed on longitudinally, while the waves are also expanding outward in all directions (spherically), as shown in Figure 1.5. It is also important to mention that air molecules don't travel from the source to your ears; they only travel a very short distance, as the energy from one vibrating molecule is passed on to the next adjacent one, then the next, and so on, in a ripple effect.

Our graph is viewing wave motion differently, as if it were a **transverse wave**, which means the wave drawn is *perpendicular* to the actual wave motion. Remember that rippling water at the pond? Those ripples were perpendicular to the pebble's vertical dive into the water.[1]

So why do we view sound this way? Compressions and rarefactions over time are much easier to view in a transverse view compared to a longitudinal one. Figure 1.6 shows a longitudinal graph of a wave (vertical lines, at top) versus a transverse view (sine wave, at bottom). If you have ever played with a Slinky, you may have seen waves of energy travel from one side to the other. If you and a friend stretch the Slinky out slightly, then your friend quickly pushes their end forward and back a few times, tightly coiled regions will "travel" through the spring toward you. Those tight bands that were created in the spring were the compressions, and the looser ones the rarefactions. Again, you can look at the longitudinal wave at the top of Figure 1.6 to see this "Slinky" effect, with the tight regions representing the denser areas. You can also see why viewing such data would be far more complex to navigate than with the transverse model.

Wavelength

A wave not only has a time interval but a physical length as well. Even though you cannot see them under normal circumstances, sound waves travel through our air medium and can be measured by their **wavelength**: the distance it takes one cycle to complete,

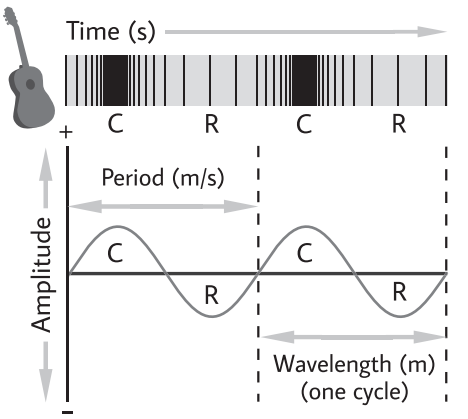


Figure 1.6 Graph of wave-measuring amplitude, period, and wavelength. Top: a longitudinal wave showing compressions (darker regions) and rarefactions (lighter regions). Bottom: transverse interpretation of the same wave, with compressions and rarefactions denoted by Cs and Rs, respectively.

in meters. In the examples throughout the book, single-cycle waveforms will be defined as mentioned earlier: ***rest → full compression → rest full rarefaction → rest***, which works great using a transverse graph. The only convenient way to measure wavelengths in a longitudinal graph is either between two consecutive maximum compressions (the *crests*), or maximum rarefactions (the *troughs*), as seen in Figure 1.6.

Amplitude

In Figures 1.5 and 1.6, normal air pressure is located at the center of the vertical axis. The change in air pressure from the wave's resting state to its maximum displacement in either direction is its **amplitude**. In other words, we are measuring the *height* of the wave. Because it is a measurement of the change in air pressure, it is also a measurement of how loud or soft a sound is perceived to be, which we will define using *decibels* in the next section.

Frequency

The **frequency** (or **pitch**) of a sound wave can be measured in cycles per second, or **Hertz** (**Hz**) (named after Heinrich Hertz, the physicist who first discovered electromagnetic waves). For example, playing the low A string on a guitar generates a 110 Hz wave, so the string is vibrating 110 times per second. The beep of a microwave oven might be 2 kHz, or 2 **kilohertz** (**kHz**), an electronic tone vibrating 2,000 times per second from a tiny speaker. In the natural world, we are constantly listening to multiple frequencies at any one time. Human speech is comprised of thousands of frequencies, and our ears are sophisticated enough to interpret all of these rapid-fire vibrations into something we can understand.

Frequencies can be **tonal**, or musical in nature, such as with the consistent tone of a trumpet, doorbell, or the wavering sound of a car alarm; frequencies can also be

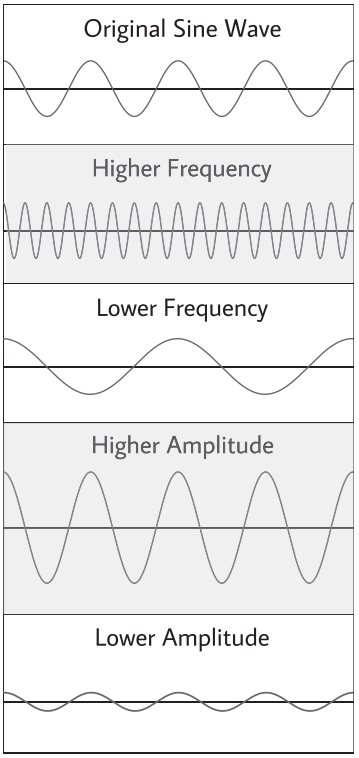


Figure 1.7 A sine wave transformed by higher and lower frequencies versus higher and lower amplitudes.

random in nature, with no discernable repeating patterns to the vibrations. In these instances, we categorize the frequencies as **noise**.

As you can imagine, changing the frequency and amplitude affects the waveform in dramatic ways. Figure 1.7 shows how altering these parameters affects its shape on a graph.

Speed of Sound

Sound is nowhere near the speed of light, but it is still *fast*. In air, it travels about 344 meters per second (1,238 km/h), or 1,130 feet per second (770 mph). Altitude and temperature can alter this speed slightly. Sound that travels in other mediums than air can affect its speed dramatically, however, which is described in the **Sound Propagation** section of this chapter.

Cue the Math!

Here are a few basic formulas which are very handy for computing wavelength, frequency, and period. You never know when you may need to make one of these calculations!

The **wavelength** (λ) of a sound is the speed of sound (v) divided by the sound's frequency (f) , or $\lambda = \frac{v}{f}$:

$$\lambda = \frac{v}{f} = \frac{\text{speed of sound}}{\text{frequency}} = \frac{344 \text{ m/s}}{\text{frequency}}$$

So, if you have a 1,000 Hz sine wave traveling through the air, its wavelength is:

$$\lambda = \frac{344}{1{,}000} = \textbf{0.344 meters.}$$

Since we know how to solve for wavelength, we can also solve for **frequency**. It is the speed of sound divided by the wavelength, or $f = \frac{v}{\lambda}$. So the frequency of a sound with a wavelength of 0.344 meters is:

$$f = \frac{v}{\lambda} = \frac{344 \text{ m/s}}{0.344} = \textbf{1,000 Hz, or 1 kHz.}$$

What about distance over time? The **period** of a wave (T) is 1 divided by its frequency, or $T = \frac{1}{f}$. Therefore, the period of a 1,000 Hz sound wave is:

$$T = \frac{1}{f} = \frac{1}{1000} = \textbf{0.001 m/s.}$$

Intensity, Decibels, and Loudness

A sound wave also has a force to it known as **power**. This is measured in **Watts** (W) per second, which are units of work over time. But what about a sound wave traveling within a physical area? In this case, **intensity** is used. This is written as:

$$I = \frac{P}{A}$$

where P = power (in Watts per second) and A = area (in meters squared). 10 W/m^2 is the maximum intensity most people can endure before physical pain is introduced, aptly named the **threshold of pain** (**TOP**). The quietest detectable sound comes in at about 1×10^{-12} W/m^2, a difference of 1 trillion units.[2] This range has far too much granularity to be practical for our needs.

Studies have consistently shown that in order to perceive a sound twice as loud as another one, its power must be increased by a factor of roughly *ten times*.[3] A logarithmic system of measurement works very well in this application. If you're unfamiliar with logarithms, take for example $2^4 = 16$. This is the exponential form for 2 (the base) multiplied 4 times (the exponent, up high), or $2 \times 2 \times 2 \times 2 = 16$. Another way to look at this is in the logarithmic form, where $\log_2 (16) = 4$. This is the *inverse* of exponential form, where we are trying to find the exponent. In essence we are saying, "How many 2s do we need to multiply in order to get 16? We need 4."

Using magnitudes of ten, this will give us a more practical level of granularity with measurements in the tens to hundreds, not trillions.

A unit of measure for comparing intensity ratios in audio and electronics is called the **Bel.**[4] We would only be measuring about 12 Bels in our audio work, which is not enough granularity. So instead, the unit we most often use is the **decibel** (**dB**), which is 1/10 of a Bel. Each Bel is perceived as "twice as loud" as the next, so for every increase of 10 decibels, there a doubling of the sound intensity level (**SIL**). It is a measurement of acoustic energy traveling through an area of space.

The decibel is a *ratio* between two levels of intensity: the intensity level of the sound we are concerned with, against the quietest detectable 1 kHz tone at 1×10^{-12} W/m^2. This value is 1 trillionth of 1 decibel, which it is effectively 0 dB and also known as the **threshold of hearing** (**TOH**). However, even those with perfectly healthy ears may have slightly different TOH; for some, the threshold may be up to ten, or even twenty decibels.[5]

Decibels for SIL are calculated as:

$$dB\ SIL = 10 \log \frac{I}{I_0}$$

We always compare our main decibel level I with I_0, which is near-but-not-quite 0 (division by 0 is a no-no, by the way).

In Table 1.1, for example, the quiet sound of rustling leaves might be 1,000 times louder than the TOH; this ratio would be 10 log ($1 \times 10^{-9}/1 \times 10^{-12}$), which equals 30 dB. A conversation might be 1,000,000 times louder than the TOH, or 10 log ($1 \times 10^{-6}/1 \times 10^{-12}$), which equals 60 dB.

Root Mean Square (RMS)

Often when listening to audio, determining the average loudness level *over time* is preferred, rather than viewing instantaneous or peak-to-peak (crest and trough) values. Through a few calculations, we can obtain its approximate average loudness, known as **Root Mean Square** amplitude, or simply **RMS**. This value can be determined by

Table 1.1 Comparison of Decibel Levels

Event	*Decibels*	*Value in W/m^2*	*Times Greater than TOH*
Threshold of hearing (TOH)	~0	1×10^{-12}	
Whisper, gentle wind	20	1×10^{-10}	10^2
Leaves rustling	30	1×10^{-9}	10^3
Quiet indoor environment	40	1×10^{-8}	10^4
Light activity in office	50	1×10^{-7}	10^5
Normal conversation	60	1×10^{-6}	10^6
Vacuum cleaner, kitchen appliance	70	1×10^{-5}	10^7
Bustling city, outdoors	80	1×10^{-4}	10^8
Lawn mower	90	1×10^{-3}	10^9
Subway	100	1×10^{-2}	10^{10}
Leaf blower, chainsaw	110	1×10^{-1}	10^{11}
Car horn from outside	120	1×10^{0}	10^{12}
Jet engine from outside at close range, **threshold of pain (TOP)**	130	1×10^{1}	10^{13}
Gunshots or large fireworks at close range, **INSTANT HEARING DAMAGE**	140	1×10^{2}	10^{14}

taking repeated samples of the amplitude as the waveform plays out, or when analyzing a digital audio file in advance.[6]

Sound Pressure Level (SPL)

Recall that SIL is used to measure the acoustical energy level of sound traveling through an area of space, in decibels. But more relevant to audio designers, we want to measure that same acoustical energy as the pressure amplitude at a point exclusively hitting our ears, or on an electronic device such as a microphone, for example. SIL can be used as a measurement for hearing, but it refers to a sound's amplitude *as it travels through a space*, not its pressure amplitude at a given point.

Fortunately for our purposes, sound intensity level can be thought of as identical to **sound pressure level** (**SPL**). As noted above, it is the force, or pressure of that energy detected either by our ears or electronic devices. From this point forward, when the term dB is used without a qualifier, it will refer to **dB SPL**. It is the *average perceived loudness* of a waveform, measured as:

$$dB\ SPL = 20\log_{10}\frac{p_{rms}}{p_{ref}}$$

where p_{rms} = the RMS power (average power) of the waveform over a given time, and p_{ref} = the near-zero threshold of sound pressure, 1×10^{-12}.

Below is a table of decibel levels, with some corresponding events given as examples. Note that at approximately 140 dB, even a quick, transient event can cause permanent hearing loss. Table 1.2 lists a few recommended exposure times to average decibel levels.

If you're curious about the loudness in your surroundings, a great iOS/Android app can be found from the NIOSH (National Institute for Occupational Safety and Health – in the USA), called the *NIOSH Sound Level Meter App*. It detects sound levels with +/–2 dB accuracy and will even calibrate to your specific microphone.[7]

Hearing Spectrum (Frequency Range)

If you were born with perfect hearing (and are still relatively young), you should be able to hear frequencies between ~**20 Hz** and **20 kHz**. But this is a mere slice of

Table 1.2 Sound Exposure Limits

dB Level	*Recommended Exposure Limit (Per Day)*
85	8 hours
88	4 hours
91	2 hours
94	1 hour
97	30 minutes
100	15 minutes
103	7.5 minutes
106	3.75 minutes

Source: NIOSH, via CDC Website.[8]

the wave spectrum. Vibrations continue in frequency far beyond our hearing, where other animals are able to sense (dogs up to 45 kHz and dolphins up to 150 kHz![9]). Eventually, the frequencies are high enough to be categorized as radio waves and microwaves, from over 20 kHz to 300 gigahertz (GHz); then they become light waves (from 430 to 750 terahertz (THz)); and finally, the more deadly X-rays and gamma rays (up to 10^{20} Hz). All of them are waves!

Equal-Loudness Contour

The human ear does not hear all frequencies at the same decibel level. Our ears favor the 2–5 kHz range best, which also happens to be the prime area for speech frequencies. With frequencies below about 250 Hz, our hearing capability diminishes steadily until we approach about 20 Hz, where sound is more viscerally felt than heard (as with a subwoofer). On the higher end of the spectrum, frequencies above 16 kHz are more of a rarefied "presence" in the air than an audible tone. And, there are some additional peaks and valleys in our listening capabilities in between.

To make sense of it all, a graph of equal-loudness contours is here to help. Carefully crafted over many decades and several hearing studies later,[10] this contour graph shows how the human hearing range is far from a linear experience.

Understanding the Data

The graph in Figure 1.8 is a series of contour lines, each representing comparisons between frequencies and loudness levels that seem equal to the human ear. For example, look at the third contour line from the bottom. To hear 20 Hz, a sine wave needs to be generated at 95 dB (quite loud, and perhaps bordering on the uncomfortable). But to hear 2.5 kHz at the same perceived volume takes but a mere 15 decibels – barely a whisper!

What Are Phons?

One **phon** is a unit of measure that represents 1,000 Hz heard at 1 decibel. Each contour in Figure 1.8 has a phon unit associated with it that is its baseline level. So, 60 phons on line 5 means the test subjects were first played 1,000 Hz at 60 dB, and then they had to compare the loudness of all other frequencies in the spectrum against that.

Waveforms

Up to now, pure sine waves have been used exclusively in the chapter examples. But there are many other basic waveform types that can be useful for audio production as well. In the early days of console game audio, all you needed were a few fundamental waveforms, such as sine, square, triangle, sawtooth, and/or noise, like those in Figure 1.9. You can also hear the first four waveform types at Chrome Music Lab (https://musiclab.chromeexperiments.com/oscillators), which is a lot of fun to play with and to hear their characteristic tones.

The classic NES console utilizes basic waveforms. It has five channels of audio, four of which support basic waveforms for game soundtracks: channels 1 and 2 are

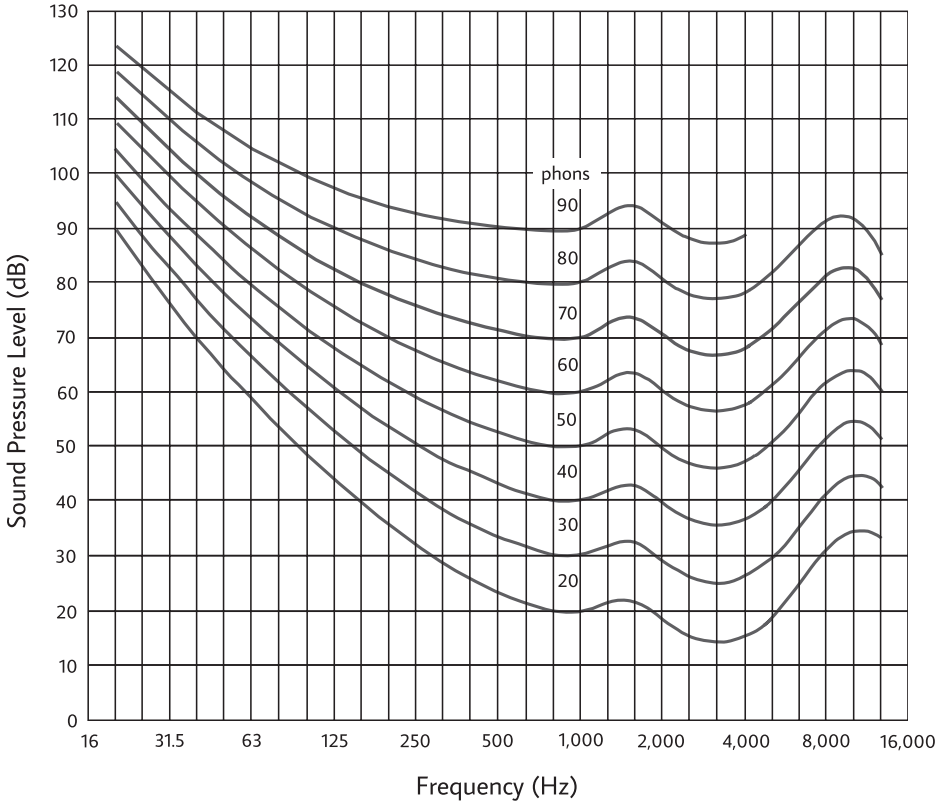


Figure 1.8 Equal-Loudness Contours from 20 to 90 phons. Image credit: British Standards Limited (BSI).[11]

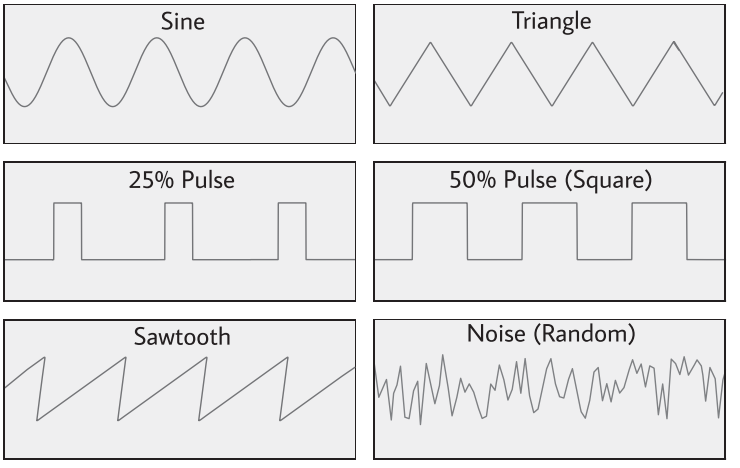


Figure 1.9 Common sine, triangle, sawtooth, pulse, and noise (random) waveforms.

for square waves which offer rich, bold tones for melodies and harmonies; channel 3 is for triangle waves, a softer-sounding tone great for bass lines or background instruments; channel 4 is for noise, which allows for kick and snare drums, among many other percussive instruments; and channel 5 is used as an "other" category, allowing for a very short digital audio sample to play (more on sampling in Chapter 2).

Sine

The simplest, purest waveform possible. A smooth tone used as the basis of many complex waveforms, in constructing new electronic sounds.

Pulse

Bold, rich, full – however you describe it, this pulse wave has a pleasing feel to it. Typically, these waveforms are measured in percentages: a 25% pulse is a rectangular wave, where a 50% pulse is known as a square wave. These are used for primary melodies on synthesizers and anywhere a commanding electronic tone is needed.

Triangle

This sounds like a cross between a sine and square wave, somewhat mellow but still enough tone to be prominent.

Sawtooth

This waveform looks and sounds like the word implies. It is buzzy, jagged, and can feel harsh if too loud. It is extremely sharp and prominent in tone.

Noise

As we have learned, noise is a series of all frequencies generated randomly. If we generate it electronically, it is really a rapid-fire series of random frequencies.

Envelope: Attack, Decay, Sustain, Release

The amplitudes of many sounds and instruments can be expressed in several distinct stages, collectively known as **envelope**. At a minimum, the envelope of a sound has two stages: an **attack**, which is the ramp-up time to reach maximum amplitude, and a **decay**, where eventually the amplitude must decrease to a lower level. For example, a bell has a fast attack and a long decay as it rings out, until its vibrational energy is depleted.

However, most envelopes are described using *four* stages: **attack** for the ramp-up time, then **decay** to fall to a level of **sustain**; from there, the amplitude holds steady at its sustain level for a length of time, and then eventually decays and ends according to its **release** time (note that sustain is the only attribute that is at a *fixed level* over time). This setup is so ubiquitous that it's referred to as **ADSR**. Most instruments in electronic music (as well as samples and virtual instruments) rely on ADSR to construct

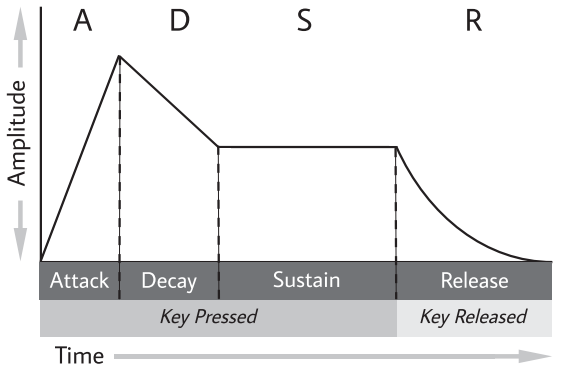


Figure 1.10 Envelope of a sound showing four distinct changes in amplitude: (A)ttack, (D)ecay, (S)ustain, and (R)elease.

their dynamics and unique characteristics: a white noise waveform with a short attack and long release could emulate a cymbal crash, for example. And as we will see in Chapter 6, many plug-ins also make use of envelopes in varying levels of complexity.[12]

In Figure 1.10, a musician presses and holds a key on a synthesizer; for the attack, the sound fades in from 0 dB to its maximum level. Then it descends slightly and remains at its sustain level, until the musician releases her finger from the key. Finally, the sound takes a short time to fade away until it reaches 0 dB again.

Timbre

Ah, the sweet sound of a wooden flute! Just by listening to it, we know that when it produces an A at 440 Hz, it sounds much different than a piano playing the same note. The first is generated by air rushing through a column of wood or metal, and the other by the vibration of a string hit with a hammer. Yet, they can both produce many of the same notes, with their own distinctive texture, known as their **timbre**.

Timbre is made up of a sound's unique envelope, combined with the magnitudes (specific volume levels) of its **harmonics** – the additional, superimposed frequencies we hear along with the original. It's what gives an instrument its rich tones and depth.

Noise

Noise is everywhere. Not just the din of the great outdoors but in all of our electronic devices, which of course includes computers and recording equipment. It's in the nature of designing circuitry, as well as from radio interference, "crosstalk" from wires placed too close together, ground loops, and more. Engineers and consumers do all they can to eliminate noise, unless you're an audio designer – then it suddenly becomes a feature!

Noise is the sound of all frequencies in the hearing spectrum, generated randomly. It is used in certain applications such as in testing electronic equipment and to smooth out errors in digital audio (described in Chapter 2). But as audio designers, we can employ noise elements in creating certain sound effects (explosions, squelchy communications, wind, etc.) and also as musical instruments in chiptunes or other analog-based electronica.

White and Pink Noise

Of the myriad of color categorizations to choose from, **white noise** and **pink noise** are the most commonly used. White noise represents all frequencies at the same volume. It has a sharp, fuzzy quality to it (synonymous with a fuzzy TV signal). It is analogous to the color white, which is made up of all colors in the spectrum.

Pink noise, on the other hand, has a slanted frequency shift to it that more closely reflects the human **equal-loudness contour**. It has a deeper tone to it, as the low frequencies are boosted considerably higher than the upper frequencies. Figure 1.11 shows a snapshot taken from a white noise and pink noise generator. The sounds are randomly distributed which is why the graphs have multiple peaks and valleys, as opposed to a straight line or curve.

Sound Propagation

As sound waves **propagate**, or travel through a medium, they can encounter all types of obstacles that interact with them in various ways. As noted earlier, sound waves will expand outward spherically in all directions, if there are no impediments to its travel. When a wave reaches the end of one medium and approaches a new one (say,

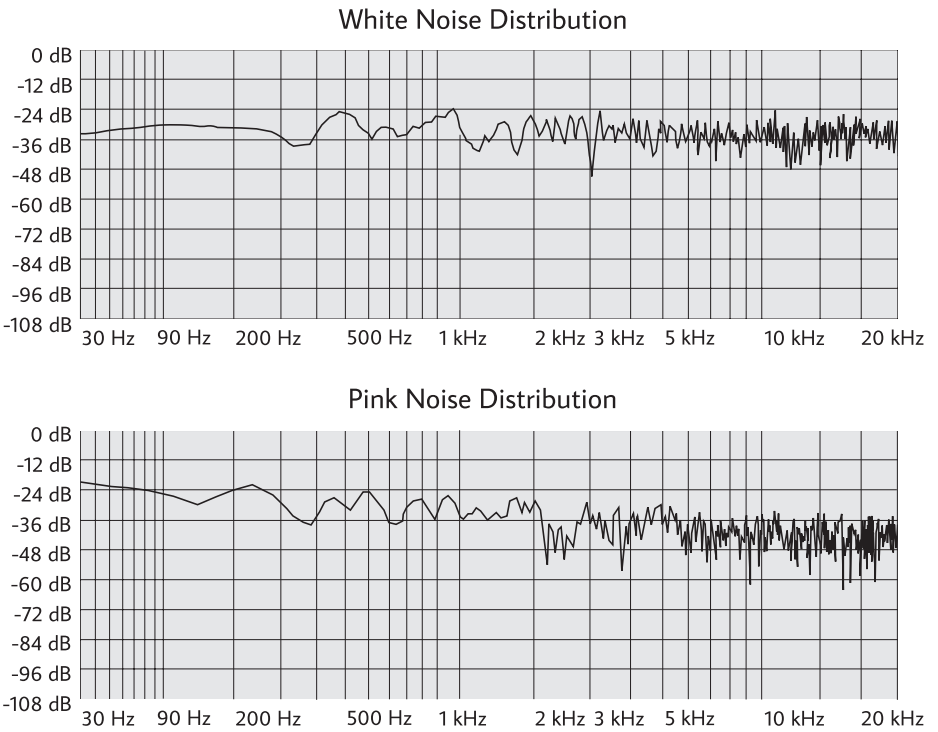


Figure 1.11 Frequency analyses of white noise versus pink noise. Note the downward slope in the pink noise, where lower frequencies are louder in volume than higher frequencies. This is closer to emulating an equal-loudness contour.

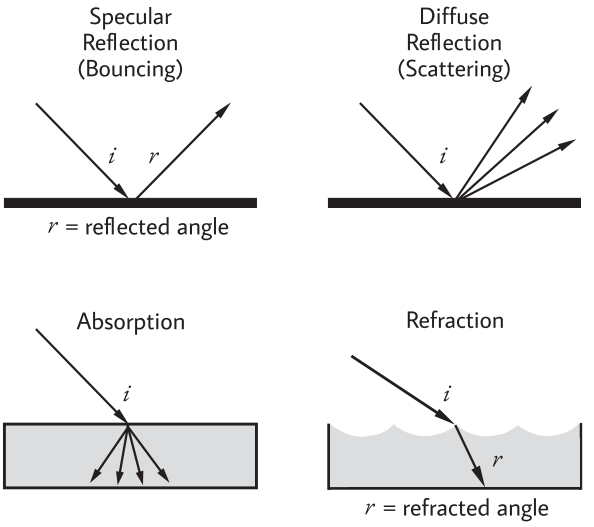


Figure 1.12 Sound propagation and various boundary behaviors.

traveling in the air until it hits a metal obstacle, or water), it exhibits a **boundary behavior**. The wave can bounce, scatter, bend, and/or become absorbed by the new medium, to some degree. Below is a brief description of each behavior type, which can also be seen in Figure 1.12.

Specular Reflection (Bouncing)

If you have ever seen or played *Pong*, you understand this basic physical behavior. When a sound wave hits a smooth, hard surface at a given angle (called the **angle of incidence, *i***), it reflects or bounces away from the surface at the exact same angle (called the **angle of reflection, *r***). You can hear reflections by clapping your hands either in an empty room or in a room with lots of sheer surfaces such as in a bathroom, which usually dedicates a significant percentage of its surface area to mirrors, tile, and porcelain. When you clap, the sound bounces off a wall and back to your ears. In a relatively empty and/or large room such a gymnasium, there are very few items in the room to deaden the reflections, so they will keep ping-ponging around the room until the wave energy is dissipated. It takes a long time for this to happen in a gym due to its large size, sheer surfaces, and lack of objects to absorb the wave (more on this in Absorption, below).

For game audio, these reflections are mainly known as reverberations, or **reverb** for short. We can use reverb for many applications in music, sound design, and dialogue, simulating all types of acoustic spaces where a player may hear audio coming from in the game.

Diffuse Reflection (Scattering)

Most surfaces in the real world aren't completely smooth. If a sound wave were to impact a rocky surface, it might encounter lots of bumps and jagged edges that cause

scattering: reflections off of the surface in multiple, random directions. The more uneven and rigid a surface, the greater chance there is for scattering, also known as **diffuse reflection** or simply **diffusion**. A good concert hall, classroom, or recording studio benefits from diffusion, if all frequencies can be evenly distributed and heard in all areas of the room. There are ways to accomplish this through applying acoustical treatments to a room, which we will see in Chapter 4.

Diffraction (Bending)

Sound can not only bounce off of surfaces, it can actually bend around them as well. This is why you could be talking to someone in one room, and another person can hear you around the corner in a different room (see Figure 1.13). Some of this effect may occur from reflections, but most is due to a phenomenon called **diffraction**. However, not all waveforms diffract successfully; they can be stopped by obstacles in their path. Lower frequency waves have longer wavelengths (20 Hz wavelengths are more than 17 meters long!) and can bend around obstacles more easily. The larger the wavelength, the larger the obstacle that is needed to block it. Higher frequency waves cannot bend around obstacles as well (20 kHz wavelengths are only about 17 centimeters long), and instead are usually *absorbed* by the obstacle. More on this in a moment.

In Figure 1.13c, the obstacle in the middle causes the wave to bend, introducing an **acoustic shadow**. This lowers the volume of certain frequencies depending on the obstacle's dimensions.

In another example of diffraction (in Figure 1.13), waves can also pass through an opening, either small or large, and expand into that opening. As with bending, larger wavelengths can pass through openings more easily than shorter ones. If you've ever seen the inside of a professional recording room, there are no visible openings once the door is shut. If there is even a small opening, sound can pass through and expand into the room (especially lower frequencies), interfering with the session.

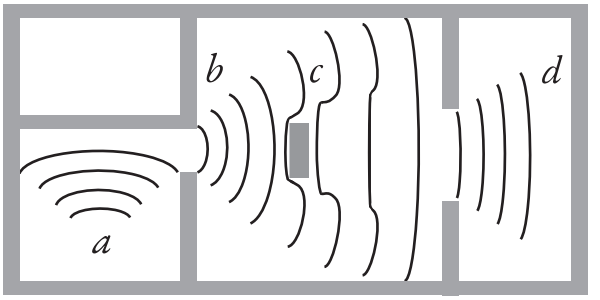


Figure 1.13 Various modes of diffraction: (a) a sound wave reaches an opening in the corner of a room; (b) enters that opening, filling the space within; (c) it then diffracts around an obstacle, with a wavelength long enough to bend around it; and (d) finally, the sound wave diffracts through another opening. The level of diffraction increases with lower frequencies (longer wavelengths) and smaller openings.

Absorption (Conversion to Heat)

This is where sound reaches a material, which can deaden its energy as it passes through a wall or window, for example. Rather than sound continuing on its journey, if the boundary material is porous or soft, then it may **absorb** the sound rather than reflect it. The energy passes into the material and is converted into heat energy. But not to worry – the amount of heat generated is infinitesimally small and poses no harm to your studio or the planet; to quote acoustician and audio educator Ken Pohlmann, "it would take the sound energy of millions of people talking to brew a cup of tea."[13]

In some cases, the material will absorb what it can of the wave, but the energy will be strong enough to pass through it. In this case, transmission occurs and the sound wave is weakened upon exiting the material. In gaming, this is an example of **occlusion** (see Chapter 12): take for example a noisy tenant in an apartment building. They may be blasting their stereo, and the low frequencies will pass through the walls, floors, and ceiling to your ears, but the higher frequencies will become totally absorbed by the building's materials, and you will not hear them. This is also the reasoning behind double-paned windows: in addition to providing some temperature insulation, the sound hitting one pane passes through it into a pocket of air; by this time, the sound is weakened and may not even have enough energy to pass through the second pane.

Refraction (Transfer)

So far, we have seen examples of sound waves only through an air medium. But sound can also travel through other mediums, including water. And did you know it actually travels *faster* in water than air? This is because water is a much denser medium than air, and the molecules are much closer together. This allows for sound waves to travel more efficiently, and in vast open spaces such as in the ocean, the waves travel much longer distances. This is how dolphins and whales can communicate with each other miles apart.

When sound from air travels into water, it is partially **refracted**, meaning that the sound is bent and spreads out due to the change in the speed of sound in water. Recall that the speed of sound in air is approximately 344 m/s; in water, due to its higher density, the speed of sound is about 1,480 m/s, more than four times faster!

Interference

When two or more sounds intersect at a given location, **interference** occurs. The waves may be in parallel with each other, or coming in from all directions. Crests and troughs combine to either make the resultant sound wave louder or softer, or to be canceled out entirely. For the sake of example, let's say two identical sine waves are playing, having started at exactly the same time. The two are **in phase** when they are in perfect alignment with each other, creating a doubling of amplitude as in Figure 1.14a. Otherwise, they are **out of phase** when not in perfect alignment with each other (Figure 1.14b).

Think of the waves' compression cycles as positive values, and their rarefaction cycles as negative values. The energy of the two waves is added together at a given intersection point and that may yield a positive value where the net amplitude increases,

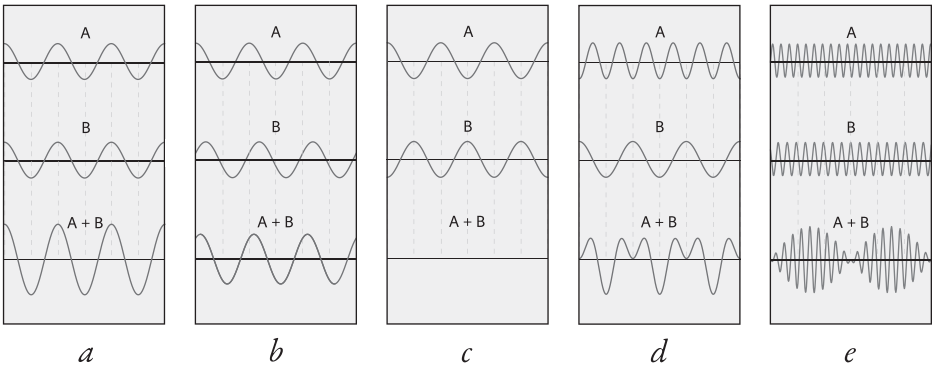


Figure 1.14 (a) Identical waves in perfect phase creating a doubling effect (100% constructive interference); (b) identical waves are 25% out of phase, causing partial interference; (c) identical waves are 100% out of phase, causing total phase cancelation (100% destructive interference); (d) the frequency of A is higher than B, causing partial interference; and (e) the frequency of A is slightly higher than B, causing beats to occur.

called **constructive interference**. Or, if their sum yields a negative value, there would be **destructive interference**, with a net reduction in amplitude. Of course, the waves could be partially in or out of phase if they did not start simultaneously and/or could be coming from various directions, so partial constructive and destructive interference would take place in this instance (Figure 1.14d). If one sound's frequency is slightly higher than another, **beats** may also be heard. This is a phenomenon where there are regular patterns of total constructive and destructive interference, causing a pulsating effect in the resultant audio (Figure 1.14e).

When the two are exactly a half-cycle apart, the crest of one wave cancels out the trough of the other, creating silence. This is known as **phase cancelation** (Figure 1.14c). In the real world, total cancelation is hard to achieve, as nearly all other waveforms are far more complex than pure sine waves. However, this is how noise-canceling headphones work. The general principle is that microphones placed outside the headphone speakers take in the ambient sound waves, and then sets them 180 degrees out of phase and plays them back into the headphone speaker. The result is your audio is easier to hear, as the outside sound is significantly reduced, all thanks to (hopefully!) an acceptable amount of cancelation.

When setting up your studio, it's important to have your speakers aligned toward your ears, so you can be in what is known as the **sweet spot**, as shown in Figure 1.15; otherwise, you will be hearing audio with partial interference. The sweet spot is actually a **phantom image**, where a new audio source seems to appear in the center between two speakers.

Doppler Shift (Pitch)

Everyone has experienced this phenomenon at some point: you hear a car or train approaching, and its engine sound seems to be higher in pitch as it approaches you, and then lower in pitch as it passes by you. How is this possible?

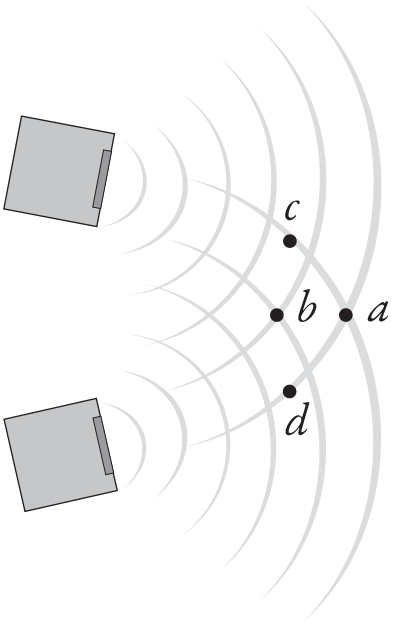


Figure 1.15 Proper seating position and alignment of speakers is important, so your ears are hearing points such as at (a) or (b) where the two waves are in phase, and their crests intersect (100% constructive interference). Otherwise, if you are seated at an incorrect position (or if the speakers are misaligned), you will hear partial constructive or destructive interference such as at point (c) or (d).

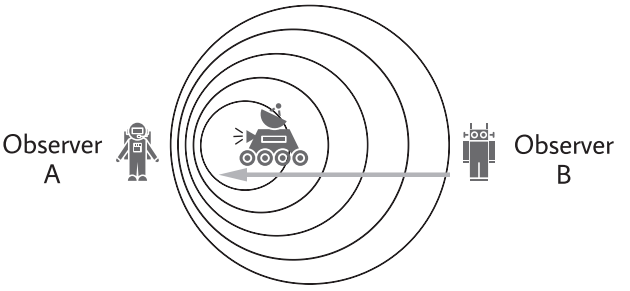


Figure 1.16 The Doppler effect for two stationary observers. When the vehicle is approaching observer A, its wave crests become closer together, creating a pitch increase. As the vehicle travels away from observer B, the crests are further apart, creating a pitch decrease.

Let's assume there are two stationery bystanders on a space station. A car is approaching observer A and has just traveled past observer B, as in Figure 1.16. As the vehicle is moving, it is generating sound. As it approaches observer A, the distance between A and the sound waves of the vehicle becomes increasingly shorter, and the sound requires increasingly less time to travel to A's ears. The crests of each wave also

become increasingly crowded together. The net effect for A is that the vehicle's sound is higher in pitch until it reaches (and just misses!) A.

For observer B, the opposite happens: the sound of the vehicle decreases in pitch as it passes by them. The vehicle's sound is moving away from B, and its sound requires increasingly more time to travel to B's ears as the distance between them increases. The crests of each wave also become increasingly spread out. The net effect for B is a decreased pitch, as it passes by and fades into the distance.

Notes

1 You would notice the ripples also had a physical, wavelike shape to them, if you could look at the water at ground level. In this specific case, the water waves have both longitudinal (the circular ripples) and transverse (the physical waves) motion to them.
2 It is truly amazing that the ear can detect such a faint level: this is air being displaced one *billionth* of a centimeter.
3 White and White, *Physics and Music*, 103.
4 The Bel is a unit of measure named after Alexander Graham Bell and originally created by Bell Telephone Laboratories to measure communication levels in telephone lines.
5 Hall, *Musical Acoustics*, 94.
6 A significant number of audio amplitudes are sampled, and each of those values is squared; then, the total is divided by the amount of sample values taken (the mean). Lastly, the square root of that quotient is determined, which is the RMS value.
7 More information can be found at the National Institute for Occupational Safety and Health Website: https://www.cdc.gov/niosh/topics/noise/app.html.
8 https://www.cdc.gov/niosh/topics/noise/factsstatistics/charts/chart-lookatnoise.html. Retrieved June 20, 2021.
9 Hard-to-find sources of the estimated hearing ranges of various animals can be located at https://www.lsu.edu/deafness/HearingRange.html.
10 More information on equal-loudness contours and various studies can be found in the paper, "Precise and Full-range Determination of Two-dimensional Equal Loudness Contours": https://web.archive.org/web/20070927210848/http://www.nedo.go.jp/itd/grant-e/report/00pdf/is-01e.pdf. The ISO 226:2003 standard for equal-loudness contours is constantly under revision and updated every few years.
11 Permission to reproduce extracts from British Standards is granted by BSI Standards Limited (BSI). No other use of this material is permitted. British Standards can be obtained in PDF or hard copy formats from the BSI online shop: https://shop.bsigroup.com.
12 Some plug-ins such as a compressor may only use attack and release, while others may use up to *five* attributes known as **AHDSR**: attack, **hold** (a sustain level between attack and decay), decay, sustain, and release.
13 Everest and Pohlmann, *Master Handbook of Acoustics*, 183.

Bibliography

Backus, John. *The Acoustical Foundations of Music: Musical Sound: A Lucid Account of Its Properties, Production, Behavior, and Reproduction*. 1st ed. New York, NY: W. W. Norton & Company, 1969.

Ballora, Mark. *Digital Audio and Acoustics for the Creative Arts*. New York, NY: Oxford University Press, 2017.

Benade, A. H. *Fundamentals of Musical Acoustics*. 2nd Revised Edition. New York, NY: Dover Publications, 1990.

Cancellaro, Joseph. *Exploring Sound Design for Interactive Media*. Clifton Park, NY: Thomson, 2007.

Everest, F. Alton, and Ken C. Pohlmann. *Master Handbook of Acoustics*. 6th Revised Edition. New York, NY: McGraw-Hill Professional, 2015.
Hall, Donald E. *Musical Acoustics*. Long Beach, CA: California State University, 1990.
Loy, D. Gareth. *Musimathics: The Mathematical Foundations of Music* (Volume 1). Cambridge, MA: MIT Press, 2006.
British Standards Institution. "Normal Equal–Loudness-Level Contours for Pure Tones Under Free-Field Listening Conditions – ISO 226:2003" Figure 1.A. BS 5228-2:2009.
Pejrolo, Andrea, and Scott B. Metcalfe. *Creating Sounds from Scratch: A Practical Guide to Music Synthesis for Producers and Composers*. New York, NY: Oxford University Press, 2017.
White, Harvey Elliott, and Donald H. White. *Physics and Music: The Science of Musical Sound*. Mineola, NY: Dover Publications, Inc., 2019.

Chapter 2

Digital Audio

Digital Audio

In the world of vinyl records, tape, and electronic instruments such as synthesizers, listening to audio is an **analog** experience; that is, the audio is a continuous stream of waveforms, with infinitely varying amplitudes over a seamless time frame. **Digital audio**, on the other hand, is a discrete representation of analog audio and time. Sound is analyzed in equally spaced snapshots over time as **samples**, and then converted to a series of carefully organized ones and zeroes called **bits**, which represent the frequency and amplitude of the original analog audio signal at a given moment. This chapter will cover foundational knowledge in digital audio, including the binary number system, sampling and quantization, recording, processing, and playback.

There are some that say analog audio is a thing of the past, while others welcome a new renaissance of vinyl and tape-based production. Whatever your opinion is, there is no getting around digital audio being part of your workflow, especially in game audio production. Let's take a look at what digital audio actually is, and the techniques designed to utilize it. While there have been volumes written on this topic, we will stick to the most important concepts here as they relate to game audio.

Binary Number System

Computers store, move, and retrieve data based on a **binary number system**. We are all familiar with our **base-10** number system: after you count up from 0 through 9, you are able to continue on to 10–19, 20–29, and so on. With each group of 10 numbers, the value on the left increases by 1: counting to 39 leads you to 40; counting to 999 leads to 1,000, etc.

The binary number system uses a **base-2** format, where only 0 and 1 are used to represent all numbers. The unit that represents either a 0 or 1 is called **bit**, and there are 8 bits to a **byte**. There is an easy way to view this number system if we create a row of bits (one byte) together:

Through technological wizardry, computers and other electronic devices use binary "numbers" (voltage stored in tiny transistors on a microchip) to perform a myriad of tasks, including the processing of digital audio. Figure 2.1 shows one byte of data as a set of eight boxes. Each box stores a bit, and each of those boxes represents a power of two, from right to left. The column on the far right represents the value 2^0 which equals 1; the preceding column represents 2^1 which equals 2; then 4, 8, and 16, up to 128. Each bit's value can only be a 0 or 1, but with 2^n representation, any number in

DOI: 10.4324/9781003218821-4

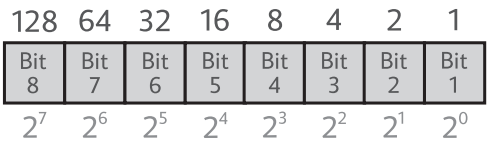


Figure 2.1 One byte of data, made up of 8 bits. Each bit represents a power of 2, with values increasing from right to left. Any number from 0 to 255 can be constructed in a byte, by assigning a 0 or 1 to each bit.

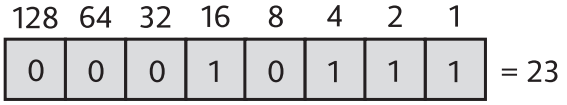


Figure 2.2 The number 23 in binary form.

our byte can be created ranging from 0 to 255. For example, the number 23 can be stored as **00010111**, where:

0 (128) + **0** (64) + **0** (32) + **1** (16) + **0** (8) + **1** (4) + **1** (2) + **1** (1) = 23.

Or, represented in binary form:

Computers are great at manipulating binary values, as the voltage in each transistor (used in processing data, memory, and storage) can either be on or off: a logical "1" or "0." As we saw above, stacking bit values together is the way binary numbers are created (Figure 2.2). In the world of digital audio, these values will commonly be stored as **kilobytes** (1 kilobyte = 1,000 bytes), **megabytes** (1 megabyte = 1 million bytes of data), and **gigabytes** (1 gigabyte = 1 billion bytes of data). To further visualize the binary counting process, Table 2.1 shows bit values for 16 numbers, 0–15.

Sampling

In the early days of motion pictures, film was used (and often still is) to capture visual events. Recording to film involves a process where pictures are taken at precise intervals, known as **frames per second** (fps). Traditionally, movies are recorded at 24 fps, though some motion pictures are now being released in 30 fps or higher. The frames on film are played back one after the other, at equal time intervals. Our eyes do not perceive the breaks between individual pictures, and so the viewing experience appears seamless.

A similar process occurs when audio is recorded digitally. Through hardware that includes an analog-to-digital converter (**ADC**), the incoming analog audio is captured as a series of **samples**. Each sample is snapshot in time, which is a binary number representation of the **amplitude** (the voltage) of the signal at that moment. The number of samples taken per second is called the **sampling rate**, and the time interval between any two adjacent samples is the **sample period**. Figure 2.3 outlines the basic idea of sampling; first, the incoming waveform (with our old friend the sine wave) is recorded digitally, with samples taken at regular intervals. Those samples are numerical values that are electronically stored. Upon playback through speakers or headphones, a digital-to-analog converter (**DAC**) converts the sample data back to its analog form. But as we will see in this chapter, there are several more steps in the ADC/DAC conversion process in order to obtain optimum-quality digital audio.

Table 2.1 Binary Numbers from 0 to 15

Binary	*Decimal*	*Binary*	*Decimal*
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

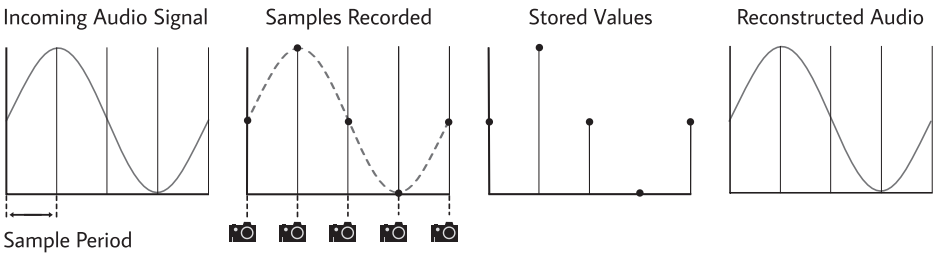


Figure 2.3 Basic sampling process (without quantization effects). The amplitude levels of an analog audio waveform are sampled at equally spaced intervals (the sample period), and then those samples are stored as discrete integer values. The original analog signal can be reconstructed later through digital-to-analog conversion. Depending on the sampling frequency, more or less samples of the wave cycle are taken.

Sampling Rates

Unlike our eyes, we need a much faster rate of speed than 24 frames per second to sample audio. Back in the 1920s, Harold Nyquist from Bell Labs[1] stated that in order to properly reconstruct digital audio back to analog, the **sampling rate** needs to be double that of the frequency range. In other words, if you want to record frequencies between 20 and 20,000 Hz, the sampling rate must be 40,000 samples per second (and a little more than that, as seen in the **Antialiasing** section, below). This is because at a minimum, both the crest and trough of a wave cycle must be sampled *at least once*, so that it can be properly reproduced later. However, there is a little more to it than this. There must also be no frequencies present above *half the sampling rate*. This value is called the **Nyquist frequency** (also known as the **Nyquist limit**). If the Nyquist frequency is exceeded, an unwanted phenomenon called **aliasing** occurs.

Aliasing

We hit the proverbial ceiling when we reach the Nyquist frequency. The higher the incoming frequency, the fewer sample points are taken per sample period. Once we reach the upper limit, the audio can only be sampled twice: once for the compression, and once for the rarefaction of the incoming wave. If a wave is sampled

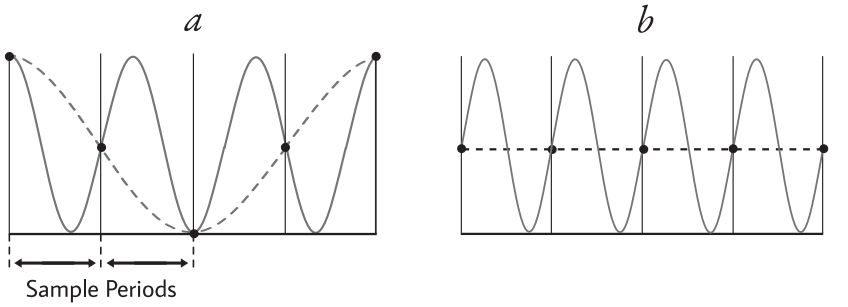


Figure 2.4 (a) and (b) Aliasing occurs when a frequency higher than the Nyquist limit is sampled. The solid waveform is the incoming sample; notice that the waves' crests and troughs appear within each sample period, instead of over two periods. The dotted lines show the aliased samples created.

over the Nyquist frequency, it cannot be properly reconstructed back into analog, instead causing **aliasing**. Aliasing can sound like clicks or pops, distortion, or even dropouts in the audio playback. Figure 2.4a shows an example of aliasing, where a frequency beyond the Nyquist frequency is sampled; when converted back to analog, it becomes the wave shown in the dotted line. Figure 2.4b shows another offender of the Nyquist frequency, whose zero points happen to line up with the start and end of each sample period, creating silence. In both cases, notice that the wave crests and troughs exceed each sample period. Remember, we need *more than double* the maximum frequency to sample properly, which means a wave's crest is in at least one sample period, and the trough in at least another. So how do we block out the unwanted frequencies?

Antialiasing (Low-Pass Filter)

The solution to this problem is **antialiasing**. Specifically, an antialiasing filter is a **low-pass filter** that can be applied to frequencies above the Nyquist limit. The filter is designed to only let waves in from the lowest frequency up to the specified highest frequency—in this case, the Nyquist frequency. It allows any frequencies set in the **passband** to come through and any in the **stopband** to be rejected. Ideally, we would like to have a "brickwall" filter, where the passband cutoff point was like an off switch. But in reality, filters are complex to design and can never fully prevent all frequencies beyond the Nyquist limit through—although they do an excellent job overall. Figure 2.5 shows our "dream filter" versus something more realistic. In the latter, there is a gradual ramp-down between the bands, called a **transition region**, where the amplitude is decreased until we reach the stopband.

This means we need to make a tweak to our sampling rates. Recall that if we wish to sample frequencies between 20 Hz and 20 kHz, we may think a 40 kHz sampling rate is needed. But we also have to consider the transition region of the low-pass filter, and so in this specific case, an additional 4,100 samples should be sufficient.[2] This gives us a 44.1 kHz sampling rate, which may sound familiar if you've ever looked at the properties of certain audio files.

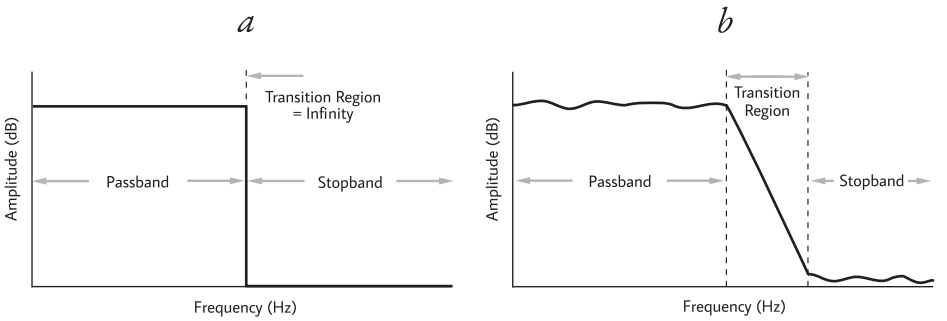


Figure 2.5 (a) An antialiasing "dream" low-pass filter versus (b) a more real-world depiction with a ramp-down transition region. The waviness of the bands in (b) is due to the nature of the electronic design of the filter.

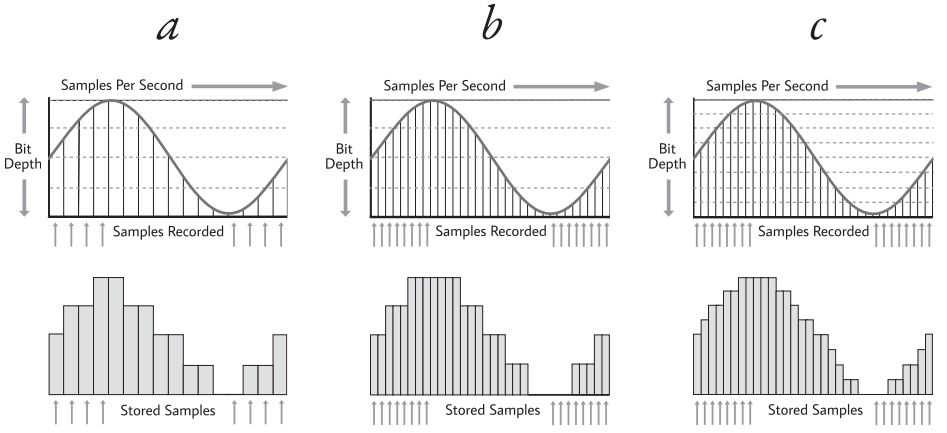


Figure 2.6 Samples recorded and stored at different rates and bit depths: (a) baseline sample rate and bit depth, (b) a higher sample rate, and (c) a higher sample rate and bit depth. The higher the both, the more the overall accuracy in preserving the original signal. The maximum stored sample value is limited by the bit depth.

Bit Depth and Quantization

In Figure 2.3, sample values are stored at discrete time intervals in samples per second, also measured in Hertz (this is the *digital* version of Hertz, not to be confused with its analog counterpart). A higher sampling rate captures the incoming analog signal with more resolution than at a lower rate (Figure 2.6).

As we saw earlier, the more bits that are stacked together, the higher a number value that can be represented. In the world of digital audio, the resolution of the amplitude effectively *doubles* with each new bit added; remember, the bits represent increasing powers of 2. So with higher bit depths, the loudness or quietness of the audio, known as its **dynamic range**, can be more accurately represented. The format is simple: 8 bits equals a bit depth of 8, 16 bits is a bit depth of 16, and so on.

Table 2.2 Sample Rates and Bit Depths versus File Size, for Uncompressed Audio Files (WAV, AIFF, etc.)

Bit Depth	*Sample Rate (kHz)*	*MB per mono minute (1 MB = 0.1024 bytes)*	*MB per stereo minute (1 MB = 0.1024 bytes)*	*Mono Bit Rate in kilobits per second (kbps)*	*Stereo Bit Rate in kilobits per second (kbps)*
8	11.025	0.631	1.262	88.2	176.4
8	22.05	1.262	2.523	176.4	352.8
16	22.05	2.523	705.6	352.8	504.7
16	44.1	5.047	10.094	705.6	1,411.2
16	48	5.493	10.986	768	1,536
24	48	8.239	16.479	1,152	2,304
24	88.2	15.141	30.281	2,116.8	4,233.6
24	96	16.479	32.959	2,304	4,608
32	96	21.973	43.945	3,072	6,144
32	128	29.297	58.594	4,096	8,192
32	192	43.945	87.89	6,144	12,288

The process of storing each sample value is known as **quantization**, and those values are binary integers. We can only store a sample value, or **quanta**, as high as the bit depth allows: an 8-bit audio is quantized into a range of integers from 0 to 255; with 16 bits, the range is from 0 to 65,535.

16-bit digital audio requires two bytes, or a **word**, for each sample. Higher bit values are still considered a word of audio, despite their size (a 24-bit word, 32-bit word, etc.). One downside to using higher bit depths is that it also increases the size of an audio file. Table 2.2 shows the bytes required for the most popular sample rates/bit depths, for uncompressed files (related to *file* compression, not audio compression!).

The **bit rate** (data rate) of digital audio can be found by multiplying the bit depth times the sample rate and is often measured in kilobits per second (kbps).

Quantization Error

A quantized value is most often an integer, so it is only ever an approximation of the incoming voltage.[3] Let's say the voltage is 0.3, and a word value of 00101010 (42) is stored. Then the voltage is 0.4, and a word value of 00101011 (43) is stored. Now the voltage is 0.35. What happens? The hardware can only choose to store 42 or 43, so it must either (a) round to the nearest integer value, up or down, or (b) truncate the second decimal place, no matter what value it is. Either way is not going to be correct, and this is called **quantization error**. The Least significant bit (LSB) has the hard job of deciding to lean toward a 0 or 1. Being one bit off in the LSB may not seem like a big deal, but if it is happening thousands of times per second, the audio is not going to sound very good! These errors pile up to create **distortion**, which is a distinct alteration of the original signal. One solution is to somehow randomize the values in our LSBs in a way that is reasonable to our ears. Enter **dithering**.

Dithering

Fortunately, a dither generator can help. Dithering is the process of adding low-level white noise to the analog signal before it is digitized. This noise, strangely enough,

diminishes the effect of quantization distortion. But how? Because the dither noise is random and mixes in with the original signal, the quantization errors now become more randomly distributed. Interestingly, the original signal is preserved, and the distortion is eliminated. However, we have gained a low level of noise. Now, nothing can be heard below the dithering (noise) level, so its decibel value becomes the new **noise floor**. At higher bit depths, noise is at a comparatively low level and is a much more preferable option than quantization distortion!. Many digital audio editors allow a setting to add dither when outputting a new file, as well as **noise shaping**, a process that balances out frequencies in the dither for an even more palatable end result.

Dynamic Range

Because higher bit depths represent more detailed amplitude values, it also means we have more granularity between the quietest sound we can hear and the loudest one. This range between quiet and loud is known as the **dynamic range**, and its value increases as the bit depth increases.

Knowing the noise floor value is important, as it affects what can be heard upon playback, and determines the overall quality level of the audio itself. The **dynamic range** of digital audio is the **maximum amplitude minus the noise floor**. It can also be determined roughly as:

Dynamic Range (dB) = Number of Bits × 6

So, each additional bit gives us 6 dB more for dynamic range. 24 bits gives us an astounding (24 × 6 =) 144 dB of dynamic range[4]; 16-bit audio has a 96 dB dynamic range; for 8-bit audio, that value is 48 dB; and for 4 bits (please, not 4 bits!), it is a meager 24 dB. This is why low-bit audio sounds as bad as you think it does. Much of the early digital audio in games was 8-bit 8,000, 11,025, and 22,050 Hz, which made for some challenging creative compromises. You can also see that if you make use of the maximum amplitude available, otherwise known as the **maximum headroom,** you can effectively increase your dynamic range. Audio consistently recorded at a lower level further decreases the dynamic range, so it always pays to record or output your audio at an optimum volume. Dynamics control is a topic of much interest in digital audio production, which we will discuss in Chapter 6.

Digital Audio Recording

Pulse Code Modulation (PCM)

The most common form of recording digital audio is through the use of **pulse code modulation** (PCM). It is a process where analog audio is sampled at specific intervals and encoded as digital audio, typically as one (mono) or two (stereo) channels.

Analog-to-Digital Recording Process

Figure 2.7 shows the steps in converting analog audio to digital.

First, the analog audio is fed into the system, which could be one or more channels of audio (for example, stereo would be two channel inputs, a left and a right). But before it is digitized, a dither generator is applied to the audio, followed by an antialiasing filter.

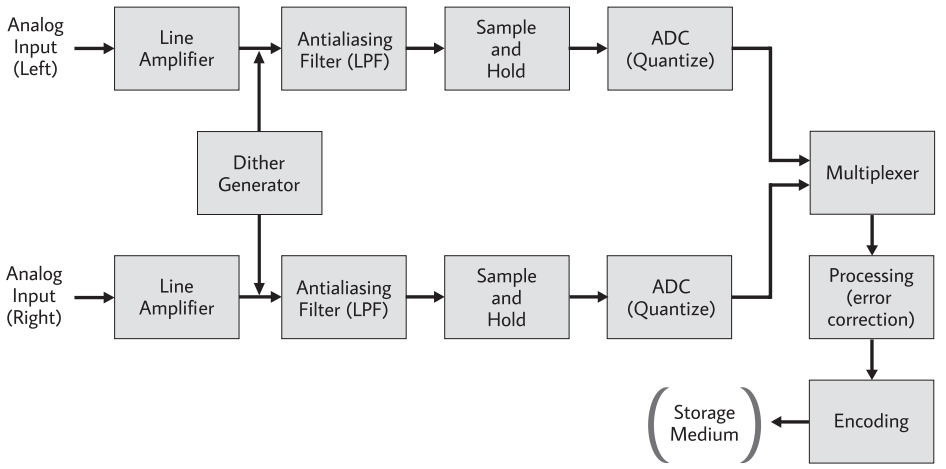


Figure 2.7 The steps required to convert stereo analog audio to digital, in a PCM recording design.
Source: Adapted from Pohlmann (2011).

The Sample and Hold circuit takes the incoming amplitude and samples it at a precise time, holding it in place while the A/D converter digitizes it. If the amplitude was not held during the conversion, it could change to any value and the digital conversion would not be correct. With the Sample and Hold circuit, time is broken into discrete segments, but the amplitudes are still analog.

The audio is then quantized into a digital value, and here both time and amplitude are now discrete.

The final encoding portion involves a few more steps. First, a circuit called a multiplexer serializes the audio into one layer; that is, if there are two or more channels of audio, they are **interleaved** into a single layer of binary data: for example, four channels of audio would interleave as 1–2–3–4-1–2–3–4 etc., one data frame (as a byte or word) at a time from each channel.

Next is the error correction stage. Without this critical step, digital audio playback could be wrought with glitches, dropouts, and other auditory problems. If the error is in the LSB, it will go unnoticed to the listener. But errors in higher-order bits (especially in the MSB) may cause significantly audible errors. Redundancy can help with this problem: the data is grouped into blocks of a certain size, and a small amount of the data is inserted into each block known as parity bits. Through complex coding algorithms, if errors in the data bits are detected, the redundant data can help repair those incorrect bits, as a kind of checks-and-balances implementation. There are many error correction schemes in use, and this is a fascinating topic in itself. If you'd like to know more, I highly recommend watching "Hamming Codes and Error Correction" from **3Blue1Brown** on YouTube to find out more.[5]

The final output is then encoded as a "raw" PCM stream (not file formatted) that can be stored into memory, disk, or other storage media. Just the actual data itself—the error correction data is not stored.

This is an example of a typical PCM encoding process, but there are other variations on this system such as PWM (pulse width modulation), PAM (pulse amplitude

modulation), and PPM (pulse parameter modulation). These other types are not generally used for audio recording, however, due to their lack of bandwidth and/or noise generation issues.

Digital Audio Playback

To convert PCM digital audio back to analog, we trace our previous steps backward, with a few slight modifications. The bitstream is demodulated (decoded) and the error correction techniques are used in the processing section. Next the data is demultiplexed, restoring the proper number of audio channels and their data streams.

The next steps are critical to ensuring the best quality reproduction possible. The DAC must convert the binary amplitude values to analog voltages. So, if 32-bit audio is being converted, that means over *four billion* different voltage values are possible, which demands extremely high precision to reproduce accurately. Errors are bound to occur, and the Sample and Hold circuit here (being used a little differently) can help. Each new analog voltage that the DAC creates is sampled and held for evaluation; if a glitch is detected, the voltage is corrected to the proper value. Lastly, an output low-pass (antialiasing) filter is applied, but it is used somewhat differently than the input low-pass filter. It first removes any frequencies above the Nyquist limit, but it is also utilized as a filter that smooths out the analog voltages. Then the analog signal is sent out to headphones or speakers, and *voilà*, we have seamless analog audio once again (Figure 2.8).

CD-Quality Audio and Beyond

When compact discs were first introduced, their audio quality was full-fidelity; that is, the human hearing range of 20 Hz–20 kHz. This is also known as CD-Quality audio, which is 16-bit, 44,100 Hz. Today, sample rates can be much higher, capturing ultra-high frequencies with many more samples per second. For example, the PS3 can play a Super Audio CD (SACD) format of 32-bit, 192 kHz audio. But in games, the audible difference is negligible, and the cost of digital space and conversion time is not worth it, yet. Typically, most games have 16 bit, 48,000 Hz audio as their highest quality, though some blockbuster games will have 24-bit audio playback as an option.

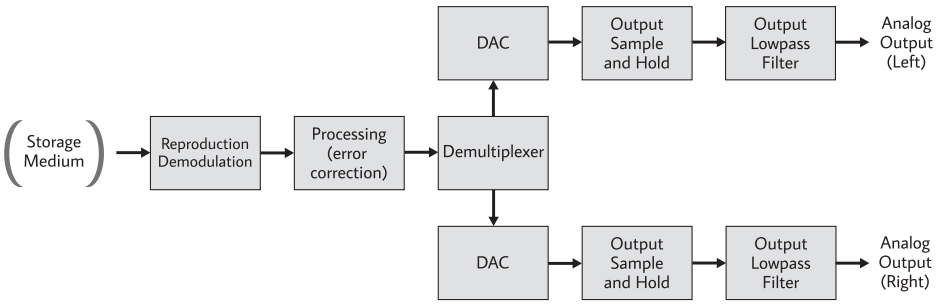


Figure 2.8 The steps required to convert stereo digital audio back to analog, in a PCM reproduction design.

Source: Adapted from Pohlmann (2011).

Notes

1 Claude Shannon also worked on sampling theory, as well as E.T. Whittaker independently, so the sampling theorem is sometimes credited as Nyquist-Shannon, or Nyquist-Shannon-Whittaker.
2 A value that dates back to the days of encoding and decoding for PCM video cassettes: https://en.wikipedia.org/wiki/44,100_Hz
3 32- and 64-bit audio files are stored as floating point but at the expense of bandwidth.
4 This number is theoretical; currently, the best dynamic range any ADC hardware can offer is about 125 dB. Source: http://skywired.net/blog/2011/09/choosing-high-performance-audio-adc/
5 "Hamming Codes and Error Correction" by 3Blue1Brown: https://www.youtube.com/watch?v=X8jsijhllIA. Retrieved 7-29-2021.

Bibliography

Aldrich, Nika. *Digital Audio Explained for the Audio Engineer*. 2nd ed. Fort Wayne, IN: Sweetwater Sound, 2005.

Ballora, Mark. *Digital Audio and Acoustics for the Creative Arts*. New York, NY: Oxford University Press, 2017.

Cancellaro, Joseph. *Exploring Sound Design for Interactive Media*. Clifton Park, NY: Thomson, 2007.

Loy, D. Gareth. *Musimathics: The Mathematical Foundations of Music* (Volume 1). Cambridge, MA: MIT Press, 2006.

Pohlmann, Ken C. *Principles of Digital Audio*. 6th ed. New York, NY: McGraw-Hill, 2011.

Watkinson, John. *An Introduction to Digital Audio*. London, UK: Routledge, Taylor & Francis Group, 2017.

Chapter 3

Music Theory for Audio Designers

Have you ever heard somebody say, "I don't like music"? Of course not! (If you have, please run in the opposite direction immediately.) All of us are drawn to at least *some* music that speaks to our soul. As audio designers – regardless of the level of our musical abilities – we owe it to ourselves to not just appreciate but to understand and at times implement audio that is firmly grounded in music theory. Perhaps we're adding sound effects that are nicely paired with the background music, or layering audio elements that must sound harmonious together. In any event, having foundational knowledge in this area is another key building block in the success of your audio creations.

A Crash Course in Music Theory

Pitch

Earlier in this chapter, pitch was given as a term to define a sound wave as vibrationally higher or lower. In music, **pitch** is defined as a discrete frequency, otherwise known as a **note**. As words are a part of language, so are notes a part of music. Notes are assigned one of seven letters in the alphabet: A, B, C, D, E, F, or G. Notes can be played in any order to form **melodies**, or together to make richer forms known as **chords**, which will be covered in just a bit.

There are 12 notes (also known as **tones**) in Western music that are ordered using a method called **equal temperament**. Here, notes are spaced equally apart from one another. And since we are only using A through G as our letter set, the additional notes need to be inserted in between at certain points (they have been mapped out in a specific way for various historical and mathematical reasons). Here are the notes as seen on a piano keyboard:

On the white keys of the piano keyboard, we see A, B, C, D, E, F, and G, also known as natural notes, or **naturals**. These were the original seven notes played in ancient times on such instruments as the Greek lyre. Notes increase in pitch as we travel to the right of the keyboard, and lower in pitch to the left. Between the white keys are a series of black keys, which are known as the **accidentals**, or **sharps** and **flats**. If you are traveling up the keyboard starting at A, the next note you will find is the first black key on the right. Notice there are two names for this key: A# and Bb, or A-sharp and B-flat, respectively. They are **enharmonic**, meaning each accidental has two names that represent the same note. Casually, it is fine to call this note either A# or

DOI: 10.4324/9781003218821-5

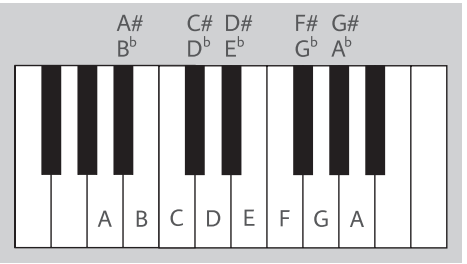


Figure 3.1 Notes of the 12-tone Western scale as seen on a piano keyboard.

Bb, and the next accidental C# or Db, etc., but as we will see when we discuss **scales** below, sometimes we must keep to a specific name for each accidental, in spreading out the alphabetical names in a scale as best as possible.

On the piano, the distance between any note and its nearest neighbor is called a **half step**. The term **sharp** also means to "go up one key" or one half-step above a note (to the right), and the term **flat** means to "go down one key" or one half-step below a note (to the left). A **whole step** is made up of two half steps. So going a half step up from A, you would arrive at A#; going a half step down from B brings you to Bb. Notice in both cases, you ended up at the same key! Follow these examples using Figure 3.1: a whole note up from F is G, a whole note down from Gb is E, and a whole note up from B is C#.

Once we travel from our first A note up the keyboard and arrive at G#, the naming convention starts all over again with a new A note. The lower and higher A notes both sound and "feel" similar; yet, clearly one is higher in pitch than the other. They happen to be a distance of one **octave** apart, and the reason they sound similar is that the lower note is twice the frequency of the higher one.[1] If our lower A is 440 Hz, then the A note an octave higher is 880 Hz; if we go an octave lower than 440 Hz, the new lower A is half the value at 220 Hz, and so on. Interestingly, if you were to play a note and its octave on a guitar, one string would be exactly half the length of the other.

Notation

Traditionally, notes and their durations are written on a **staff**, comprised a series of lines and spaces to place them in. As seen in Figure 3.2, a staff has five lines with four spaces in between, and the placing of specific notes depends on their pitch. Lower notes are placed lower on the staff, and vice versa.

There can be one of several different symbols applied to the beginning of a staff that change the pitch values of the lines and spaces, called a **clef**. Generally, music resides in high, middle, and low registers of pitch, and a clef is the symbol that tells the performer which register to be in. You will mainly encounter two types of clefs, the **treble** ("G") **clef** and the **bass** ("F") **clef**. In Figure 3.2, the treble clef is up top and the bass clef on the bottom.

The treble clef is known as the G clef, as its circular loop is centered around the G line. The F clef gets its name for the two dots placed between the F line. Notes are placed on the staff in order of performance, from left to right. If note values go

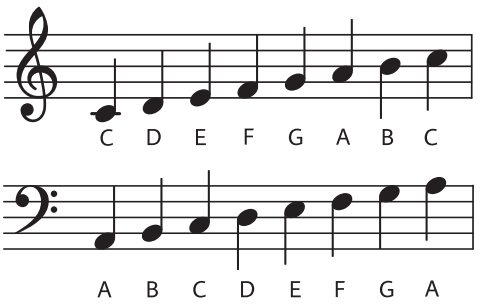


Figure 3.2 The C major scale in the treble clef, and the A minor scale in the bass clef.

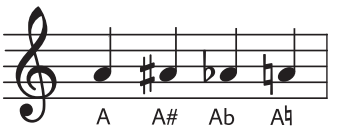


Figure 3.3 A natural, A sharp, A flat, and returning the accidental to make A natural again.

beyond the staff in either direction, a short, horizontal **ledger line** is added for the note. In Figure 3.2, the first C note in the treble clef falls below the staff, so a ledger line is added there.

If a natural note such as A needs to become G# (sharp) or Gb (flat), the appropriate accidental symbol is applied, as in Figure 3.3. When a vertical bar is reached, that is the end of a **measure** (also known as a **bar**), which we will learn more about later in this chapter. If a note is made sharp, flat, or natural, that note will keep its value until the end of a measure, unless it is changed again before that.

Natural notes are denoted with the ♮ symbol (as seen in the last example of Figure 3.3), but unless that note was previously marked as a sharp or flat, it is not normally used.

Often times, music is composed on a wider range than a single staff can handle, especially when scores are written for left- and right-hand playing on a keyboard or for the various instruments in an orchestra. A **grand staff** can be used to view these notes more clearly. In the grand staff, there is also an invisible line between the treble and bass clef that represents **middle C**. Figure 3.4 shows middle C on an 88-key piano, as well as all of the positions and frequency values for A notes for further reference.

When composing electronically, music can be written either via digital notation or with notes represented as individual pitch values and their durations using a **sequencer** (more on this in Chapter 4).

Scales

A **scale** is a specific collection of notes. The collection of all 12 notes (A through G#) is in every octave and is known as the **chromatic scale**. This is the master scale that encompasses all notes in music. Normally, when a complete scale is written or

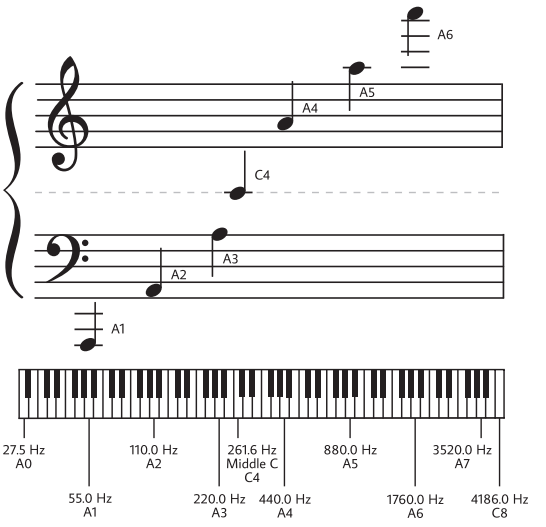


Figure 3.4 The grand staff with its invisible middle C line (dotted) as well as some useful reference notes and their frequency values, shown both on the staff and on an 88-key piano.

performed, the octave note is added to the end as it gives a sense of completion; we will resolve our scales in that fashion here.

The two most common scales in Western music are the **major** and **minor** scales. The major scale has a feeling often described as happy or uplifting, while the minor scale could be described as feeling sad or serious. Major and minor scales are two types of **diatonic** scales, with the word *diatonic* meaning they are made up of seven notes (and eight if you add the octave). They are a specific combination of five whole steps and two half steps each. Going up the staff or keyboard, we can look at the intervals between each pair of notes, which is either a whole step or a half step.

In the topmost example of Figure 3.6, the notes go up the C major scale: C, D, E, F, G, A, B, and the C octave note added for completion. Notice the number of (W)hole or (H)alf steps between each note; you can also refer back to the keyboard in Figure 3.4, or the note intervals in Table 3.1 to count along. The pattern for any major scale is **W, W, H, W, W, W, H**. The A minor scale is A, B, C, D, E, F, G, and the A octave note, and for any minor scale, the pattern is **W, H, W, W, H, W, W**.

Intervals

Any two notes in a scale can be played melodically (separately) or harmonically (together). The distance between them in half steps is called an **interval**, which is an important concept that will be used to define note relationships going forward.

Table 3.2 lists the names of each interval along with relevant information:

The term **perfect** denotes that the interval has no major or minor counterpart; it exists on its own. There is also the **tritone** interval (it is named "tritone" as it is made up of three whole tone intervals), which also goes by two other names: an augmented 4th (or +4) and a diminished 5th (or °5). To **augment** an interval is to raise it by one half

Table 3.1 Whole and Half Step Intervals for C Major and A Minor Scales

	C Major Scale			*A Minor Scale*	
Note Interval	*Number of Half Steps*	*Interval*	*Note Interval*	*Number of Half Steps*	*Interval*
C to D	2	Whole	A to B	2	Whole
D to E	2	Whole	B to C	1	Half
E to F	1	Half	C to D	2	Whole
F to G	2	Whole	D to E	2	Whole
G to A	2	Whole	E to F	1	Half
A to B	2	Whole	F to G	2	Whole
B to C	1	Half	G to A	2	Whole

Table 3.2 Intervals and Distances in Half Steps

Interval Name	*Shorthand Name*	*# Half Steps Away from Unison*	*Examples Using A*
Perfect Unison	U, or P1	0	A to A
Minor 2nd	m2	1	A to A#/Bb
Major 2nd	M2	2	A to B
Minor 3rd	m3	3	A to C
Major 3rd	M3	4	A to C#/Db
Perfect 4th	P4	5	A to D
Tritone, aka Augmented 4th or Diminished 5th	A4 (+4) or d5 (°5)	6	A to D#/Eb
Perfect 5th	P5	7	A to E
Minor 6th	m6	8	A to F
Major 6th	M6	9	A to F#/Gb
Minor 7th	m7	10	A to G
Major 7th	M7	11	A to G#/Ab
Octave	O, or P8	12	A to A

step; to **diminish** an interval is to lower it by one half step. The tritone interval has a very dissonant-sounding feel to it and used to be omitted from early music – it was known in the Dark Ages as the "Devil's Interval." Think pipe organs and Halloween, and there's your tritone![2] When you hear it, you may feel like you want the interval to resolve up to a 5th or down to a 4th.

Harmonic versus Melodic Intervals

When two notes are played at the same time, a **harmonic interval** is created. When played separately over time, a **melodic interval** is created.

Compound Intervals

When an interval is made beyond the one-octave range, it also becomes a **compound interval**. Functionally speaking, it is similar to its equivalent interval one octave in size or smaller (known as a **simple interval**). So, the interval C2–D3 is a 9th but can

also be called a compound 2nd (C2–D2). The interval A3–D4 is a 10th but can also be called a compound 4th (A3–D3), etc. We're basically subtracting 8 (the octave) from the higher number.

Consonance and Dissonance

To say that an interval has **consonance** means it feels **resolved**, or "stable," meaning pleasant to hear. If the interval had **dissonance**, it would feel **unresolved**, and "unstable", meaning it may sound like we want the interval to **resolve** to another interval. There is some wiggle room on these terms, as it sometimes depends on their context within a song.

Consonant intervals are unison, m3, M3, P4, P5, m6, M6, and octave.

Dissonant intervals are all the rest: m2, M2, A4/d5, m7, and M7.

Key Signatures

In most cases, the **key** of a song is the scale that it is centered around. Chords and melodies can change, but its unifying key holds true throughout the piece. It is also the place that composers can return to for a sense of resolution, musically speaking.

What's more, using a **key signature** shows us the song's main scale, by defining all of its accidentals just after the clef. At any time, you can override this key signature by adding another accidental or natural before a note, which will last until the measure ends.

The accidentals are written exclusively as all sharps or flats. The best way to determine which accidentals to use can be found by looking at a **note circle**, also known as a **circle of fifths**, as each note on the circle is a musical fifth apart from one another.

It works like this: if your song is in a major key, then beginning at C (at the top left of Figure 3.5), move clockwise around the circle to find the number of **sharps** for that

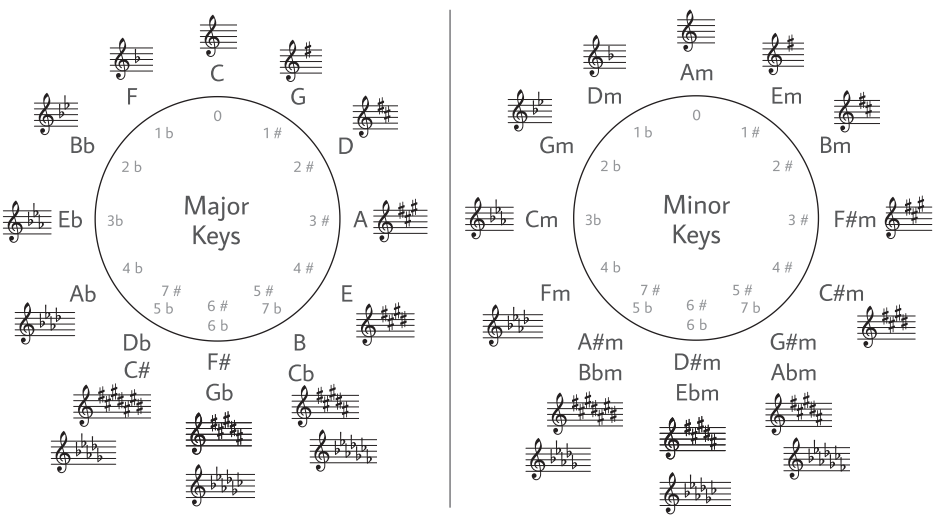


Figure 3.5 The circle of fifths, split into major (left) and minor (right) note circles for easier viewing.

key. Or, move counterclockwise to find the number of **flats**. For example, if your song is in G major, move ahead clockwise to G and you'll see we have one sharp in that key (F sharp). If your song is in B flat major, move counterclockwise until you locate the key of B flat. Here, you'll see that this key signature uses two flats (B flat and E flat).

For minor scales, another note scale is used, in the minor note circle of Figure 3.5. Here, you are starting at A minor and moving in either direction as above. So, A minor has no flats or sharps, B minor has two sharps, and E flat minor has six flats.

You'll also notice in the major note circle that there is a "C flat." Why not just write "B"? This is because the scale needs to be described **enharmonically** – that is, we must have one unique letter for each note in the scale. We've already used B flat, so we're left with referring to the new flat by an unused letter with the same meaning.

Here's something very interesting: when comparing a similar location in both note circles, the number and position of the sharps and flats for any major and minor pair remain the same. For instance, the C major scale is equivalent to the A minor scale, with no sharps or flats; the Gb major is the same as the E flat minor scale, with the same five sharps and five flats, and so on. The two circles are a minor third apart; another way to describe it is that C major's **relative minor** is A minor, and A minor's **relative major** is C major.

Key Signatures for Game Audio Designers

As audio designers, it's really handy to know what key a song is in. This can define what key or frequencies your most important sound effects might center around, such as for pickup sounds, special events, or user interface (UI) buttons, to name a few examples.

A Closer Look at Scales

Contrary to popular belief, there are far more scales at our disposal than just major and minor. As noted earlier, these are also part of the **diatonic modes** (*mode* is Latin for "method"). There are a total of seven modes which are scales, and if we were to use only the natural notes to create them, they would be ABCDEFG, BCDEFGA, CDEFGAB, DEFGABC, EFGABCD, FGABCDE, and GABCDEF. While this pattern of scales seems straightforward at first glance, it is worth experimenting in these seven modes to hear the wondrous change in feel and emotion each one can create. Within each, the harmonic center changes, as well as the interval relationships between the notes. Please refer to Figure 3.6 for the diatonic modes, all in the key of C. The octave note for each scale will be added after a "|" bar symbol below.

To hear what each one sounds like, hold down a low C note with your left hand on a music keyboard, then try out the first scale (C Ionian) from Figure 3.6 on your right hand. Next, hold down a low D note and try the next scale (D Dorian) on your right hand, and so on, listen to the harmonic tensions between each low note and with its corresponding scale, and hear the power of the modes!

Ionian

This is the major scale we all know and love. Playing the Ionian scale in the key of C, more commonly known as C Ionian, it is CDEFGAB|C, with no sharps or flats. Pure and simple!

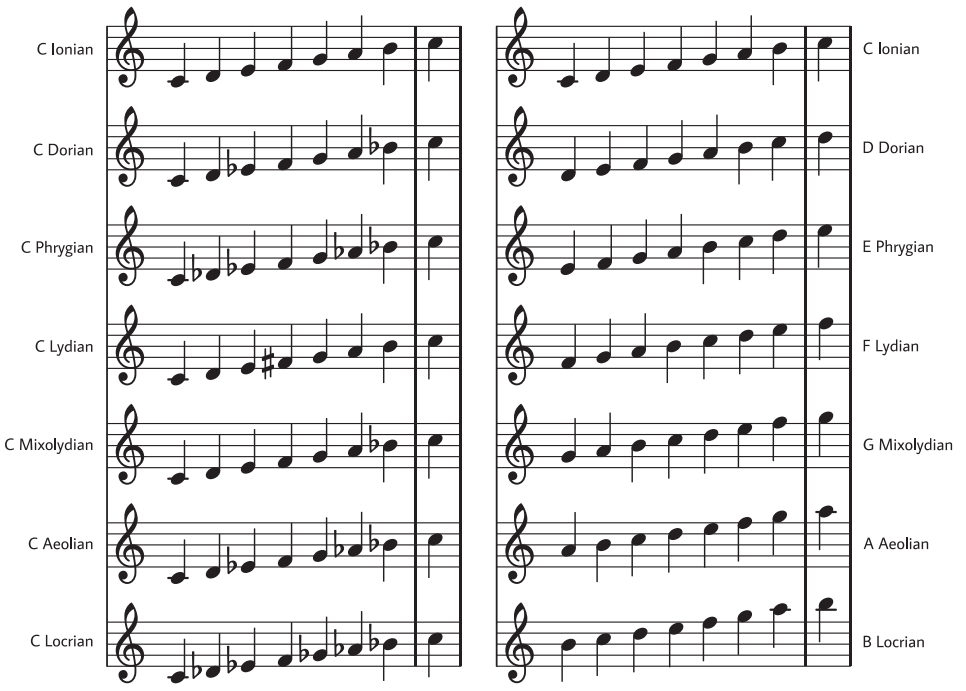


Figure 3.6 Diatonic modes in the key of C.

Dorian

This scale, DEFGABC|D (D Dorian), sounds "almost" minor and would be, if the B was a half-step lower: here, the B note is a major 6th instead of a minor 6th. C Dorian has two flats, Eb and Bb.

Phrygian

This also sounds similar to a minor scale, except here with EFGABCD|E (E Phrygian) there is a minor 2nd instead of a major 2nd. C Phrygian has four flats, Db, Eb, Ab, and Bb.

Lydian

A musical friend of mine once said that "Lydian is the mode that can save the masses." It is a beautiful and haunting scale with a sharp 4th, FGABCDE|F (F Lydian). C Lydian has one sharp, F#.

Mixolydian

This is also similar to a minor scale, except there is a flat 7th instead of a major 7th: GABCDEF|G (G Mixolydian). C Mixolydian has one flat, Bb.

Aeolian

This scale is the classic natural minor scale we're all familiar with, ABCDEFG|A (A Aeolian). C Aeolian has three flats: Eb, Ab, and Bb.

Locrian

Possibly the most mysterious sounding of the modes, Locrian is BCDEFGA|B (B Locrian). C Locrian has five flats: Db, Eb, Gb, Ab, and Bb.

Melody and Motifs

A **melody** is a distinct series of notes in a musical piece. It may represent a character's main **theme** (the melody's main subject) or a passing interlude in a song, for example. A **motif** is a musical "hook" that has a short tonal or rhythmic pattern to it. It is repeated throughout a song, often as part of the melody.

Harmony and Chords

If more than one note is played simultaneously, they are in **harmony** with one another. Whether a particular harmony is pleasing or not is subjective, but the two most common harmonies in Western music are major and minor thirds. The former is normally associated with happy, good, or pleasant moods, and the latter with sad, evil, or more serious moods.

If three or more notes are played simultaneously, a **chord** is formed. Chords are based on scales, making it easier to build them. Basic major chords involve the root (first note), major third, and fifth, such as C, E, and G – a C major chord. Basic minor chords involve the root, minor third, and fifth, as with C, Eb, and G. Chords can become quite complex, involving higher-order notes in a scale. Table 3.3 shows some

Table 3.3 Common Western Chords

Description	*Chord Name with C as Root Note*	*Structure with C as Root Note*	*Emotional Association*
Suspended 2nd	Csus2	C-D-G	Neutral, inoffensive, agreement
Major third	Cmaj	C-E-G	Happy, good, lighthearted
Minor third	Cmin	C-Eb-G	Sad, evil, serious
Suspended 4th	Csus4	C-F-G	Anticipation, suspense (waiting to resolve to a major or minor chord)
Diminished	C°	C-Eb-Gb	Horror, fear, sudden surprise, shock
C major 7th	Cmaj7	C-E-G-B	Loving, pleasant, happy, content
C dominant 7th	C7	C-E-G-Bb	Slight tension, anticipation
C minor 7th	Cmin7	C-Eb-G-Bb	Dark, serious, confident, angry
C major 9th	Cmaj9	C-E-G-B-D	Somewhat brighter than a major 7th
C minor 9th	Cmin9	C-Eb-G-Bb-D	Somewhat less dark than a minor 9th, increased anticipation

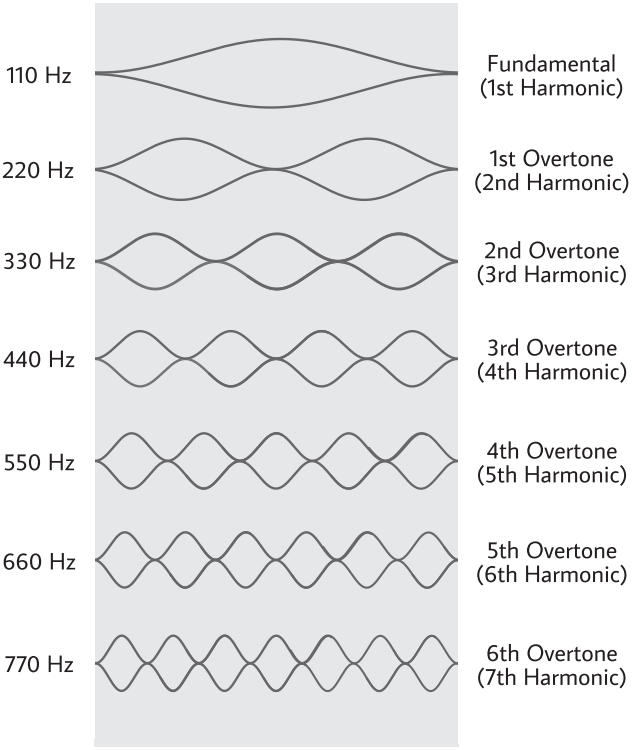


Figure 3.7 The harmonic series of 110 Hz (note A).

popular chords in Western music, along with their emotional associations (in this author's opinion).

Harmonics and Overtones

Every sound, save for a pure sine wave, has **harmonics** associated with it. These are frequencies that vibrate at integer multiples of the **fundamental** (original, or root) frequency. We may only hear specific harmonics depending on the envelope and overall character (woody, metallic, buzzy, etc. quality) of the sound, but typically the fundamental is the dominant harmonic that is heard.

The first harmonic is always the fundamental; the second will vibrate *two times* faster than the fundamental; the third, at *three times* the fundamental; and so on, theoretically forever. We have all heard harmonics in everything from guitars, pianos, flutes, the human voice, and electronic instruments, to name a few sounds. Here is the harmonic series from the fundamental to sixth harmonic, for the note A at 110 Hz (Figure 3.7).

Harmonics can be heard most clearly in a tuning fork. Figure 3.8 is a graph of an A 440 Hz tuning fork ringing out. You can see the dominant frequency is the fundamental at 440 Hz, followed by second harmonic at 880 Hz, the third harmonic at 1.32 kHz, the fourth at 1.76 kHz Hz, and so on, in multiples of 440 Hz.

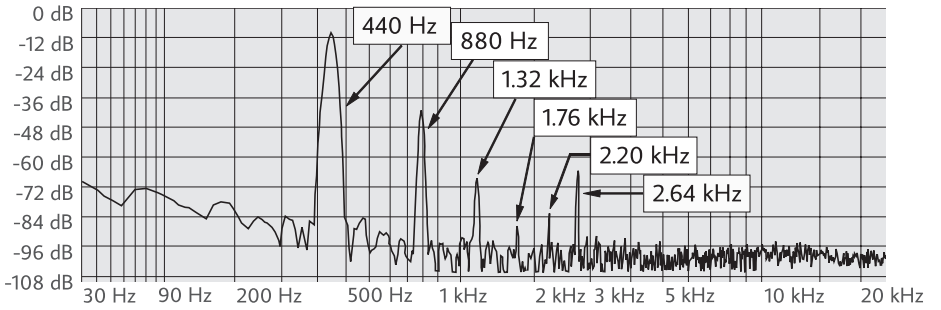


Figure 3.8 A 440 (A3) Tuning Fork Frequency Analysis. The fundamental is prominent at 440 Hz, followed by its additional harmonics (880 Hz, 1.32 kHz, 1.76 kHz, etc.).

Table 3.4 Harmonics and Overtones of 110 Hz (A1)

Frequency	*Overtone*	*Harmonic*	*Value*	*Note*
110	n/a	1st	Fundamental	A1
220	1st	2nd	Octave	A2
330	2nd	3rd	Perfect 5th	E3
440	3rd	4th	Octave	A3
550	4th	5th	Major 3rd	C#4
660	5th	6th	Perfect 5th	E4
770	6th	7th	Minor 7th	G4
880	7th	8th	Octave	A4
990	8th	9th	Major 2nd	B4
1.1 kHz	9th	10th	Major 3rd	C#5
1.21 kHz	10th	11th	Tritone	D#5
1.32 kHz	11th	12th	Perfect 5th	E5
1.43 kHz	12th	13th	Minor 6th	F5
1.54 kHz	13th	14th	Minor 7th	G5

Overtones are any audible frequencies over the fundamental. Think of overtones as being "over the main tone." In the tuning fork figure, note that the overtones begin after the fundamental: the first overtone is the second harmonic, the second overtone is the third harmonic, and so on. If this all sounds confusing, Table 3.4 clearly lays out the relationship of frequencies, harmonics, and overtones, again using the note A1 at 110 Hz as an example.

Beats and Rhythm

Up until now, you have seen one type of note for representing pitch. But to represent the length of a note, we need to understand the division of **beats**. Let's say we know that four beats occupy one measure, in even intervals. If you tap along to your favorite tune, chances are it has four beats to the measure as well (also called **common time**, as seen below). We have seen **quarter notes** in all of the chapter examples so far, as indicated by a solid dot with a vertical stem. If our four beats are quarter notes, then

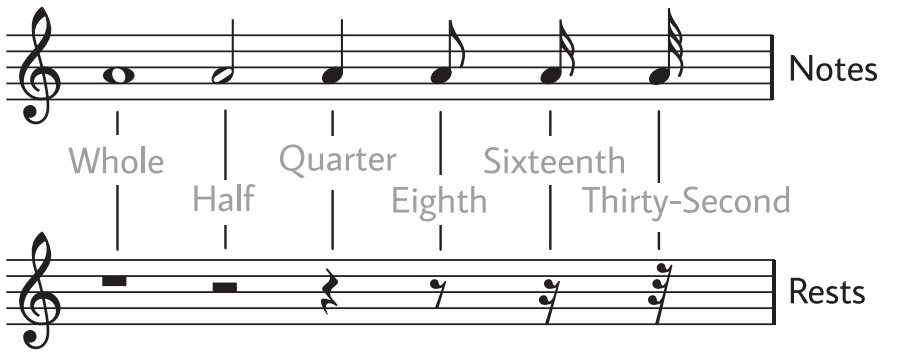


Figure 3.9 Various note and rest values on a staff.

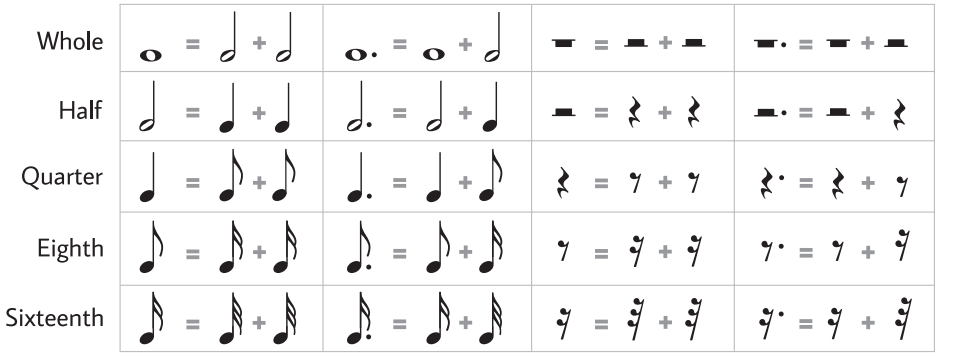


Figure 3.10 Note and rest equivalents.

we know that four of them fit into each measure |1–2–3–4– | 1–2–3–4– |, etc. While each percussive note (a snare drum, for example) has a short duration, each musical note must have a duration indicated by its symbol at the top of Figure 3.9.

Conversely, *not* playing a note is also indicated by a **rest** value, seen at the bottom of Figure 3.9. Adding a dot symbol to the right of the note or rest adds an additional half value to it. For a more detailed look, Figure 3.10 shows all of the common note and rest values.

Tempo and Time Signatures

How fast or slow is a song? The pace of a musical piece is described with **tempo** and can be defined with **beats per minute (BPM)**. This is a precise value used in electronic-based composing including MIDI devices and Digital Audio Workstations (DAWs). Traditional scores may also use standard tempo descriptions (in Italian) to describe the speed, in a general range of pacing, as seen in Table 3.5.

Finally, we just need to know how many of a particular note adds up to one measure. Normally, the quarter or eighth note is used to define this value, known as our **time signature**, located at the beginning of the first measure. The number of beats per

Table 3.5 Common Tempo Descriptions

Tempo (approx. BPM)	*Italian Name*	*Description*
20–40	Grave	Extremely slow, dirge-like
40–60	Largo	Very slow, broadly
60–76	Larghetto	Somewhat slow, broadly
66–76	Adagio	Slow pace, stately
76–108	Andante	Walking speed
108–120	Moderato	Moderate speed
120–156	Allegro	Fast, briskly
156–176	Vivace	Fast, lively
168–200	Presto	Extremely fast, blazing speed!

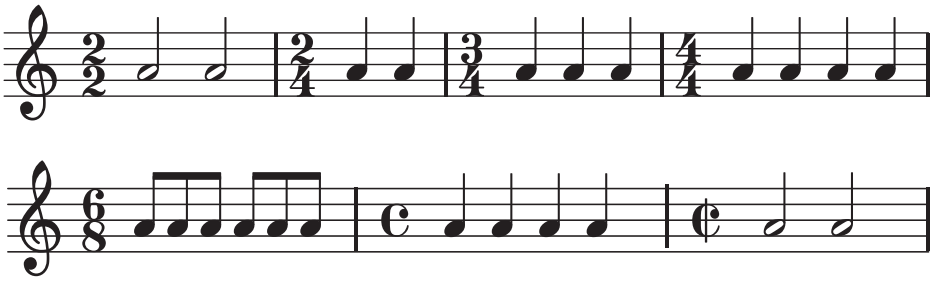


Figure 3.11 Common time signatures, including common time (as noted by the C symbol) and cut time (the C with a vertical slash, adjacent).

measure is listed first, then beneath that value is the type of note that gets the beat: 1 for whole, 2 for half, 4 for quarter, 8 for eighth, and so on. For example, a time signature of 3/4 means there are three beats per measure, and the quarter note gets the beat. A time signature of 2/2 means there are two beats per measure, and the half note gets the beat. Of course, all kinds of notes and rest can be used at any time, so long as they add up to the value determined by the time signature with each measure; so if you are in 4/4 time, all of your notes and rests must be equivalent to four quarter notes per measure.

Several examples are given in Figure 3.11. The 4/4 time signature has been so commonly used in music history, it has its own symbol, a capital letter C, also known as **common time**. 2/2 is also very common and has its own symbol, a capital C with a vertical slash through it; it is also referred to as **cut time**.

The time signature need only be displayed once per staff, per page of music. However, if the time needs to change, ideally the new value appears in the affected measure, permanently overriding the old time. If no time signatures appear at all on a page, it is assumed to be common time.

Ties, Slurs, and Stems

Sometimes a note must be played across two or more measures, and a **tie** is used to group them together, seen in Figure 3.12. If two *different* notes are played across

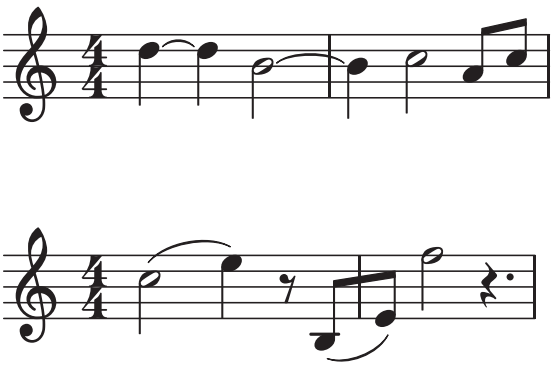


Figure 3.12 Examples of ties (top) and slurs (bottom).

measures, or are to be held together as close as possible in the same measure, a **slur** is used, also in Figure 3.12.

On the subject of stems: if a note is placed on the middle line of the staff, the stem points down instead of up (as seen in Figure 3.5).

Conclusion

The world of music and music theory is vast, and this chapter serves to familiarize game audio designers with the basics. There is much, much more to explore, but this should be a good starting point in your musical adventure!

Notes

1 Using the seven-note Greek scale of ABCDEFG, Pythagoras discovered that if he added an additional string half the length of A, it was a higher-register version, or octave, of the lower A.
2 For more history on this spooky interval, check out https://www.npr.org/2017/10/31/560843189/the-unsettling-sound-of-tritones-the-devils-interval.

Bibliography

Benade, A. H. *Fundamentals of Musical Acoustics*. 2nd rev. ed. New York, NY: Dover Publications, 1990.

Cancellaro, Joseph. *Exploring Sound Design for Interactive Media*. Clifton Park, NY: Thomson, 2007.

Clendinning, Jane Piper, and Elizabeth West Marvin. *The Musician's Guide to Theory and Analysis*. Vancouver, BC: Langara College, 2022.

White, Harvey Elliott, and Donald H. White. *Physics and Music: The Science of Musical Sound*. Mineola, NY: Dover Publications, Inc., 2019.

Chapter 4

Production Tools

Your Project Studio

It's time to set up the project studio!

But where to begin?

You may already have a computer, or a spare laptop, and some earbuds but not much else at the moment. You may also be wondering: "how do I get audio into my computer? What about good microphones or speakers? And what's the best software to use, for what I want to accomplish?"

When it comes to setting up your studio, there are so many options available it can be dizzying. But I hope to alleviate your anxiety here, and to give you a tour of the most crucial pieces of gear. Be careful, though, when you get paid for your first game audio gig – it's going to be tempting to spend the proceeds on newer, better equipment. Then you might do that again after the next gig, and the next, and … well, I'm afraid at this point you have succumbed to the sickness that afflicts many an audio designer; you are now officially a *gearhead*!

Hardware

Computer and Storage Considerations

The first thing we need to do is secure some hardware for our studio, and computers are the perfect place to start.

Desktop versus Laptops

The home desktop computer has been a mainstay of productivity since the late 1970s, with faster processing power always around the corner. More recently, laptops and notebooks have surpassed desktops in sales, and it's no wonder, as they are effortlessly portable and still quite powerful computing devices relative to desktops. But there are some significant trade-offs: laptops are more expensive than desktops, when comparing their CPU speed, memory, and audio/visual capabilities. Laptops also have more limitations on expansion compared to desktops, and repairing or modifying them is much more difficult, in terms of working within its physical space; anyone who has ever seen the overly crowded innards of a laptop can appreciate the work involved. Also, if you find yourself hooking up a dozen or more peripherals and cables to a laptop, things can get a bit unwieldy.

DOI: 10.4324/9781003218821-6

Tablets and Smartphones

Devices offering lower cost and even more portability include tablets and smartphones. There are perfectly acceptable audio production tools available in this domain such as GarageBand, FL Studio Mobile, Reason Compact, and Beatmaker, to name a few apps. But CPU speeds, memory, and connectivity to external peripherals will be even more limited than with desktops and laptops.

The CPU

The Central Processing Unit, or CPU, is the engine of your computer. It communicates with the operating system and any applications that are running, processing, and outputting their instructions. The faster the CPU, the more instructions that can be handled, and since today's processing speeds are now in the Gigaflops (billions of floating-point operations per second) – or even Teraflops (*trillions* of operations per second) on gaming systems like the Sony PlayStation 5 – getting a fast CPU is going to speed up your applications and overall computing power. The trade-off is price; the fastest CPU available for a system is generally cost-prohibitive, unless you absolutely need the best processor out there for, say, gaming competitions or computing quantum entanglement. For a home project studio, the second- or even third-best CPU might just be fine.

Memory (RAM)

When looking at a computer for audio design, CPU speed is obviously a concern, but an even more important factor may be **memory**. Of all the places data is processed, moved, stored, or retrieved, memory wins the speed contest every time. When data needs to be accessed frequently or quickly, it gets stored in **random access memory** (**RAM**). Access times are measured in nanoseconds (ns), as opposed to milliseconds (ms) with a mechanical hard drive, so it often helps alleviate bottlenecks in access times depending on the application. Standard RAM has access times between the 50 and 70 nanosecond range. To put it in perspective, 1 nanosecond is 1,000,000 times faster than a millisecond, so there is quite a difference in speed versus mechanical hard drives.

The CPU is still faster than RAM, which needs to keep up with the CPU's instructions; so, a small amount of a superfast memory called **cache** acts as a buffer between the CPU and RAM. It stores the most frequently used and important instructions and sends them to the CPU, with the fastest speeds measuring around the 2-nanosecond range. Cache can exist on the computer's motherboard and also on the CPU itself. The more cache you have, the more efficiently the computer can operate.

Whether you're looking at a desktop or laptop system, you should strive to buy as much memory as you can afford. Memory is a home for operating system processing and also for any applications that are running simultaneously. This includes audio software, which can quickly eat up any remaining RAM resources.

Hard Disk Drives (HDDs) and Solid-State Drives (SSDs)

Hard disk drives (HDDs) are drives with several stacked magnetic disks that spin inside of it. They are accessed through a mechanical arm that can read and write data to them.

Solid-state drives (SSDs) have no moving parts and are more closely related to memory chips than mechanical drives. The access time (the time taken to seek out data) of the fastest mechanical drive might be 5 milliseconds, but a run-of-the-mill SSD could clock in at around 50 microseconds. Bandwidth is important as well, which is a way we measure how much data can be accessed *at the same time*; think of bandwidth as highway lanes, and the number of cars traveling on it as your disk data. A superfast HDD might be able to handle 500 MB/s, whereas a comparable SSD could offer 7,000 MB/s (that's 7 GB/s).[1] While SSDs are faster than HDDs, they are still slower than memory, as they can't be directly accessed by the CPU.

Both HDDs and SSDs can be internal to the computer, or external to it; internal drives tend to be cheaper in price as less complex enclosure and connectivity is needed, but portability is lost. Byte-for-byte, SSDs are also much more expensive than HDDs due to manufacturing costs, but they are far less prone to malfunctions and data loss.

For audio production, it's always a good idea to let the operating system and your apps run on the main system drive and to use at least one separate drive for your audio data. If you can afford an SSD for the latter, then go for it!

Nobody Told Me about the Fans!

There is one major issue when purchasing any computer for audio work: its external noise. Desktops and laptops require fans to cool off their ultra-hot CPU and other electronics. There will be moments when you are on a laptop, and suddenly it sounds like it's preparing for liftoff; or when you require absolute silence in your studio, but that lone muffin fan on your desktop never stops whirring. It's worth investigating how many fans your computer-to-be will have, and how loud they can get.

Audio Channels and Tracks

Before we dive further into audio hardware and software, a few quick definitions are in order. You will often encounter the term **channel** in audio specifications, which is a single physical or logical path that audio travels on. Then there are **tracks**, which can contain multiple channels at once, a term often used with **digital audio workstations** (**DAWs**) – a tool that allows the user to record, edit, mix, and output digital audio. **Instances** are related to channels, but the term is normally used when describing the real-time playback of one audio file played multiple times in succession (like a flurry of pickup item sounds, footsteps, or rounds of ammo).

The original *Pong* audio was one channel, which is also called a **mono** channel, or monaural audio. Your favorite game soundtrack probably has a physical left and a right channel (your headphones, maybe), which is **stereo** audio. An HDMI cable on the Nintendo Switch offers **surround** audio – 5.1 surround in this case, which is six physical channels. Digital audio editing software can handle many concurrent **tracks** (comprised of one or more **channels** each), and game consoles like the Xbox Series X are capable of hundreds of simultaneous audio channels. Clear as mud? Don't worry, you'll pick it up!

Audio Interfaces

At the heart of your computer-based recording setup is the **audio interface**, which serves two purposes. First, it is a way to bring audio from the outside world into your

computer, converting the analog input to digital. Second, it sends digital audio from your computer out to your speakers and headphones, converting it back to analog. Often, there is a third benefit: the interface is likely an upgrade from any existing audio device internal to your computer. There are some exceptions, but generally the audio chip inside your computer is part of the main circuit board, which can pick up all sorts of noise, hum, and other electronic interference from the components surrounding it, even from the chassis of the computer itself. On many systems, if the volume is high enough you may even hear noise generated with every mouse click and keystroke!

Audio interfaces operate so cleanly and quietly because all the audio processing is done within it, using high-end circuitry that is completely decoupled from the computer. Some interfaces also provide volume metering, with LEDs showing the input/output dB levels. Others add MIDI capability (the Musical Instrument Digital Interface protocol – more on this in a bit), which allows a MIDI-compatible keyboard or other instrument to be used in conjunction with music composition software.

A typical audio interface may look like the one in Figure 4.1. All you need is one input and output to get started, but higher-end interfaces allow for 16 or more simultaneous channels of audio in and out of the computer. Audio can be brought in one recording session at a time via a digital audio workstation (DAW), and then stacked up to work in concert together. Or you can import audio files into your DAW and they will all play back to your ears, all without any additional noise.

On the front panel are the analog inputs with input volume adjustments, the main speaker output volume, and a headphone output with volume. On the back are the speaker outputs, a supporting MIDI input and output, and the I/O connection to your computer interface output. Now when you add your DAW software, you can have a complete recording and playback solution for a basic project studio.

There are several ways to connect the interface to your computer, but two popular options are USB3 or 4 (PC) and Thunderbolt 3 or 4 (Mac), which all happen to use the same cable type, USB-C (see Figure 4.20).

Latency in Audio Interfaces

Ah, that dreaded L-word, **latency!** It is a serious issue you will encounter at some point in your production life. In the project studio, audio latency is measured by the time it takes an analog audio signal to go from the interface into the DAW, then back out of the DAW to the interface, and finally converted back to analog again to reach your ears. The reason for the delay is due to the interface's specific circuitry and any resource limitations. If you have a latency of less than 20 milliseconds (ms), your ears may not even notice the delay. But beyond ~100 ms and you will get frustrated quickly; imagine selecting a command in your DAW that records or plays back audio, then waiting ¼ second later to hear it. Or, singing into a mic with headphones on and hearing even a 50 ms delay between your vocal cords and your eardrums. Not good!

All interface models differ considerably, and those with a higher price tag offer lower latency times and higher-quality ADC/DAC circuitry. Many interfaces allow

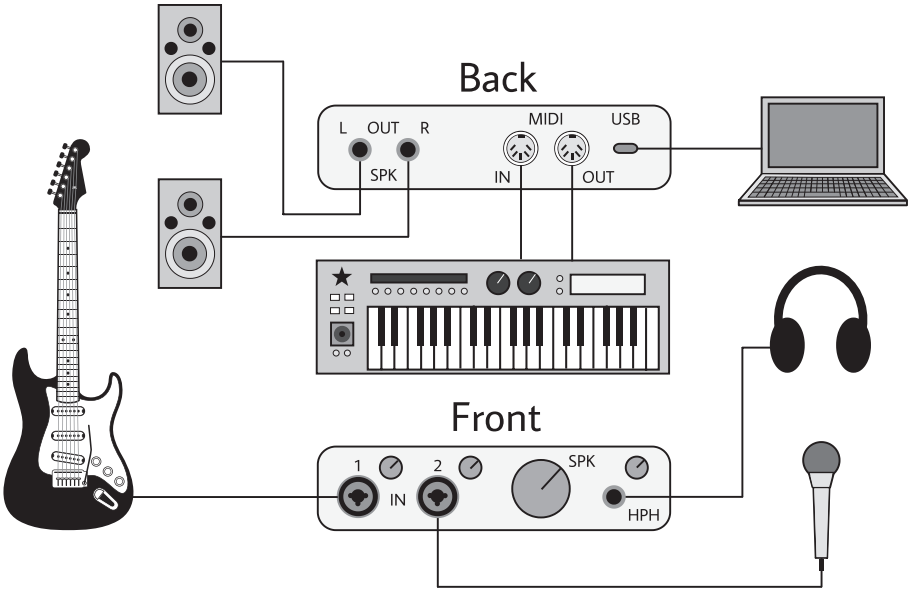


Figure 4.1 An example of an audio interface and typical setup. The front panel has inputs for microphones and instruments, as well as volume controls for headphones and back panel outputs; the back panel has outputs for speakers, plus MIDI throughput and a USB-C or Thunderbolt interface for the computer.

you to adjust latency with accompanying driver software, but at the expense of throughput rate, CPU resources, and/or audio quality.

Recorders

Whether you are recording music, sound effects, or your own voice, the computer is often the main recording device. It receives signals from the audio interface via a microphone or a direct input, such as with a guitar. Two channels recorded together is usually the norm, depending on CPU speed and storage media type.

If you prefer not using the computer to record audio, there are plenty of other hardware-based solutions. Many of these devices offer multiple inputs and outputs, such as the Zoom R8 8-track recorder,[2] or the TASCAM DP-30SD 8-Track PortaStudio.[3]

Portable Recorders

Another essential recording device is the portable recorder, where a handheld device can be used to record everything from the great outdoors to city streets, podcasts, or live music. Typically, portable recorders have a long battery life, removable media such as SD or XD cards, and an external interface to a computer via USB, like the

Zoom H1n or the iZotope Spire Studio. They also have at least one or two analog inputs and outputs, including a headphone jack (Figure 4.2).

Hardware-Based Workstations

If you decide that you don't want to move computers, interfaces, and recorders around, or you prefer the convenience of an all-in-one system, then dedicated workstations are a good choice. This also applies to live performances, where the convenience of portability and increased fault tolerance (e.g., fewer software crashes) are needed. Hardware-based DAWs such as the TASCAM DP-24SD 24-track digital PortaStudio[4] offers recording, mixing, and editing tools (Figure 4.3).

Figure 4.2 The Zoom H1n digital recorder (left) and the Zoom R8 (right), which is a hardware-based digital recorder and sampler with mixing and editing features (Courtesy Zoom).

Figure 4.3 The TASCAM DP-24SD 24-track digital PortaStudio (Courtesy TASCAM).

External Signal Processors

Audio can be modified or enhanced with changes to volumes, frequencies, and effects processing, such as creating an echo. For this purpose, the audio is sent through a cable into an external **signal processor**. This is specialized hardware that receives the audio signal, alters it in some desirable way, and sends the processed signal back out. Most of these devices will mount into a standard 19-inch-wide rack enclosure (Figure 4.4).

The beauty of having external processors as opposed to software-based ones (which we will see later in this chapter) is that they normally don't crash, or become obsolete. They are stand-alone devices that will always perform their function. The downside is they are more expensive than software-based processors, generate heat, and are far less portable.

Among the hundreds of devices available, the big three are the EQ (equalizer), compressor (dynamics controller), and reverb (reverberator, or echo):

EQ: This enhances and/or reduces certain frequency levels. Applying an EQ can be done with knobs, buttons, or sliders to achieve the desired frequency characteristics.

Compressor: This controls the overall dynamic range of audio. It can be used to boost low-level signals while keeping a lid on its maximum volume level.

Reverb: This is essentially an echo device, creating a series of reflections you can dial in to fit your needs. Common parameters include room type and size (e.g., a small room or large concert hall), and decay time (how long the reverb takes to fade away to zero volume).

Figure 4.4 An example of rack-mounted audio signal processors.

Mixers and Signal Flow

Mixers

A **mixer** (or **mixing board**) is a device that combines several channels of audio, with the option to control the volume of each channel, at a minimum. Most mixers also provide EQ, panning (left-to-right speaker positioning), and the ability to send and return signals from an external source, such as an effects processor. They can handle as few as two channels or as many as 100, with the larger systems known as a **mixing console** or **desk**. More expensive options include complex signal routing, multitrack outputs, and onboard effects processors.

Analog mixers are still in demand, but digital mixers have become an increasingly popular choice as well. With the digital option, noise is kept to a minimum, and the mix outputs can be sent directly to your computer without the need of an interface; the digital mixer itself takes on that additional role. There are also analog-digital hybrids, offering the "warmth" of analog channels but using digital routing and outputs. Bluetooth connectivity with other devices is an option on such consoles as the Mackie Onyx, allowing a wireless device to stream into a mix channel as well.

Below is an example of a typical "classic" analog mixer. If you've never used or even seen a mixer before, you might think you'll be trying to land a jumbo jet with the controls, but as you will see it's not as frightening as it looks. A real-world mixer will actually have several more buttons and knobs than this, with several options for adjusting and routing signal flow, and this will vary among different manufacturers and product lines. But the console in Figure 4.5 is a streamlined mixer, demonstrating the most universal features.

Input Section

If you take a vertical slice of the mixing channels section, you have a **channel strip**. That's almost everything you need to know about the mixer already! Let's look at the channel strip controls from the top down (see close-up in Figure 4.6):

Gain: This is the amount of volume boost or cut given to the incoming signal, before it is sent through the rest of the channel strip. Ideally, the signal should be at a manageable level, not too low or too high in volume. The "U" above the gain knob means **unity gain**, meaning no increase or decrease to the incoming volume has been made.

EQ: Adjusting the equalization for all channels ensures the proper weighting of frequencies in the mix. You might want more thump in your percussion, or more sparkle to your synth sounds. Our console uses a classic 3-band Perkins EQ (also known as "British EQ") with key frequency areas set to boost or cut. Each band boost or cuts +/–15 dB, and if the knob is centered, no change is made. The high-frequency knob adjusts everything from 12 kHz and above; the low-knob handles everything from 80 Hz and lower; and the midrange has extra functionality: it is a "sweepable" midrange, where you can set a center EQ point between 100 Hz and 8 kHz with one knob; then with the next knob it can be adjusted from +/–15dB at its peak, creating an adjustable bell curve shape on a frequency graph.

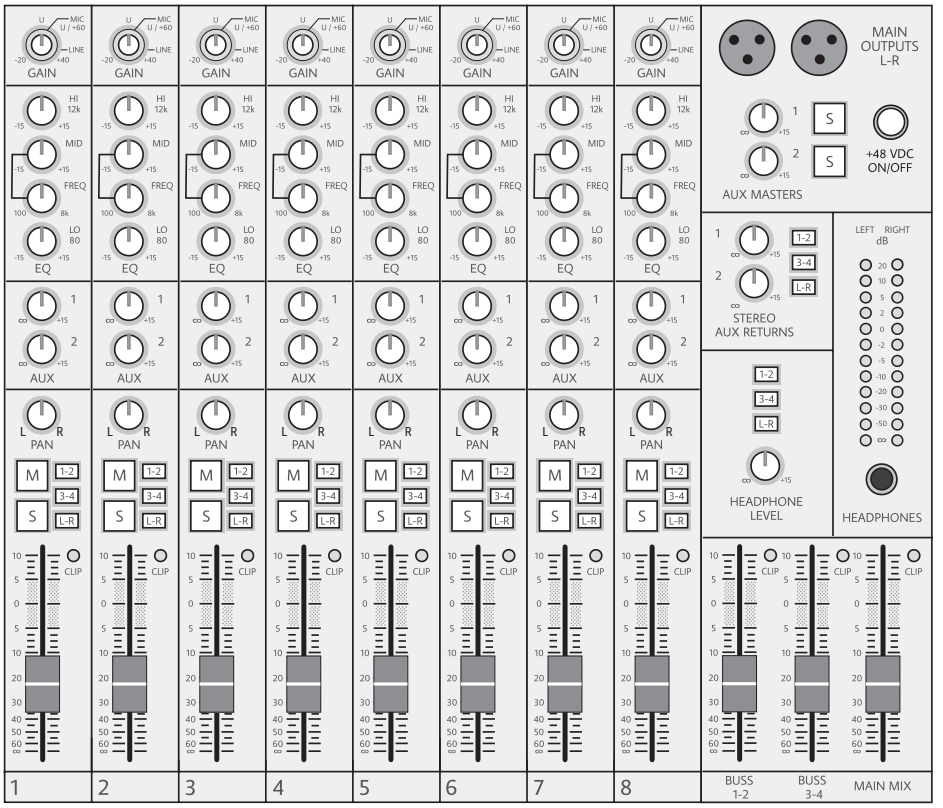


Figure 4.5 A basic eight-channel mixing console.

Aux: An auxiliary knob sends the channel signal to an external device, such as an effects processor. The processor then returns the signal, now with effects, back into the mixer. The Aux controls can be used in other ways: for example, sending the channel signal to speakers on a stage, so performers can hear their instrument at the desired level – a different mix than the one going to the main speakers. More on the auxiliary signal path in the next section.

Pan: This knob simply adjusts the position of the audio, from the left to right speaker. Leaving the knob straight up keeps it in the center (both speakers at equal volume).

(M)ute/(S)olo: Mute silences the channel, and solo button mutes all other channels that are not soloed.

Routing: Any channel can also be sent to a particular output group, either for listening in headphones or to specific output jacks on the back of the mixer; with this particular channel strip, there is the choice or sending a channel to Buss 1-2, Buss 3-4, and / or the main stereo (Left-Right) mix. For more details, see the *Output Section*, below.

Level: This is the final stage, and the master volume control for the strip. The volume control is called a **fader**, which adjusts volume from total silence ("minus

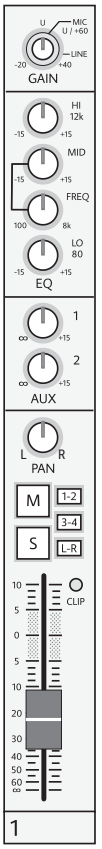


Figure 4.6 Channel strip from an analog mixer.

infinity" symbol) to +10 dB, in this case. A small space below the fader is usually reserved for notes (written on a removable adhesive such as masking tape) called a **scribble strip**. Channel 1's scribble strip might have "GTR" written on it, for guitar.

Nice-to-Haves: Other mixers have convenient options like an Aux ← → EQ button, which changes the order the signal is processed in; you may want to EQ your signal after reverb is applied, or vice versa. Other options include motorized (recall) faders, more auxiliary controls, and buttons for parts of the signal path to be either pre- or post-fader.

Output Section

To the right of the channel strips is the output section of our mixer. Some mixers have **busses**, each a special audio channel where any number of channel strips and auxiliary signals can be routed to. Busses are normally used for multitrack production or for live stage situations, where multiple speakers are needed. Our mixer has four busses which can be selected per channel with push button controls. As an extra feature

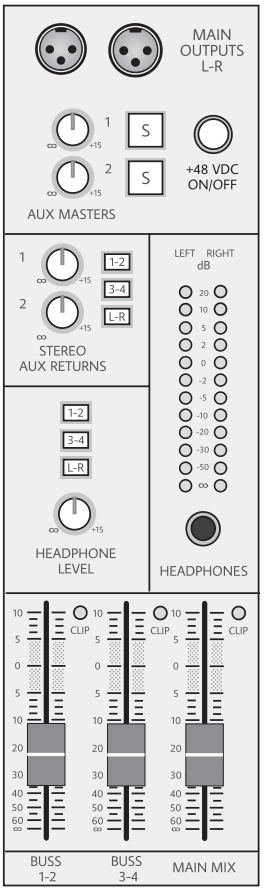


Figure 4.7 The mixer output section.

with our mixer, we can also hear the main L-R mix, busses 1 & 2, and/or 3 & 4 in our headphone level control.

The main outputs consist of two circular ports, each with three inputs. These are called **XLR** jacks and receive XLR cables, one for the left speaker and one for the right (see the *Cables and Connectors* section below).

Below the main outputs (XLR jacks) are the Aux Master controls. The Aux 1 and Aux 2 knobs are used for the final auxiliary volume adjustments before they are sent out of the console. Many times, a signal processed by an effects unit becomes stereo, so each auxiliary return is also in stereo, if needed. The Stereo Aux Return knobs control the incoming stereo signal before it is sent back into the main mix. These returns can also be routed to the busses and main mix.

On some devices (and later when we visit DAW plug-ins), you will see knobs or faders for adjusting "dry" and "wet" signal levels. Each channel's signal before effects processing is called a "dry" signal, sent to the main mix, and the aux returns are stereo "wet" signals, also sent to the main mix. They combine to form a (hopefully!) pleasing sound.

Phantom Power

Most mixers also provide **phantom power** (48 volts of direct current) to certain types of microphones that require it, sent at the push of an on/off button.

Control Surfaces

With modern DAWs, it has become a popular choice to choose a **control surface** instead of a mixer. It may *look* like a mixing console, complete with faders and knobs, but the difference is that your DAW's features (as well as features from other audio-based software tools) can be controlled directly from the surface, with basic channel strip, MIDI, and transport functions being among the most commonly offered features. The PreSonus FaderPort 8 (in Figure 4.8) is an example of a universal control surface that works with many DAWs; in addition to the abovementioned controls, it also has touch-sensitive motorized faders with recall ability, as well as digital scribble strips for each channel, and the ability to edit many common plug-in settings.[5] Other control surfaces offer modular expansion, as with the Monogram

Figure 4.8 The PreSonus FaderPort 8 Control Surface (Courtesy PreSonus Audio Electronics/Fender).

Creative Console, which allows you to add or remove any of its available control modules.[6]

Microphones

Microphones, or mics for short, are a type of **transducer**; they convert one form of energy into another. The acoustical energy the mic receives is converted into electrical energy, which is then amplified as a usable signal.

There are an amazing variety of microphones available for all sorts of general or specific applications. But which is the right one for the job? We will cover the basic types here, but for specific mic selection and placement, I highly recommend the book *Microphone Techniques* by David Miles Huber and Philip Williams.

Common Mic Types

Dynamic

The **dynamic mic** (also known as a **moving coil mic**) has a plastic, circular **diaphragm** that vibrates in and out as it receives sound waves. A metal wire coil is attached to the diaphragm, so the two move in tandem. The coil is surrounded by a strong magnet, and the fluctuations of the coil within the magnetic field create a tiny amount of electrical current, its strength being relative to the velocity of the coil. The current is sent out of the mic by two wires, where it can be amplified and sent to a speaker. Dynamic mics are of a rugged construction and can handle a lot of wear-and-tear. As such, they do not capture delicate fluctuations in sound; this may be suitable for live applications, where a more consistent volume level may be desirable. There is also an excellent **rejection** behind and around the mic, meaning sound not directly in front of the diaphragm will fall off greatly in volume – also good for certain recording applications. One possible drawback is they are not as sensitive to very high frequencies, as it begins to taper off at around the 10 kHz range (Figure 4.9).

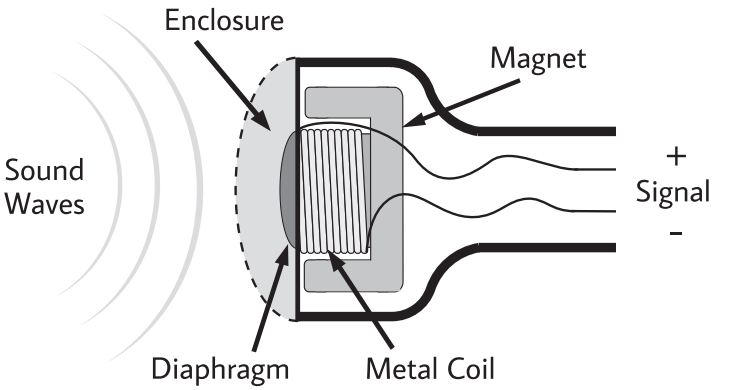


Figure 4.9 Diagram of a dynamic mic.

Condenser

One of the most popular studio mic choices, the **condenser mic** operates a bit differently than a dynamic mic. There is still a diaphragm, but the coil is replaced by an **air capacitor**, comprised of an electrically charged front and back metal plate. The diaphragm is connected to the front plate, and the back plate is fixed in place. As the diaphragm moves in and out, it moves the front plate with it, causing fluctuations in air pressure between the two plates. This also varies the electrical charge the air capacitor holds, and this electrical value is amplified and sent out through two wires. The diaphragm-air capacitor combination is more sensitive to air pressure changes, and so is a great choice for detailed vocal or instrument recordings, or for recording sounds that have a more tactile quality to them.

Condenser mics require a power source in order to use it. This power can be supplied by an internal battery, or sent straight into the mic cable from external phantom power. This can be supplied by a mixing board, audio interface, or any other device that connects to the mic. Just be careful not to touch wire leads using phantom power! It is normally 48 volts of direct current, which can cause personal injury (Figure 4.10).

Condensers are more delicate than dynamic mics, so care must be given not to drop them, or for something to strike the mic enclosure, such as an errant drumstick!

Ribbon

In a **ribbon mic**, a corrugated metal strip (the "ribbon") acts as the diaphragm *and* the coil, which makes it extremely sensitive to sound fluctuations. The ribbon is surrounded by a magnet, and vibrations in the ribbon cause changes to the magnetic field, creating electrical current like the dynamic mic, which are sent out by two wires.

Most ribbon mics do not require phantom power (an *active* ribbon mic needs it, but if you run phantom power to a non-active ribbon mic, you run the risk of destroying it!). Most generate a voltage comparable to dynamic mics. But the magnets in some ribbon mics are less efficient, so their output is much lower; these require a "step-up" transformer that boosts the signal internally.

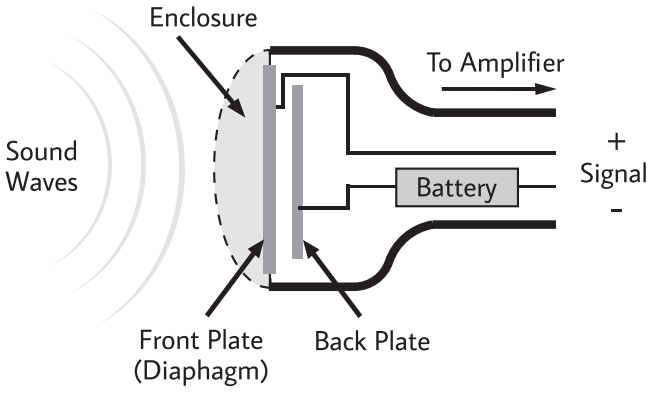


Figure 4.10 Diagram of a condenser mic.

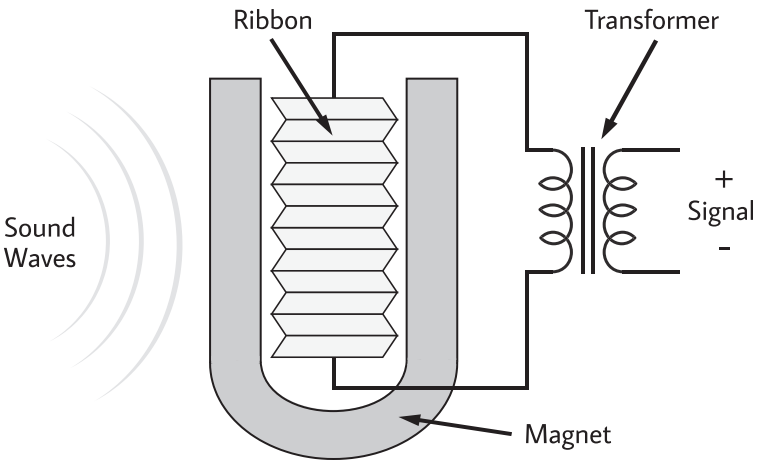


Figure 4.11 Diagram of a ribbon mic.

Ribbon mics have an appealing "warm" characteristic when recording, with a slight boost in the lower frequency range. They also have more sensitivity to higher frequencies than a dynamic mic, which is great for recording piano, strings, vocals, and other instruments having a broad frequency spectrum.

To capture such dynamic nuances, the ribbon is designed to be quite light and delicate, which unfortunately makes it easily susceptible to damage. Even if the manufacturer says they're rugged, you should still go out of your way to protect it (Figure 4.11).

With any mic, you should at least cover the mic head when not in use; for longer-term storage you should keep it in a foam case, and somewhere that is resistant to vibration, heat, and moisture.

USB/Digital Mics

The mic types listed above also come in models that can connect to a computer, via a USB or other serial connector. You can even plug in directly to your smartphone or tablet: the Shure MV88 mic is a stereo condenser for iOS, while the Zoom AM7 works for Android devices. There are also wireless options like the Samson Go, which is good for when a lavalier (clip-on) mic is needed; its receiver also has a USB output for PC, iOS, and Android.

Ambisonic Mics

The ambisonic audio format dates back to the 1970s, but only recently has the technology been made more popular through 360- and VR-based audio. The mic uses condensers placed in an array, such that audio is recorded on the horizontal and vertical planes. Upon playback, the audio is decoded to match whatever speaker array is available; the format represents the *directionality* of sound, and not physical speaker channels themselves. Stereo playback is the minimum output, but with larger speaker arrays the listener can experience *spherical* surround sound, including sound above and below them.

Pickup Patterns

A microphone's sensitivity to sound in any direction is known as its **polar pattern** (also known as its **pickup pattern**), which can be visualized on a **polar response graph** in 360 degrees. The graph shows the relative sensitivity of the mic, or its **directionality**, measured in decibels within concentric circles. It also shows where the mic is **on-axis** or **off-axis** at any given location. The **axis** of a mic is a measurement of its sensitivity; on-axis is where sound can be picked up, and off-axis is where sound is rejected by the mic. The further the on-axis reading is from the center of the graph, the more sensitive it is at that location (in dB) (Figure 4.12).

Most mics have at least one on- and off-axis point, with a few exceptions. Here is a short list of the most common mics, as seen in Figure 4.13:

Cardioid: Latin for *heart-shaped*, the cardioid mic is the most common pattern that is 0 degrees on-axis, and 180 degrees off-axis, giving optimum rejection behind the microphone.

Supercardioid: This has a narrower pickup pattern than cardioid, which means more rejection in front of the mic. There is also some directionality behind the mic.

Hypercardioid: A similarly narrower pickup pattern than cardioid, but this also has the most rear directionality available in the cardioid family.

Omnidirectional: An omni mic captures everything around it and is plotted as a circle on the polar response graph. The whole mic is on-axis, which is excellent for recording environmental ambience in 360 degrees (on a single channel), or as a cost-effective solution for miking an entire band. Just place it in the center of an ensemble, and everyone can be recorded at once.

Bidirectional (figure eight): This one is easy to remember, as its pickup pattern is shaped like a figure eight. A great choice for recording two people on opposite sides, such as for an interview or a live performance from two voice artists.

Shotgun: For extremely narrow targeting of a sound, shotgun is the way to go. It's like shining a flashlight on a surface in a dark room: it shines clearly and brightly where it is aimed but only within a tight radius. The example in Figure 4.13f is called

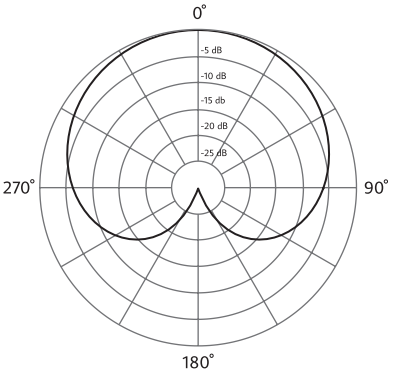


Figure 4.12 A polar response graph, showing directionality in 360 degrees. Relative mic sensitivity is measured in increments of 5 decibels per circular level, starting from the innermost circle at −25 dB to the outermost circle at 0 dB. This graph shows a cardioid mic, which is 0 degrees on-axis and 180 degrees off-axis.

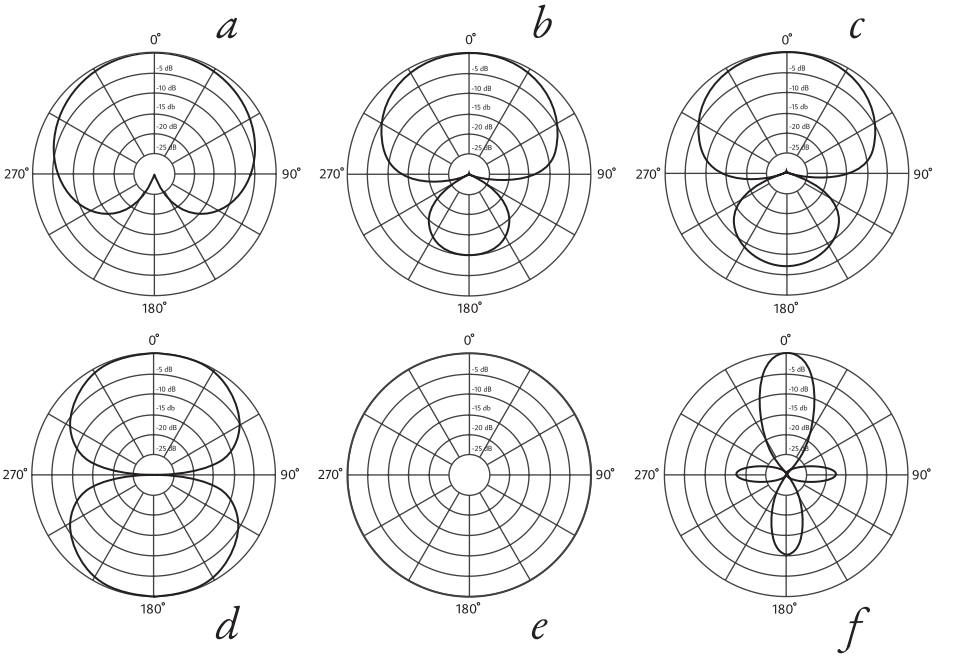


Figure 4.13 Various mic pickup patterns: (a) cardioid, (b) supercardioid, (c) hypercardioid, (d) bidirectional (figure eight), (e) omnidirectional, and (f) lobar shotgun.

a lobar shotgun as there are four "lobes" to the pickup pattern, but the main on-axis point is still 0 degrees with superior rejection around it. Now you can record that bird tweeting 50 feet above you with laser-like focus!

Frequency Response

In addition to its polar response graph, a mic also has a **frequency response** graph associated with it. This shows which frequencies the mic is more or less sensitive to (in dB), normally ranging from 20 Hz to 20 kHz. Some mics will even come with a polar response graph using multicolored overlays, if there are significant changes to the polar pattern across the frequency spectrum.

There are hundreds of different microphones commercially available that are designed for a myriad of applications: for example, a kick drum mic might have an increase in the lower frequencies, while a mic for vocals may have an increase in the 4–6 kHz range, the area of speech intelligibility known as *presence*. Other mics strive to have a completely flat response across the spectrum, which may be more expensive due to the engineering complexities to achieve that flatness.

Peak Amplitude

Peak amplitude is the maximum amplitude that can be recorded before the signal becomes distorted, or worse, damages the microphone components. An acceptable

peak amplitude might be 120 dB or greater. It is sometimes called **Maximum SPL** (sound pressure level).

Noise Floor

The **noise floor**, or **noise level**, is the amount of actual noise in the mic electronics. All electronics have some self-generated noise, and no signal recorded can be heard beneath this level, which is usually very quiet in a professional mic – on the order of 20 dB or less. Some manufacturers also call this the mic's **self-noise**.

Dynamic Range

When considering a microphone to buy, the **dynamic range** is another important consideration. It is the range between the quietest and loudest decibel level the mic can receive (you can find it by taking the peak amplitude minus the noise floor). This value can be vastly different from one mic to the next, but an acceptable range would be at least 110 dB.

Signal-to-Noise (S/N) Ratio

The **signal-to-noise ratio** is a bit more complex to define, but it represents the amount of audio signal received by the mic, compared to its noise level; in other words, the mic receives audio at a loud enough volume level, such that the noise is perceived as comparatively nonexistent. The higher the S/N ratio, the less self-noise is heard in the mic under normal conditions.

S/N ratios are only used for active mics, i.e., those that require power, which are mostly condensers. An acceptable S/N ratio might be 70 dB or higher.

Mic Extras

The most common add-on to a mic is a signal **boost** or **cut switch**. In either case, the typical amount of change to mic sensitivity is +/–20 dB. A switch that cuts incoming dB is also known as a **pad switch**.

Many mics also offer a **High-Pass** and/or **Low-Pass filter** (HPF/LPF) switch. As noted previously, an HPF allows higher frequencies than its cutoff to pass through, and an LPF does the same for lower frequencies. So, a 16 kHz LPF would attenuate anything above 16 dB, according to its frequency response graph. HPF is also known as a **low cut** or **low rolloff**, and LPF as a **high cut** or **high rolloff**.

Mic Spotlight: The Blue™ Spark

Let's take a look at the Spark mic from Blue Microphones (Figure 4.14).[7] Here are the technical specifications from the manual:

The Spark is an excellent choice for an entry-level, professional-grade mic. It is a cardioid condenser with a dynamic range of over 119 dB and a signal-to-noise ratio of 73 dB. It has greater sensitivity between 30 and 60 Hz, and again from the 5 to 10 kHz range. The Spark can handle a maximum SPL of 136 dB while its noise floor is

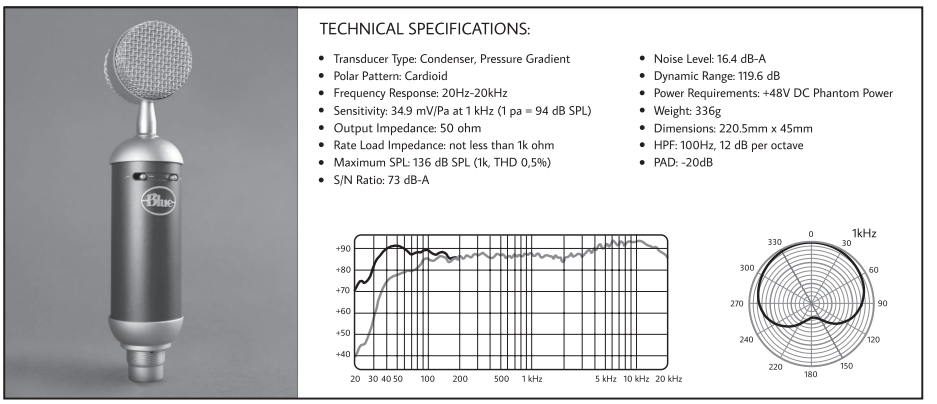


Figure 4.14 The Blue Spark condenser mic, with its frequency response and pickup pattern (Courtesy Blue LLC/Logitech).

16.4 dB. Other pluses include a High-Pass filter (HPF) with a 100 Hz cutoff, and a –20 dB pad switch.

Pop Filters

There may be a problem with your vocalist's *p*'s and *b*'s; these consonants can "pop" from the mouth and create a lot of energy in your mic. Try holding a piece of paper a few inches from your mouth and vocalize a strong "puh!" and "buh!" toward it. See what happened? The paper should have moved quite a bit. Now imagine your mic diaphragm reacting to these low-frequency **plosives**. They will be recorded much louder than normal, ruining your session (they can be difficult to edit out later). There may also be a problem with *s*'s; they may come out as **sibilant** (of a very high frequency) compared to the rest of the spectrum.

Enter the magic **pop filter** to the rescue! It is a circular frame that contains two layers of mesh material (usually a specific type of cloth), with an air gap in between. It behaves a bit like the double-paned window example from Chapter 1, but it is more porous: plosives reach the first material and the energy is weakened; then they reach the air gap, and by the time they hit the second mesh layer, the energy level is roughly on par with the rest of the frequency spectrum. The high frequencies are also absorbed by the mesh material, with their volume levels reduced slightly.

Sometimes pop filters are used simply to keep any vocal gunk from hitting the mic (yuck) (Figure 4.15). It's much easier to clean a pop filter than a mic diaphragm!

Speakers (Monitors)

A **speaker**, also called a **monitor**, operates much like a microphone but in reverse. The electrical signal is sent through a magnet with a coil wrapped around it. The coil vibrates, making the speaker cone (the diaphragm) vibrate. Bass frequencies are diminished in smaller speakers, as the iron magnet and speaker cone need to be large enough to generate audible, low-frequency energy. This is why subwoofers are so

Figure 4.15 A pop filter in use in a recording studio.

heavy! In any case, all speakers require some power to recreate acoustical sound (Figure 4.16).

Frequency Response

When purchasing monitors, we are looking for a relatively flat frequency response. You will end up fighting with the monitors if there are frequencies that are inaccurately reproduced. In some cases, speakers may be too small to play frequencies below, say, 80 Hz. In this case, you may end up mixing your audio incorrectly – you can't mix audio you cannot hear! This is where the addition of a **subwoofer** to your monitor setup is ideal.

Active versus Passive

A **passive** monitor is one that requires an external amplifier. An **active** monitor has its own built-in amplifier. While passive monitors tend to cost less than active ones, if the external amplifier sends too strong a signal, the monitor can become damaged or inoperable. Active monitors often have onboard electronics that handle excessive volume levels or power surges, using a fuse or circuit breaker.

Two- and Three-Way Systems

Monitors having more than one speaker require the frequency spectrum to be split between them, and the electronic circuit that handles this is called a **crossover**.

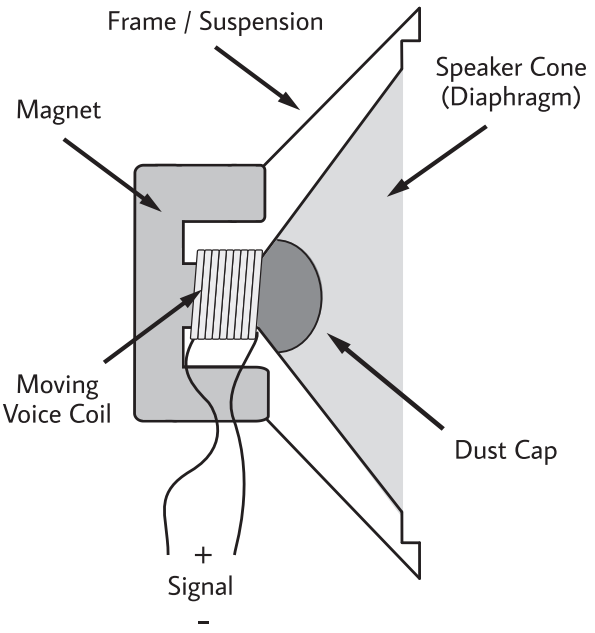


Figure 4.16 Diagram of a basic speaker.

Two-way systems have a **tweeter** handling higher frequencies, and a **woofer** for lower frequencies. There are also three-way systems with a tweeter, midrange, and woofer speaker configuration.

Surround Sound

Surround sound is supported by an increasing number of games each year, with some variations available in the format. All of the speakers in an array handle a frequency range of roughly 100 Hz to 20 kHz, except for the **subwoofer**, a speaker that only reproduces low frequencies below the 100 Hz range (though some can handle up to 200 Hz).

The surround system naming convention is X.Y.Z., where X is the number of standard speakers, Y is the number of subwoofers, and Z is the number of high (ceiling) speakers.

5.1 is the most common system, with front and surround left and right speakers, a front center speaker, and a subwoofer. A 7.1 system adds two surround speakers to the middle left and right areas of the listener, and when using a format such as Dolby Atmos, the addition of two ceiling speakers makes a very immersive 5.1.2 or 7.1.2 system.

If you have a subwoofer, it can be placed anywhere, but the ideal location would be slightly off the floor and a few feet to the left or right of your monitors. Some audio designers prefer placing the subwoofer in a corner, but be aware that this generates even more bass response (which may be preferable depending on your needs).

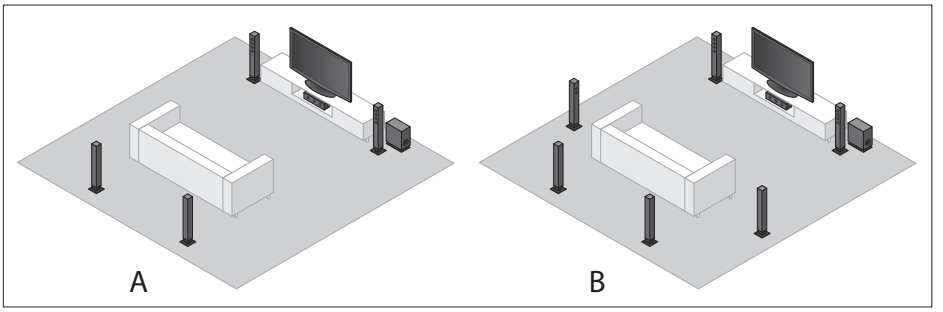


Figure 4.17 Examples of (a) 5.1 and (b) 7.1 surround sound systems.

On gaming systems such as the Xbox X/S, PlayStation 5, and the Nintendo Switch, surround audio is sent via an HDMI cable for 5.1, or an HDMI 2 cable for 7.1. HDMI cables can carry up to eight channels of uncompressed PCM audio; other formats such as digital theater systems (DTS) carry digitally compressed channels of audio through a single *optical* or *coaxial* cable (see Figure 4.20 for connector types). With newer technologies such as Dolby Atmos, audio is transmitted as *objects* that can move throughout your room, irrespective of speaker array types, with Atmos codecs even supporting headphone-based listening.[8]

While there are many delivery formats available, the most popular are Dolby Digital, DTS, and Dolby Atmos (Figure 4.17). If you have never experienced gaming in surround, any of these formats are sure to pleasantly surprise you!

Speakers versus Headphones

The year 1985 marked the time that gamers could finally experience stereo sound, thanks to the Commodore Amiga home computer. Four years later, the NEC Turbo Grafx 16 console arrived with stereo outputs (and six channels of audio), followed by the Sega Genesis (stereo with ten channels of audio). Ever since, we've been enjoying soundscapes both with stereo speakers and headphones. But do they sound the same? No!

With headphones, bass frequencies are perceived louder as compared to speakers, due to the closer proximity of the waves to our ears. So, which is better to mix in? There are purists on both sides, so feel free to choose your personal preference. But professionally speaking, if you're mixing for a game soundscape, you should consider providing both mixes if possible. On the developer side, this may be as simple as applying an EQ filter to reduce the lower frequencies (compared to the speaker mix), when the user selects headphones as the audio source.

Stereo Positioning

As we saw in Chapter 1, proper monitor positioning is important, in order to avoid phase interference. Your monitors should be equally spaced apart from where you sit, slightly tilted inward, and at about your ear level. You'll want to create a triangle between the two monitors and yourself, like the illustration in Figure 1.15a. Experiment

with adjusting the monitor height and angles, as well as your seating height. You may find some interesting differences in the sound depending upon these tweaks! When you feel like you're hearing the audio accurately, you've hit the sweet spot.

Headphones

As a game audio designer, getting an accurate pair of headphones is a must. Look for name brands that have specifications of 20 Hz–20 kHz, and as flat a frequency response as possible. That said, gamers have endless varieties of headphones, headsets, and earbuds, and you may need to consider this for your in-game mixes.

Closed versus Open Back

Open back headphones have small gaps around the ear cup, that allow the outside air to pass through. This provides a pressure release for low-frequency energy and creates a more "live" environment within the headphones (as opposed to sterile isolation), which can be preferable for general mixing or listening. They are not so good if you are trying to work in a busy office or prevent your audio from emanating outside. Closed back provide solid isolation from the outside world and have a punchier bass response. For critical listening applications, the isolation of closed back headphones is often preferred. However, some audio designers may feel fatigued after hours of listening to audio in such an enclosed environment.

In the game audio world, this may mean using closed back headphones to craft a particular sound effect, carefully examining a creature voice, or working on a specific instrument in a music soundtrack, and using open back for mixing soundtracks or in-game soundscapes. But again, it comes down to personal preference. (I like the precision of using closed back, but that's just me.)

Headphones can connect to your device via an audio cable, USB, or Bluetooth.

Earbuds and In-Ear Headphones

Most earbuds are placed just outside of the ear, letting in outside ambience. Again, this can be good for a "live" environment, but bass response is diminished. Custom-fit earbuds can be obtained from boutique manufacturers, but of course they will be much more expensive than the one-size-fits-all generic brands. In any case, the generic variety can get quite uncomfortable after long use, and may frequently fall out if you're not perfectly still.

In-ear headphones go just inside the ear canal, for a completely immersed experience. These are often made of silicone that fit snugly and can stand up to hours of use. Look for a good manufacturer that will offer several size options to choose from in a set (Figure 4.18).

Acoustic Material

Absorbers

If you were to step into a large, empty room (bare floors and walls), you'll likely hear lots of reverberation. Throw in a carpet, a full bookcase and some furniture,

Figure 4.18 Mixes will sound different through a variety of headphones, headsets, and earbuds.

and those reflections will almost disappear. These objects are absorbing many of the room's reflections.

In your game audio production space, it's an excellent idea to get some absorption. A carpet is a must, and beyond that, **acoustical foam tiles** can be hung from the walls that will make a huge difference in your listening experience. Trying to mix with reverb ping-ponging around your space is a recipe for disaster! Ideally, you shouldn't have too much acoustical treatment (say, more than 50–60%) or it may begin to sound unnatural and **anechoic**.

Absorption of really low frequencies (under 250 Hz) requires thicker material. **Bass traps** are a kind of specialized foam or other composite material that can be wedged into a corner of your room to soak up excess bass frequencies. These can be expensive, in which case a sofa or mattress will do in a pinch.

Diffusors

To complement absorption, we can also break up reflections with **diffusors** – specialized tiles that evenly scatter a reflection into smaller, multiple ones. Not only does this make the reflection weaker but can also mask some of its reverberation. If you're sitting in front of your speakers, then ideally these should be placed to the wall behind you, as that's where the first reflection is made. If budget allows, you can also place them to your left and right walls, parallel to the speakers (Figure 4.19).

Cables and Connectors

Balanced versus Unbalanced Cables

There are two main categories of audio cable: **balanced** and **unbalanced**. An unbalanced cable has two wires inside: the first wire carries the signal (the "hot" wire)

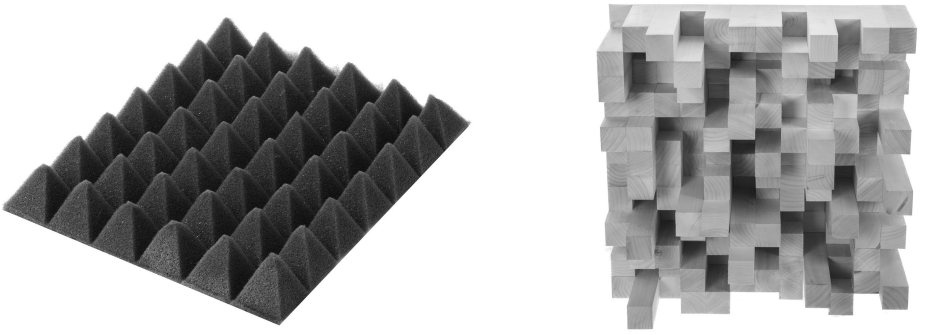


Figure 4.19 An acoustic foam tile (left) and wooden diffusor tile (right). Note the intricate pattern in the wood used to break up and scatter reflections.

surrounded by a plastic shielding to keep out external noise and interference. Then a second wire, the ground, is wrapped in a mesh around that shielding. This completes the electrical conduction and also helps to keep noise from being introduced into the signal.

However, this is only good for short-length cable runs before noise and distortion gets picked up. Let's say your unbalanced signal path travels out of your synthesizer into a mixer, then out of the mixer to two external processors, then out of the mixer to your audio interface. If any of the audio cables pick up noise, it will be carried through the rest of the signal path. If that noise appears near the start of the path, it will only get worse down the line. Your signal path is only as strong as the weakest link in the chain, and if that weak link is an unbalanced cable, it will affect the final signal.

Unbalanced cables may suffice if you are only using a few audio cables of very short length (for a total signal path of 10–12 feet, as a rough estimate). The main advantage of unbalanced cables is that they are less expensive than balanced. On the outside (see Figure 4.20a), the metal connector uses the connector's tip for the signal, followed by its sleeve for the ground. A small, black plastic spacer separates these two signals from contact with one another (the plastic spacer does not carry signals). It is called a **TS** or **tip-sleeve cable**.

Balanced cables have three wires: the first and second carry the signal, and the third is the ground wire. But here, the second wire carries the signal 180 degrees out of phase (really, flipped polarity). From an electronic audio device, it is flipped *before* entering the cable. During use, noise is introduced into the cable just like with the unbalanced, into both the first wire and the one out of phase. Then at the receiving device, a transformer flips the 180-degree signal back into phase, *along with its noise*. So, the signal is preserved, but the noise through the magic of phase cancellation gets removed completely! And our main signal is doubled in amplitude. As shown in Figure 4.20b, the connector has a tip for the main signal, a ring for the out-of-phase signal, and the sleeve for ground. It has two plastic spacers separating these three signals, and is called a TRS or tip-ring-sleeve cable.

OK that's great, but what about **headphones**, you ask? Well, there are some exceptions. Headphones carry three wires: one for the left signal, one for the right, and the

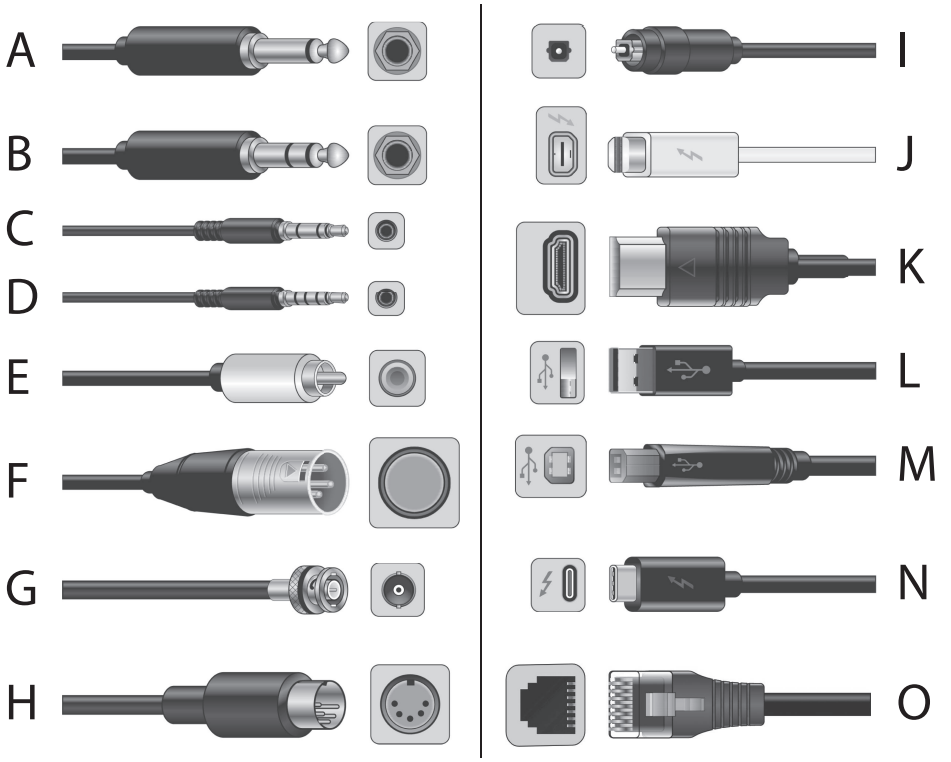


Figure 4.20 Various types of cables used in audio production.

ground wire; but nothing is out of phase here. Even though it is a TRS configuration, these are still unbalanced cables.

And what about **headsets**? Those have *four* wires: one for the left and right signals, another for the microphone, and one for ground. Headsets use unbalanced cables with a tip-ring-ring-sleeve or TRRS connector.

Useful Cables and Connectors

There are several audio cable standards, the most common being the following:

A) 1/4″ Tip-Sleeve (TS): Unbalanced instrument and **patch cable**. A patch cable is a cord that routes a signal from one electronic device to another (i.e., you can "patch" your guitar cable to an amplifier to hear the boosted audio signal).
B) 1/4″ Tip-Ring-Sleeve (TRS): Balanced instrument and patch cables.
C) 1/8″/3.5 mm Tip-Ring-Sleeve (TRS): Stereo outputs from headphones, audio interfaces, instruments, etc.
D) 1/8″/3.5 mm Tip-Ring-Ring-Sleeve (TRRS): Stereo outputs + microphone (headsets or speaker-mic combo).
E1) RCA/phono: Radio Corporation of America's connector format, which is a single channel much like the TS connector, originally designed for RCA's phonograph

(record player) outputs. Typically come as a stereo pair, white for left channel, and red for right channel. It is also known as a **coaxial cable**, inside which there is a solid wire in the center, an insulator in between, then a metal wrap for shielding (ground). Also used to carry digital compressed surround signals such as DTS.

E2) S/PDIF coaxial: The Sony-Philips Digital Interface is a format that allows digital audio to be transmitted on coaxial (RCA) cables, or on fiber optic cables (see I1, below). S/PDIF was originally designed to support two uncompressed 16-bit 44.1 kHZ and 48 kHz channels, or two channels of compressed audio for 5.1 and 7.1 surround formats, such as DTS (Digital Theater Systems).

F1) XLR: A 3-pin balanced audio connector commonly used for microphones, or for professional-grade audio signals. It carries one channel of audio.

F2) AES3: Also known as AES/EBU (Audio Engineering Society/European Broadcast Union), is the equivalent of SPDIF/e using an XLR cable, but supports two channels of audio up to 24-bit, 48 kHz uncompressed audio.[9]

G1) BNC (Bayonet Neill–Concelman): A coaxial cable with a "bayonet"-like center pin on each end. They are mainly used for synchronizing digital clocks between professional audio devices such as interfaces and some electronic instruments.

G2) AES3: The BNC-formatted version of AES3.

H) 5-pin DIN: A connector that supports the MIDI (Musical Instrument Digital Interface) protocol, a communication link between musical instruments and computing devices. A detailed look at MIDI follows in the next section.

I1) S/PDIF optical: The TOSLINK cable was designed by Toshiba Corporation, originally to connect CD players to stereo receivers. Uses fiber optics to carry the digital signal, at a rate between 125 Mbps (megabits per second) and 1.2 Gbps (gigabits per second). It is cross-compatible with S/PDIF electrical (see E2, above).

I2) ADAT Lightpipe: Uses a TOSLINK cable but a different datastream protocol. It can transmit up to eight channels of uncompressed audio at 24 bits, 48 kHz, or four 24-bit channels up to 96 kHz.

J) Thunderbolt 1 and 2: Used for high-speed data transmission, including digital audio, for Apple Mac computers.

K) High-Definition Media Interface, HDMI/HDMI2: The former (HDMI 1.4) carries 5.1 surround audio, and the latter (HDMI 2.1) carries 7.1 surround. Their connector types are exactly the same, but HDMI2 requires an *Ultra High Speed HDMI Cable* certification. There are also mini- and micro-HDMI connectors for both types.

L), M), N) USB: The Universal Serial Bus is one of the most widely used formats available for data, with digital audio being no exception. USB is designed to be backward-compatible, so no matter how fast the data throughput is, the cable must work for any device that uses it. Among the dozens of shapes and sizes, the three most common cables are USB-A, USB-B, and USB-C, respectively. The fastest standard is USB4, with speeds of up to 80 Gbps, using the type C cable.

N) Thunderbolt 3 and 4 for Apple Macs, which happen to be the same shape as USB-C. They both have a 40 Gbps data rate. Thunderbolt 3 also supports 4K video, and Thunderbolt 4 supports 8K video.

O) Ethernet, Category 7. Used for high-speed data i/o of multitrack audio and related data. It is also backward-compatible with previous Ethernet standards Cat-6 and 6a, Cat-5 and 5e, and Cat 3.

Patchbays

What if you need to switch outputs and inputs frequently? If everything is hard-wired to everything else, it makes re-cabling difficult and time-consuming. If you don't want to go behind your desk every time you need to switch to different speakers, or keep rerouting your drum machine back and forth to different channels on your audio interface, then you will need a **patchbay**. All of your inputs and outputs meet at this central hub: cables from all of your devices plug in behind the patchbay, which are accessed in front via short-length **patch cables**. How you set it up is up to you, but if you don't want a menagerie of spaghetti cabling, set up all of your devices' outputs on the top row and inputs on the bottom row. Short-length patch cables (preferably balanced) connect one output to one input, wherever you need to reroute a signal.

The standard project studio patchbay has 48 1/4″ balanced jacks (inputs) on the front and back, in a 19-inch rack space format, with 24 jacks on top and 24 on bottom. Larger studios with more equipment often use Tiny Telephone (TT) patchbays and cables – the same format the telephone companies once used for their switchboards – that fits 96 jacks into the same 19-inch space, but in the back are either a series of specialized connectors, or leads that require your wiring to be hand-soldered into place. TT is much more expensive than TRS, but both provide the same balanced connections from point to point. Balanced connections are preferred to prevent signal degradation.

As for setting up the patchbay, it's totally up to the workflow you like best. For example, when using effects, some audio designers prefer patch points (patchbay inputs and outputs), while others prefer to keep the connections as auxiliary sends and returns through a mixer (Figure 4.21).

MIDI Basics (a Hardware + Software Combination)

As mentioned briefly in the introduction, 1983 was a milestone year for electronic music. Finally, there was a protocol for electronic instruments and computers to communicate with one another, called the Musical Instrument Digital Interface, more famously known by its acronym **MIDI**.

A typical application is to connect a MIDI-enabled music keyboard to a computer. The composer records their performance into the computer, but they are only recording the *data* about their performance: the instrument type, volume level, velocity (how hard or soft a key was struck), panning, sustain pedal usage, and so on.

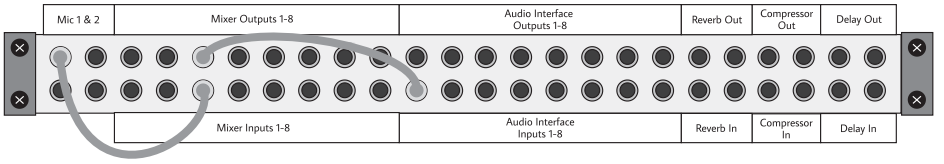


Figure 4.21 Illustration of a patchbay, with outputs at top and inputs at bottom.

There are many benefits to this process over traditional recording. First, the file size of a MIDI performance is incredibly small compared to WAV or AIFF files. One minute of full-fidelity stereo recording takes up about 10 MB; a MIDI performance of the same length may take up as little as 10 kilobytes, depending on instrumentation and performance data. Second, the MIDI file can be edited: notes can be added or removed, or the tempo and pitch changed – in fact, any parameter in the MIDI file can be modified as you desire. Third, performance-based composition is optional, as music can be manually scored via notation software like Avid's *Sibelius*, or *Finale* from MakeMusic. Music can even be "stepped" in one value at a time, using MIDI note data for all the purists out there! Fourth, MIDI allows musical ideas to be shared rapidly between composers, who can collaborate more easily if they share similar samples or virtual instruments – samples that can be mapped to a MIDI keyboard (more on this in a moment). And with the advent of a recent upgrade, there are sure to be more benefits to come.

At Last, MIDI 2.0

Technology always evolves rapidly – that is, except for MIDI. For nearly 40 years, the MIDI protocol had remained the same, still able to keep billions of MIDI-compatible devices speaking the same dialect but showing its age as it entered the 2000s. Thankfully, in January 2020, the MIDI Manufacturer's Association announced the release of MIDI 2.0, which offers two-way communication between devices (now with one cable!), greater memory access for features such as dynamic profiles (mapping parameter sets for specific instruments), and increasing its 7-bit values to 32-bit, greatly enhancing the resolution of performance data. The best news is that the protocol is backward-compatible, so the world's MIDI devices can keep on communicating with one another.[10]

Channels and Interfaces

MIDI uses channels to determine which devices are in the network, and where to send and receive data. Instruments can exist as physical hardware or software, in any combination. The original protocol could not exceed 16 channels. But with MIDI 2.0, there are now 16 groups of 16 channels each, for a total of 256 channels of simultaneous data. MIDI cables also once exclusively used round 5-pin DIN connectors, but nowadays they use USB or even Ethernet cables, the latter used for carrying the signal to greater distances.

Though many audio interfaces have MIDI support, there is also the option to use a **MIDI interface** in your setup. Many have versatile connectivity options like iConnectivity's *Mio* series, which offer 5-pin, USB, and Ethernet ports all on a single device. The MIDI interface works as the traffic cop for all devices, channels, and MIDI data flow.

MIDI Devices

MIDI-equipped hardware includes devices that have their own sound capabilities such as drum machines and synthesizers. There are also MIDI controllers,

which are devices that send and receive MIDI information but have no sounds of their own; those would be supplied by other connected hardware or software-based instruments. Controllers include MIDI keyboards, drums, breath controllers (for woodwind-based performances), or grid controllers such as the *Launchpad* series from Novation. The latter are comprised of a series of color-backlit, pressure-sensitive square pads in a grid layout, which are MIDI triggers that can be used to play samples or virtual instruments. For example, drum, vocal, and instrument samples could be loaded onto various pads, and then tapped with your fingers for a customized performance.

Sequencers

Composers and sound designers alike can create MIDI performances using a **sequencer**, which records MIDI events from other MIDI keyboards, drum machines, and sound generators. MIDI data such as instrument selection, note on/off, duration, pitch, and volume can be recorded, edited, and manipulated into any tempo, key, or time signature, among many other operations. In this way, music and sound are crafted and manipulated in ways that would otherwise be quite time-consuming or even impossible.

Many hardware-based models are available including Korg's Electribe and Elektron's Oktatrack sequencers, but software versions exist as well and are also included as part of some DAWs.

Samplers

As we saw in Chapter 2, samples are series of recorded snapshots at regular time intervals, capturing a sound wave in digitized form. If we wanted to perform a sound effect on a MIDI controller, or audition it at a certain pitch, we could use a **sampler** to do this. Audio is recorded and mapped to one or more keys on a MIDI controller; it can then be performed in a MIDI sequence, or recorded as new audio material (a sampled human voice played as a three-part chord, for example) (Figure 4.22).

There are also sampler and sequencer capabilities in many audio software applications, which we shall see in the next section.

Figure 4.22 The Novation Launchpad Pro and Launchkey MIDI Controllers (Courtesy Novation).[11]

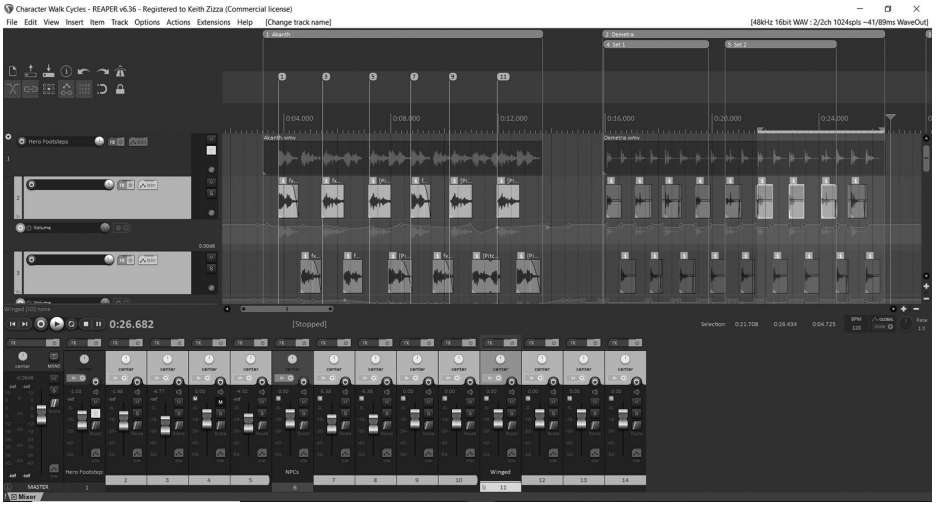


Figure 4.23 Image of a digital audio workstation (REAPER Software from Cockos, Inc.).

Software

Digital Audio Workstations (DAWs)

Not too long ago, audio software came in multiple "flavors": sequencers, editors, samplers, signal processors, and mastering tools, to name a few (Figure 4.23). Today, most professional audio software can be considered a DAW that contains some or all of these features. There are many reputable brands that have stood the test of time, for example: Pro Tools (Avid), Cubase (Steinberg), Ableton Live (Ableton), Studio One (PreSonus), and REAPER (Cockos). Each has a style and feel of their own that may be preferable to your workflow; I highly recommend trying out several DAW trials or demos if you can (in the next two chapters our focus will be on REAPER, which is free to evaluate). As game audio designers, the DAW is a crucial component of our work. Here are some popular features of a typical DAW:

Editor/Mixer

Used to set up digital audio (and sometimes MIDI) on a timeline, and regions of audio can be trimmed, cut, copied, pasted, and so on, often with video support. Multiple tracks of audio are the norm, ranging from 16 tracks to hundreds. Effects processing can also be applied to one or more audio selections, busses (if available), or on the main mix.

Sequencers

Like their hardware counterparts, sequencers are required for the recording, playback, and editing of MIDI events and music production. There are a wide range of sequencers that may focus on live performance, loop-based editing, and/or notation-based composition. Anything beyond an entry-level sequencer usually offers basic

video support for syncing compositions to visuals. And as mentioned earlier, many DAWs offer this capability natively.

Plug-ins

Plug-ins are add-on software tools that work within the DAW and process digital audio, either in-line with the audio itself (for live playback), or processed for later use (rendering). Because many of them are signal processors, they are sometimes referred to as "effects" or "FX." But plug-ins can also be samplers or virtual instruments (see below). Popular options include compression, EQ, reverb, delay, and modulation, all which we will cover in Chapter 6.

Samplers

Like its hardware predecessor, the sampler records and digitizes incoming audio and can be used as a MIDI instrument for sequencing or live performance. Samplers can be DAW-based like IK Multimedia's *SampleTank*, or added on as a plug-in, such as Serato's *Serato Sample*. Many samples can be recorded and mapped to a MIDI controller, either as different sounds or many musical notes that form a virtual instrument (below).

Virtual Instruments

Used for live performance or for sequencing, a **virtual instrument** (VI) is a collection of samples that are mapped to MIDI notes and are accessed via a GUI (graphical user interface). It can operate within a DAW as a plug-in or as a stand-alone application. There are VIs that are comprised of symphonic instruments (brass, woodwinds, strings, percussion) that painstakingly sample all of the notes of an instrument including its variety of articulations: for instance, a violin VI requires all of its playable notes but also bowing techniques (*pizzicato*, *col legno*, *arco*, and many more) as well as dynamics (*pianissimo*, *forte*, etc.).

Electronic VIs model the schematics of its own analog circuitry. One example is with Vangelis's soundtrack to *Blade Runner*, which showcased dreamy synth sounds from the Yamaha CS-80 analog synthesizer; today it would cost you more than $20,000 USD to buy a used one, if you can even find one in good shape; they also weigh about 200 pounds and go out of tune quite often! In this case, it's much better to use the VI, in Arturia's *CS-80 V*. The interface even looks (and functions) like the real thing.

Mastering

In music, sound, and dialogue production, **mastering** is the final phase of post-production where finishing touches are added. A collection of songs might need to be balanced frequency-wise, to feel like a cohesive musical experience. Or upon a final listening session, you may decide that a set of character voices may need some brightening in the high frequency range, while others need a low-frequency boost, and so on. Much can be achieved through dynamics and equalization, and mastering tools help facilitate the process. They are available as expensive hardware options

(a rack-mounted mastering compressor can cost thousands of dollars) and also as a stand-alone software or plug-ins for a fraction of the cost. Some notable examples include IK Multimedia's *T-RackS* series and *Ozone* from Izotope.

Batch Processing

Audio designers are prone to editing multiple copies of things. Some games have thousands of lines of dialogue, for example, and how can we create individual files from raw data in a timely fashion be created into individual files? Or what if you'd like to take 5,000 sound effects and add an EQ filter to them? Or just convert from WAV to MP3? With a **batch processor**, you can select the files you want to process and export, and with a click of a button it can get the work done while you focus on other things.

Sound Generation Software

Sound effects can also be created digitally with software. *GameSynth* from Tsugi (in Figure 4.24) allows the user to create and render procedurally generated sound effects, including modular synthesis, particle effects, impacts, footsteps, and random variations per sound. Sounds can be built with modular components, or drawn in to create instant effects. The software can be coupled with their proprietary runtime engine that acts as middleware, harnessing GameSynth's capabilities during live gameplay.[12] *Sound Particles* by the company of the same name uses a 3D graphics engine for designing and processing complex sounds, as well as auditioning

Figure 4.24 The Yamaha CS-80 as a virtual instrument, as seen in Arturia's CS-80 V virtual instrument (Courtesy Arturia).

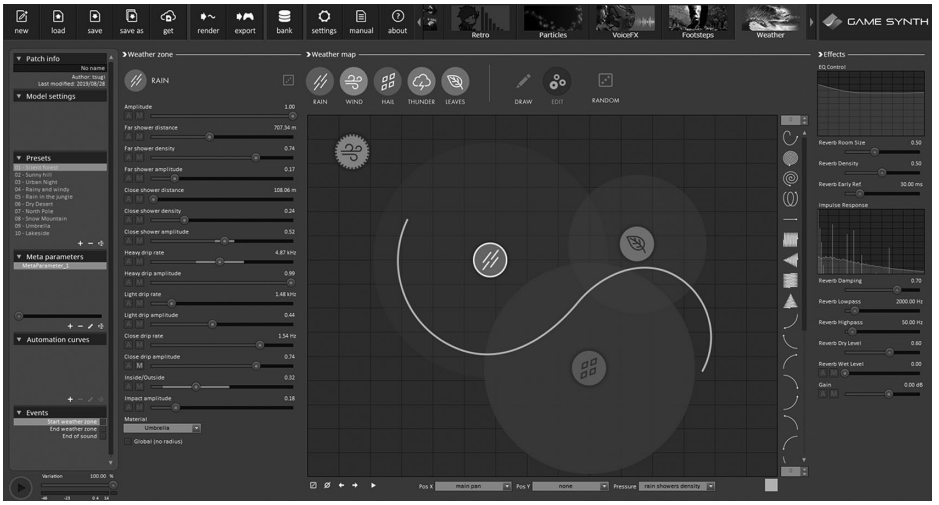


Figure 4.25 GameSynth from Tsugi allows the user to create a multitude of procedurally generated sounds, all through the use of a graphical interface (Courtesy Tsugi GK).

sounds in a spatial audio mode. You can even add head-tracking through the use of ambisonic microphones (Figure 4.25).[13]

Notes

1 Pdf. "SSD VS HDD: Which Is Best for You?" Intel. Accessed August 17, 2022. https://www.intel.com/content/www/us/en/products/docs/memory-storage/solid-state-drives/ssd-vs-hdd.html.
2 Zoom. "R8 Recorder: Buy Now." ZOOM. Accessed August 17, 2022. https://zoomcorp.com/en/us/digital-mixer-multi-track-recorders/multi-track-recorders/r8/.
3 "DP-03SD: Overview: TASCAM - United States." TASCAM. Accessed August 17, 2022. https://tascam.com/us/product/dp-03sd/top.
4 "DP-24SD: Overview: TASCAM - United States." TASCAM. Accessed August 17, 2022. https://tascam.com/us/product/dp-24sd/top.
5 Ltd., Magnolia International. "Product: X-TOUCH One." Behringer. Accessed August 17, 2022. https://www.behringer.com/product.html?modelCode=P0CAP.
6 "Workflow: Audio and Music Production." Monogram. Accessed August 17, 2022. https://monogramcc.com/workflows/audio/.
7 Blue Spark Mic and images © Blue (Baltic Latvian Universal Electronics, LLC)/Logitech.
8 "Appendix C – Dolby Atmos Delivery Codecs." Dolby, 2021. https://learning.dolby.com/hc/en-us/articles/4406039180564-Appendix-C-Dolby-Atmos-Delivery-Codecs-.
9 Robin, Michael. "Technical Notes – 1 the AES/EBU Digital Audio Signal Distribution Standard." McGill University. Accessed August 18, 2022. http://www.cim.mcgill.ca/~jer/courses/comparch/assignments/03/as1/aes-ebu.pdf.
10 "Details about MIDI 2.0™, Midi-Ci, Profiles and Property Exchange." The MIDI Association, 2020. https://www.midi.org/midi-articles/details-about-midi-2-0-midi-ci-profiles-and-property-exchange.
11 2021 © Focusrite Audio Engineering Limited. All rights reserved. Game Audio Fundamentals has not been endorsed by Focusrite. Novation, Launchpad, Launchkey and the N logo

are registered trademarks of Focusrite Audio Engineering Limited in the United Kingdom and other countries.

12 "GameSynth: Tsugi: Software for Creatives." Tsugi GK, 2022. http://tsugi-studio.com/web/en/products-gamesynth.html.

13 Fonseca, Nuno. "Sound Particles." Sound Particles, 2022. https://soundparticles.com/products/soundparticles/overview.

Bibliography

Cancellaro, Joseph. *Exploring Sound Design for Interactive Media*. Clifton Park, NY: Thomson, 2007.

Huber, David Miles, and Philip Williams. *Professional Microphone Techniques*. Emeryville, CA: Mix Books, 1999.

Huber, David Miles, and Robert E. Runstein. *Modern Recording Techniques*. 9th ed. New York, NY: Routledge, 2018.

Marks, Aaron, and Jeannie Novak. *Game Development Essentials: Game Audio Development*. Clifton Park, NY: Delmar Learning, 2009.

Pejrolo, Andrea, and Scott B. Metcalfe. *Creating Sounds from Scratch: A Practical Guide to Music Synthesis for Producers and Composers*. New York, NY: Oxford University Press, 2017.

Thompson, Daniel M. *Understanding Audio: Getting the Most Out of Your Project or Professional Recording Studio*. 2nd ed. Boston, MA: Berklee Press, 2018.


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Part II

Soundscapes and Disciplines


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 5

DAW Essentials, Part I

Editing, Workflow, and Rendering

Emergence of Digital Audio Production

Back in the late 1980s/early 1990s, reel-to-reel tape was still the dominant medium of choice in audio production. Edits were done using razor blades, a chopping block, and splicing tape. Digital audio was gaining in popularity, though it was still quite expensive to put together a system for professional recording and editing. Today, we have shifted away almost entirely from multitrack tape and embraced the digital domain, with increasingly innovative technologies allowing us to discover new creative processes free from the confines of analog mediums – and reinventing what it means to truly be an "audio designer." As with any discipline there are best practices, and with digital audio production, ultimately you will find a workflow that works best for you; it will likely evolve over time as you continue to hone your skills.

Choosing a DAW

A **digital audio workstation**, or **DAW**, is a place you'll spend a lot of time in as a game audio designer. At a minimum, it is a domain where digital audio can be recorded or imported and added to a timeline, possibly mixed with one or more other audio assets, then exported as a new audio file. Most DAWs will also offer multitrack editing and mixing and effects processing; some DAWs will include additional features such as batch processing (converting a list of files according to a set of user-defined tasks), scripting, video support, and music composition tools.

DAWs provide a **nondestructive editing** environment, so that splicing up media items has no effect on the original material; your sound files are safe on your media drive, regardless of how you manipulate them (unless you explicitly state in the DAW to overwrite media on your drive), so you can slice and dice up audio to your heart's content. Similarly, you can make multiple copies of the same media in a DAW without duplicating it externally.

DAWs also feature **nonlinear editing**, such that you can duplicate, trim, cut, copy, and paste your audio anywhere along a timeline, including layering multiple tracks. Once you are satisfied with the results, you can **render** out the session, meaning everything you hear from start to finish in your project will be exported as a new audio file.

DOI: 10.4324/9781003218821-8

An Introduction to REAPER

Why REAPER?

There are many solid, industry-leading DAWs available to use. But as a game audio designer, REAPER (**R**apid **E**nvironment for **A**udio **P**roduction, **E**ngineering, and **R**ecording) holds a special place in my heart. There are so many good reasons to use REAPER, and here are a few:

- It is free to evaluate for a generous amount of time (but yes, please purchase a commercial license when you start making revenue from your work!).
- It is extremely robust; crashes and hangs are few and far between.
- The install size is remarkably small, and it is currently supported on Windows, Mac OS, and Linux.
- REAPER provides useful and frequent updates. Somehow, they are always finding new features to add or improve upon, fixing any known bugs, or enhancing workflow.
- It comes with free plug-ins, which include effects, dynamics controllers, EQs, and MIDI tools; at last count, there are over 250 in all!
- Many audio professionals in the game industry have adopted REAPER, so you'll already have a big plus in your skillset by learning it.
- It is completely customizable, from its controls and commands to the interface and even scripting through the ReaScript language.
- It supports a large number of file formats, both to import and to render, and offers customizable batch processing, which is great when needing to render hundreds – or thousands – of audio files.
- Nearly every process can be automated in real time, including effects parameters (e.g., EQ filters, fader levels).
- There are several ways to perform just about every action, and there are always new and useful features around the corner.
- The companion book *Up and Running: A REAPER User Guide* (covering version 6) has excellent, practical examples of REAPER in action (https://reaper.fm/userguide.php). The author Geoffrey Francis has provided special permission for the use of some of the guide's visuals and tables in this book. Though the PDF is available on the REAPER website, nothing beats a printed, bound version to explore; the REAPER 6 User Guide is available at www.lulu.com.

REAPER in Game Audio Production

Many game audio folks swear by using REAPER for all of their audio work: creating sound effects, soundscapes, dialogue editing and processing, syncing to video, and even music composition and production. While REAPER is easy to use, it is also quite deep, allowing you to take advantage of its advanced features as needed.

While it is beyond the scope of this book to explain how to fully use REAPER, this chapter and the next will cover the basics for importing media, editing, mixing, using plug-ins, and rendering audio files, relating to game audio production. As mentioned earlier, *Up and Running: A REAPER User Guide* is a definite must to check out.

At over 400 pages, it's an excellent reference for specific commands, plug-ins, and workflow examples. Lastly, there are some fantastic video tutorials that can be found online, including those on the REAPER website and on YouTube (try searching on "REAPER Basics" or "REAPER Tutorial").

Getting Started

You can download REAPER at https://www.reaper.fm. Locate the version of 6.x you need, either the 32- or 64-bit install for Windows, Mac, or Linux. Once you've installed it, double-click on your desktop REAPER icon and we can begin! The key commands that follow will be Windows-based, but on a Mac, you can substitute the **Command** ⌘ key for the **Windows** ' key, and the **Option** key for **Alt**.

Overview

The first thing you'll see is a relatively empty space that should look something like this (Figure 5.1).

Adding some audio content (which will be covered in a moment) and labeling some of the main interface elements, we have the following (Figure 5.2).

We will refer back to this example often, so go ahead and toss in a bookmark here!

Input Monitoring versus Direct Monitoring

When you are recording live audio into your computer-based DAW, an **input monitor** allows you to hear what you are recording *after* the computer received the signal and sent it back to the audio interface. Any live audio traveling through your DAW and back to you is going to introduce some latency. Some audio interfaces offer "zero latency," which means they are providing a **direct monitor** through an output such as

Figure 5.1 REAPER 6.x default setup for a new project.

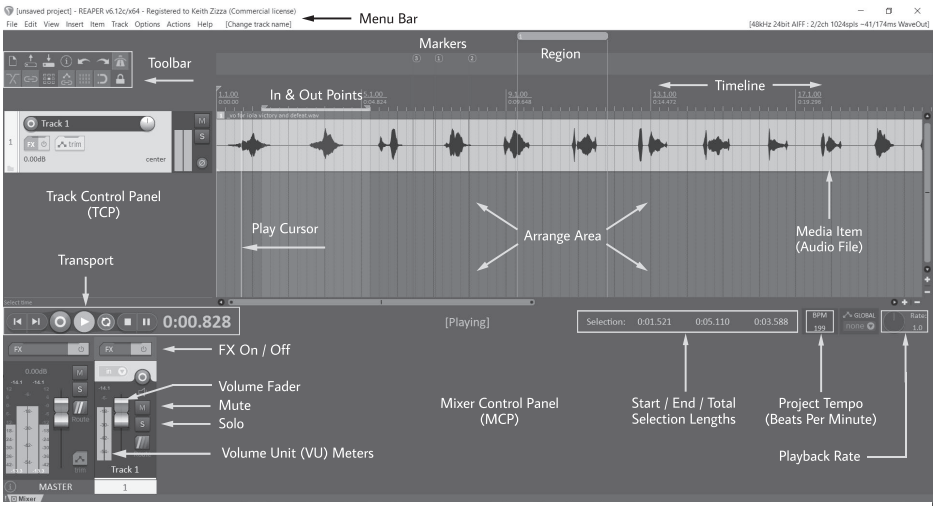


Figure 5.2 REAPER 6.x with a track and audio added, along with labels for some of its main interface elements.

the headphones: you're listening to the audio signal immediately after it reaches the interface, but *before* it travels to your computer or other destination device. It's actually *near*-zero latency, as the signal still needs to travel through the circuitry of the interface first, but this could be measured in the single-digit number of milliseconds.

For example, if you want to add vocals to your song, you might record using direct monitoring so you could hear yourself in headphones delay-free, but if you are also trying play back music on the computer to sing along with, that audio will be delayed once again because of latency. If the latency issues can't be resolved (buffers and samples be darned!), some DAWs try to bridge the problem by offering a direct monitor on/off toggle when recording; there are also audio interfaces that allow you to dial in a blend of direct-versus-input monitoring to find a "happy medium" of sorts.

The Main Toolbar

Let's take a look at the main toolbar icons in Figure 5.3, starting from left to right in the top row.

Top row: New project, Open project, Save project, Project Settings, Undo, Redo, Metronome toggle.

Bottom row: Auto Crossfade toggle, Item Grouping toggle, Ripple Editing options, Move Envelope Points toggle, Grid lines toggle, Snap toggle, Locking toggle.

The transport is relatively straightforward with the usual controls, from left to right with corresponding key commands in [brackets], we have:

Go to start of project [**Home**], Go to end of project [**End**], Record [**Control+R**], Play [**Media Keyboard Play**], Toggle Repeat [**R**], Stop [**Media Keyboard Stop**], Pause [**Media Keyboard Pause**], and Current time (which can be set to frames, samples, minutes and seconds, seconds, or beats). Play/Stop toggle can also be controlled by pressing [**Spacebar**].

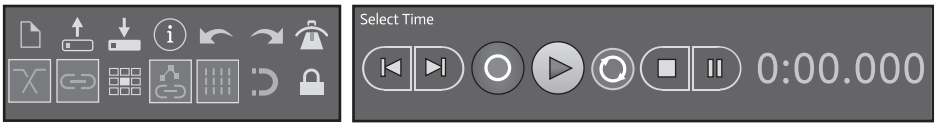


Figure 5.3 REAPER main toolbar and transport.

If a toolbar icon has a green background, that means it is **on**. Make sure that the Metronome toggle is off, as well as the Snap toggle.

The Timeline

REAPER 6.x defaults to a format that uses bars and beats (for music-based production), but we are going to switch to a minutes-and-seconds format. Hover your cursor up to the timeline, right-click and a popup will appear for ruler type; choose either **Minutes and Seconds** or **Seconds**.

Tracks and Channels

REAPER can have unlimited tracks (dependent upon your computer's limitations), and up to 64 channels *per track*, but for our purposes, each channel will be either mono or stereo. You can create a track automatically by importing media (below), or by selecting **Insert → New Track**, or by pressing **[Control+T]**. It so happens that REAPER is very smart about handling media items: you can have a mono, stereo, and multichannel audio file on the same track. You can even throw MIDI data and video on that track as well!

Importing Media, and Resources

To add audio to your project, select **Insert** and locate the file(s) you wish to add. If more than one is selected, you will be prompted to add them all on one track, or multiple tracks.

If you need some sample audio content, head over to the **Companion Website** under **REAPER Audio Files**, in the Student Exercises → Exercise Materials section. I also highly recommend **www.freesound.org** as a resource.[1] It's free to sign up, and audio content is free to use so long as you cite the creator(s) via the **Creative Commons** guidelines.[2] And who knows, perhaps in time you'll contribute to the freesound archive as well!

Recording Audio

Many of the REAPER exercises on the **Companion Website** involve pre-recorded audio. But if you wish to record your own material, simply arm the track you want to record on by pressing the track's red **Record** button, then the main **Record** button on the transport. Stop the recording by pressing transport Record button again, or **[Spacebar]**. If you choose the latter, REAPER will ask you (through a popup window) if you would like to **Save**, **Delete**, or **Rename** the recorded media.

Editing 101

Below is a list of basic editing features for your reference, which can also be found in the ***Up and Running*** guide (Table 5.1).

REAPER also has smart editing features that allow for more complex manipulation of media items, as shown in Table 5.2. For some of these features to work properly, you will have to disable **Options → Loop Points Linked to Time Selection.**

At this point, you should go ahead and import an audio file into REAPER. It can be just about anything – we'll be using it to examine some of REAPER's basic editing features.

REAPER Icons

Some of the most common icons you will see while editing are shown in Figure 5.4, with a corresponding legend in Table 5.3.

Table 5.1 REAPER Basic Editing Features for Media Items

To do this …	*You need to do this …*
Select media items	To select a single item, **Left click**. To add more items to the selection, **Ctrl left click**. To select a range of adjacent items, **Left click** on first item in range, **Shift left click** on last.
Split items(s) at edit cursor position	To split all items, make sure no item is selected, press **S**. To split one selected item or a selection of items, press **S**. Item FX will be applied to both items unless Media Preferences option **Duplicate take FX when splitting items** is disabled.
To join split items together (heal)	Select items. Choose **Heal splits in items** from the context menu.
To join selection of items (glue)	Choose **Glue** from the context menu (renders to new file).
To move item(s) with contents	Select item(s). Use Num Pad 4 (left), 6 (right), 2 (down), 8 (up).
To slide contents within item	Select item(s). Use Num Pad 1 (left), 3 (right).
To move item but not contents	Select item(s). Use Num Pad 7 (left), 0 (right).
To delete selected items(s)	Press **Delete** or chose **Remove items** from context menu.
To cut/paste selected item(s)	Press **Ctrl X**. Reposition edit cursor, press **Ctrl V**.
To copy/paste selected item(s)	Press **Ctrl C**. Reposition edit cursor, press **Ctrl V**.
To cut that part of selected item(s) defined by time selection	Press **Ctrl Shift X** or choose **Cut** selected area of items from context menu. Reposition edit cursor, press **Ctrl V**.
To copy that part of selected item(s) defined by time selection	Press **Ctrl Shift C** or choose **Copy** selected area of items from context menu. Reposition edit cursor, press **Ctrl V**.

Reprinted with kind permission from Geoffrey Francis.

Table 5.2 REAPER Smart Editing Features for Media Items

To do this with an item or selection of items …	*You need to do this …*
Select a single item	Click on the item.
Select part of item	Left drag on background area in arrange view to make time selection, then click on the item.
Select several items or part of several items	Make any time selection. Right drag to marquee items, or **Ctrl** click for non-adjacent items.
Delete selected item or items	Select item(s), press **Delete**.
Delete part of selected item or items	Make time selection of item(s). Press **Ctrl Del**.
Move item/items, ignore any time selection	Select item(s) and drag.
Move item/items, ignore snap and time selection	Select items and **Shift** drag.
Copy selected portion of item or items	Click on one item within time section and **Ctrl** drag. **Shift Ctrl** drag to ignore snap.
Copy entire item or selected items (ignore time selection)	Click on one item outside time section and **Ctrl** drag. **Shift Ctrl** drag to ignore snap.
Move item contents (within item)	**Alt** drag.
Adjust item pitch fine	**Shift Alt** drag up or down.
Render item to new file	**Ctrl Alt** drag.
Copy item, pooling MIDI source data	**Shift Ctrl Alt** drag.

Reprinted with kind permission from Geoffrey Francis.

Table 5.3 Legend for Common REAPER Editing Icons

A	Time and item select	**N**	Crossfade width
B	Select and move	**O**	Crossfade move
C	Adjust window size horizontally	**P**	Crossfade curve type
D	Adjust window size vertically	**Q**	Trim start of media item
E	Adjust window size diagonally	**R**	Trim end of media item
F	Adjust window size diagonally	**S**	Trim end/start of media item pair
G	Insert point on envelope	**T**	Adjust start offset (slip edit)
H	Drag and drop media item	**U**	Move through timeline
I	Delete item	**V**	Time compress/expand from start of media item
J	Adjust offset	**W**	Time compress/expand from end of media item
K	Adjust start fade	**X**	Time compress/expand end/start of media item pair
L	Adjust end fade	**Y**	Pencil tool
M	Zoom	**Z**	Eraser tool

Volume and Panning

There are several possible stages where volume can be adjusted. Referring to Figure 5.6, pressing **V** will expose the volume envelope for the selected track, which is a panel with a green envelope. The **P** key will expose the panning envelope in a similar fashion, using an orange envelope. Press **V** or **P** again to hide either envelope. To add

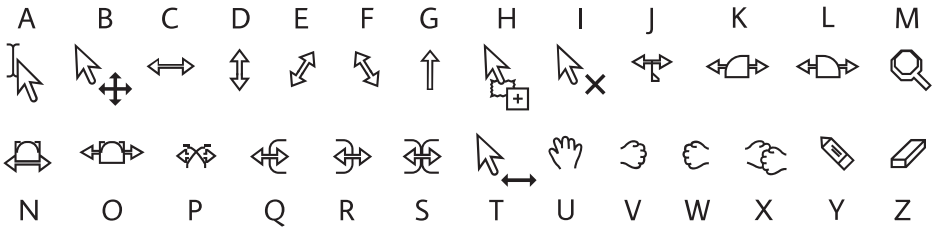


Figure 5.4 Common REAPER editing icons.

Figure 5.5 REAPER Track on TCP with Volume and Panning.

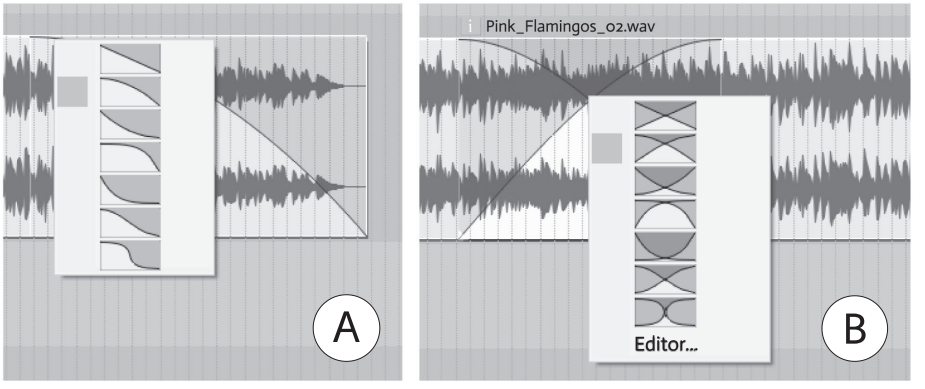


Figure 5.6 Fade types for (A) a single media item and (B) a pair of crossfaded media items.

a point, **[Shift] double-click** on an envelope; **Alt click** will remove a point. Right clicking on an envelope will present several selection options you can experiment with.

Moving to the top of the media item (your audio clip) will make the cursor turn from a pointer (Figure 5.5A) to an up-and-down vertical arrow (Figure 5.5D); **left click + drag down** and a thin horizontal red line will appear, as seen in Figure 5.6. This is another way to adjust the volume for a specific media clip. You can also go

to the top left or right corners of the audio clip to fade in and out; these fades can be extended within the media item by hovering the cursor near one, and **left click + drag right** for the fade in, and **left click + drag left** for the fade out.

Lastly, each track has its own volume fader, as seen in Figure 5.2. This is the final volume stage for the track before it gets sent to the **Master Output** (the track on the far left, in Figure 5.2). All tracks are routed here, and it acts as the master control for everything in the project including volume.

Fades and Crossfades

Fades in most DAWs come in several varieties. REAPER defaults to an equal power (curved) fade in/out, but you can experiment with your media items by trying various fade types including linear, a variety of curved fades, and an "s" curve option, as seen in Figure 5.7A. Hover over a fade, and when the icon turns to a curve (Figure 5.5K, L, or O), right-click on a fade and select the shape you prefer.

When two media items intersect, they will **crossfade** if the **Auto Crossfade** toggle (in the main toolbar) is on. With crossfades, the media item on the left fades out while the one on the right fades in. You can **right click** on the center of the crossfade (the cursor will turn into the icon in Figure 5.5P) and select the crossfade type. A more complex menu will appear if you choose "**Editor…**" (the **Crossfade Editor**) at the bottom of the list, and there, you can tweak the fade pair anyway you like.

To stretch the crossfade in either direction, hover to its center, **left click + drag left or right** (the icon in Figure 5.5N will appear). To move the entire crossfade, hover to its center, then **[Shift] left click + drag left or right** (the icon in Figure 5.5O will appear).

A Word (OK, Two Words) on Volume: AVOID CLIPPING

Whenever you can, it's best to avoid **clipping**, i.e., going beyond the "0 dB" reading on your track meters. REAPER will be forgiving of peaks that go beyond 0 dB, and this is actually fine for individual tracks now and then, but I would never recommend it on the Master Output. We can't really go past 0 dB. At that point, all of our digital audio bits are effectively "1"s and we can go no further! If you render a WAV file, for example, where you consistently peak over 0 dB, the audio data may be compromised, glitches and errors may be introduced, and so on. There is no telling what will happen on the playback systems of game consoles, mobile devices, PCs, and Macs, that all have different tolerances to audio files of this nature. Aside from simple volume attenuation, a couple of ways to avoid this situation will be offered in the next chapter.

Markers and Regions

Pressing **M** creates a marker; you can **left click** on the marker number and drag it to reposition it, or **double click** to rename it. It's a great way to label a certain media item or upcoming section of the project.

Regions can be created by left-clicking and dragging an in and out point on the timeline and pressing **R**. The region size can be changed by left-clicking and dragging on either side. You can rename it by pressing **Shift + Double Click**; simply

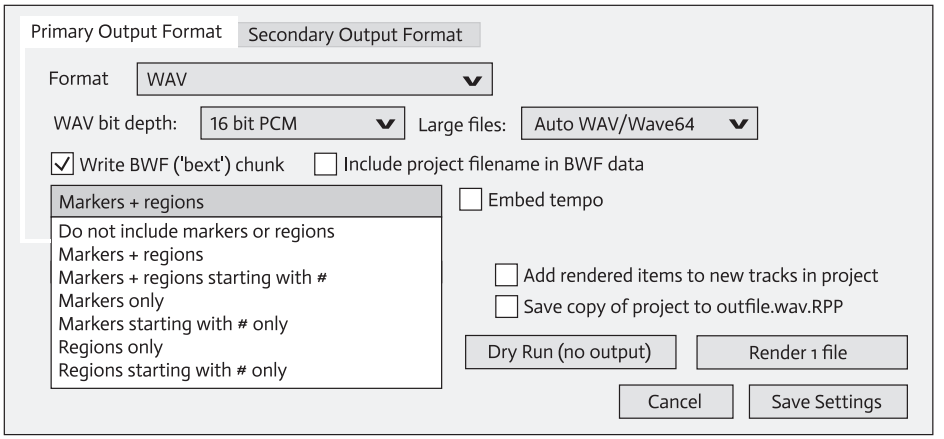


Figure 5.7 The dropdown menu for saving markers and regions in a WAV file (bottom left).

left clicking on it twice will select the region, which is convenient to start playing audio at that area of the project, or to only **Render** that region (as we will see later in this chapter).

Markers and regions are great not just for being a locator in your REAPER project but also as something a programmer can utilize later in-game. Specifically, there is an option to save markers and/or regions in WAV files, which can serve as a new file position to jump to, or as locations for looping, shown in Figure 5.7.

Splitting and Trimming

As noted in Table 5.1, splitting an item is as easy as hovering the cursor over the desired split point and pressing **S**. To trim the start or end of a media item, hover over to the left or right edge of the item (at about 25% of the height of the clip or lower) until the cursor changes to a double horizontal arrow over a curved bracket (Figure 5.4Q and R). Here, left click and drag left/right to adjust the trim size. If you have two media items perfectly adjacent to one another (perhaps a media item that was split), select both and hover over where they meet, and the cursor will change to a double horizontal arrow over a double-curved bracket (Figure 5.4S). Left click and drag to move the split point between the two items.

Time Compression and Expansion

One powerful feature in the REAPER editor is the ability to speed up/slow down a media item *without changing the pitch*. What is this sorcery? It is called **time compression (and expansion)** and can be applied to your media item simply by hovering your cursor over its left or right edge, and while holding down the **Alt** key, left-click and drag the mouse to the left or right. Your cursor should turn into a grip symbol, seen in Figure 5.4V or W, and grow or shrink the media item itself. Making

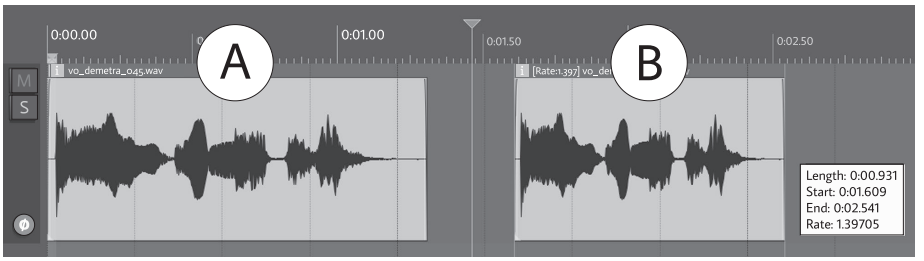


Figure 5.8 Example of time compression and expansion of an audio item. Note the increase in (B)'s playback rate on the bottom right, due to compressing the audio length.

it smaller results in time compression, and thus speeds up the audio. Similarly, making it larger results in time expansion, but this has mixed results (Figure 5.8). Removing time slices via compression usually works pretty well, but when expanding, REAPER has to create new audio in between time slices of the media item, which can result in "burbly" artifacts or other strange phenomena. If you're not happy with the results, you can modify the method of time compression/expansion by going to your media item's properties (**F2** key), locating the dropdown menu for "Take pitch shift/time stretch mode" and selecting a different method. But anything overly time expanded won't sound that well, unless you're going for something glitchy-sounding!

Folders (Busses)

In REAPER, tracks can be routed into a **Folder**, a central hub that other tracks are a part of. On a traditional mixer these are known as **busses**, as we saw in Chapter 4. A REAPER Folder is considered a **parent** track, and the tracks inside of it the **children**. So, if you add volume or panning envelopes, or effects processing to the folder, the child tracks are affected as well as in Figure 5.9.

At a minimum, Folders are also great for keeping your tracks organized. Your birds, wind, and cicadas could be sent to an Ambience Folder, and your cymbal, hihat, and timbales could live in a Percussion Folder, for example.

In Chapter 12, you will discover that certain software tools may call busses something else entirely! But the concept of parent–child track grouping remains the same.

Rendering Audio Files

When we are satisfied with our work, it's time to export it as a new audio file, otherwise known as **rendering**. This is where we have options like setting the outgoing sample rate, bit depth, and file format. There is even an option to apply dithering and noise shaping, but this is primarily used when converting 24 bit and higher audio down to 16, for a smoother transition on downsampling. There is also a Normalize feature which outputs your audio to a particular loudness level (which we'll see more of in Chapter 6).

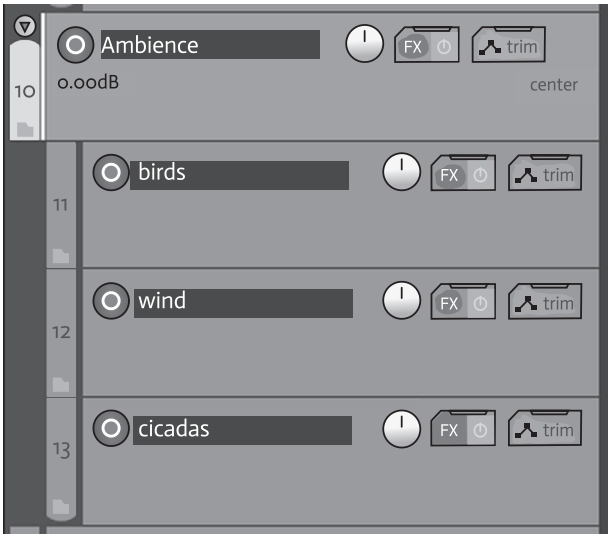


Figure 5.9 Left-clicking the folder icon on the bottom left of Track 10 makes it a folder; it becomes a parent track, with tracks 11–13 here now its children. Clicking the folder icon again restores the child tracks to normal.

Audio File Formats

Which Format Is Right for My Project?

There are many file formats to choose from, but essentially, they boil down to two main categories: **compressed** and **uncompressed** (oh no, there's that term compression again!). Compression in this context means the data in the file has been reduced, in order to minimize its size.

Uncompressed

This is a great choice for short-length audio files that can be loaded directly into memory. You may have lots of footsteps, weapons, and emotes that need to be played *a lot* during gameplay. Loading them into RAM ensures the fastest access and playback times possible, but the files need to be relatively short (measured in seconds, not minutes!) to conserve this valuable storage resource. The size of a full-fidelity stereo file (16 bit, 44.1 kHz audio) is about **10 MB per minute**. The most common uncompressed audio formats are **WAV** (Waveform Audio File Format, originally from Microsoft) and **AIFF** (Apple's Audio Interchange File Format). Both use PCM (pulse code modulation, from Chapter 2) to encode sample data. WAV is the predominant format that most game engines support. Both formats have a maximum file size of 4 GB, but another uncompressed format, Apple's **CAF** (Core Audio File), does not.

Compressed

These are better to use for longer-playing files, and/or when memory storage is scarce; compressed files only take up a tiny bit of RAM as they are played back in

small chunks, but this comes at a slight CPU cost. Full-length songs, or ambience that runs for minutes at a time, either stereo or multichannel, can benefit from a compressed format. The first packet of data is loaded into RAM and played, then the next packet is loaded as the previous one is discarded, and so on. This repeats until the file completes playing. Its file size depends on its **bit rate** (measured in kilobits per second, or **kbps**). Generally, 128 kbps is acceptable for most purposes, but full-fidelity audio begins at 160 kbps; higher values are less compressed and improve in quality, with a maximum value of 320 kbps. A 128 kbps stereo file is about **1 MB per minute**.

Lossless versus Lossy

Audio files that are **lossless** means they contain audio as it was recorded, without any reduction or coloration to the content.

Lossless formats include **FLAC** (the **F**ree **L**ossless **A**udio **C**odec), **ALAC** (**A**pple **L**ossless **A**udio **C**odec), and **WavPack**, to name a few. Lossless files are compressed, much in the way a ZIP file is compressed; upon restoration, all of the data is intact. The files aren't used much in games, however, as there isn't a lot of code support in game engines for them, and they take extra time to decompress. WavPack can contain audio of up to 32 bits, 24 bits for FLAC, and 16 bits for ALAC.

Compressed formats are **lossy**, which means audio data has been removed (such as frequencies overlapping) to reduce file size. The higher the compression, the more data is removed, and audio quality suffers. The most common two formats are MP3 and OGG Vorbis (from Xiph). Some game engines such as Unity support OGG files, and OGG is the preferred choice over MP3 because (1) OGG files encode data more efficiently than MP3 files,[3] and (2) MP3s cannot seamlessly loop due its format structure, adding a small amount of blank space at the start of each file. Other lossy formats include **AAC** (Advanced Audio Coding) and **WMA** (Windows Media Audio), both of which improved upon MP3 quality at the same bitrates. There are also lossy formats that are encoded specifically for surround sound, below.

Stereo versus Mono

This depends on how the audio will be used in your game. If it needs to be positioned or movable in a 2D or 3D world, **mono** is the preferred choice, as it can travel anywhere between your speakers. Stereo is best for fixed-in-place audio such as with music playback.

Multichannel (Surround) Formats

If you need to render multichannel audio for something like a cinematic or cutscene, the most popular multichannel audio formats are WAV and AIFF, Dolby Digital, and DTS. Dolby Digital and DTS require a license and encoder system. Dolby has several varieties of encoders; the venerable AC-3 format is compatible with most systems but only handles 5.1 surround; Dolby Digital plus (E-AC-3) handles up to 15.1, and Dolby Atmos can handle an even more immense array of floor and ceiling-based speakers, for deeper sound immersion.

Ambisonics is another format that is on the rise, but this is beyond the scope of what is covered in this textbook. However, there are some great books on spatial audio available; I recommend starting with *New Realities in Audio: A Practical Guide for VR, AR, MR and 360* by Stephan Schütze.

Last but Not Least: Actions and SWS Extensions

Nearly every command you can issue in REAPER is an **Action**. You will notice there is even an Actions menu item: want to open a file? Create a region? Toggle ripple editing per track? It's all there! We have the convenience of menu-based commands and keyboard shortcuts, but they all start as an Action.

There are a series of additional Actions that go beyond the basics, which together are called the SWS (Standing Water Studios) Extension – a collection of open-source features that add to the already voluminous list of REAPER Actions. We'll cover an example or two using SWS Actions in Chapter 6; if you'd like to check them out, please go to https://sws-extension.org and download the version you need for Windows, Mac OS, or Linux. They will seamlessly integrate into Reaper once installed.

Beyond keyboard shortcuts, Actions are also the basis for creating macros and scripts, though this is a topic for another book. If you are interested in automating some REAPER tasks, however, I highly recommend reading *REAPER Plus! The Power of SWS Extensions*, another informative guide by Geoffrey Francis. It can be found at https://www.standingwaterstudios.com/download/REAPERPlusSWS171.pdf (or a printed version at https://www.lulu.com). You can also check out the *The Reaper Blog* for up-to-date SWS tips and tricks (and much more) at www.reaperblog.net.

Finding Your Workflow

Learning the basics of DAW editing and mixing are vital to your success as a game audio designer. You should strive to achieve a lucid connection between your creative inspirations and the tools you use to realize them. The faster the translation from your mind to the computer, the more effortless it will be to express your ideas in the digital domain.

Recommended Reading and Viewing

Up and Running: A REAPER User Guide,
REAPER Plus! The Power of SWS Extensions, and
ReaMix: Breaking the Barriers with REAPER, can all be found at https://www.lulu.com/spotlight/glazfolk.

The Reaper Blog: https://www.reaperblog.net.

The Reaper Blog YouTube Channel: https://www.youtube.com/c/thereaperblog.

Kenny Gioia's REAPER YouTube Channel: https://www.youtube.com/c/REAPERMania.

Notes

1 https://www.freesound.org.
2 https://creativecommons.org/tag/freesound/.
3 As stated by Xiph.org: https://xiph.org/vorbis/.

Bibliography

Francis, Geoffrey. "Up and Running: A REAPER User Guide." REAPER, May 2020. https://dlz.reaper.fm/userguide/ReaperUserGuide611.pdf.

"What's the Real Difference between.WAV, .AIFF, .MP3, and .M4A?" iZotope, May 10, 2021. https://www.izotope.com/en/learn/whats-the-difference-between-file-formats.html.

Thompson, Daniel M. *Understanding Audio: Getting the Most Out of Your Project or Professional Recording Studio*. 2nd ed. Boston, MA: Berklee Press, 2018.

Chapter 6

DAW Essentials, Part II

An Overview of Plug-ins

Plug-ins and Signal Processing

One of the more puzzling aspects of DAW editing is knowing where and when to use **plug-ins**, a family of **digital signal processors** (**DSPs**), colloquially known as "effects" or "FX." Broadly categorized, these include **Dynamics control**, **Equalization** (**EQ**), **Reverb** and **Delay** (**Echo**), **Modulation** (changing the signal over time), **Pitch Control**, and **Analysis** tools.

Plug-ins can be applied to any REAPER track, including the Master Output. Multiple plug-ins can be stacked up on a track, meaning the signal travels through the first plug-in, then the next, and so on. This is known as an **effects chain**, discussed in more detail later in this chapter. Upon rendering, the final output will be processed just the way you hear it in the DAW.

So many mixes start out with great promise, but ultimately fall short due to the overuse (or underuse) of plug-ins. This is largely due to a lack of understanding of how they actually work; in fact, we will see that the underlying concepts of many plug-ins can be traced back to the fundamentals that are covered in Chapters 1 and 2.

Why Use Plug-ins?

Plug-ins are used to enhance, process, or restore audio in some way. Most of the time we are seeking to accentuate or reduce certain frequencies, give our audio a volume boost, or apply some special effects to a sound. In any case, great care should be given whenever adding plug-ins to our project. Your souped-up audio may sound great in headphones or speakers on its own, but once in the game, it may have a different sonic quality to it.

Plug-ins are also used in real time through game and audio engines, which may have their own proprietary effects packages. When used live during gameplay, they can keep the total mix from getting too loud, enhance or reduce frequencies on-the-fly, and create space with reverbs and delays, among other processes.

Where to Find Plug-ins in REAPER

Adding plug-ins is easy in REAPER! From any track, look in the Track Control Panel for the **FX** button. It's to the right of the Track name and Volume knob (Figure 6.1). Click on it, and you will see a whole host of magnificent plug-ins and plug-in categories (Figure 6.3).

DOI: 10.4324/9781003218821-9

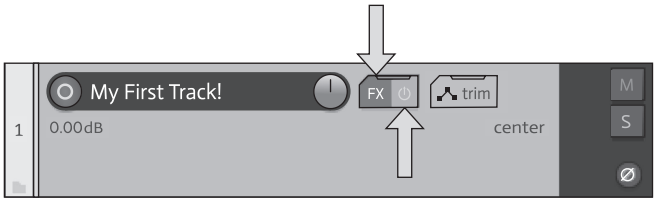


Figure 6.1 The Track Control Panel (TCP). Clicking on the FX button window opens up a plethora of REAPER plug-ins. FX can be enabled or disabled via the button to its right.

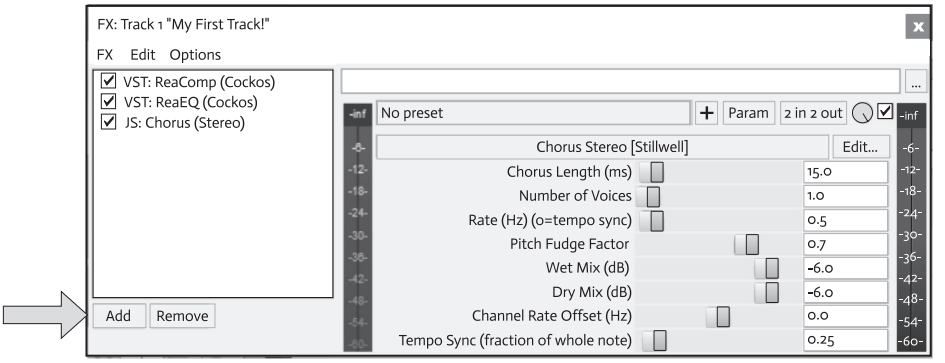


Figure 6.2 The FX Popup Window. Clicking on Add displays all available plug-ins (upper left of image).

If you click on a plug-in and then click Add, it will be sent into the FX popup window, and into your FX list for that Track (Figure 6.2).

Once you have a plug-in on a track, the FX button will light up green. Plug-ins on a track can also be disabled by unchecking them in the FX popup window (Figure 6.2), or toggling the enable/disable button to the right of the FX button (Figure 6.1).

Effects Chain

This list of effects is known as the **effects chain** for that track. You can add as many plug-ins as you like (be careful!) and change the order your audio travels through them (starting from the top downward), simply by left clicking and dragging a plug-in to the desired position.

There are too many REAPER plug-ins to cover in one chapter, never mind an entire book. As of version 6.3, there are about 250 plug-ins to choose from!

Once we learn about the capabilities of plug-ins, we can then try to adopt a "Zen" approach to using them, where less is more. My philosophy is this: *use the least amount of processing that gets the job done*. In a complex project with multiple tracks and plug-ins, it can be hard to track down problems with an overuse of EQ, compression, and the like.

For each category of game audio, individual files can be rendered with plug-ins used on tracks in REAPER; but depending on the game engine (and/or audio engine) used,

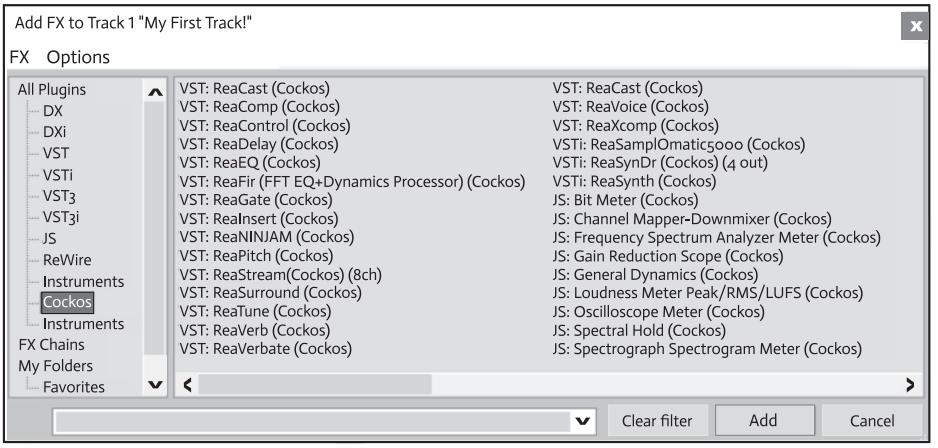


Figure 6.3 The Add FX to Track Window.

we can also add plug-ins *within the engine itself* which are applied to the real-time mix, for additional control over the soundscape during live gameplay.

Common Plug-in Controls

Wet and Dry Mixes

Among the vast array of plug-ins available, there are a handful of common features that many share. Two of them are **wet** and **dry** mix controls. The wet mix is the output volume of the processed signal, and the dry mix is the output volume of the original (input) signal.

Plug-in (FX) Chains

As seen in Figure 6.2, plug-ins can be stacked together by using the Add and Remove buttons; a check box can be toggled on/off to activate/deactivate a specific plug-in in the chain and left-clicked and dragged up/down to change its order.

Presets

Just above the center of the plug-in in Figure 6.2 is the preset menu. Opening this will show you all the saved presets in a dropdown menu, and you can select any one you want from there.

Save/Load

To the right of the preset bar is a small box with a "+" symbol. Clicking on this opens up a menu to save or delete a preset.

"Sidechain" EQ

Many plug-ins will also contain a **sidechain EQ**, offered as a High-Pass and Low-Pass filter. Setting these filters will determine which area of the frequency spectrum utilizes the plug-in.

Dynamics

Some of the most important plug-ins we have in our toolbox are dynamics controllers, which deal with volume adjustment. At their most basic, they can perform a simple, linear volume boost or cut, or on the more sophisticated side, they can make moment-by-moment adjustments, keeping a lid on a signal's **dynamic range**: the range between the softest and loudest possible volume.

Dynamics Control for Game Audio Production

Applying the proper dynamics to our game audio, we can make sure the various categories – music, sound, and dialogue – can each be heard well during gameplay. Sometimes controlling the minimum and maximum volume levels for each category, or even individual sub-categories of audio, can ensure a more controlled mix.

When you're playing your favorite game, you might be hearing growls, guns, lasers, dialogue, user interface sounds, and a dramatic orchestral soundtrack with lots of highs and lows in volume. Great care had to be taken to ensure you could hear everything well, and it all started with harnessing the dynamics. **Normalizing**, **compression**, **limiting**, and **gating** are all relevant dynamics plug-ins, described below. For use during gameplay, a compressor can be added to the master track to prevent the mix from clipping.

Normalizing

Peak Normalization

Sometimes all we need is a simple volume boost. **Peak normalization** raises the volume of an audio file to a fixed amount, usually the maximum allowed (0 dB). So, for example, if a REAPER media item (an audio file) had its highest peak at –6 dB, processing it with a peak normalization of 0 dB would bring everything up internally by +6 dB, as seen in Figure 6.4. Remember, *we can't get louder than 0 dB*; all of our binary digits are maxed out with 1's at the loudest possible volume, so we have literally hit the digital ceiling. In REAPER, we can select a media item, right-click and from the popup menu select **Item Processing ➔ Normalize**. This will find the highest peak and raise it up to 0 dB, bringing everything else up with it linearly. If you select two or more items and choose **Item Processing ➔ Normalize (Common Gain)**, the highest value between ALL selections will be brought up to 0 dB, bringing everything else up with it linearly across all media items.

RMS and LUFS Normalization

There are other times that we need to "rein in" the dynamics of an audio file. Perhaps some creature dialogue has shouts that are too loud, and they utter some important

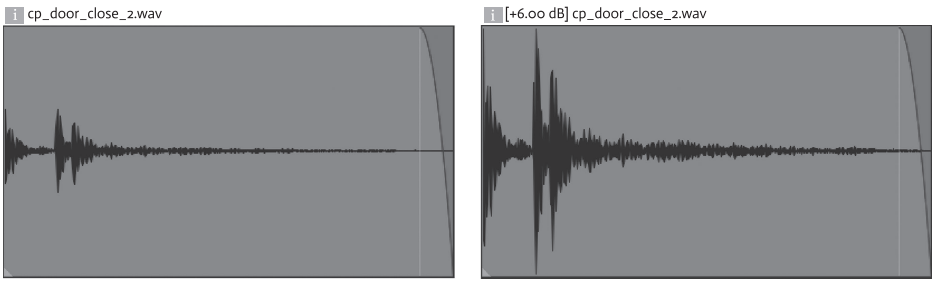


Figure 6.4 Peak normalizing an audio file from −6 dB to 0 dB.

passages that are too soft; we'd like to bring up some of the quieter speech but tamp down those shouts. You could apply some volume envelopes as a quick, practical solution. But what if you had 10,000 lines of dialogue to handle in this way? If you don't want to get repetitive stress injury and lose your sanity, some **loudness normalization** may be just the trick. Up until recently, the most common type of loudness normalization was **RMS-based**. This is a sort of "contoured" normalizing where we wish to maintain a consistent, perceived loudness level. This technique looks at peak loudness values in a digital audio file, adds them up and divides them to yield an average level.

A more recent loudness format that has largely replaced RMS is the **LUFS** format, which stands for **L**oudness **U**nits relative to digital **F**ull **S**cale (full sale being 0 LUFS = 0 dB = the loudest possible digital audio). Recall that RMS determines average loudness over time, as a function of electrical power. When using LUFS, it is still average loudness over time, but using a more sophisticated process that is based on human hearing perception. 1 dB = 1 loudness unit (LU), and we exclusively use the term "LU" to describe units of loudness.

Roughly speaking, game audio soundscapes may fall between –16 and –24 LUFS, but a standard has been recommended for the industry, according to the Sony Worldwide Studios Audio Standards Working Group (ASWG):

1 The loudness standard for home-based entertainment (consoles and home computers) should be **–23 LUFS**, with a +/–2 LU tolerance; and
2 **–18 LUFS** with a +/–2 LU tolerance for mobile entertainment.

They also mention that the audio should be measured for as long as possible to get an optimum average level. Also, the measurement should not be limited to any particular category of audio (music, sound, or dialogue), but rather the mix should be taken as a whole.[1]

Normalizing Options

You can normalize via several options, by right-clicking on the audio clip and selecting Item Processing ➔ Normalize Items (peak/RMS/LUFS); from there, enter your normalization type, with one of the following selections:

LUFS-I: Integrated; the entire file is considered in calculating LUFS. Anything under –70 LUFS is ignored in the evaluation, as well as certain quick, transient bursts of audio in the mix (a weapon impact, loud computer beep, etc.).

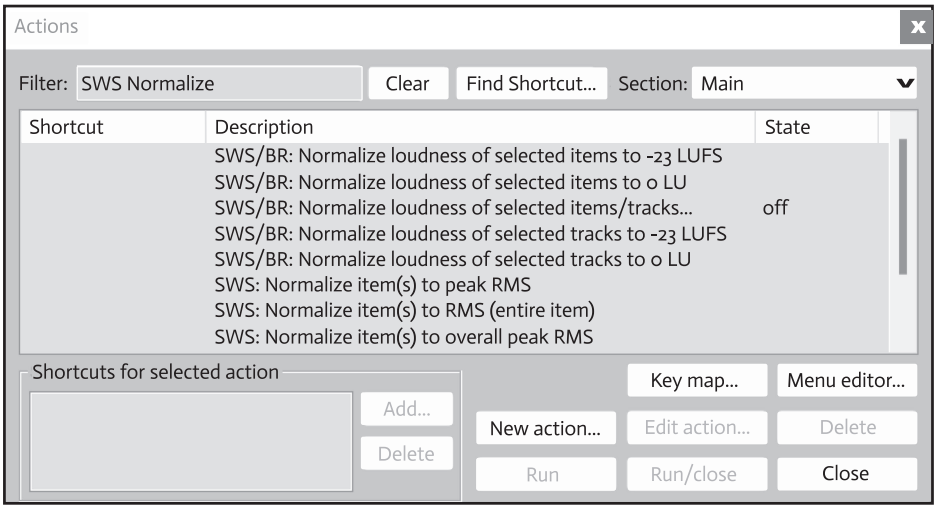


Figure 6.5 SWS Normalize Actions.

LUFS-M max: Momentary, using 400-ms time windows of audio to calculate loudness.

LUFS-S max: Short-term, using 3-second time windows of audio to calculate loudness.

RMS-I: Integrated; the entire file is considered in calculating RMS loudness.

Peak: Normalizes using the peak amplitude over the entire file.

True Peak: Normalizes using the peak amplitude over the entire file, sending back the peak values to you *as they will exist once the digital audio is rebuilt to analog.* This is more accurate, and the better choice than "traditional" peak.

There are also several SWS Normalize Actions. Simply select **Actions**, search on "SWS Normalize" and select the desired Action, as seen in Figure 6.5.

When we listen to an individual track, it might be sitting around, say, –16 LUFS. But as you add more tracks to your project, even if every track is –16 LUFS, the combined output rises – you are generally summing waveforms together via constructive interference – making the mix louder overall. For this reason, it's a good idea to (1) later, record a video capture of your game, (2) import the audio into REAPER, and (3) place a loudness meter on the Master Output to check your levels. A good plug-in to start is with REAPER's **JS: Loudness Meter** plug-in. Click on the Master Output **FX** button and add the plug-in there (Figure 6.6).

Normalizing on Render

Lastly, REAPER has a Normalize function in the **Render** window. There is much debate about whether or not you should normalize your project "sound unheard," but this is entirely up to you. You can do a **Dry Run** which yields important data about the rendered file, a sort of audition for the real thing.

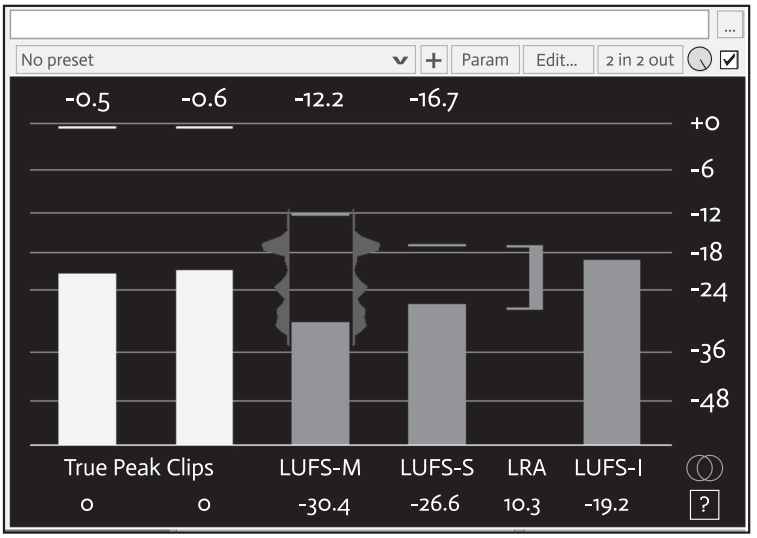


Figure 6.6 The JS: Loudness Meter plug-in, showing True Peak and LUFS values.

Upon selecting the **Normalize...** button, the **Normalize Render** window will appear; you can select from the same options as with Item Processing ➔ Normalize, as well as set your dB loudness target (Figure 6.7).

Compression

When we speak of **compression** in audio, it can mean several things: the opposite of rarefaction; a compressed file format (like MP3); or in this case, a **compressor** is a plug-in that reduces the dynamic range of audio.

Compressors can be quite complex, having many tweaky bits like "sidechain EQ" and "knee size" but these are details that can be found in the REAPER guide and experimented with.

Here is all you need to know if you're just starting out:

The two main parameters in a compressor are the **threshold** and the **ratio**. Let's say you have some very dynamic dialogue that needs some moderate compression. The threshold is the magic line in the sand that, if crossed, alerts the compressor to turn on and do its thing. It is measured in decibels, from minus infinity dB (total silence) up to 0 dB. As the audio plays, the compressor waits and waits some more until ... *bam*! The audio passes the threshold. Then, the compressor turns on. It will push the offending audio back down [x] dB, according to the **ratio**, a measurement of input versus output. So, a 1:1 ratio will do nothing, as the input is equal to the output. But a 2:1 ratio will compress: for every 2 dB over the threshold, only 1 dB is allowed to pass through. A 10:1 ratio means that for every 10 dB over the threshold, only 1 dB is allowed to pass through. You can see that as the ratio increases, the dynamic range becomes heavily compressed.

With all this squashing of dynamic range, the overall signal volume will decrease significantly and become harder to hear. But fear not! A handy parameter in many

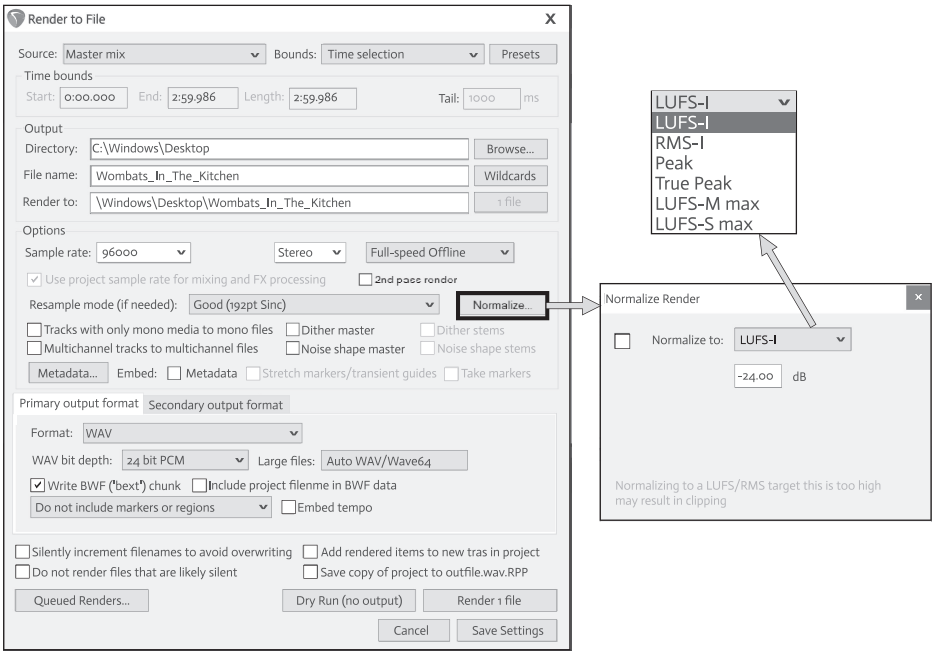


Figure 6.7 The Render and Normalize Render Windows.

compressors called **make-up gain** provides a quick volume boost to the affected audio, bringing it back up to an acceptable listening level. Take a look at **ReaComp** in Figure 6.8; there is a lot to take in here! Of interest are the threshold and ratio but also the **Auto Make-Up** check box, which boosts the overall volume to an acceptable output level. The **Limit Output** checkbox in the very bottom ensures the output will not exceed 0 dB.

There are many external plug-ins that offer compressor types such as *Tube*, *Optical*, *Field-Effect Transistor (FET)*, as well as blends of these models. Each has unique characteristics that give audio different dynamic qualities, and it is worth experimenting with these models if they are available to you. A great place to start is with free plug-ins, at sites such as https://plugins4free.com, or https://www.audiopluginsforfree.com. Search on "Tube," "Optical" (or "Opto"), and/or "FET," and see what you find!

Limiting

A **limiter** is really just a compressor that uses a 10:1 ratio or greater, severely compressing any audio that meets the threshold. In effect, it creates a sort of "brick wall" against loudness. Most of the time, limiters are the go-to when we don't want the master level in a mix to exceed a certain volume level.

There are simple limiters that let you set a threshold and be done with it; most limiters offer an additional volume boost (check out the **JS: Soft Clipper/Limiter** plug-in) and others that automatically boost the volume as you lower the threshold (as with **JS: Master Limiter**). Most offer some sort of "look ahead," where the audio is analyzed

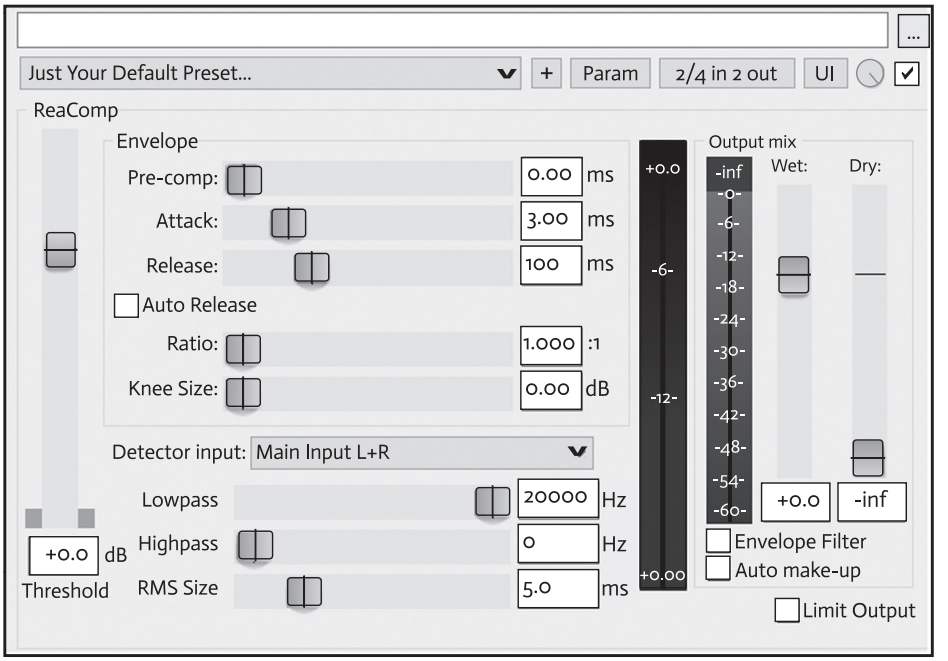


Figure 6.8 The ReaComp Plug-in. In this example, the compressor is set to a −3.5 dB threshold; audio that exceeds this level will be compressed at a ratio of 5:1. Make-up gain is automatically applied when the "Auto make-up" box is checked, near the bottom right corner.

at least a few milliseconds ahead, allowing the limiter to compensate for loudness control without going beyond the threshold accidentally.

Gating

Imagine you just recorded a great character voice artist. Everything sounded fantastic in the studio, but when you listened to it the next day you noticed lots of low-level noise in the background. Someone had a fan on the whole time! Fortunately, you could only hear it when the character *wasn't* talking. Can the audio be saved? Probably yes! The best way to fix this in REAPER is by adding a **noise gate**, aptly titled **ReaGate**.

The threshold is set to a decibel level you determine, and then when audio reaches that threshold or below, it disappears – it is completely muted. **Attack** (fade in) and **Release** (fade out) envelope times can be adjusted in milliseconds, creating smoother or more abrupt dynamic transitions for when the gate opens and closes, respectively. The **Hold** parameter keeps the gate open for *n* milliseconds after the signal falls below the threshold (Figure 6.9).

Distortion

Sometimes we want to purposefully grunge up our audio. A glitchy computer interface? Some crunchy guitar riffs? Distortion can help with that! As the names imply, it

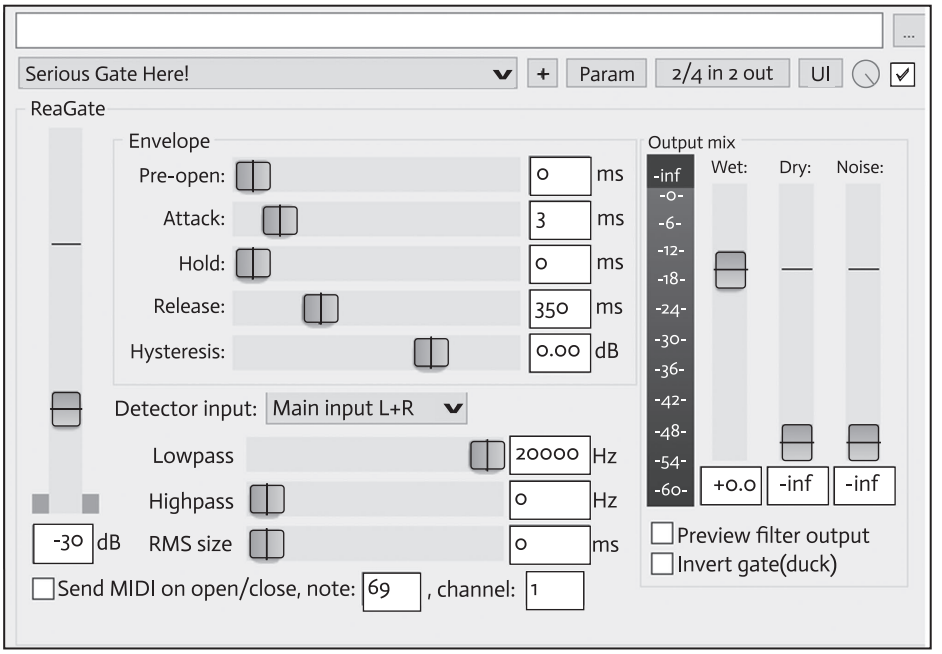


Figure 6.9 The ReaGate plug-in, set to a threshold of −30 dB.

makes the signal sound distorted, gritty, grungy, or clipped, whichever adjective you choose to describe it. In any event, your dynamic range and signal clarity can both be ground into a pulp here. Give it a go with REAPER's **JS: Distortion** plug-in!

Equalization

When volume is adjusted, we are bringing the entire frequency spectrum up or down. But often times, audio will only need specific frequencies adjusted, rather than sweeping volume changes across-the-board. **Equalization**, or **EQ**, is another crucial plug-in category that deals with changing the frequencies of audio. With EQ we are usually doing at least one of three things to our audio: repairing it, enhancing it, or separating it from other audio content in a mix.

Equalization for Game Audio Production

Everything needs to sit right in a soundscape, not just by making changes to dynamics but by also raising or lowering specific frequencies within our audio categories. For example, music may need to "thump" more in the lower frequency range, or require more "brightness" in the higher frequency range. Sometimes dialogue needs an audibility boost, and bringing up the midrange frequencies may help with this. An arrow's impact into wood may sound too "muddy" so we might bring down the lower midrange frequencies to make it sound a bit clearer.

EQ can be applied to audio files through a plug-in, or during gameplay; in either case, a good strategy is to create pockets of frequency ranges for different channels.

Table 6.1 Frequency Ranges and Their Descriptions

Hz	*Range*	*Description*
20–60	Sub-bass	Very low frequencies that are less tonal and more visceral (feels "rumbly"). Low-frequency energy (LFE) that is good for kick drums, low percussion, bass synths, explosions, giant footsteps, and weapon impacts.
60–250	Bass	As the name suggests, instruments with low frequencies benefit here (taikos, bass, low brass or woodwinds) as well as many weapon impact sounds, and lower-register dialogue. LFE continues in this range to about 150–200 Hz. Often described as "thumpy" or "punchy."
250–600	Lower midrange	This range is also jokingly known as "mud," as lots of elements in a mix pile on thick here. Drums, voices, footsteps, etc., all contribute to this range and may need some attenuation to avoid sounding too "boxy."
600–2k	Midrange	A critical mixing area: speech audibility begins here, but also some buildup from musical instruments and sound effects (the midrange is a very commonly shared area among music, sound, and dialogue). Too much midrange and your mix may have too much "honk" to it.
2k–4k	Upper midrange	Speech audibility is at its peak here. Caution should be used in boosting too much, as (1) the listener's ears will quickly become fatigued if this range saturates the mix, and (2) you can also accidentally cause lisping to dialogue.
4k–6k	Presence	Speech audibility continues here, and a slight boost for dialogue can really help it cut through the mix. Boosting these frequencies can make the mix feel closer to the listener; reducing too much and it will seem dark, or distant.
6k–20k	High range	An area of brightness, clarity, or "air" for instruments and upper-range sound effects, like magical spells (especially for sounds with crystalline or shimmery qualities). Great for user interface (UI) sounds as well: any higher-frequency clicks or beeps that need to be heard clearly in the mix can benefit from a boost here. Too much can add more hiss or noise to the mix.

For example, UI sounds may have their low frequencies removed completely with an EQ plug-in, while explosions may have their highs reduced to emphasize the lower and middle ranges. Often these frequency increases and decreases need not be so drastic. But by separating channels out this way, even by just a bit, we can suddenly increase the clarity of a soundscape. Multiple sounds may no longer be overlapping on certain frequencies as much, which again helps with audibility and may even decrease listener fatigue. There are no clear-cut rules on where frequency ranges begin and end, but Table 6.1 finds some common ground among many recommendations.

Common EQ Filter Types

An EQ **filter** defines an area on the frequency graph where frequencies can be boosted or cut in volume, or removed completely. Often these filters take the shape of adjustable **bands** used to change frequency volumes.

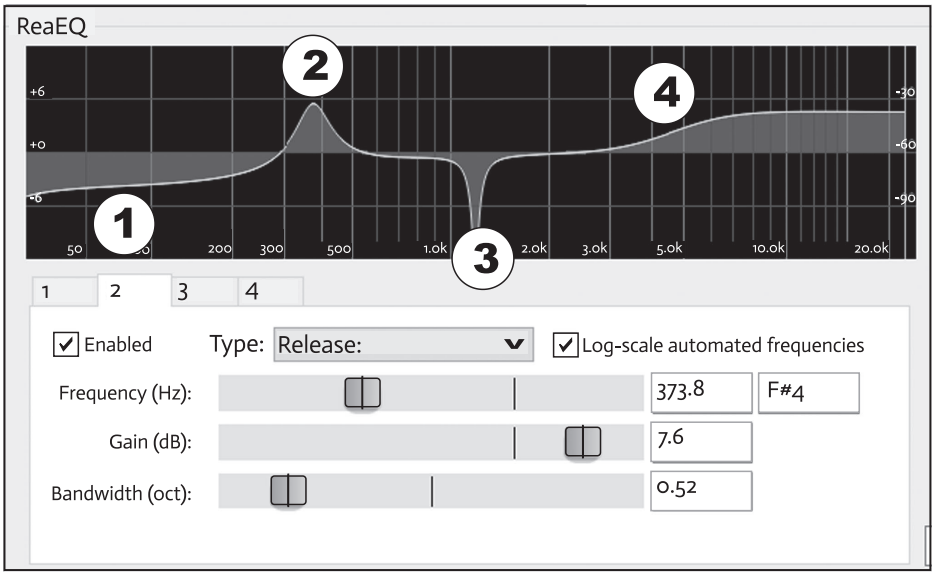


Figure 6.10 Various filter types in the ReaEQ plug-in.

Figure 6.10 shows the following filter types:

1 **Low shelf**: Boosts or cuts the gain (volume) of all frequencies at or below the cutoff point.

2 **Band:** A bell-shaped curve that boosts or cuts the gain of frequencies, with its peak at the center point.

3 **Notch (band reject)**: An inverted bell-shaped band that removes frequencies entirely, with its peak at the center point.

4 **High shelf**: Boosts or cuts the gain of all frequencies at or above the cutoff point.

Then, in Figure 6.11, we have Low- and High-Pass filters:

1 **High-Pass filter**: Removes all frequencies at or below the cutoff point, allowing higher frequencies to pass through.

2 **Low-Pass filter**: Removes all frequencies at or above the cutoff point, allowing lower frequencies to pass through.

The Pass filters are different from Shelf filters, in that their gain cannot increase or decrease; they are only used for masking and preserving frequencies.

Note that in both Figures 6.10 and 6.11, there are three parameters below the graph.

Frequency: Selects the center frequency for the band or filter.

Gain: Boosts or cuts the volume; not used for all filter types, such as with a Notch or Pass filter.

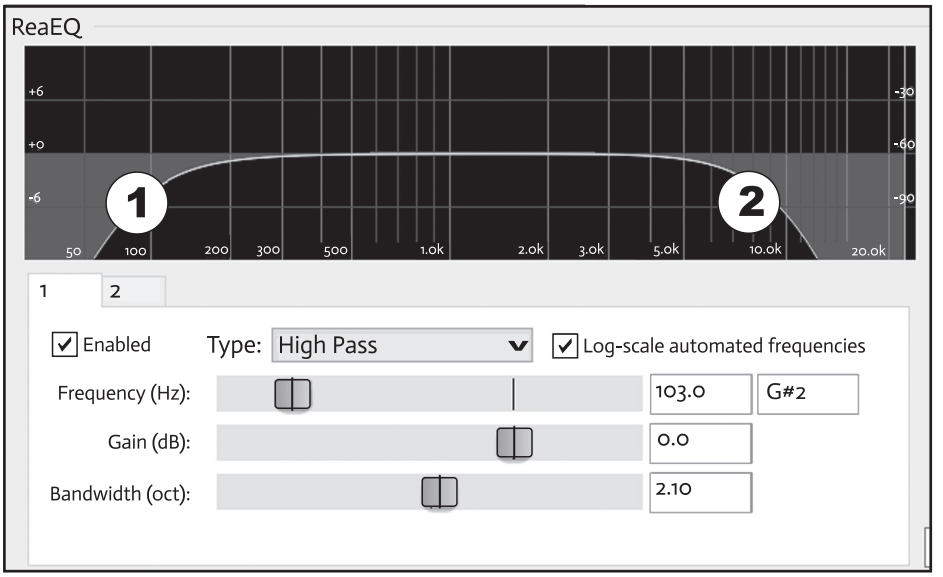


Figure 6.11 Low- and High-Pass filters in the ReaEQ plug-in.

Bandwidth: Narrows or widens a band; steepens or flattens a filter's slope, such as with a Shelf or Pass filter.

Consider the EQ graphs in Figure 6.12. There's nothing inherently wrong with having a high number of bands; in fact, REAPER has some *11-band* presets that can be quite effective for instruments like drums. And sure, they look really cool to show off to your friends, or on the cover page of an article you're writing about EQ. But my experience has been that if you're needing 5+ bands for *everything*, either your source material was badly recorded, or you aren't making good use of your time.

Will any gamer notice a sound effect that benefited from 14-band EQ versus 3-band EQ? Game audio production can be fast and furious sometimes, and efficiency is key. You can't be saddled with multiple bands on every EQ, across 20 tracks of audio; you'll never finish the project! It takes time and practice, but you will gradually discover ways to streamline your work with EQ, and plug-ins in general. That said, don't be afraid to experiment whenever you get the chance.

Help! Where's That Frequency?

If you feel that the EQ of your audio is "off" somehow, here's a tried-and-true method to find out what's going on: create a narrow band filter on an otherwise flat EQ graph as in Figure 6.12b, and make the amplitude (gain) reasonably high. Then, sweep the band left-to-right, and then right-to-left. Doing this slowly a few times should reveal a frequency range that jumps right out at you; that's likely the offending range to reduce. Or, you could use this technique to find an area that desperately needs boosting as well.

Another trick is to employ a High- or Low-Pass filter to get rid of a whole swath of unwanted frequencies. This is great for things like preserving the sound of recorded ambience – you can keep those gentle lake ripples and bird tweets, but roll off the low

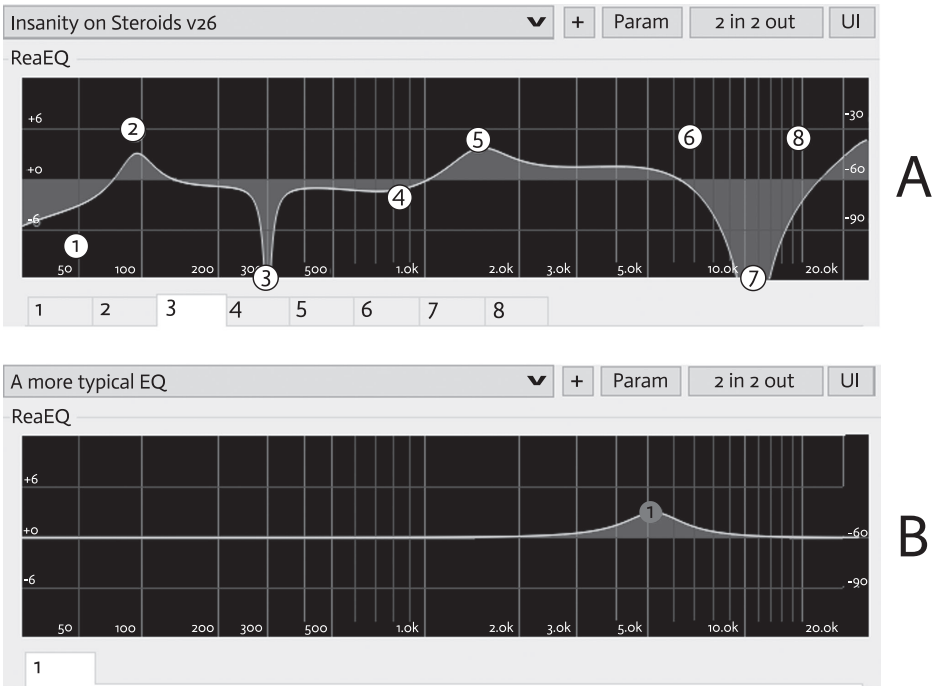


Figure 6.12 (a) A very flashy, attractive-looking EQ, and (b) a more plausible EQ in a real-world situation – not ostentatious, just getting the job done.

frequencies entirely (using a High-Pass filter), and that truck rumbling by in the distance is removed. Or, keep your lawn mower engine recording intact, but remove the sound of the high-pitched alarm going off nearby (using a Low-Pass filter). Sometimes we also want a sound to add as LFE to enhance or "sweeten" a different one; using the Low-Pass filter can make that happen.

You can also use Low- and High-Shelf filters to achieve the same results as the Pass filters (Low Shelf = High Pass, and vice versa), but remember that the Shelf also controls the gain of the chosen frequencies.

Reverb

As we saw in Chapter 1, **reverb** (reverberation) is the pile-up of reflections that we hear bouncing off multiple surfaces. If we play a snare drum in a gymnasium, we will hear the initial "dry" hit of the drum first, followed by many ping-ponging reflections of the hit, bouncing off the gym walls, floor, and ceiling, all eventually decaying to zero energy. This decay time is also known as the **reverb tail**.

Reverb for Game Audio Production

Need the sound of water drops in an icy cave? Footsteps in a tunnel, or roars from the interior of a shipping container? Reverb can simulate the dimensions and reflective characteristics of many environments. It can even be used to great effect (no pun intended)

to enhance the dialogue of monstrous titans or otherworldly beings, work magic on an acoustic guitar, or provide a sense-of-place to any sound heard by the listener.

Like we saw with Dynamics and EQ, Reverb can be "baked in" (rendered to) an audio file via plug-ins. It can also be simulated within game and audio engines in real time, which we will see in more detail in Chapter 12.

Types of Reverb

The characteristics of various spaces can be modeled using a reverb plug-in. The main preset categories are:

Room: Simulating a small room or studio, with close reflections and short decay times.

Hall: Larger spaces such as a concert hall, which have more distant reflections and longer decay times.

Chamber: These presets are modeled after reverb chambers, spaces in recording studios with high-reflective properties. Sound is piped into a chamber or stairwell, and the effect is picked up on another mic in the room to be sent back to the control room.

Plate: Simulating the real-world technique of sending audio onto a vibrating metal sheet, or plate. Audio is projected onto the plate, which vibrates and creates reverb, and picked up by another mic to be sent back to the control room. A bigger plate means a longer reverb decay time.

Spring: A similar process to chamber and plate, but with an array of springs. Longer springs mean longer decay times.

Convolution reverb: This is used for simulating customized spaces, such as a cave, a forest, a guitar amplifier cabinet, or a specific concert hall. Convolution reverb has no built-in presets, but instead relies on **impulse responses** (**IR**), which are recorded samples of the acoustic properties of various spaces. Using some mathematical techniques, the convolution reverb plug-in can take your input signal and process it through an IR, to give you the reverb of that particular space. Neat!

Convolution reverb is made in one of two ways: either by recording a 20–20 kHz sine wave sweep in the space you desire, or a quick, plosive sound that generates a wide swath of spectral content all at once: a take clapboard (and, *action*!) or a balloon popping will do the trick. Next, that audio information needs to be imported into a special IR plug-in. In the case of the sine wave sweep method, specific software may be needed to convert the audio to an IR. The end result is usually a WAV file that starts with the 20–20 kHz frequency burst, and the corresponding reverb that now covers the frequency spectrum.

The powerful **ReaVerb** plug-in can be used as a traditional reverb plug-in, or as convolution reverb. The latter has some built-in synthetic IRs, and a File option to import your audio to become an IR. I highly recommend checking out the REAPER Effects Guide for a detailed look on how to use this plug-in, which can be found at https://user.cockos.com/~glazfolk/ReaEffectsGuide.pdf.

There are lots of convolution reverb plug-ins and impulse responses out in the wild, but some IRs are in proprietary formats (specific to certain audio software or bundled exclusively to work with specific plug-ins), and others require conversion from one format to another. Typically, though, most IRs are stored as WAV files.

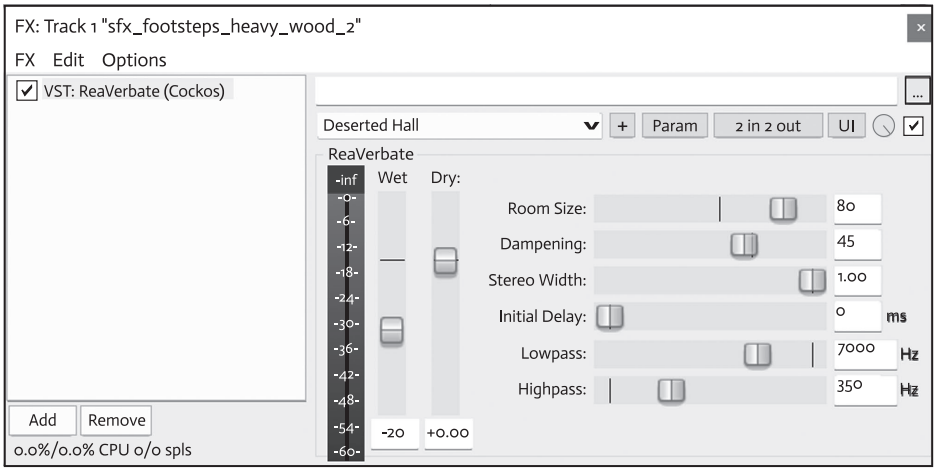


Figure 6.13 The ReaVerbate plug-in, with its easy-to-use controls.

Some free convolution plug-ins that may also include several IRs can be found at https://bedroomproducersblog.com/2019/03/18/free-convolution-reverb-vst/. If you're on the hunt for some free IRs, a good place to start is www.voxengo.com/impulses.

REAPER's ReaVerbate plug-in is a good place to get started for a basic room- or hall-style reverb, with some easy-to-use parameters:

Room size: The higher the number, the larger the room. Things really get noticeable around 70.

Dampening: Imagine acoustic panels in this room that dampen, or suppress, the amount of reflections ping-ponging around. It is a reduction of the **late reflections**, aka the **reverb tail**.

Stereo width: 1.0 is normal stereo, 0.0 is summed to mono (left and right channel combined into the center of the stereo image), –1.0 is reverse stereo.

Initial delay: How long the first reflection takes to return to the listener.

Lowpass and Highpass: Adjusts which lower/higher frequencies are allowed to be processed. Experiment with these to hear some interesting EQ! It's often used to shut out muddy (low) reverb frequencies to provide more clarity in processed dialogue or music stems, for example (Figure 6.13).

A word of caution with reverb: A little goes a long way. It may sound cool to cake it on while mixing, but unless the intent is to sound over-processed, placing it into your game with other elements will feel out of context. Figure 6.14 illustrates this, applying about 80% reverb with 50% dampening to a door-closing sound effect.

Delay

Imagine shouting "SOUNDSCAPE!" into a canyon. You'd expect to hear it come back at you as an echo, over and over again, until it dies out. That's a type of delay! A delay is an effect where the original signal is repeated within an adjustable interval of time, decreasing in volume over an adjustable period. It can be a long echo or an extremely short interval (10 ms, for example).

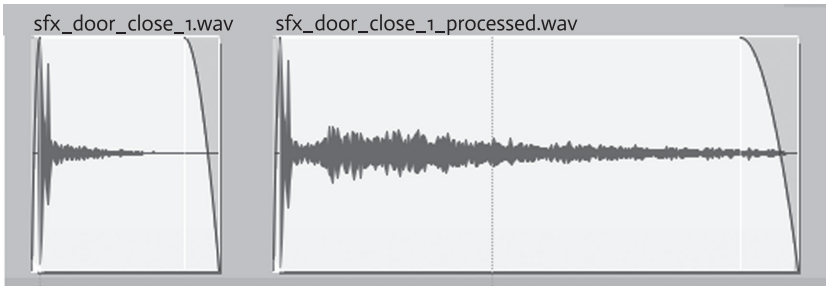


Figure 6.14 An example of a sound effect with and without reverb (using ReaVerbate). Note the long reverb tail that appears, in this case due to a very large room size value.

Delay for Game Audio Production

Delays can work wonders for environmental effects such as large outdoor expanses, but it works for supernatural and science fiction sounds and dialogue as well. It can also be added to song stems on melodies, chords, and rhythm parts, synced to the song's beats per minute or a more specific timing (eighth note, quarter note, etc.). Just take it easy on the wet mix levels.

There are a few types of common delay effects.

Flanging: A delay time less than 15 ms (described in more detail in the Modulation section, below).

Doubling: A delay time between about 15 and 35 ms.

Echo: 50 ms and up, but usually in the 250–1,000 ms range.

Typical delay parameters include Length (the delay interval, in ms) and Feedback (the decay time, in ms). Some delays also include High- and Low-Pass filters, and the inclusion of multiple delays in a single plug-in. REAPER's ReaDelay has all of these, and you should experiment with it to hear its full potential.

Modulation

This family of plug-ins covers a wide variety of signal processes, where the source material is altered in several ways. Some modulation effects are purely ultra-short delays, or may use special filtering to enhance the source with interesting spatial textures. Other effects use a **low-frequency oscillator (LFO),** normally a sine wave, over the source material. The LFO in this case is a **carrier** signal that either controls the source's volume or pitch, or acts as an interference pattern that creates some dramatic results.

Modulation for Game Audio Production

Ah, this is getting good! The use of modulation plug-ins for game audio is well-suited for sci-fi, fantasy, and horror genres, especially when it comes to dialogue processing. Droids, aliens, and all your favorite creatures can benefit from using these plug-ins. Sound design can also benefit from modulation when simulating machines, computers, weather, vehicles, weapons, and more. Modulation can imbue and transform the

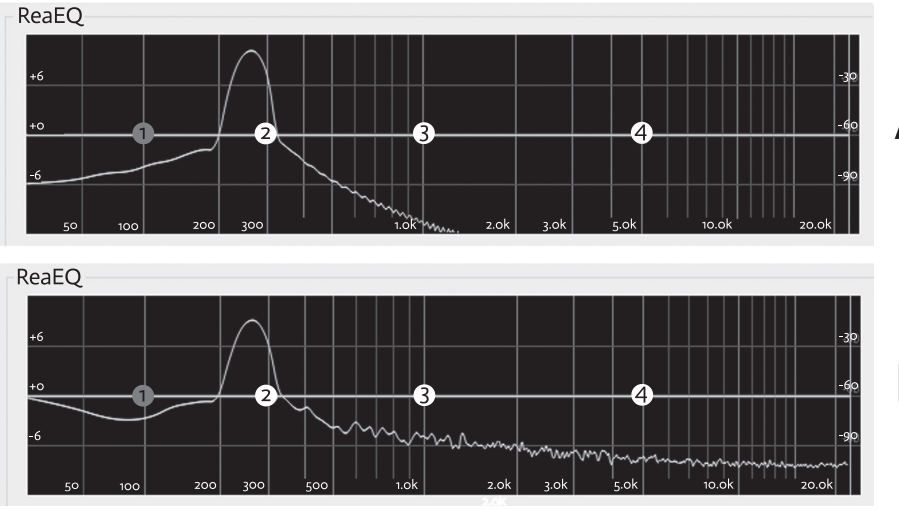


Figure 6.15 (A) A 261.63 Hz (Middle C) sine wave dry, and (B) with Chorus effect. A copy of the source in the wet mix, altered slightly in pitch and with a short delay, causes the resultant signal to have shimmery, vibrant overtones.

source material by creating rich sonic textures, motion, and depth, and can be used either as a stand-alone plug-in or combined with other effects. There are some modulation plug-ins available for live use via game or audio engines, but more variety will be found in REAPER's plug-in library (Figure 6.15).

Chorus

One or more copies of the source is made, then placed into the wet mix with slight variations in pitch. The pitch shifting is achieved using a carrier (an LFO, at a **rate** usually between 0.1 and 5 Hz), and by shortening or lengthening the delay time between wet and dry mixes (the **chorus length**, or **delay**, in ms). These slightly detuned copies create ever-changing phase interference leading to a lush, shimmery sound. The chorus delay time is between 15 and 35 ms; our brain cannot perceive this short delay time but instead hears one composite signal.

Flanger

This also has a shimmery quality to it, but for different reasons than a chorus. It is considered a standard delay effect as well as modulation, with a delay time of less than 15 ms. This short interval causes phase interference that produces a **comb filter** in the mix, producing its own swirling, interesting effect (if you look at the wet mix on a frequency graph, the waveform looks much like the teeth of a comb, hence the name).

Phaser

Instead of using delayed copies of the source, a phaser uses a special **All-Pass filter** that allows all frequencies to pass through it, but then alters the *phase relationship*

between the source and the copies. The phase shifts create frequency notches that travel in an up-and-down manner across the spectrum, creating a sweeping effect. It can be adjusted by changing the LFO **rate** and **depth**.

Amplitude Modulation

The volume (amplitude) of the source is controlled by an LFO, quickly rising and falling repeatedly. This is known as a **tremolo** effect.

Frequency Modulator

The pitch (frequency) of the source is controlled by an LFO, giving the sound a wavering quality. This is known as a **vibrato** effect.

Ring Modulator

The input and carrier (the latter being an LFO) are multiplied together, but we only hear the sums and differences of the new frequencies that are created. The carrier frequency is removed upon final output, yielding some interesting effects.

Mixing with Plug-ins

Ask a dozen audio designers what the proper order should be of your plug-in chains and you'll get a dozen answers. But here's the order I recommend, and one that has worked quite well for me over the years.

1 *Dynamics – Balancing Volume*
   If you can get a mix sounding great with just volume changes, you're a rock star! This may involve simply moving fader values into place, or some fancy volume envelope changes per track. It could even involve adding compression where needed, though there is great debate over whether EQ should come before compression; it really depends on the individual content. Just be aware that if you EQ first, any frequencies you boost may be squashed back down by the compressor.
2 *EQ – Balancing Individual Frequencies*
   Now we really need to start listening: muting and soloing tracks for individual sonic characteristics, referencing the entire mix, and making EQ changes, sparingly at first. When mixing, there is a tendency when to start EQ-ing everything simply because we can. If you do need to EQ, first start with only what absolutely needs adjusting. Also, we are predisposed to boosting frequencies when we should really favor reducing them. Using this type of subtractive EQ method keeps the mix from becoming too loud overall.
3 *Spatial Processing – Adding Depth and Imaging*
   Once dynamics and EQ are under control, special effects and spatial imaging can be added. Imagine how difficult it would be to EQ a mix that already had heavy reverb on it! Panning, reverb, delay, and modulation can all be applied in this final stage.

Using Volume Envelope with Effects

The volume envelope on a track not only changes its output level but also changes any effect levels on that track. However, adjusting the volume on the **media item** itself does not change the effect levels. You can think of the red volume line as **gain** on a mixer, which occurs **pre-effect** and **pre-fader**, and the green volume envelope as **post-effect** and **post-fader**. This is most recognizable using effects with potentially long tails, such as reverb or delay – here, the effects can continue long after the media item has been played.

Plug-ins: Some Final Words

This chapter is about familiarizing you with the main categories of plug-ins available and what they do, but rules are made to be broken. How they are used is entirely up to you (maybe you want to use a pitch control plug-in before EQ-ing, for example), but some specific plug-in strategies can be seen in later chapters. We can tweak digital faders and knobs until the cows come home, but at the end of the day, using your ears is the most important tool of all in making final decisions.

While plug-ins are great at enhancement and restoration, you can't enhance frequencies that don't exist in the mix, nor can you restore dynamic range to audio that has already been squashed into a brick. Do the best job you can at recording or finding existing source material, then see if you can make your sonic creations shine with your plug-ins.

Recommended Links

SWS REAPER Extensions: https://www.sws-extension.org

The REAPER Blog: https://reaperblog.net

REAPERCockosEffectsSummaryGuide:https://user.cockos.com/~glazfolk/ReaEffectsGuide.pdf

Additional information on Choruses, Flangers, and Phasers: https://www.izotope.com/en/learn/understanding-chorus-flangers-and-phasers-in-audio-production.html

How to Use Reverb: https://www.izotope.com/en/learn/reflecting-on-reverb-what-it-is-and-how-to-use-it.html

Note

1 http://gameaudiopodcast.com/ASWG-R001.pdf, and https://ubm-twvideo01.s3.amazonaws.com/o1/vault/gdc2013/slides/824764Garry_Taylor_Loudness_and_how.pdf, and https://www.audiokinetic.com/library/2016.2.6_6153/?source=Help&id=monitoring_signal_level.

Bibliography

Elmosnino, Stephane. *Audio Production Principles: Practical Studio Applications*. New York, NY: Oxford University Press, 2018.

Francis, Geoffrey. "The REAPER Cockos Effects Summary Guide." REAPER, March 2016. https://www.cockos.com/~glazfolk/ReaEffectsGuide.pdf.

Francis, Geoffrey. "Up and Running: A REAPER User Guide." REAPER, May 2020. https://dlz.reaper.fm/userguide/ReaperUserGuide611.pdf.

Huber, David Miles, and Robert E. Runstein. *Modern Recording Techniques*. 9th ed. New York, NY: Routledge, 2018.

Kody, Arthur. "Understanding Chorus, Flangers, and Phasers in Audio Production." iZotope, August 6, 2021. https://www.izotope.com/en/learn/understanding-chorus-flangers-and-phasers-in-audio-production.html.

Owsinski, Bobby. *The Mixing Engineer's Handbook*. Burbank, CA: BOMG Publishing, 2017.

Thompson, Daniel M. *Understanding Audio: Getting the Most Out of Your Project or Professional Recording Studio*. 2nd ed. Boston, MA: Berklee Press, 2018.

"What Is Reverb? The Reverb Resource for Audio Engineers of All Levels." iZotope. iZotope Education Team, June 16, 2020. https://www.izotope.com/en/learn/reflecting-on-reverb-what-it-is-and-how-to-use-it.html.

Chapter 7

Understanding Soundscapes for Games

I hear lush, ambient music drawing me into this place. Here on the Groundbreaker *(a colony ship converted into an orbiting space station), the hum of scanners and machines are emanating all around me, and the din of patrons and workers milling about is filled with bits of muffled conversation. All the while there is a low rumble, letting me know I'm on a very large craft.*

As I walk past the station promenade, I approach a diner called Auntie's Kitchen. *There, Greasy the robot chef welcomes me with a mechanical grunt. We talk. "May this unit dispense proteins for your ingestion?" it asks in a burbly, monotone computer voice. "Let's see what you've got," I say. I am sent to a menu screen where I buy some* Auntie Cleo Catch-Up, *a tomato-flavored paste "with 37 herbs, spices, and flavored additives!" I hear the satisfying "cha-ching!" sound of an old-fashioned cash register, letting me know the purchase is complete. I leave and head back to my trusty ship in the docking bay, the* Unreliable. *The hatch opens with a familiar, hydraulic "ka-chunnnk." ADA, the ship's AI, greets me with her usual sarcasm: "Pick up any more strays while you were out?" she asks in a synthesized tone.*

This is a segment of gameplay experience from Obsidian's *The Outer Worlds*, a vast sci-fi epic where you can follow a main story arc and also choose from a plethora of side quests (Figure 7.1). Each location offers different layers of music, sound effects, and dialogue that inform the player of what's happening all around them. At an enemy base, a soldier guarding the main doors might suddenly say, "what was that?" if you are sneaking up and are about to be discovered. A character may change their opinion of you for better or worse, and the conversation suddenly turns in a different direction. Or, the music may rise and fall based on your circumstances – whether you're resting on your ship, or engaged in combat with a particularly angry *raptidon*.

This blend of audio unfolding in real time is known as the **soundscape**. When implemented correctly, it has the ability to immerse, inform, and entertain us. During development, however, **there must be a balance between the aesthetic and technical decisions made, in order for a soundscape to fully succeed**.

A composer can craft the perfect interactive score, but if there aren't seamless transitions between musical segments, or the volumes are inconsistent, the experience will be punctuated with unintentionally jarring moments. Lines of dialogue could be implemented perfectly and in all the right places, but if the performances are "phoned in" and flat, immersion and entertainment are broken. If sounds that are supposed to provide clues or feedback aren't heard clearly, gameplay outcomes could be affected.

DOI: 10.4324/9781003218821-10

Figure 7.1 Approaching Auntie's Kitchen in The Outer Worlds. *(Obsidian/Screenshot by Keith Zizza).*

The audio designer must own their soundscape and work with the entire team to ensure its success: that includes designers, artists, animators, programmers, writers, producers, and testers (and anyone else I've missed.). This becomes especially important as the game gets closer to the development finish line.

As players, we know the very best soundscapes immediately **transport** us to the game world we temporarily inhabit. This is the result of careful collaboration between audio designers and the rest of the development team. A great soundscape is also relatively **transparent**, in terms of our awareness of its implementation; it should not distract us from the game itself in any way. Otherwise, we will be jolted out of that Magic Circle of participation, and it will have revealed itself to us.

The last time you watched a scary movie, and the tension of the scene increased until it was palpable, I bet you gradually leaned forward in your seat, your teeth may have clenched a bit, and your heart rate may have increased. Was it the visuals that caused this? Yes, partially. But I guarantee that the audio had a lot to do with it as well! It's the same principle with game audio; it too can pull at our emotional strings. Tension and mood can be adjusted when needed, and all the while we take the soundscape for granted – never questioning it but always immersed in it.

Our first thought when playing a videogame may be that it's the *visuals* that inform our responses, before anything else. We process visual information faster, right? Well, it turns out that audio is, in fact, the quicker sense. According to neuroscientist Seth Horowitz, in his fascinating book on aural perception called *The Universal Sense*, humans can interpret changes to visuals up to 25 times per second, but changes in auditory events can be detected at least *200 times* per second.[1] Keeping this in mind, we have many opportunities for providing immediate feedback to

the player, and several events can occupy different regions of the frequency spectrum simultaneously.

The soundscape plays a critical part in drawing us further into the game, playing with our emotions (perhaps again our heart rate) and helping to suspend our disbelief. I always joke with my students about my experience of installing *Skyrim* for the first time: "I started at 5 pm on a Friday 'just to make a character,' and *poof*, suddenly it was Sunday night … and now I'm knee-deep in empty soda cans and covered in Cheetos dust!" Clearly, the soundscape of *Skyrim* had a huge part in keeping me immersed, informed, and entertained (and meeting my daily requirement of MSG).

"Nothing" Is Everything

As someone who has been working in the game industry since the 1990s on dozens of projects, I can attest that one of the best compliments we can get as audio designers is: nothing.

That's right.

When nothing is wrong with the soundscape, there will be no complaints. There may be excitement about the game soundtrack, but never have I heard, "listen to the dope EQ on this dialogue!" or "these UI sounds are on point!"

When something is wrong with the soundscape, it immediately pops up in a player's ears. The specifics of *what* is actually broken may not be quantifiable by players or your development team, though. A common response might be, "well, you're the audio designer … you tell me." And it's our job to fix it. This is also a good opportunity to open up a conversation about game audio with the team: describing some of your techniques, the various game audio disciplines, test methods, and offering a bit of new vocabulary, to facilitate their understanding of your process. And a great place to start is with an overview of the soundscape.

Telling a Story through the Soundscape

You already know from playing a game that the audio experience is never a linear one. For instance, even if there is only one tune that plays in an endless loop, and no dialogue is present, sound effects are generated in a **nonlinear** way each time you decide to collect a powerup, slay a goblin, or match three tiles in a row.

Part of what differentiates a game's **nonlinear** experience from a linear one is that the player is allowed to make choices. With each new gaming session, there is an opportunity to play the game differently. Audio researcher KC Collins notes that a player has "agency and freedom" to choose how to play, thus providing them with "authorship" over how the soundscape unfolds. Unlike film and television, where music, sound, and dialogue are a singular, custom-made construct, the elements of game sound, music, and dialogue are built for a soundscape in real time. The player, in this sense, is telling their own story through a live mix of auditory events.[2]

On the development side, the amount of audio content in a game is known quantity, of course. But as to *what*, *where*, *when*, *why*, and *how* audio is heard during gameplay – that is less predictable. As audio designers, we make our best guesses on what will be heard at any given time, and under what conditions. We can design systems to implement audio in a way that best serves the game, but there isn't a perfect

fit for every situation during gameplay. But over time, some best practices have arisen for creating a balanced and more predictable soundscape, which we'll look at in more detail in Chapters 12 and 13.

Our very first order of business is to determine what the various categories (and subcategories) of audio are. Creating a list of these items will reveal some elements that fit into more than one place, or seemingly nowhere at all – perhaps being in a class by itself. Having a few gray areas in your list will always be the case and that's OK. These special-case items may require some extra attention during the implementation process, but their discovery is precisely why we're making the list in the first place!

Your method of categorizing the following items may be somewhat different, but here is a thorough listing for your consideration.

Soundscape Components – An Introduction

Sound

Sound effects can help immerse, inform, and entertain us in many ways. Often, a blend of simultaneous sounds can provide specific feedback, a sense of place or time, or invoke an emotional response at just the right moment. Sound is often the category that makes up the bulk of the game's audio content.

Cutscenes

For games, the definitions of **cutscenes** and **cinematics** have become interchangeable.[3] These are non-playable movie sequences in the game which can help drive the narrative and maintain continuity between levels or chapters. Cutscenes can be provided as an introduction, as transitional moments between levels or chapters and winning or losing the game. While music and dialogue can play a big part here, great sound design can also be a highly influential and expressive part of the cinematic experience.

Environmental Ambience

An immersive soundscape may call for some environmental or ambient sounds. The low rumble of a starship's engine, the reverberant wind inside a cave along with water drops and a rat scurrying by, or the clamor of a baseball stadium between hits are all examples of ambience. Weather effects can also be included here, such as thunder and rain, a sandstorm, or a tornado.

Characters

Whether it be a human, alien, or Formula 1 race car, every character has a personality, and with it, a set of sounds that defines them. Movement animations for a character usually need at least one set of motion-based sounds, and possibly more if they're ground-based and moving across different terrain types (dirt, stone, metal, wood, etc.). They may also need emotes, idle sounds, special attack sounds, getting hit reactions, and if they're humanoid, let's not forget their William Shatner-esque death knell!

Objects

Any object may require multiple sounds: a door may need opening, closing, locking, unlocking, and breaking down sounds. Items may need sounds when picked up or put down, or when you're moving around gold, weapons, scrolls, and armor in your inventory. Starting a campfire may trigger an object sound (the crackling of burning wood), or perhaps crafting a new rune from several smaller stones causes a magical event to be heard (forming the new object makes a *ka-pow! sound* followed by a shimmery, crystalline effect).

Pickups

What's more satisfying than the sound of Mario collecting a row of coins, or the *bwooop* heard when you eat a bonus fruit in *Pacman*? This is a special case of an object sound and is often an abstract sonic representation of the item picked up. Stylistically, pickups can be of a positive or negative nature, or anywhere in between.

User Interface

Any time you hear the sound of a button pressed, slider moved, or the beep of a heads-up display (HUD), you're listening to user interface (UI) sounds. This may also include audio for scoring points, extra lives, popup messages, alerts, and indicators. They are usually non- diegetic, meaning they are not heard in the game world per se but are rather just for our ears alone (more on this below).

Music

From its humble beginnings as simple tones, game music has evolved dramatically over the course of five decades. The 1990s saw the adoption of MIDI soundtracks, sampled audio, and MP3s to allow for many more songs per game release. The 1990s also introduced off-the-shelf interactive and adaptive music tools through the use of **middleware**, also known as **audio engines** (which we'll explore in later chapters), and today, blazing-fast processors on several platforms can support multichannel music with ease. There is also generally more media storage available per game, allowing for multi-hour soundtracks. In examining game music further, we find that it can be broken into several categories.

Cutscenes

Here are any musical events for the game, such as the developer logo or title sequences, introduction, transitional cutscenes, and any end sequences.

Setup

This is a relatively short, looping track that normally plays during the main menu and any setup screens.

Background/Standard Gameplay

This is any music heard during the normal modes of play. Background music could be calm and ambient at times, with breaks in between so as not to fatigue the listener.

Battle/Challenge

The music here matches the challenge presented to the player. It could sway in different musical directions for aesthetic purposes but also or suddenly change in intensity according to how close you are to winning (or losing).

Characters

Unique themes for each character can play an important part in solidifying their personality and purpose. Their music can be repeated and interwoven throughout a story or even intertwined between multiple characters.

Stingers

You open the secret vault door, and – Dum, dum, duuuuuum! – cue the dramatic orchestral flourish! These short, musical moments are called stingers, which can happen at any place in gameplay, whether it be surprise moments, item pickups, or short, transitional win and lose sequences.

Dialogue

The voices of humans, creatures, and omnipotent beings can add depth and realism to a game, and can be organized into several types.

Cutscenes

Dialogue in cutscenes can help propel the main story along, help develop a character's narrative, or provide assistance as a tutorial or providing clues for gameplay events.

Narration

A narrator may be needed to introduce the game or prepare the player for a mission. In any case, the narrator is a central figure, one that is an anchor of sorts among all of the game's dialogue, and who should be carefully chosen.

Characters

Every story needs interesting characters, and your game may have one or more that have lines of dialogue when interacting with other characters, objects, or even talking to yourself ("I don't have that key yet!").

Tutorial

Often times, a narrator or character will guide us through the various modes of gameplay and mechanics and will require lines for many different situations.

User Interface

Sometimes UI sounds will have accompanying voice cues, like your cybernetic adjutant (aka Terran Advisor) in *StarCraft II* calmly informing you that "your forces are

under attack" or "nuclear launch detected." If the voice is on-camera, panning it to the appropriate spatial location can make it more convincing.

Dialogue as Sound Effects

For more incidental voice cues like "hey!", "take that!", or "ouch!", these are sometimes categorized as sound effects, especially if they are used during brief character actions or reactions.

As a side note, some of the challenges faced with dialogue production are hiring multiple voice artists, juggling recording schedules, and **localization**, where we must translate the dialogue into multiple languages for international release.

Soundscape Propagation

Now that we've examined the most common audio layers, we need to think about how those layers will work as a whole – how this soundscape will be heard by the player. Our soundscape has spatial dimensions to it and can be presented in a variety of ways.

2D Audio

A 2D audio setup provides a listening experience within a stereo (left and right channel) field. 2D audio elements can either be **fixed** or **positional**, and either mono or stereo.

Fixed audio is firmly set in place anywhere in the 2D field, such as with a stereo game soundtrack, whereas a UI or HUD-based sound effect may be positioned at wherever its visual indicator is on screen, usually in mono so our ears can easily locate its position. The sound of a bird flying across the screen can utilize a 2D mono sound effect attached to it. A 2D game may also have no audio panning whatsoever, with music in stereo but everything else as "straight up" mono sounds that feel like they're coming from the center of the speakers.

You can also think of 2D audio being presented at the front of the screen on a flat plane; there is no true depth (no third dimension) to the audio. However, the illusion of 3D audio can be achieved simply by adjusting the volume based on the depth of the object of attention on screen, a technique sometimes called "2.5 D."

3D Audio

3D audio is **spatialized**, such that it has a position anywhere in a spherical field surrounding the player. A vehicle approaching the camera will increase in volume and decrease as it passes by; similarly, if the player is moving circularly around a band of creatures, their grunts and growls will pan around the listener accordingly. The **Doppler effect** is also implemented in games that demand a more accurate 3D audio experience.

Diegetic, Non-Diegetic, and Trans-Diegetic Audio

The word *diegesis* is Greek for "narrative" – something described that is part of the story itself. When we refer to **diegetic audio**, we are referring to any sound heard that

is part of the game world itself. These could be any player or NPC sounds, weapons, and character dialogue, to name a few examples. Each of these would still be diegetic if the objects associated with the audio are not visible on screen. The sound of seagulls and crashing ocean waves may be heard against the still image of beach scene, and even though there are no animations for these elements, it is implied that they exist in that moment. A humming generator we cannot see behind a wall, or the far-off echoes of a howling wolf are also diegetic.

A game's soundtrack and its narrator are two examples of **non-diegetic audio**, as they are not heard by anyone within the game world (other than yourself, being external to the game). However, if an in-game character turns on their car radio and music starts blaring, or if a narrator is present on screen and delivering a monologue, these are diegetic audio elements. Other non-diegetic audio includes setup menu controls and some UI button sounds like player preferences and game options.

On rare occasions, audio may be **trans-diegetic**. This means that the sound morphs from diegetic to non-diegetic (or vice versa). The whoosh of a jet flyby used as a transitional effect from one scene to actually seeing the jet flyby in the next scene makes it a trans-diegetic audio element. In Bloober Team's *The Medium*, the player (Marianne) discovers an old radio in a fallout bunker; when she turns it on, a song ("Fade" sung by Troy Baker and Mary Elizabeth McGuinn) is playing, and if the character moves around the bunker, the music fades in and out from diegetic to non-diegetic, depending on her location.[4]

Accessibility

We should also keep in mind that some players are visually impaired. Not everyone has 20/20 vision or can see the entire color spectrum. These players may need to rely more on specific audio cues, so with this in mind, soundscapes are that much more crucial to get right.

Conclusion

A great soundscape should be seamlessly woven into the game itself. Though this may be hard to qualify, it should feel like an effortless listening experience. It must surreptitiously inform, immerse, and entertain the player, and do so consistently throughout the gameplay. Behind the scenes, of course, great care and effort needs to be taken to ensure the best aesthetic and technical elements exist and are in balance with one another. The practice of *using the least amount of audio required to serve the game* is also part of this philosophy; any more or less than that is no longer serving the needs of the game itself.

Notes

1 Seth S. Horowitz, *The Universal Sense How Hearing Shapes the Mind* (New York, NY: Bloomsbury, 2013), 98.
2 Karen Collins, *Game Sound: An Introduction to the History, and Practice of Video Game Music and Sound Design* (Cambridge, MA: MIT Press, 2008), 4, 135.

3 Before scripted in-game events were widely used, cinematics used to refer to the highest-budget movies made for the game, such as the introduction and win/lose events, and cutscenes were the minor or transitional moments in between gameplay. But over the years, the two terms have come to mean the same thing, encompassing all noninteractive movie sequences.
4 *The Medium* Easter Egg: *Fade*, sung by Troy Baker and Marianne McGuinn: https://youtu.be/91rojfDZ99w?t=145.

Bibliography

Collins, Karen. *Game Sound: An Introduction to the History, and Practice of Video Game Music and Sound Design*. Cambridge, MA: MIT press, 2008.

Horowitz, Seth S. *The Universal Sense: How Hearing Shapes the Mind*. New York, NY: Bloomsbury, 2012.

Marks, Aaron, and Jeannie Novak. *Game Development Essentials: Game Audio Development*. Clifton Park, NY: Delmar Learning, 2009.

Chapter 8

Sound Design

The Importance of Sound

From the ambience of an alien planet to the clamor of a medieval battle, sound design wields great power in captivating our imaginations. Ever since the beginning of audio recording in the early 20th century, sound has helped tell a story to the listener, transporting them to a particular place and time and (hopefully!) entertaining them along the way. In audio-only mediums such as radio, podcasts, and audio books, sound is tasked with a much greater responsibility to enhance the narrative. This is especially the case for completely imagined settings in fantasy and science fiction stories. Classic productions such as NPR's *Star Wars* radio drama made clever use of Ben Burtt's sound effects from the original movie trilogy, incorporating it in 13 half-hour episodes. This played a large part in the project's success, in concert with John Williams' score and voiceovers from the original cast members.[1] And of course, the *Star Wars* films and television series all benefit from custom audio that helps define its universe – the sounds are permanently fixed in place, in a linear progression.

As we saw in the previous chapter, videogames make use of a **nonlinear** audio presentation. Specific sounds can be heard at any time, repeat themselves, get cut short or played at different speeds, layered upon each other, or even played backward. And because sound design normally plays a larger part of the game mix, it must provide a generous amount of feedback, engagement, and enjoyment, all in real time. Where to begin?

A Unified Aesthetic

Suppose your game is set on Mars, and a new colony is arriving to build the planet's first city. What should the buildings sound like? There could certainly be a variety of sounds, like oxygen generators, terraforming machines, and transportation systems, to start with few ideas. Stylistically speaking, it makes sense that the sounds should all "fit" together in some way. This can seem a bit intangible at first, and to some degree it is highly subjective. But we can make some best guesses by first hearing if the sounds feel compatible together in a mix. Chances are your project is still early in development, and audio may not be supported yet; you can always create a mockup of the sounds in your DAW, choreographed as if you might hear them in-game. Also, sharing your sound design concepts with the team is extremely helpful, as you'll get

DOI: 10.4324/9781003218821-11

some good feedback in determining the overall aesthetic. If time permits, create a few different styles for each type of sound that you can audition to the group. To help with the decision-making even more, you can play the sounds against any screenshots or relevant concept art from the project. If no art assets exist yet, any inspirational art from other sources will do!

How Much Sound?

The amount of time spent on creating sound effects also depends on practical considerations, namely the game's production schedule and budget. The sound designer may have two weeks or two years to work on a project, and more than one person may be needed to complete the audio. Bear in mind that compared to music and dialogue, sound usually involves the broadest scope of implementation within a game, as well as the bulk of the audio files, otherwise known as **audio assets**.

It is also important to consider the logistical and technical aspects of sound design. Some first questions are: how much sound does the game actually need? What limits should be placed on the total number of assets created? When should the player hear sounds, and how many sounds should play at once? This is where some competitor research pays off. The development team has begun designing a game; surely, it is drawing inspiration from similar titles before it. The sound designer must seek similar insights: watching gameplay videos, playing videogames hands-on wherever possible, and carefully studying their soundscapes.

With this information, they can better estimate the sound design budget, which includes:

- Any research up front;
- The potential number of sound assets needed;
- The approximate time required to produce them; and
- The approximate time required to implement them into the game.

Sometimes a minimalist approach with sound design is very effective. Playdead's *Limbo* is a haunting puzzle platformer, utilizing a spooky black-and-white visual scheme with silhouetted characters and an eerie soundscape. At times, only environmental sounds like wind, birds, and waves in a pond are heard. But in other areas, a giant spider may appear that suddenly injects terrifying, buzzy drones and rumbles into the mix (in addition to the spider's own creepy thuds as it crawls toward you).[2] According to composer and sound designer Martin Stig Andersen, sounds that were most important for the player to hear were given preference, and only occasionally, so as not to become annoying during gameplay.[3] Overall, the soundscape in *Limbo* is brilliantly implemented and serves the visuals and gameplay well, all while using a small set of sounds.

Conversely, Witch Beam's puzzler *Unpacking* has the player opening boxes and decorating their home, producing over 14,000 sound effects for moving around cleaning supplies, books, clothes, home electronics, and all sorts of everyday items (which are **Foley**-based, a discipline we'll talk about shortly).[4] Each item also has many slight variations in sound when interacted with, which feel like tasteful additions to the game's calming aesthetic (Figure 8.1).[5]

Figure 8.1 Unpacking has a vast amount of customized sound effects, which makes this puzzle game a delight to listen to (Witch Beam/Screenshot by Keith Zizza).

Designing Audio with Purpose

Before any sounds are created, the sound designer must consider *where* they should exist in the game and *why*. An easy trap is to simply add sounds *ad hoc*, irrespective of form and function. This approach could cause them to lose sight of their original plans, if any – not to mention it being an organizational nightmare. Rather than having a throw-it-in-and-see-what-sticks attitude with game sound, it's always best to think about its objectives first. This way, sound effects are added with *purpose*.

The Roles of Game Sound

Just as we observed with the high-level soundscape analysis in Chapter 7, soundscape, **each sound effect needs to serve the game first,** above all other reasons. With this in mind, let's take a look at the key objectives of sound in games.

Create an Immersive Setting and Mood

Sound can help to draw the player into a scene, whether it be a specific location or period in time. It should also conform to the central narrative of the game. For example, the din of distant city traffic, electronic outdoor advertising, and haunting synth drones combine to form a convincing rooftop ambience in *Cyberpunk 2077*.[6] The steady sounds of flowing water and nautical machinery forms the underlayment of the ambience in *Bioshock*, where the player explores the now-derelict underwater city of Rapture. The listener cannot help but feel they are part of this submerged habitat, thanks to the subtle complexities of the environmental audio.[7] Even the simplest sound design can be effective, such as bird tweets and a gentle wind during a round of *Wii Sports: Golf*; at times, this is all we hear on the course, yet it works perfectly to immerse and set the overall laid-back tone of the game (Figure 8.2).[8]

Figure 8.2 Sound design can go a long way in describing a scene, especially with the environmental ambience heard in Bioshock (2K Games/Screenshot by Keith Zizza).

Deliver an Entertaining Experience

Let's face it, sometimes it's just fun listening to ear candy. It could be the sound of a finishing move from your favorite character in *Super Smash Bros. Ultimate,*[9] or the satisfying plethora of frequency-rich sounds playing all at once when a treasure chest is opened in *Diablo 3*, revealing a bounty of loot.[10] Most players enjoy the initial surprise of an interesting sound effect and look forward to hearing it repeatedly. That said, other sounds can certainly benefit from variety at times, especially ones that are repeated often: multiple explosions can feel more random by adding variations with slightly different debris trails, or the *twang* of a wooden bow can benefit from a few pitch and volume variations.

Add Tactile Realism

You know it when you hear it: that satisfying *chunk, glink, or pow* of a sound effect you just triggered in the game. It could be the sound of a hollow shotgun shell bouncing off the floor, or the squeak of sneakers on a parquet floor during a fast-paced basketball match. Often, making game elements seem real can be as simple as capturing the real-world sounds of items represented in the game, like starting a Formula 1 car and revving its engine.

There will also be times when a sound must be convincing, and yet does not exist in our own reality. A light saber from the *Star Wars* universe, Spider-Man shooting a web from his wrist, or the emotes from the various creatures in *Monster Hunter* are a few examples. In Remedy Entertainment's *Control*, Jesse uses her supernatural abilities to manipulate and destroy a variety of objects. Creaks, breaks, and impacts from all sorts of materials can be heard when she interacts with them, but these sounds are often sonically distorted and altered in ways that make hearing them viscerally satisfying – as if we always expected them to sound this way.[11]

Provide Critical Feedback

Getting precise feedback from the game is important across every genre. In a first-person shooter (FPS), the sound of having no ammo left (click! click!) can alert the player to quickly switch to another weapon. In an RPG like *Skyrim*, the sound of swishing blades swinging between walls can help time a nail-biting dash through them, and the muffled growl of a *Draugr* just around the corner can quickly send a fight-or-flight message.[12] Activating a Color Bomb in *Candy Crush* wipes out all the candies of any one color on the board(s), and the multiple scoring sounds enhance the feedback mechanism (and sounds fun too!).[13]

Connection to the Brand

Creating the overall "look and feel" of a game is important, and sound plays a crucial part of this strategy. Establishing a sound design style helps give the game its unique identity, and hopefully one that builds some long-term equity with its fans over time. This brand identity is heard immediately in such game franchises as *Mario*, *Sonic*, *Angry Birds*, *Plants Vs. Zombies, Halo, The Sims*, and *League of Legends*. They all have their own "signature sound" that gamers can get cozy with, at a high level of quality and consistency they can come to expect with each new release.

Sound Components: A Closer Look

A general list of soundscape components was introduced in the previous chapter; here is a more detailed look at how the sound components are structured, and some contextual examples of effective sound design in these areas. Later in Chapter 12, we will look at how these components can be implemented.

Cutscenes

These can be **prerendered** as stand-alone movie files (MP4, WMV, and AVI formats are among the most popular), or generated **in-game**, meaning they are scripted events using the game engine itself. Technology has progressed to the point where in-game cinematics are becoming a more popular option, for two reasons: scenes are easily modifiable with a change of script, and they also facilitate a smooth changeover into gameplay.

One example of a prerendered cinematic with effective sound design is the introduction for *Death Stranding*. About halfway through the scene, we see Sam Bridges riding on his motorcycle in a rocky remote area. He tries to escape a storm (of the supernatural variety) and there are many crows flying and cawing above him. The mix of the crows and the storm, coupled with the rich engine sounds of the motorcycle and the terrifying, invisible sound of nearby Beached Things (BTs) make for an amazing soundscape. The fact that the audio was edited and mixed with precision makes it an even more palatable listening experience.[14]

Then there is *Machinarium*, a bespoke steampunk puzzle game with an all-sound design soundscape: during the intro cutscene, a transport ship travels across a bleak planet, dumping your robot character onto a high-piled junkyard heap. The cutscene

is a scripted series of events utilizing the game engine (originally created in Flash), so the transition into gameplay is a seamless experience.[15]

Environmental Ambience

This is typically a long-playing, stereo-looping sound accompanied by several shorter, positional mono sounds triggered at certain intervals. For example, the din of a bustling forest at night may be comprised of the stereo sound of light wind and crickets; then, one or more layered, panned mono sources are triggered every so often. So for an ambient forest looping sound, some supporting mono sources might include an owl hooting, a wolf howling, or even bats flying past, moving from one location to the next (with its path randomly decided at runtime).

In Children of the Nile, I used an ambient loop for each distinct type of terrain on the map, which could also be changed during time-of-day. On the sands you would only hear a constant low-level wind, day or night. But on a floodplain during daylight hours, you would hear a gentle wind and cicadas, coupled with songbirds, geese, or falcons; then in the evening hours, the ambience would change over to the sound of wind, crickets, and the occasional jackal howling in the distance.[16]

Whether it's a gentle wind, lightning strikes, or a tsunami, environmental ambience can also change with seasons, weather events, distance, or other special moments.

Actions, Reactions, and Objects

Characters

As introduced in Chapter 7, the sounds for the player and other characters are often thought of as a set and are organized as such, both in concept and implementation.

Let's say the plan is to have a small number of sound variations for a dungeon traveler: footsteps should be at the top of the list, at a minimum. Next on the list might be at a couple of alternating action sounds for repetitive abilities such as jumping, landing, and basic combat animations, including sounds for getting hit (their reactions). Special attacks, spellcasting, launching/receiving a buff or debuff, and a death sequence ("*Ahh!*," followed by weapons and armor clanging on the ground) could all remain unique. If there's still some room left over, idle animations are a great place to add some variety and humor – spinning a yo-yo, tapping a foot, or clearing their throat to get the player's attention – these are all possibilities that are worthy of a sound effect or two.

But of course, the more variations that are created, the more realistic and entertaining the experience of hearing the character will be. And there are ample opportunities for unique sounds with NPCs (non-player characters). In *Minecraft*, mobs (monsters) have their own sound sets for being idle, attacking, getting hit, dying, and/or a few special abilities here and there. There's nothing like hearing the reverse-groaning sounds of Endermen idling nearby; if the player happens to look at them directly, their groans turn into piercing shrieks, and it's fight-or-flight time![17]

In another example, the sound design for a card game should include, well, lots of variations for shuffling, dealing, placing, and turning over cards (which seems obvious but is frequently overlooked). It's also a great opportunity for some original recording work.

Yet another example is with a first-person shooter (FPS), where the player doesn't actually see themselves, but perhaps only their hands using a weapon. There are several opportunities here for off-camera actions and reactions including footsteps, emotes, and breathing variations.

The category of objects can consist of just about anything, either through direct interaction with the player or otherwise: the sound of a race car they're driving, along with the crash of the racetrack wall they just hit; the tumbling of dice on a craps table; a sailboat gently rocking back-and-forth in a bay; the creaking of a wooden door opening; or the impact on a shield being struck by an orc's battle-ax.

Challenges for Abstract Environments

If a game is set in a world with nontraditional constructs, coming up with new sounds may be challenging. For example, there are certain sonic expectations when the player is using a sword and shield (bereft of any magical enhancements), but what about a game that is totally removed from reality?

Geometry Wars has the player, an outline of a two-dimensional spaceship, fighting enemy shapes. It uses futuristic UI and gameplay sounds, including an interesting transitional effect: when a new fleet of shapes warps in to attack the player, a short but heavily modulated synth sound plays. To emphasize this effect, the music and all other sounds cease playing until the warp sound ends, making it feel even more powerful.[18] Similarly, *Thomas Was Alone* uses simple waveform types for the sounds of basic shapes bouncing and moving through the game.[19] Another example of abstract sound design can be heard in *MirrorMoon EP*, a planetary exploration game. The haunting synthesized soundtrack often seems like more sound design than music, and the sound effects are also constructed from rich synth textures.[20]

Some games have abstract-looking tiles or gems that require unique sounds. In the match-3 game *Marvel Puzzle Quest*, the heroes and villains of the Marvel Universe are portrayed in battle, using the abstraction of opaque, multi-shaped tiles on a grid to determine who is winning or losing. This required a mix of realistic sounds (the heroes in combat), and more abstract sounds for the tiles, constructed of rudimentary shapes like circles, diamonds, and squares. I created their sounds from a blend of mahjong tiles dropping, ceramic plates breaking, and glass shattering, which yielded an effect that ultimately made the cut (no pun intended!). The tiles also generated sounds for dropping, shuffling their order, bombs activating, and serving as a conduit for the various special abilities of characters (Figure 8.3).[21]

Pickups

Pickup items often require multiple **instances** of the same type of sound. When *Little Big Planet's* Sackboy collects a row of similar objects, he moves fast enough past the second and third one before the first even has time to finish playing.[22] Each successive instance needs to be considered carefully, as they take playing them takes up additional resources and a part of the soundscape.

Pickups can also benefit from being musically tuned, to better integrate them with existing background music or other melodic soundscape elements. In *Subway Surfers*, the characters Jake, Tricky, Fresh, and Yutani all ride the subway rails, grabbing

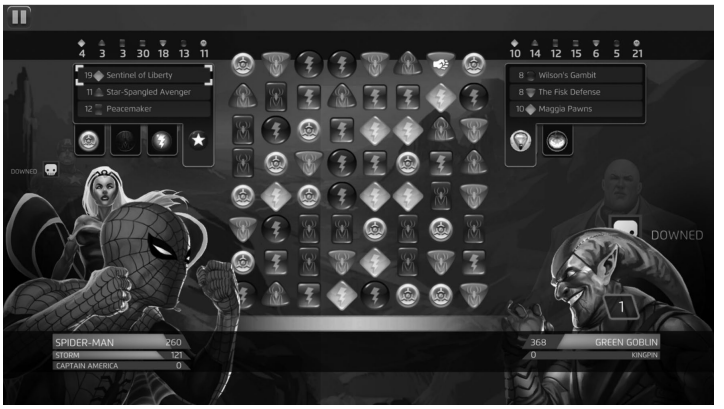


Figure 8.3 The tiles in Marvel Puzzle Quest provide tactile audio feedback to the player when they are moved, activated, or broken (D3 Publisher/Screenshot by Keith Zizza).

tokens along the way that each generate a pleasing sound effect. Though collecting a burst of tokens produces ascending chromatic notes, the single token pickup sounds are tuned to a C note (with a lower G to fill it out a bit), matching the same key signature as the soundtrack (C major).[23] It just wouldn't feel as connected to the mix if the pickup were tuned to, say, a dissonant F# (or even worse, B – only one semitone apart).

User Interface

These are usually pegged for high-priority playback in the mix, where they must be heard to alert the player – either as a prompt, message, or response from an action taken by the player, such as selecting an item or pressing a button. UI sounds are typically designed to cut through a soundscape mix using a variety of techniques which we'll see later in Chapter 12. One initial observation is that they tend to work well in the higher frequency range, away from the bulk of the frequency spectrum utilized by music, sound, and dialogue. They can be either quick one-shot sounds like a button click or incoming message alert to looping audio such as a heartbeat to indicate low health. UI sounds normally take up a small amount of RAM, and one or two simultaneous instances of UI sounds are fine in most cases.

But by no means is there a requirement to use high-frequency beeps and clicks – the sounds just have to be heard clearly within the soundscape. In Arkane Studio's *Deathloop*, for example, all of the UI sounds were created predominantly using an electric guitar, including use of the input jack, harmonics, and even playing the strings using a double bass bow.[24]

Sound Disciplines

Field Recording

A game may require some sort of environmental ambience, and it can benefit from original **field recording**, also known as **remote recording**. Field recordings can sound

wonderful in the context of your interactive scene, if time and budget permit: a bustling city block, a forest, the beach, a calm field in the springtime, or the dramatic ups and downs of a thunderstorm (recorded from a waterproof vantage point!) can all add depth and realism to a scene. As mentioned in the previous chapter, a stereo, long-playing loop can serve as a "bed" in mix, while other, shorter ambient sound effects can accentuate it.

The takes can then be downloaded via USB or media card to a computer, where they can be backed up and edited in a DAW. There, compression and EQ can be added as necessary, along with removing anomalies like unwanted rumbles (a truck hitting a pothole during your peaceful meadow recording), a city pedestrian getting a *little* too close to your mic (breaking up with their partner on the phone – this actually happened during one of my recordings!), or a bird that decides to screech repeatedly, wherever you seem to relocate. Of course, I'm fine with birds in nature recordings. But they are heard just about everywhere and can tangle up many a session – so much so that there is now a plug-in for such a conundrum, called *DEBIRD* from BOOM Library, that automatically detects and eliminates them from recordings.[25] Now if only someone could invent *DE-motor*.

Foley

When sound for movies became a viable technology in the late 1920s, there were many scenes that required more than just background music and dialogue. Mics placed on soundstages weren't that great at recording clear or detailed audio, and something needed to be done to ensure specific sound effects were heard in the final production. Enter Jack Foley, the screenwriter, producer, and sound designer (and even stuntman!), who popularized the art of recording sound synchronized to visuals. He would watch any scene that required human sounds and imitated the actor's movements, creating a believable sonic performance. The technique has been named in his honor and is used to this day.

Foley can be any live performance that replicates a sound, whether it's in the scene or not. It is grouped into three categories: footsteps, props, and clothing. The most common Foley effect is footsteps, but hand Foley is also popular, either for the hands themselves (clapping, snapping fingers, etc.) or through the manipulation of objects, such as turning a squeaky doorknob or loading a shotgun. Foley can also convey events that aren't seen, like the turning of a wind machine to simulate a blustery ambience, or the shaking of a giant aluminum sheet to simulate thunder. Foley can also be more abstract, in the creating of sounds used in fantasy or science fiction realms, or for when a specific action needs to be over-the-top: up-close weapon impacts or cartoon hijinks, for example.

It doesn't matter how dramatic or ordinary Foley is, so long as it's adding value to the final medium. It may seem simple at first glance, but Foley is not an easy task! Great care and skill are needed to match an actor's movements, as well as making use of any relevant objects. A **Foley artist** watches a scene and mimics the action, generating multiple takes in the process. For footsteps, a **Foley pit** is often used, which is a group of isolated surfaces for simulating walking, running, and the like. If there is a scene of an actress wearing boots and walking on dirt, the Foley artist will do the same, using comparable footwear on a dirt surface in the pit. Their

Figure 8.4 The author performing some Foley horse clops, Monty Python-style!

subtle step variations in cadence, pitch, and dynamics are what gives the performance authenticity.

For games, Foley can add tremendous realism to character animations, either for movement on various surfaces or for the handling of objects (ala *Unpacking)*. It's important to note that while a character's walk animation may loop over and over (known as their **walk cycle**), that doesn't mean there should only be a single repeating footstep sound attached to it, ad nauseum. Though the pitch and volume of a single sound be changed during gameplay, its general *timbre* can still become quite repetitive and draw unwanted attention to itself. To remedy this, the character should be supplied with several Foley footstep sounds, with their pitches and volumes slightly randomized during gameplay. Suddenly, a handful of sounds has now become hundreds; the ear "lets go" of any pattern recognition and allows the immersion to take hold, at least in its own small part here (Figure 8.4).

Foley can still be used in the traditional way for games, where the artist is choreographing events live in a particular scene, as for a cinematic. But for most in-game actions, it is not always known when the sounds will be generated, so they are recorded as stand-alone sound files that are triggered when that action occurs.

Creating Magic with the Mundane

When a Foley artist is tasked to create something extraordinary, they must get creative with it! Take, for instance, dragon wings: since they don't exist, the artist might try flapping a large piece of canvas up and down, or a large trash bag; it could be processed in their DAW with some downward pitch shifting and perhaps some compression and EQ. Or, for an ancient dungeon door opening, they might discover that a concrete block slowly dragged across the pavement (with a little reverb added in postproduction) does the trick. The mighty *swoosh* of a broadsword might just be a long stick recorded at close range, with some DAW post-processing.

In *Mortal Kombat*, some of the most sinewy, visceral impacts were created by destroying all manner of produce (but please be advised of graphic content in the video reference at the end of this chapter).[26] And this is a tradition that hearkens back to early horror movies. Bone breaks can be simulated by snapping celery or carrots, or

even crunching up a large pinecone. The sound of slimy creatures or other gory fare can be had by mushing up wet pasta between your hands (very gooey!).

I highly recommend reading *The Foley Grail* by Vanessa Theme Ament, for deeper insights into the recording process. There, you'll find interviews with several Foley artists, along with some great theory and techniques from the author herself.[27] Some other great Foley wisdom can be found in *The Sound Effects Bible* by Ric Viers.[28]

Custom Recording

If the Foley artist is not performing their craft in the studio, they may still want to make use of the studio environment for original recordings that aren't synced to anything in particular, like the sound of a household appliance in operation, or bringing in your neighbor's parakeets to do some chirping.

Sometimes sound effects are needed that don't necessarily exist in nature. What does a teleportation machine sound like? Or an undead curse spell? To facilitate the creation of a more unique sound effect, synthesis tools may be used, which are either hardware or software-based (as we saw in Chapter 4). Combined with existing recordings in a DAW or on their own, the sky's the limit on what can be accomplished here.

Source Material

If the sound designer is faced with a quick turnaround, they can rely on sound effect libraries, and/or sound effect marketplaces, in order to meet production deadlines. Having "off-the-shelf" source material at hand is a great go-to option; many library sound files have multiple takes of one event, so there are often variations to choose from within a selected file (chirp 1, chirp 2, etc.). The challenge is often finding the sound required, especially if the filename doesn't match the material (e.g., a dog bark effect with the filename of "sfx_emote_animal_24.aiff"). It's more reliable to search on the sound file's **metadata** (the search keywords), which will more likely lead us to it. Software tools like Soundminer, Basehead, and Soundly can all perform powerful audio file searches using metadata, while also providing features such as on-the-spot auditioning.

Sound Effects Libraries

Perhaps the sound designer is working on a sci-fi shooter and needs a lot of futuristic weapon audio. This might be the time to consider obtaining source material from a **sound effects library**, where they can find entire categories of sounds, either predesigned to perfection or basic "raw" elements to assemble on their own (for example, a polished bow-and-arrow sound effect might have the draw and shoot elements already combined into one sound file, as well as files of the separate elements). Some libraries may have as few as 100 sounds, or into the tens of thousands. Two of the longest-running sound effects companies are Sound Ideas and the Hollywood Edge. Both offer many professional libraries spanning a variety of topics – Foley, cars, ambience, cartoon effects, monsters, weapons, and the list goes on (and on). There are also excellent libraries offered from SoundStorm, the BBC Library, and Soundrangers.

Some newer companies that also offer an amazing array of useful sounds for games are the BOOM Library, Gamemaster Audio, and FuseHive. But of course there are many more than I have listed here, and this book can only be so long!

Sound Effects Marketplaces

If the user (the sound designer or game company) needs to go beyond what they currently have in their personal library of sounds, this is a great place to start. **Online marketplaces** allow the user to license only what they need. A nuclear blast? No problem. Slide whistle? They've got your back. There are many excellent marketplaces, but the one that's been around the longest is Sounddogs.com (since 1997), with many of the major Hollywood studios' sound libraries included, along with their parent company, Sound Ideas. They also offer a subscription service for up to 2,000 downloads a year. Soundsnap is another great choice, with nearly half a million sounds as of this writing, and an unlimited download option. Other marketplaces include BOOM Sound Effects, Epidemic Sound, artist.io, and MixKit, to name a few. Each offers different licensing terms and/or subscription options, so they must be considered carefully.

Licensing

When purchasing a sound effect from a library or marketplace, the user (a sound designer or game studio) does not actually own the sound outright. They are in essence buying a **sound effects license**, which is the *right to use the sound* in a game, and they must carefully look the **licensing agreement**. Most sounds are royalty-free, meaning the user only pays for the license and not a percentage of each sold copy of your game. The agreement may state that the user is allowed to use a sound effect for only one project or for a lifetime on as many projects as they wish. Again, there are all slightly different agreements from company to company, so they must be read carefully.

"Free" Sound Effects Licensing

The largest online database of free sound effects and samples can be found at Freesound (www.freesound.org). Other popular free databases are SoundBible (www.soundbible.com) and ZapSplat (www.zapsplat.com); all three sites have royalty-free downloadable sounds through Creative Commons (CC) licensing.

A widespread misnomer about Creative Commons (CC) audio is that because the media is free and can be shared, no credit is necessary. In fact, there are several levels of CC licensing – all are free, but some require more attribution than others; some audio cannot be used for professional use at all. For example, a license of CC0 (Creative Commons Level 0) audio is free to use without citing the author, while CC-BY means the user must cite the author as the source. A CC license of NC means no commercial use, and CC-ND means that the user cannot distribute the work in an altered form. CC-BY, CC-NC, and CC-ND licenses can all be appended to a single work (e.g., CC-BY-NC-ND would be the most restrictive license) (Table 8.1).[29] More specific attribution information and examples can be found at https://creativecommons.org/licenses/ and www.audiocommons.org.

Table 8.1 Audio Commons Attribution Levels

Type	*Description*
CC0	You may copy, distribute, and alter the original audio in any way you see fit, for personal or commercial use. You do not need to give credit to the original author.
BY (Attribution)	You must give full credit to the original author if you use this audio for any project.
SA (ShareAlike)	You cannot modify the audio under a different license. For example, if you edit or remix audio with a CC license, you must share (distribute) the work using the same CC license.
ND (NoDerivatives)	You can make copies of the file and distribute it freely. However, the audio may not be altered in any way.
NC (NonCommerical)	This audio cannot be used commercially, under any circumstances.

Building a Custom Library

In addition to using licensed effects, a sound designer can also accumulate their own recorded material over many years, resulting in a personal library they can draw upon for different projects. If recording becomes a weekly habit, then with patience and persistence they will discover quite a bit of material has been recorded in a few years' time.

The Audio Asset List

Knowing where specific game audio is located can provide an oasis of calm during the heat of production. This applies to all sound effects, as well as to any music and dialogue for a project. During a fast-paced schedule, however, files are created but might never be catalogued, or they become catalogued but their status related to the game is not recorded. This is why an **audio asset list** is essential. It can be as detailed as needed, but I recommend starting simple.

Let's say your audio team is working on a game called **Cosmic Penguins: Operation Deep Freeze**. Ideally, the audio asset list should contain at least the following:

- **Title** of the list, including the name of the developer and the project ("*Antarctic Studios/Cosmic Penguins – Audio Asset List*")
- **Item name/label** for the audio (*penguin laser blast 1*)
- **Filename** (*cp_sfx_penguin_laser_01.wav* – project initials followed by audio type, in this case, sound effects)
- **Description** of the audio itself ("*a short pulse, high- to low-pitch sweep*")
- **Is it in the build, or repository?** (Using version control software or a singular file ***repo***sitory)
- **Status?** (Not started, in progress, placeholder, or completed)
- Has it been **integrated** into the game? This is when an asset is incorporated into the audio middleware and/or referenced somewhere in the game code.
- Has it been **tested** during gameplay**?** Is it **working** as planned?

The first few items can be set up in a spreadsheet such as Excel or Google Sheets:

	Antarctic Studios		
	Cosmic Penguins Audio Asset List		
	Item	*Filename*	*Description*
SOUND FX			
Weapons	Penguin laser blast 01	cp_sfx_penguin_laser_01.wav	A short pulse, high- to low-pitch sweep
	Penguin laser blast 02	cp_sfx_penguin_laser_02.wav	A short pulse, high- to low-pitch sweep
	Penguin laser blast 03	cp_sfx_penguin_laser_03.wav	A short pulse, high- to low-pitch sweep
	Penguin laser blast 04	cp_sfx_penguin_laser_04.wav	A short pulse, high- to low-pitch sweep

With additional status columns to the right of the Description field:

In Repo	*Status*	*Integrated*	*Tested*	*Works*
Y	Placeholder	Y	Y	Y
Y	Placeholder	Y	N	?
N	In progress	N	N	N
N	Not started	N	N	N

Note the naming convention of the file above: "cp" for our game, *Cosmic Penguins* (your studio may have more than one game, after all), "sfx" for sound effect, "laser" for the type of sound, and "01" representing the variation (we may need some slight variations of the laser sound, to avoid monotony). The **underscore** "_" characters glue the name together. Why bother with underscores? Because many game and audio engines (and repositories) require *no_spaces_within_filenames*, otherwise the way they parse the name will not work properly.

I strongly recommend keeping only ***one*** *copy of the list*, either in a repository or Google Sheet that can be updated and referenced by anyone on the team; at some point, another set of eyes will need to check it out as well.

The full sample audio asset list can be found on the **Companion Website**. It also includes dialogue lines (covered in Chapter 10) and production times for each item (covered in Chapter 11).

Preproduction

There are a few things to get in order before we begin!

Gather Materials and References

First, determine what equipment you'll need for sound design and the project in general. Do you have a dedicated computer, speakers, headphones, and any other necessary

hardware or software we covered in Chapter 4? Also determine what you have on hand for existing source material from libraries from your own previous work.

Also, are there **game design and visual references** for the sounds being created? Researching design specifications, concept art, mood boards, in-game 2D/3D art and any gameplay video captures are of tremendous benefit in creating the right sounds for the job. Even at this very early stage, working closely and communicating with the development team is essential. The best place to start is with the lead game designer, who can provide the necessary information to get started on sound design, as well as any additional personnel to talk with.

Recording Checklist

If a sound designer is creating original recordings, they must ensure that they have the right gear for the job: at least one microphone, and any accessories that go along with it. For example, if they are doing some field recording with a digital condenser mic, bringing along some extra batteries, media storage, and a windscreen is mandatory. If they are recording Foley, the necessary props should be located ahead of time, with a dedicated recording workspace, clearing away anything that might get damaged in the process. That wet pasta I mentioned earlier does *not* clean up well, especially if it accidentally flies onto the wall!

Production

Outdoors

Most mics can be fitted with a windsock, also known as a *dead cat* (we field recordists use such scientific terms!). For smaller recorders such as a Zoom H1n, a smaller version is needed, known as a *dead kitten*. I had purchased some of these for my game audio classes, and you should have seen the purchaser's reaction when I told them, "I'd like to order 12 dead kittens, please."

A sound designer must always be careful, and no situation can be taken for granted. Batteries could die unexpectedly; a data card to record on might have been forgotten; it could start raining when sunshine was expected; or, one may be chased by angry seagulls, as was the case with me. I was once doing some field recording at a commercial pier for a fishing game and saw a few seagulls just milling around the dock. I turned myself (and my recorder) toward them, and they walked toward me, so I walked a little bit away, but they kept walking faster and faster and the next thing I knew, I was RUNNING AWAY FROM THEM at top speed! Animals can be unpredictable, no matter how cute one thinks they are. It may also be tempting to provoke an animal to moo, bark, or meow, but field recordists out there, please be humane and just don't do it. If it's not meant to happen, there's always existing material out there to draw from.

Also, permission should obviously be obtained before recording on private property and preferably before arrival.

Indoors

Just as you would do outdoors, you should again get permission to record indoors, keeping equipment packed away until you need it. And a word of warning: many

Table 8.2 Foley Ideas

Footsteps on grass	Stepping on old magnetic cassette tape (if you can find some!), or a sod square
Footsteps on snow	Stepping on cornstarch in a leather pouch, or glove
Footsteps on grit	Stepping on coffee grounds on a hard surface, like wood or concrete
Gore, juicy impacts	Close-up mouth noises, squeezing a tomato, hands in wet pasta, a generous wad of hair gel, or Jell-O
Breaking bones	Crushing a pinecone, snapping celery or carrots
Flying insects	Tiny fan with foam blades, making contact with paper and slowly alternating fan angles
Horses	Clomping coconut halves on desired surface; for the reigns, manipulating a couple of belts or a keychain with your hands
Whooshes	Swinging a long stick or dowel quickly back-and-forth, or in a circle (careful not to hit the mic!)
Monsters	Human clicks, grunts, growls and screeches; scraping metal, rub an object (glass pane, drum head, etc.) to generate a squeak or groan
Body impacts	Hitting a very thick book, side of meat (with gloves) or punching the palm of your hand. Layer different impacts and add a low-frequency thud, like dropping a large stone on dirt or pre-made LFE sound, for full effect
Small animals or insects	Manipulating twigs on desired surface; use toothpicks for insect legs

digital recorders have the dubious distinction of looking like stun guns, so you should never, ever march up to a counter with one unannounced.

Foley-Specific Considerations

A dynamic mic is not going to cut it for Foley recording. A condenser mic will work much better at capturing sonic nuances, and it's these details that will help "sell" the Foley to the listener. A matching pair of condensers is sometimes used for a fully immersive stereo recording.

Try alternating takes, varying mic placement from close to medium range (a few inches versus 1–3 feet away). For footsteps, if you're not getting the performance to sound right, or you don't have enough space to walk on, try using them with your hands and perform them in-place. Table 8.2 lists a handful of Foley ideas that have proven useful in the studio.

Sonic Spotlight: Joanna Fang, Senior Foley Artist at Sony PlayStation

KZ: What is a typical day like as a Foley artist for games?

JF: I usually start my morning bright and early with about an hour at the gym working on cardio and weight training. Foley can be physically and mentally exhausting, it's important to keep a good regimen and to know one's limits. After my morning workout, I clock in and either have an early AM standup with relevant projects or begin planning my day with the Foley mixer. By 9:30 AM, we're off to the races going after footsteps, clothing movement, or props. We take breaks when we need to, reset

the stage, or simply stretch in between cues before having lunch. Returning to the studio, we touch base with any relevant questions that may have come up during the first half of the day before getting back into the flow and shooting out the remainder of the Foley cues. At the end of the day, I status report with our sound teams, adjust schedules, and preview assets for the next day of Foley. Out of the stage, I typically spend my evenings playing guitar or games. I typically sleep a full eight-hour cycle to keep my mind and ears refreshed for the next day of Foley.

KZ: Where do you find your inspiration?

JF: Inspiration for Foley comes from everywhere but more specifically, from other people. It's impossible to be a good Foley artist without being a great listener of sounds and stories that exist all around us. It's very critical as a Foley artist to listen for sounds that we take for granted or simply do not exist in our reality. There's an entire language of sound culture through the media we consume that shapes our perceptions of what props, footsteps, or clothing should sound like. However, forensically sampling reality never yields enjoyable results for a listening. Refining the way we communicate feeling, movement, and emotions to our audience requires being deeply inspired by the way we perceive experiences that transcend material reality. To that end, it is crucial especially as future game audio professionals, to consider that inspiration does not come from a 1:1 pairing of sonic realism to physical modeling. We can be effortlessly inspired and creative if we choose not to be myopic, we should all aim to expand our worldview through others. We should prioritize dropping needless, homogenous, and pedantic sound scoring with bold, flavorful, and meaningful expression.

KZ: What do you find the most challenging about designing Foley for interactive use?

JF: The biggest challenge that separates interactive Foley from linear Foley is quantizing the performance into granular chunks. Most Foley artists learn to walk gracefully and aim to flow from one sound to the next. However, interactive systems are looking for granularity, so finding ways to break apart the movement and mechanics of a Foley performance without compromising the sound takes precedence. To that end, learning a wide variety of footstep walking techniques and breaking down behavioral patterns into bite-sized chunks require a reimagining of a traditional Foley artists' instincts. The other fundamental challenge for gameplay Foley is to produce a variety of prop and footstep performances without creating a variant that pokes out. If a game is likely to loop and randomize ten footsteps, each step should sound unique enough for variety without any standing out to the point of annoyance.

KZ: What should an aspiring Foley artist do to begin their career?

JF: Aspiring Foley artists should try and get as much experience shooting and syncing to visual media as well as examining and immersing themselves in the sounds around them. Foley is an art form that relies on your body as an instrument. It pulls from theater, music, dance, and the fine arts. Any cross-disciplinary experience that can refine your skill in interpretation and execution of sound is a great way to build your fundamentals. As a career path, I highly recommend aspiring Foley artists to keep an open mind and try to tap out the rhythms of footsteps or the trickling of rain. There will always be more opportunities to work on and practice Foley for students' films and independent pictures. It's key to build a

vocabulary in a specific media before transitioning talents and skills into the interactive realm.

Postproduction

In a DAW, a sound effect can be combined with other audio elements in parallel to enhance it. For example, a troll's footstep may consist of a heavily compressed footstep sound that is pitched down an octave, and on track two, it is sweetened by the higher-frequency sound of small pebbles and debris being impacted. The two complement each other and make for a more audibly enjoyable sound. The overall length of the sound can be made shorter or longer with careful editing, and any additional effects processing applied before rendering. Below are some postproduction considerations for using dynamics, EQ, and various effects.

Managing Dynamics

Volume adjustments, compression, and/or normalization to files can be used to produce an acceptable dynamic range for sound effects.

Transient Stacking and Timing

Transients are short, high-amplitude peaks in an audio file or mix. It could be a door slam, bullet firing, or one footstep much louder than the others. If you have sound made up of multiple tracks in your DAW, look to see where the transients are throughout the mix. If there are too many stacked together, your ears will hear it as one transient, which may or may not be the intended result. And without dynamics control, this can cause unwanted clipping or distortion. A better approach is to stage transients, even if they are just ~10 ms apart. This reduces the chance for clipping and will help with clarity in this area of the mix.

Managing EQ

Frequency Slotting

Sounds for ambience, characters, objects, UI, and the like may all play simultaneously during gameplay. In this case, certain categories will share ranges of frequencies in the mix that might make discerning their content less clear. As frequencies pile up in the mix, they may also have an additive effect that doubles or even triples the mix volume within a certain range, causing clipping and distortion. Or, they may cause a subtractive effect, where frequencies start to cancel each other out.

It's impossible to know all cases for multiple sounds, but a common strategy is to take the most important categories of sound and "lock in" a fixed frequency range for them. This technique is known as **frequency slotting**. For instance, you may decide your UI sounds can do without low-end frequencies, so all of them might have a High-Pass filter applied before rendering. Or, a giant's footsteps may not need ultra-high frequencies but could keep a range between 20 and 5 kHz. Obviously, you should not overdo slotting for every category, but it pays off in the soundscape during gameplay

if you can isolate a few areas for processing. The mix will have more clarity, and players will most certainly benefit from the enhanced experience.

When editing in a DAW with some stacked tracks, you may run into a similar situation with too many of one frequency range piling up, or the mix just isn't coming together right. In this case, try making the track EQs mostly complimentary in nature, with some generally favoring the low frequencies, some mid, and some high, as needed.

Enhancement via Effects Processing

Enhancing sounds with special effects can be great fun, but be mindful of over-processing. When you've got more than a few plug-ins on a single track, or too much of any one plug-in, especially multiple reverbs or modulators, it might be time to rethink your strategy here.

Mastering

The last stage of audio processing is **mastering**. This is the final touch-up and polish given to an audio file, though it often involves the bulk processing of multiple files. Final volume balancing between sound effect files, dynamics control, EQ, and/or use of other plug-ins on a file all occur here, but nothing too dramatic should happen to it – otherwise, you should consider going back to your original DAW project and try remixing it there first.

There are two philosophies regarding where to perform mastering: some professionals prefer creating a separate DAW project and importing their near-final work into it, while others prefer adding mastering plug-ins directly into their working DAW project on the Master buss track, to make their work "mastering ready" when it's time to render it. Either way is acceptable, but you should try sticking to one method to avoid confusion when dealing with multiple projects.

On the **Companion Website**, you'll find some sound design exercises to try, which include adding dynamics control, EQ, and effects.

File Formats

Sound effects for games are normally rendered as WAV, AIFF, MP3, or OGG files. In Chapter 12, we'll see which files may be right for us, in terms of sampling rate, bit depth, number of channels, and in some cases, bit rate. We'll also take a look at what formats supported in leading game engines and audio middleware.

Testing Considerations

Placeholders: Temporary, Yet Critical!

Earlier in this chapter, I said that we should not add sounds haphazardly. Well, we can forgive ourselves when it comes to **placeholders**. These are temporary audio assets that are beneficial for position in a soundscape, in getting a sense of the overall mix – but also help to ensure that the implementation for those sounds is correct. Without placeholders, the risk remains of not having enough time to (1) iterate on sound creation

within the development cycle of the game and (2) test their implementation properly. Upon hearing placeholders for the first time in the game, they may not work as the sound designer first envisioned and may require an alternate method of implementation.

Programmers have deadlines like the rest of the team does, and plenty of their time for implementation and testing needs factoring in. Placeholders can be simple enough to create, and it may help to make them obvious. When I worked on Sierra's *Lords of Magic*, the placeholders we created for characters were employees' voices saying things like "walking, walking," "attacking," and "dying." They were not only funny to listen to, but they were also implemented early with their functions verified, and we were motivated to replace them quickly. I highly recommend making placeholders as early in development as possible.

Sound-Testing Methods

Testing can take place throughout the development cycle but especially when the game is getting close to release. **Test plans** are created that may involve playing the game as one "normally" would and listening for the expected sounds in the process. Testing may also include sounds when approaching the **limits** of the game, for example, the correct sound playing when inventory is maxed out, for attaining the highest possible score, for acquiring every single object in a level, or for combat sounds when having the most NPCs in your party. There is also **edge case** testing for sounds heard that are outside of normal play: casting a ranged spell beyond the furthest boundary of a map, crafting an item with obscure materials, or letting a level 1 wizard attack a level 99 arch demon in the final dungeon level. And of course, there is testing of all of the various components and systems of sound implementation, which may include things like ensuring that the various layers of sound (including all sound-based channels) are working properly; that ambience is triggered for all the right environments; and that sequences of sounds for animations are firing in the proper order, to name a few examples. We'll look at implementation in more detail in Chapter 12.

Sonic Spotlight: Trevor Dikes, Lead Sound Designer at Avalanche Studios Group

KZ: How important is original field recording to a game?

TD: Original field recording brings a uniqueness to a game's soundscape that can really help establish a game's audio identity, but people shouldn't feel like they are doing lesser work when using sound libraries, that's absolutely not the case. Sometimes schedules and budgets don't allow for field recording but when it does I think it can be a very valuable tool.

KZ: Any general advice when going out to record?

TD: Be prepared to go off plan at a moment's notice. You might go with the intention of recording specific sounds but on the day it isn't working out. Don't waste the trip, see what other interesting sounds you can get recordings of, they will always come in useful.

KZ: How do you audition the material once it's recorded?

TD: I try to do it while the recording session is still fresh in my mind. One thing that I find really helps is that most audio devices will allow you to add markers while

recording, I use that to mark moments of importance. I also take a photo of the recording location to add as a reference to the DAW session.

KZ: Is there a field recording you're especially proud of for a recent title?

TD: I've been able to get some really nice ambient recordings out in the beautiful nature in Sweden. I've spent days hiking through the nature reserves recording ambient sounds and once even came across a woodpecker pecking away at a tree.

KZ: Has there ever been a situation during field recording that's surprised you?

TD: During a winter in Sweden, a lot of the local lakes around my area were frozen over. I went out with my gear and came across a spot where the ice was starting to melt as the temperature was increasing and it was making these stress-cracking noises, they were really subtle but they sounded great.

KZ: What are some of the responsibilities of a lead audio designer?

TD: Helping drive and maintain the game's audio direction, critically assessing audio content, providing feedback and mentoring other sound designers, assessing technical requirements and potential risks are some of the key responsibilities.

KZ: What skills are most important to have?

TD: Aside from sound design and understanding audio implementation and middleware, communication is really important, as is scoping and prioritizing tasks, planning and understanding how other disciplines and teams work and how that aligns with audio are important skills to have.

Notes

1 John, Derek. "That Time NPR Turned 'Star Wars' into a Radio Drama – and It Actually Worked." *NPR*, December 18, 2015. https://www.npr.org/2015/12/18/460269884/that-time-npr-turned-star-wars-into-a-radio-drama-and-it-actually-worked.
2 "Limbo – Full Game Walkthrough (NO Deaths)." YouTube. Bolloxed, February 26, 2015. https://www.youtube.com/watch?v=1ie19_GXAAw.
3 lostchocolatelab. " 'Limbo' – Exclusive Interview with Martin Stig Andersen." Designing Sound: The Art and Technique of Sound Design, August 1, 2011. https://designingsound.org/2011/08/01/limbo-exclusive-interview-with-martin-stig-andersen/.
4 "Unpacking | Steam Full Gameplay (No Commentary)." YouTube. Unbuliebubble, November 21, 2021. https://youtu.be/7mYTFtQXNnY?t=58.
5 Notis, Ari. "Hit Puzzle Game Unpacking Features 14,000 (!) Audio Files Replicating Ordinary Sounds." Kotaku. Kotaku, November 4, 2021. https://kotaku.com/hit-puzzle-game-unpacking-features-14-000-audio-fil-1848000220.
6 "Cyberpunk 2077 | Night City Ambience 👀🎧 Rooftop Waterfront (4K Ultrawide)." YouTube. ASMR Destiny, December 13, 2020. https://www.youtube.com/watch?v=8ubB4AR8lUM.
7 "BIOSHOCK REMASTERED Full Game Walkthrough – No Commentary (#Bioshock Full Game) 2016." YouTube. RabidRetrospectGames, September 16, 2016. https://www.youtube.com/watch?v=nFjMkFwB1ck.
8 "Wii Sports Golf: 9 Holes – 22 (Theoretical Score)." YouTube. AwesomeGamerland, January 2, 2018. https://www.youtube.com/watch?v=cqGqNPpqhPo.
9 "Super Smash Bros Ultimate Gameplay – No Commentary." YouTube. Gameplay Cache, March 1, 2019. https://youtu.be/Z4ummET1dxw?t=216.
10 "DIABLO 3 ACT 1 PART 2 FEMALE WIZARD Walkthrough No Commentary." YouTube. OGV – OnlineGamersVault, November 20, 2017. https://youtu.be/4O4bmmmkMQI?t=446.
11 "Control Dev Diary 07 – Behind the Sounds." YouTube. Remedy Entertainment, May 24, 2019. https://www.youtube.com/watch?v=-eDkR2opC7Y, and "The Sound of Control." YouTube. In Development, January 27, 2021. https://www.youtube.com/watch?v=ojn-TOKyBTs.

12 "Skyrim – Longplay Main Quest Full Game Walkthrough (No Commentary)." YouTube. Loopy Longplays, November 24, 2019. https://youtu.be/nr62-GnrrOs?t=5189.
13 "Candy Crush Saga Gameplay No Commentary Levels 20–30." YouTube. Bazzy Gaming, May 4, 2020. https://youtu.be/eLSA-qPe7ig?t=731.
14 "Death Stranding – Opening Cinematic." YouTube. GameSpot Gameplay, November 7, 2019. https://youtu.be/h898oNXJ7Oo?t=136.
15 "Machinarium – Intro." YouTube. Brainstormando Canal, June 4, 2014. https://youtu.be/acB7RNYl-JA?t=7.
16 "Children of the Nile – Gameplay (PC/UHD)." YouTube. SergiuHellDragoonHQ, September 16, 2019. https://youtu.be/V6lAMhi7LXE?t=480.
17 "Minecraft – ALL Mob Sounds (1.7) [HD]." YouTube. ZoomBee, August 22, 2014. https://youtu.be/LR5FSDsx8fE?t=155.
18 "Geometry Wars 3: Dimensions Evolved – Playthrough/Longplay – (PC, PS3, PS4, Xbox 360, Xbox One.)." YouTube. 10 min Gameplay, November 22, 2020. https://youtu.be/E13_biUR0LY?t=7239.
19 "Thomas Was Alone Longplay [Full Playthrough, No Commentary, Original Audio]." YouTube. Retro Game Hacks, January 14, 2017. https://youtu.be/uMYgmpO-aoY?t=1887.
20 "MirrorMoon EP (Gameplay)." YouTube. RazrBit, September 16, 2013. https://youtu.be/cwgPn0EpA2M?t=33.
21 "12 Days of Marvel (Full Event) | Marvel Puzzle Quest." YouTube. Webb 859, January 4, 2020. https://youtu.be/d7gefotz5_U?t=1145.
22 "LittleBigPlanet | PS3 HD | Full Game Playthrough Walkthrough | No Commentary." YouTube. Full Playthroughs, December 29, 2015. https://youtu.be/Yyr5CS0DwHI?t=579.
23 "Subway Surfer Gameplay No Commentary." YouTube. January 24, 2018. https://www.youtube.com/watch?v=3jEpiw87n5U.
24 Walden, Jennifer. "Designing Deathloop's Grungy, Retro Sound with Audio Director Michel Trémouiller." A Sound Effect, December 28, 2021. https://www.asoundeffect.com/deathloop-sound/.
25 "DEBIRD: Bird Noise Removal Tool by Boom Library." BOOM Library, November 30, 2021. https://www.boomlibrary.com/sound-effects/debird/.
26 "The Secret to the Gruesome Sounds in Mortal Kombat Is Exploding Vegetables." YouTube. VICE News, August 20, 2019. https://www.youtube.com/watch?v=IYS0rPYjW28.
27 Ament, Vanessa Theme. *The Foley Grail: The Art of Performing Sound for Film, Games, and Animation*. New York, NY: Routledge, 2022.
28 Viers, Ric. *The Sound Effects Bible: How to Create and Record Hollywood Style Sound Effects*. Studio City, CA: Michael Wiese Productions, 2008.
29 License information can be found at https://creativecommons.org/licenses/ and www.audiocommons.org.

Bibliography

Ament, Vanessa Theme. *The Foley Grail: The Art of Performing Sound for Film, Games, and Animation*. New York, NY: Routledge, 2022.

Koskinen, Kari. "Creative Commons: What Licence to Use and When?" Audio Commons, January 4, 2019.

Marks, Aaron, and Jeannie Novak. *Game Development Essentials: Game Audio Development*. Clifton Park, NY: Delmar Learning, 2009.

Stevens, Richard, and Dave Raybould. *Game Audio Implementation: A Practical Guide to Using the Unreal Engine*. New York, NY: Focal Press, 2017.

Viers, Ric. *The Sound Effects Bible: How to Create and Record Hollywood Style Sound Effects*. Studio City, CA: Michael Wiese Productions, 2008.

Zdanowicz, Gina, and Spencer Bambrick. *The Game Audio Strategy Guide a Practical Course*. New York: Routledge, Taylor & Francis Group, 2020.

Chapter 9

Music

Music

The Power of a Great Score

Music has the power to influence our emotions and enhance our everyday lives. If a song we haven't heard in ages suddenly reaches our ears, it can transport us right back to the time and place we first connected with it – including the way we felt at that very moment so many years ago. Game music has a similar effect; it can immerse us in the virtual worlds we inhabit, entertain us, and even apprise us of our surroundings. And sometimes, as part of the very best soundscapes, it can make a connection to our very soul.

The first continuously playing game soundtrack was composed for Taito's *Space Invaders* in 1978, which is a repeating motif of four descending bass notes (quarter notes in D minor: D, C, B flat, and A). As you destroy the alien invaders, the music increases in tempo, continually building tension.[1] Today, there are still minimalist scores that celebrate an earlier gaming era, but there are also soundtracks lasting multiple hours, arranged and fully orchestrated for epic gaming experiences.

Think of the one game you played whose soundtrack influenced you in a meaningful way: a soundtrack you cannot live without hearing, one you crave listening to again and again, even outside of the game itself (maybe you even bought the limited edition vinyl!). It has lasting charm and personality, which transcends the game as well, becoming a part of you – that's the magic of a great game score (Figure 9.1).

The Roles of Game Music

A game composer has special creative challenges over traditional scoring: the music may also require many hours of lasting appeal, perhaps with modular sections that can be played back in multiple combinations. It may also require playback at a range of intensity levels, through altering the levels of its instrumentation in real time. No small order!

As seen below, there are many ways in which music can enhance the player experience, whether it be an underscore for a setup screen, or a visceral, dynamic soundtrack for an all-out battle.

DOI: 10.4324/9781003218821-12

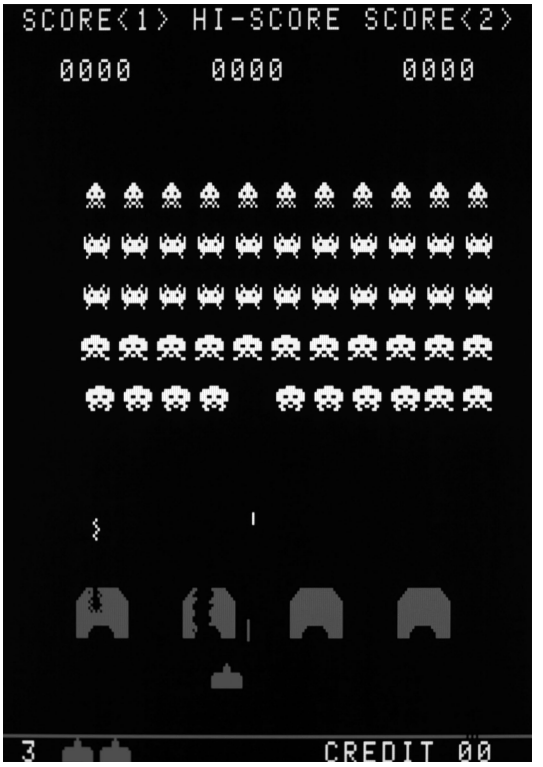


Figure 9.1 The first continuous music soundtrack dates back to Space Invaders from 1978 (Alamy Stock Photo/ArcadeImages).

Create an Immersive Setting and Mood

Right away, we have an opportunity for the soundtrack to immerse us, and tell us where and when our game is taking place. 1890s England? Proxima Centauri? The Jurassic Age? Music can help transport us there.

Game designers may want to convey awe, anticipation, fear, joy, or sorrow in a particular location or moment, further adding depth and immersion. The composer can write a piece that conveys a sense of place, but there is also the potential for music to elicit specific emotions from the player. The ethereal soundtrack in *Journey* pairs beautifully with the desert landscape you inhabit, conveying a sense of grandeur and mystery.[2] Musicians performing in a tavern from *The Elder Scrolls V: Skyrim* strengthen our sense of place in a medieval fantasy world (another mead, please!).[3] And Ellie and Joel playing guitar in *The Last of Us 2* create touching moments between the characters and story line.[4]

Provide Critical Feedback

Music can also help set the pace of a game, making any moment more relaxing, challenging, or downright stressful. The *Mario Kart* series is a great example of this,

where on the final lap, the music tempo increases dramatically, informing the player that the race is about to end (so they'd better be in first place soon).[5] Music can also change styles in an instant, to synchronize with the game mode or tension level: the player may be listening to the relaxed, ambient melody of a single woodwind, which suddenly changes to a dramatic theme complete with pounding taiko drums and vocal shouts – letting the player know a battle is about to begin. Or, the song's instrumentation can be varied: in the case of *Wii Play: Tanks!*, certain tanks that appear in a level can augment the game's main four-bar song loop by adding a metronome, cymbals, and/or timpani drums in multiple combinations. Though the song is lighthearted, the buildup of instruments matches the difficulty level, which helps to inform the player they'd better stay sharp![6]

Establish Your Brand

If the composer's job is to create a memorable theme for a game character, or for the game itself, they also have an opportunity to associate the theme with the studio's brand. Then, if a sequel or related project is in the works, the motifs can once more be revisited.

Over the years, game franchises like *Legend of Zelda* and *Super Mario Bros.* have updated their main themes with each new game release, whether the melodies appear in an opening sequence or are heard during gameplay.[7] A good theme engrains itself into the player's memory and can remind them of great gaming experiences to think back on for years to come. They will always associate it with the brand of your game franchise.

Deliver an Entertaining Experience

The composition of a piece itself can have entertaining moments, like a lightning-fast melody, pleasing chord changes, or even distorting the music in some way; if your car in *Mario Kart* is squashed or struck by lightning, the soundtrack also sounds "squashed" until you snap back to normal.[8] Inclusion of the player in a musical performance is also very enjoyable. In the pirate-themed action game *Sea of Thieves*, the player can pick up an instrument such as a hurdy-gurdy or concertina and begin singing a shanty, while nearby NPCs can join them in perfect sync (Figure 9.2).[9]

Game Components – A Closer Look

Having a variety of music should do more than just meet the game's basic requirements. The soundtrack should have a distinct purpose, hold up to multiple plays, and provide feedback if necessary. Below are the key areas where game music can have an impact.

Cutscenes

Logo/Brand Identity

This is usually a short tune accompanying the opening logo animation (or still image) faded up and out. This is also an opportunity to establish the brand of the game

Figure 9.2 When performing a shanty tune in Sea of Thieves, fellow NPCs can join in the fun with their own instruments, playing along in perfect sync (Microsoft Game Studios/Screenshot by Keith Zizza).

studio or the product itself. Sometimes a particular musical hook is all it takes, to connect to the studio's identity.

Game Introduction

Music can play a powerful part in establishing the game narrative. Opening music can help reinforce prominent events, introduce character themes, and set up motifs paired to the game world itself. It is usually part of an opening cutscene.

Transitional Moments

Transitional cutscenes often provide breadcrumbs for upcoming narrative events, and the music here can certainly emphasize these moments. In the city builder *Pharaoh*, for example, a new transitional cutscene is presented when the player reaches a new milestone in the ancient Egyptian era; the music plays a large part in illustrating the setting and scope of their empire.[10] Alternately, music can be part of a reward-based transitional sequence, providing a well-deserved and entertaining break in the action. The melodic pieces in the *Pac-Man* series of games are a classic example; they offer some lighthearted visuals of Pac-man and the ghosts, along with some musical ear candy to boot.[11]

End Sequences

Here is a chance to compose a grand win or lose finale, depending on the player's fate. The more interesting, the better! In the classic game *Oddworld: Abe's Oddysee*, Abe the Mudokon (the player) must free at least 50 of his brethren from the forced labor practices of the evil Glukkons; if so, the end sequence shows the freed Mudokons joining in a circle to summon a lightning spell, which destroys the Glukkon leader Molluck; Abe is then rescued, and the musical score is appropriately climactic and

triumphant. Otherwise, a cutscene is shown sending Abe into the meat-packing plant, *Soylent Green*-style! The music here is fittingly tense and malevolent.[12]

Setup

Typically, music for the main menu/setup screens is a looping tune, anywhere from 30 seconds to 2 minutes in length, with a steady pace and consistent mood. The tempo and tone of the menu music in *Saints Row* remains even-keeled in every respect (Eric B. & Rakim's legendary hip-hop song, *I Ain't No Joke*),[13] while the haunting menu tune from *Galactic Civilizations II* provides the feeling that you're about to enter a grand space opera, combining orchestral and synthesized instrumentation.[14] In any case, setup music should be interesting to listen to but never a distraction while preparing for the game.

Background/Standard Gameplay

This category of music is the most important one by far, as the player will spend the most time listening to it. Some games have a single tune that plays in a loop, while others incorporate custom programming or middleware to make the music change in different environments. As we will see later in the Nonlinear Music Systems section, there are a variety of ways to extend the music's "shelf life," including the preservation of its transparency to the player.

It is often subtle and unnoticed, yet is always helping to set the mood, scene, or the player's current situation. You can think of background music as an "underlayment" to the rest of the soundscape: it is providing atmosphere, interest, and depth, while at the same time never drawing much attention to itself. Still, a driving beat can achieve an ambient effect if it's relatively consistent; most of the pieces I composed for Tilted Mill's *Caesar IV* soundtrack are built as light orchestral arrangements, but I often included a driving snare and bass drum as a reminder that the game is set in ancient Rome after all, and that you are ever-slowly but surely building an empire.[15]

Battle/Challenge

Upbeat or prominent music can scream to the player "get busy!" in the more tense moments of gameplay. Pulse-pounding rhythms and distinct melodies can work here, along with a louder overall volume. This music is frequently broken into smaller segments, in order to serve up the correct ones to play for variety and feedback purposes. In Sierra's *Lords of Magic*, I broke the battle music up into many loopable segments, on an evolving scale between light and dark factions. Whomever was winning the battle at any moment had their music playing, with increasingly dramatic segments playing the closer they were to winning.[16]

Characters

Heroes and villains can all benefit from having a theme. From the *Star Wars* movies came the themes for Luke Skywalker, Princess Leia, and Darth Vader, and from the game world came the themes for Princess Peach (Mario), Link (Zelda), Crono

(Chrono Trigger), Professor Layton, Nate (Uncharted), Guile (Street Fighter), Sephiroth (Final Fantasy), and countless others. Good themes are not just memorable but can also be shaped into different arrangements and keys (major versus minor, for example) as characters find themselves in different situations.

Stingers

These are short musical bursts that provide immediate feedback in the form of a bonus score, trap, or other important events. They are often catalogued as sound effects, depending upon the nature of their use. One of the most famous is the alert stinger from the iconic stealth game *Metal Gear Solid*, heard when the player is accidentally seen by the enemy. After it plays, a longer, looping alert theme follows.[17]

Win and Lose Cues

As part of winning a battle, leveling up, or reaching a new milestone, a victorious tune may play. Conversely, if we are defeated or time runs out, a tune signaling our loss may be cued up instead. These are of higher significance than the stingers previously described. In either situation, win and lose cues are quite short, typically only a few seconds in length.

Linear Music Systems

The linear method of music playback is straightforward: tunes may play in a linear song list ala "jukebox" mode, from first song to last. They may also be randomized, where any tune can play in any order, including repeats. Or they can be shuffled, where tunes are randomly chosen but a unique one is always selected, until the list is exhausted and then the list is shuffled again. Each tune can also be looped (e.g., the song for mission 58 could loop until the player starts mission 59). Music can also change when the player enters a new game mode: for example, different music for the main menu, gameplay, win and lose screens, and so on. But the key takeaway here is that each tune sounds the same way every time the game is played.

Nonlinear Music Systems

Interactive versus Adaptive Music

There are a variety of ways to describe how nonlinear music unfolds during gameplay. For our purposes, we can divide it into two main types: **interactive music** and **adaptive music**. These terms are still in a state of flux, with game developers, music software and hardware developers, and authors sometimes interchanging or even combining the terms. For our purposes here, though, we will call music that relies primarily upon direct user input ***interactive***, and music that changes with indirect or no user input ***adaptive***. Either can be considered **dynamic music**, i.e., music that changes its level of instrumentation, mood, and/or tension level, throughout gameplay.

We can think of interactive music as changing based *directly* through player input. Take, for instance, the player unlocking a door on a dungeon level, which leads to a

descending staircase. They take the stairs via pressing a button on their game controller, and the music transitions into a more intense version of the current song. The player has altered the music directly by providing input to the game: pressing a button to move from one area to the next.

Adaptive music is music that changes, or *adapts*, to the current game state. It is based on *indirect* decisions (or even no decision at all) made by the player. For instance, ambient music may be heard when the player is in a jungle habitat. They are simply standing in place, when suddenly, a panther leaps onto them from above! The music adapts to the situation, now quickly changing pace from ambient to high energy. The player just-so-happened to be in a dangerous part of the map, where there was a greater random chance of predator attacks. So, the player has changed the music indirectly by their mere presence, in this case. They didn't have to know this part of the jungle was more dangerous, and the attack was determined by a virtual dice roll.

These systems allow for the soundtrack to play out in various ways, creating greater replayability, and keeping the music interesting and fun to listen to for the player.

There are certainly times when music can fall somewhere *between* being interactive or adaptive which should be determined on a case-by-case basis.

Horizontal Resequencing

Within dynamic systems, there may be a need to segue from one type of music to another, based on gameplay events or user input. For example, ambient music switching to battle music, then switching back to ambient music when the battle ends. This technique is known as **horizontal resequencing**.

Here, the music is broken up into reasonably short **segments** (as WAV, OGG, or MIDI files) stitched together at the correct times during gameplay. The segments can either be crossfaded or bumped up against one another. The entire score can also exist as one large file with several location markers, so specific points can be accessed programmatically. Markers can also denote the start and end times of each section of the music, whether it be for environmental ambience, combat, and so on – or as loop points to continually play one region of music until the next change is needed (Figure 9.3).

Horizontal resequencing is commonly used in many games. In *Chrono Trigger*, a peaceful trip into a forest can quickly turn deadly. The player normally hears a pleasant underscore, which changes to high-energy battle music when encountering enemies. Once the scoundrels are defeated, the soundtrack returns again to the underscore (Figure 9.4).[18]

In *Jetpack Joyride*, acquiring a vehicle temporarily changes the music from a side-scrolling action loop to an even more intense song, which is written to complement the unique vehicle's personality. The vehicle music segment has an introduction, followed by another segment that continually plays in a loop (thanks to a marker saved in the WAV file) until you crash the vehicle. The original side-scrolling music segment then resumes after a brief fade in.

Resequencing Tips

The segments should have **tails** wherever possible, meaning any finishing moments that continue out after the last beat in a measure. This includes elements such as an

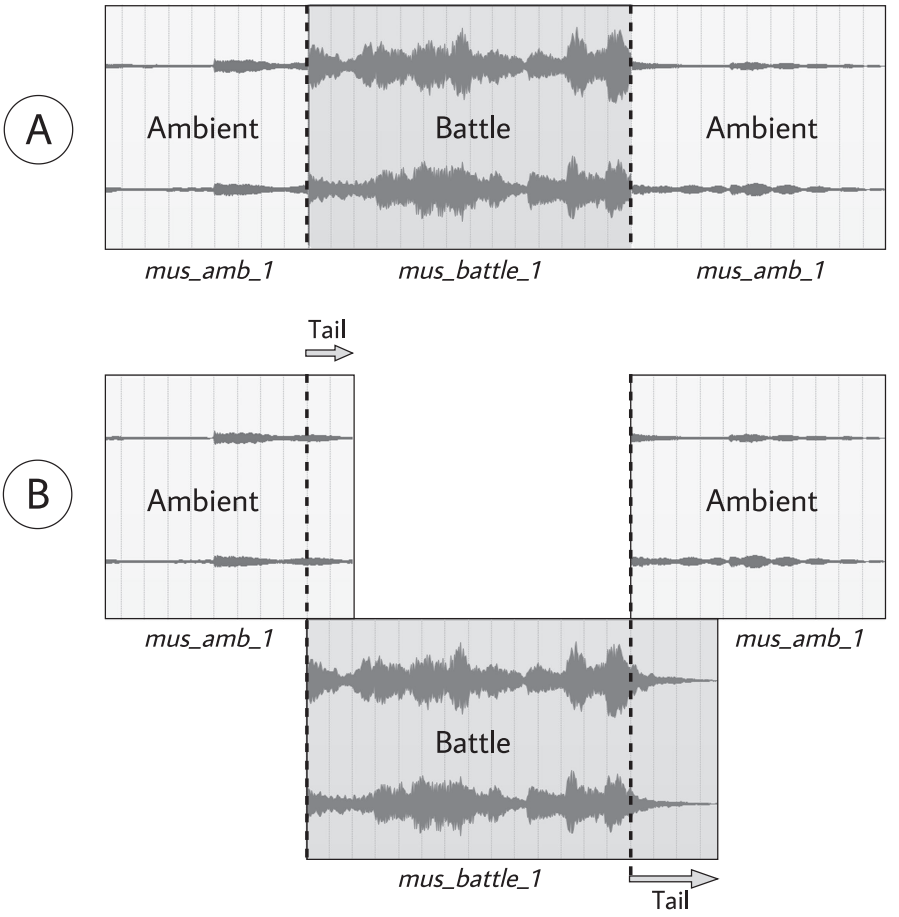


Figure 9.3 Examples of horizontal resequencing, where (a) segments are bumped up against one another seamlessly, and (b) segment tails are allowed to play out after the next music changes for a smoother listening experience.

Figure 9.4 Horizontal resequencing is an important part of the game experience in Chrono Trigger (Square/Screenshot by Keith Zizza).

ending cymbal flourish, a string instrument's release envelope, or letting the segment's **reverb** ride out at the end (aka a **reverb tail**), to help make the changes between segments feel less abrupt, as in Figure 9.3b.

It is also common practice to provide aesthetic variations between music modes (e.g., battle1a, 1b, and 1c); this helps to avoid over-repetition of looping sections.

Fades, Crossfades, and Transitions

One benefit of horizontal resequencing is that there is no need to match tempos between one segment and another. However, you must decide whether or not the music changes immediately when one game mode ends and another begins. A quick **fade out/fade in** between segments is a possibility, or you can try a **crossfade** of the two for immediate results. You can also choose to let the segments play out in their entirety, in which case, they should be very short – so as to line up faster with game events unfolding.

Sometimes timing the music to precisely match all game modes may sound awkward: for example, gentle ambient music that instantly jumps to aggressive battle music, or crossfades that musically seem odd. Another solution is to insert **transitional music** between sudden changes. These are very short segments of no more than a few seconds in length. If the transitions still sound awkward, try synchronizing the music changes to the measure, or at least to the beat (Figure 9.5).

Tempo Matching

Of course, horizontal resequencing allows for tempo matching as well. Here, letting the end of segments ride out is just as important, in order to sustain continuity between them. Crossfades may be too awkward here, but certainly tempo-matched transitions can work well.

Vertical Reorchestration, and Stems

In the editing world of music, a song is normally broken up into **stems**, which are groups of tracks mixed into a smaller subset of tracks. This makes editing life easier,

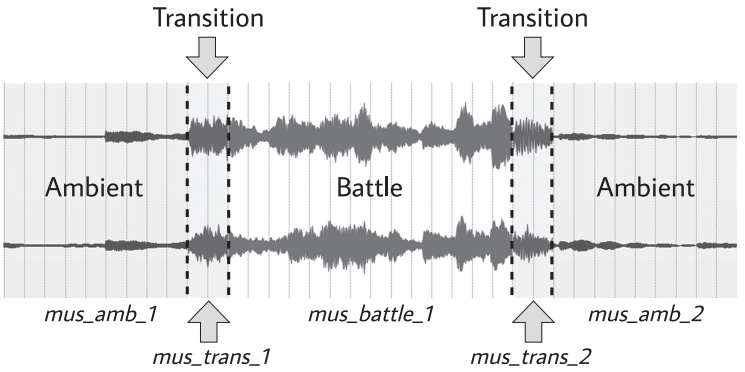


Figure 9.5 Horizontal resequencing, with transitional music in between major music changes: from Ambient → Transition 1 → Battle → Transition 2 → to Ambient once again.

offering greater ease of control over the entire song mix but also control for in-game use, as we will see in just a moment. Sometimes a large number of tracks become a stem (strings, for example), or just a single track that becomes its own stem, such as a bass guitar. In other situations, audio for a cutscene may make up three stems: music, sound effects, and dialogue. Just like a buss (Folder) in REAPER, changing volume, EQ, or other DSP in the buss will change all tracks that are part of it, as shown in Figure 9.6.

Imagine this situation: you have just composed a fantastic three-minute looping song, but you've discovered it's too heavily orchestrated for the more relaxed moments in gameplay and you've run out of time to create more music. What can you do?

If we break the song down into a few stems: drums, percussion, bass, chords, and melody, as in shown Figure 9.7 – and play them all simultaneously – your music loop still sounds the same, but by adding or removing different stems during playback (e.g., by soloing and muting them, or fading them in and out), you can create moments of tranquility, tension, or excitement. Suddenly, your three-minute song has

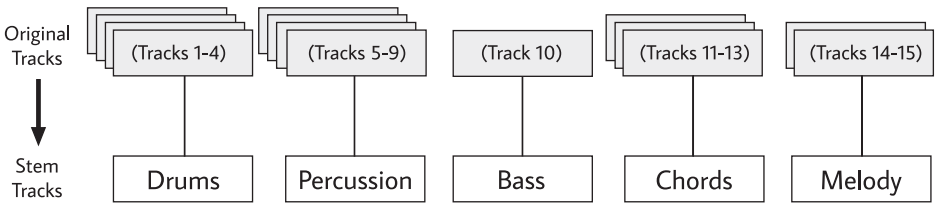


Figure 9.6 Individual tracks (top) can be organized into groups known as stems (bottom), making mixes easier to control.

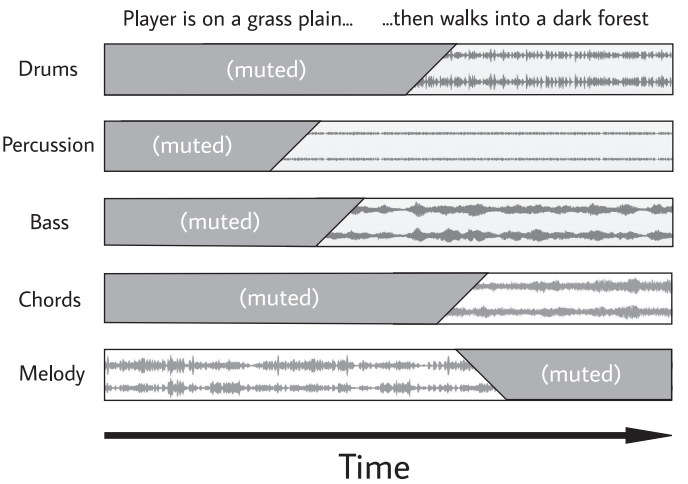


Figure 9.7 Using vertical reorchestration, stems can be turned off and on as needed, to control the soundtrack's instrumentation and tension level. In this example, stem volumes are gradually faded in and out for a smooth, consistent listening experience.

been stretched to more than an hour of unique music! This technique is called **vertical reorchestration** (also known as **vertical layering**).

In the sci-fi classic *Portal 2*, the soundtrack's layers are added and removed depending on how you solve a particular puzzle. In an interview with the game's composer Mike Morasky, he says that the number of simultaneous instruments heard depends on which portions of the puzzle you've solved in a particular space, "with each additional piece of music actually coming from the device that is participating in the activated game play mechanic." This helps to give the player a sense of success but also in control of the music itself. Morasky goes on to say, "Most of the interactive music is also positional so that as you move through the space you also change the mix and volume of the music you are hearing, which invites explorations of the space as well."[19]

Reorchestration Tips

A good rule of thumb is to try to make the overall composition feel natural on its own. An easy trap is to over-compose each part. Instead, compose the entire score in one sequence first, as a whole piece and then isolate the segments needed and render those out in stems (usually as WAV or AIFF files), muting and soloing them to understand how various stem combinations work together.

Also, start with the fewest number of stems possible to reorchestrate. You can always expand upon them later, but a lower number is faster to produce, and in the game will take up less space and fewer CPU cycles.

Blended Techniques

Often times, using horizontal and vertical techniques together is used, for maximum flexibility in driving the gameplay and narrative. In *Sackboy: A Big Adventure*, the eponymous protagonist is trying to stop a villain from turning Craftworld into a dark dystopia. As Sackboy travels from one level to the next, the music adapts to fit the surroundings (a jungle, a sci-fi realm, etc.). Horizontal resequencing is used to change the mood (e.g., a sunny day that turns into a rainy one), or to seamlessly transition to a final resolution at the end of a level. Vertical reorchestration is used to create aesthetic variation in the soundtrack but also to increase tension at certain moments.[20]

Visit the **Companion Website** and try your hand at some horizontal and vertical techniques!

Generative Music

At a much higher level of complexity than horizontal or vertical techniques, **generative music** (also known as **algorithmic music**) is procedurally generated and can be based on a multitude of game parameters. It does not necessarily require player input. Game states, player health, location, elapsed time on level, or inventory capacity are a few elements that may determine how the music unfolds. But no matter what the constraints are, the soundtrack will never quite sound the same between any two plays. This process can greatly extend the original plan for compositional ideas. If you wrote

Figure 9.8 Adventuring to the beat in Crypt of the Necrodancer (Brace Yourself Games/Screenshot by Keith Zizza).

music for an interplanetary exploration game, for instance, you may toil over creating a one-hour soundtrack, and perhaps work up an even greater sweat writing two hours' worth of music. Maybe your soundtrack works for a couple of planets – but what about the remaining 18 *quintillion* ones? For *No Man's Sky*, the band 65daysofstatic worked with the game's audio director to create generative music, which the band described as "atomized and rendered in a great number of variations" and determined algorithmically in the game, just as the game's worlds and its inhabitants are.[21]

Rhythm-Based Approaches

Games that drive player actions to the beat of the music may not employ any of the aforementioned techniques. One example is *Beat Saber*, where songs are comprised of neon blocks that travel toward the player, and they must slash them with a pair of red-and-blue sabers to the song's beat. In another notable example, the retro-inspired *Crypt of the Necrodancer* has the player exploring a tomb; in order to move or attack an enemy, the player must do so precisely to the beat of the music, or suffer negative outcomes. As the tempo increases and enemies mount, decisions become more challenging to make (Figure 9.8).[22]

Preproduction Considerations

While we'd love to have untold hours of original music for our next game, the scope of music production needs to be determined before getting underway. This first includes the amount of time allocated to create the complete soundtrack. As we have seen, this can be lengthened by using horizontal and vertical techniques. However, that work by itself will take considerable time to get sounding right: the music must retain a high level of quality from one mood to another, or when stems must be removed or added. Composers must also determine whether or not they are the solo artist on the project, or if they will be collaborating with other musicians. Additionally, they must decide upon using either real or virtual instrumentation (or both).

Original versus Licensed Music

There are times when original music is required for a game and/or licensed music. The difference is that original music is created by an in-house or freelance composer, versus licensed music where negotiations are made (e.g., a licensing fee from the composer, group, or music publisher) to add songs that have already been written, often times from a well-known individual or group. Occasionally, both types of music will exist in a game: for example, the *Grand Theft Auto* series famously incorporated original and licensed music, the latter heard by tuning into various radio stations in your vehicle.[23]

Deciding on Composition Method

Composers have many ways to score videogame music. This includes live performances of real or virtual instruments, either using a MIDI sequencer or a digital multitrack recorder.

Creating MIDI-based compositions can be done in several ways: by step sequencing (manually adding notes, duration, and other MIDI information in a sequence, which can be painstakingly slow); by notation-based composition using tools such as Finale, Sibelius, or MuseScore (the latter is a free, open-source option)[24]; or by recording a live performance in a sequencer, as played on a MIDI instrument. The process of a live MIDI performance also includes going back and fixing mistakes (eliminating a wrong note, for example), redoing specific sections of a performance, manually inserting new notes, or recording additional MIDI data such as volume or panning, to name a few items (Figure 9.9). Having this flexibility is the beauty of composing with MIDI!

Internal versus External Studios

If the composer is handling all the music and production by themselves, great! They can stay in their project studio and create everything in one place, which gives them

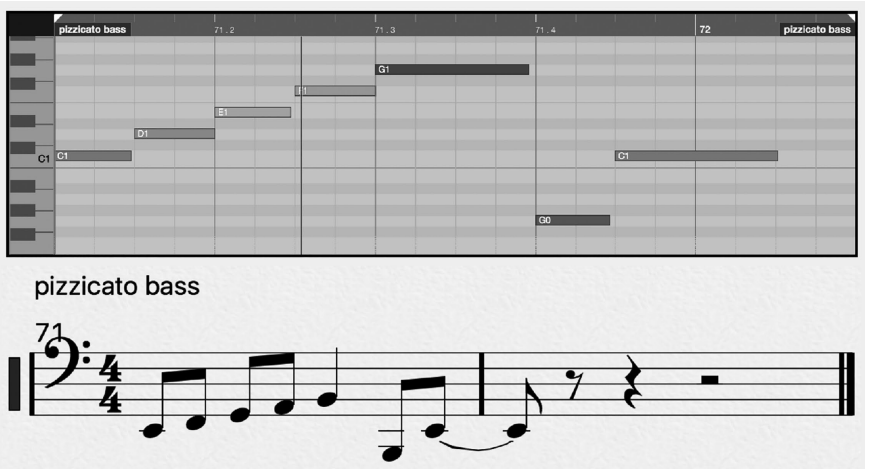


Figure 9.9 MIDI sequence data represented in two ways: in "piano roll" mode (top) and the MIDI notation equivalent (bottom) (Cubase by Steinberg/Screenshot excerpts by Keith Zizza).

total control and convenience. If performers are required outside of their own studio, the performers' fees have to be factored in, plus the recording studio time. Some commercial studios charge hourly, half-day, or full-day rates, and they may also have a separate fee for a studio engineer, which will be handling the setup and recording of the performers. It will likely be more expensive, but the benefits include recording in an exceptional acoustic space, access to the highest-quality equipment, and working with accomplished studio personnel.

Deciding on Instrumentation

Which instruments are needed for a game soundtrack? It could be an all-electronic score for a science fiction adventure, or perhaps an orchestral ensemble for a medieval fantasy epic. In many cases, the instrumental arrangement is dictated by the genre of game. For example, a game about combat in feudal Japan will require instruments like the Koto (plucked strings) and Shakuhachi (a woodwind instrument) to lend realism and immersion to the soundtrack.

Real versus Virtual Instruments

Whether or not real or virtual instruments are chosen depends on budget as well as any time constraints. Studio musicians usually charge an hourly or day rate, and these costs can add up fast as more musicians are required. In a fast-paced development cycle, the score may be written and performed by a single composer using VIs, which eliminates studio rental time and additional performer's fees. The trade-off is the VIs can't exactly match the quality and nuances of a real performer. Some VIs sound more convincing than others; a piano is much easier to sample into VIs than say, a saxophone or acoustic guitar. The latter two have many more articulations and tones that cannot be easily replicated in a MIDI performance.

Hybrid Approaches

Some soundtracks contain a blend of MIDI and live performers. This method can save a considerable bit of time and money and still have a realistic quality. Even the addition of a single expressive, realistic instrument can make the mix come alive. This was achieved in the remix of the *Pharaoh* soundtrack I originally created with Henry Beckett back in 1999. In *Pharaoh: A New Era*, composer and arranger Louis Godart expertly transcribed the soundtrack and updated the MIDI instrumentation with new virtual instruments. He also very thoughtfully replaced one or two MIDI instruments from each piece with authentic ones, such as a violin, duduk (a woodwind instrument), qanun (a plucked string instrument), and live percussion.

Spotting Sessions

Before production begins, the composer often has a **spotting session**, where they confer with the game designer (and other relevant team leads), regarding the creative direction and the various types of music needed for the project. In addition to examining concept art and the overarching game design, the composer can also play

through any available game cutscenes, levels and modes, or review video captures of the game-in-progress. From there, they can begin to locate the appropriate musical references, listening to relevant material from similar games, and also any related material from albums, movies, and other media. They can also get a sense of the project scope, including estimates of the individual pieces needed, potential soundtrack length, and the musical palette itself.

Production Process

Iterative Production

Now that the designer has imparted their vision for the music, the composer gets to work right away, sketching out rough musical ideas for review and further exploration, as time permits. First, drafts of the setup and gameplay music can be added to the game-in-progress. These may be roughed out as temporary MIDI sequences (known as a **temp tracks**) using a single instrument, such as a piano or a string; this is just to get the right motifs set up, and basic song structure. The music will undoubtedly evolve with the project as a whole, with the temp track removed early on in production, and new, more final layers built over time.

Music for setup, gameplay, stingers, and win/lose events can follow the above process; however, cutscenes demand a special iterative process, as seen later in this section.

From MIDI to Live Orchestra

In the case of requiring live orchestral music, the composer will score everything with virtual instruments in a MIDI sequencer first as a mockup, then an **orchestrator** can be hired to translate those MIDI compositions into a suitable format for live performers. The orchestrator takes MIDI sequence data and converts it into **score sheets** for an individual performer, ensemble, choir, or full orchestra. The score sheet shows the performer's sheet music along with that of the accompanying performers.

As you can imagine, the budget to hire an orchestra is quite expensive, not just for compensating the musicians but again for the studio and engineering time. With "AAA" titles (i.e., major publishers with high budgets), however, many performers can be brought into the project. For instance, Composer Bear McCreary employed nearly 150 orchestra musicians and vocalists for 2018's *God of War* soundtrack, which yielded stunning results.[25]

Scoring in the Project Studio

Setup, Gameplay, and Stingers

When recording live instruments, a software- or hardware-based DAW is used; mixing can be done on-the-fly or saved after all tracks have been recorded. Usually, a preliminary mix is set up, just to get each track volume and EQ set to an approximate position.

As mentioned earlier, it's not uncommon to have a blend of real and virtual (MIDI) instruments together, and most DAWs can easily accommodate this setup.

Transposition and Tempo Changes

Sometimes, a song may not feel right for the game; aesthetically, it may seem to fall flat, or not "gel" during gameplay. If the instrumentation and song structure are OK, then **transposing** the song may be an option. This is a feature in MIDI sequencing where the key signature of the entire song can be instantly changed, for example, from C major to E major, or from A flat minor to B major. Another practice is to change the tempo of the song. Sometimes a change of **beats per minute** (**BPM**) can make all the difference in connecting with gameplay.

Tempo Mapping

A particular piece may require different tempos as dramatic moments unfold. These moments can either be recorded separately, then joined together later in a DAW, or they can be recorded as one contiguous MIDI sequence. In the latter case, **tempo mapping** can be used. This is a feature found in most sequencers that allows for multiple tempo changes during certain measures or times. Tempo changes can be added manually as numeric data or drawn as curves or lines in a tempo-based graph over time.

Scoring to Cutscenes

If a composer had to wait until a cutscene was finalized before they could score it, there would never be enough time to complete it. Instead, they begin right away, first examining the art team's **storyboards** – a set of sketches that illustrate the movement of each scene, which are eventually joined together in a video to demonstrate timing and overall flow. The art team then adds or removes scenes, and perhaps adjusts the timing in certain areas. At this point, the composer can begin sketching out the music, either just with a single MIDI instrument such as a piano or string, then later filling in the gaps with the proper musical palette and instrumentation.

Next, the art team creates an **animatic**, which is a rendered, low-poly version of the storyboard video decided upon earlier. Only a minimal amount of effort is invested in the animatic, to ensure the flow and pacing of the cutscene works. The composer can benefit by looking at the **shot list**, which is a list of scenes to be rendered in the animatic. There may be further adjustments to adding and removing scenes, as well as adjusting any scene timings. Team feedback is extremely beneficial at this stage.

The composer may see, for example, that one mood needs to be created from scenes one to three in the shot list, and another mood for scenes four and five, etc. They can further develop any initial ideas into a more cohesive musical palette, making instrumentation choices that are close to final.

Next comes the final rendering of the cutscene and several iterative passes at the music, getting the scene timings just right and obtaining any end-stage feedback from the team.

Obtaining Useful Feedback

"Too many notes!" said the emperor to Mozart in the movie *Amadeus*. Ah, to receive constructive feedback! Within the full spectrum of audio disciplines, I have found

music to be the most subjective, with people having the strongest opinions about what the "best" music is for the game.

More often than not, the team's feedback will not be what the composer hopes for. Of course, this is because not everyone is going to understand musical terminology, let alone what the music production process entails. Common non-feedback includes: "I don't know, I just don't like it"; "it sounds… off"; and the facepalm-worthy "I'll know it's right when I hear it."

Rather than get upset or cast blame on someone else, it's the composer's job to facilitate better communication. They can improve the quality of feedback by requesting specific music references, as mentioned earlier (movies or music examples from YouTube, Spotify, etc.), and to also ask what the team likes about the music that has been already created for the game, if any. Sometimes, it turns out that just one specific instrument is providing an interesting texture to the music, or that a particular style, melody, or tempo is appealing.

Copying the Impossible: The "Steeping Problem"

One common, unfortunate situation is that developers will put a temporary piece of music in the game, which lingers there for several weeks or even months. When the composer comes along to replace it, there is a sense that their work isn't living up to the temp track, no matter how effective the work really is. I've come to call this the **steeping problem**: like tea in hot water, the team has become *so* accustomed to the temp music that it's become inseparably steeped into the gameplay experience; any deviation from the way its dynamics, chord changes, or melody feels against the game sounds off the mark. This can be a hard problem to break through! But before you go up against something like *Temp_Music_Erase_Later_1.mp3*, some polite discourse should take place between you and your team regarding this very phenomenon.

Postproduction

Mixing and Editing

Here, you are often working with a much larger project than with sound design. Music is normally broken down into stems (similar to vertical reorchestration), and individual tracks are first volume balanced, followed by EQ or compression, then any spatial processing such as panning, reverb, or modulation effects. But we're not quite done yet! Render your mix and shelve it until you're ready for the **mastering** process.

Mastering

Just as we did with sound design, mastering is the final polish of our mix. In addition to this, music must be mastered considering *all* songs in the game. This often entails careful adjustments between song volume levels, EQs, and compression/limiting levels.

Here are two quick tips for mastering music: first, set up a new DAW project, importing rendered music files from your past projects (known as **premasters**) and master them all in one place. Second, if you feel like the mastering process is overwhelming,

try collaborating with another audio-centric person whose ears you trust! They may hear peculiarities in the mix you've overlooked or have become used to hearing.

Rendering Music Files

Once mastering is complete, songs can be saved as uncompressed WAV or AIFF files, then rendered later into compressed formats if necessary, such as MP3 or OGG.

Save Your Stems Now, but Also for Later

As with any audio content you create, it's vital to save your work for future access. However, unless your game requires vertical layering, you may have ignored saving (rendering) your music stems. Your sequencer or notation files are saved somewhere, which is great, until your virtual instruments don't work on your new PC anymore. Or the virtual instruments got taken off the drive accidentally. Or your computer had a meltdown. Or you've had to switch from PC to Mac (or vice versa), or a different operating system, and the sequencer files are now incompatible. Yikes!

Saving stems allows you to have master copies of the songs' main instrument groups that you can bring into any DAW for mixing, editing, or rendering anytime you like.

Testing Music

As with sound design, extensive testing is required to ensure music is working in all the right places. On the aesthetic side, for example, there is the question of continuity: Do the songs fit well together as a collection? Are they placed in good locations on the game level? Do they set the mood, and serve the gameplay and visuals well? Have they been mastered to your satisfaction? Does the game require more music? Longer or shorter lengths? and so on.

Then there is the technical side: Are the songs functioning properly in-game? Any horizontal resequencing or vertical reorchestration systems to check? Are they using up too much memory or CPU cycles? Any fade in/out/crossfading systems that need to be verified? Remember that more ears on the soundscape is important, so be sure to get external feedback early and often.

<Sonic Spotlight: Wilbert Roget, II, Composer>

KZ: How do you approach composing for a game like *Call of Duty: WWII* or *Mortal Kombat 11*? Is there normally a lot of back-and-forth with the development team during music production?

WR: We usually begin by defining the overall sound of the game – the composer is typically asked to write a main theme which guides the tone, instrumentation, and motivic basis of the rest of the score. On *Call of Duty: WWII*, for instance, we spent a full month iterating on the main theme before moving on to the in-game score. In cases like *Mortal Kombat 11*, I was commissioned along with several others to write a custom demo piece – once they hired me, this piece became the main theme as well as an index for a few other character themes.

As I continue writing the score, I try to reference as much of the in-game art, story, gameplay, and dramatic themes as possible. The developers review every piece and send their feedback, which can range from one track being immediately approved, to another requiring countless revisions and rewrites before everyone is happy. One of the biggest challenges in scoring games is this iteration – it can be heartbreaking when a composer works very hard on a cue, following directions precisely, only to have it bounced back for unpredictable reasons. It's important to remember that most of our favorite scores likely followed this pattern as well, with their composers handling the situation while remaining creative and motivated throughout the process.

KZ: Do specific tools matter as part of your composition process?

WR: Game composition is an incredibly competitive field, and to be successful, one needs to seek out every advantage possible while developing a unique personal style. We have to strike a balance between mastery of a specific style or field, and being versatile enough to capitalize on opportunities when they arise. For some this means learning sound design or synthesis, learning one specific DAW or another, learning to record and program your own sample libraries, or learning in-game audio tools, game engine scripting, possibly even coding – I've seen all these skill sets lead to work in this field. All of these skills require tools in the form of software or hardware, and it's up to you to decide which appeal to you enough to invest in them. Generally speaking, expertise in a few tools is more advantageous than simply having experience in many.

Currently, the DAW doesn't seem to matter, although sound designers are gravitating more and more toward Reaper for its flexibility, audio editing, and scripting capabilities. As a composer, I use Reaper for similar reasons, as much of my writing process involves audio editing and musical sound design. I use Native Instruments Kontakt as my main sampler and develop my own custom sample libraries for almost every score. I also frequently use hardware synthesizers in my musical sound design, both for their sonic power and depth as well as the workflow and mindset involved. And I have a few mics set up at all times so that I can quickly record an instrument, percussive Foley, my own voice, or anything else that comes to mind. Lastly, in terms of plug-ins, I have many different software synths as well as unusual sound-mangling tools, granular synthesis tools, and the standard glut of way-too-many reverbs, EQs, compressors, and other mix tools. These days there are astonishingly many free or cheap VST plug-ins that can easily compete with the most expensive gear out there!

Notes

1 https://www.museumofplay.org/games/space-invaders/.
2 "Journey – Gameplay/Playthrough (No Commentary)." YouTube. IAmSp00n, March 21, 2012. https://youtu.be/bkL94nKSd2M?t=888.
3 "Sleeping Giant Inn | Skyrim Ambience and Music | 1 hour." YouTube. Ambient Being, September 13, 2019. https://www.youtube.com/watch?v=F1WgUuTghEU.
4 "Ashley Johnson Sings Take on Me | The Last of Us Part 2 Behind the Scene." YouTube. Kingusureiyā, July 3, 2020. https://youtu.be/joXNz0RCXCM.
5 "MARIO KART 8 DELUXE Full Game Walkthrough – No Commentary (Mario Kart 8 Deluxe Full Game All Cups)." YouTube. RabidRetrospectGames, August 4, 2020. https://youtu.be/SLptdIpW1yQ?t=89.
6 "The Surprisingly Complex Music of Wii Play's Tanks!" YouTube. Scruffy, January 10, 2022. https://youtu.be/NkBXgcN3fXo?t=69.

7 "Evolution of Super Mario Bros. Theme Song (1985–2018)." YouTube. Shiromi, August 29, 2018. https://www.youtube.com/watch?v=iEHgX_h8lYU; "The Evolution of the Legend of Zelda Main Theme (1986–2017)." YouTube. 4865alex, October 25, 2018. https://www.youtube.com/watch?v=lCgAmISmnSA.
8 "How All 25 Characters Sound Like Distorted in Mario Kart Wii." YouTube. Those Amazing Games, July 21, 2021. https://www.youtube.com/watch?v=3ztXo6oYfXg.
9 "Rocking the Plank: The Music of Sea of Thieves | Inside Xbox." YouTube. Xbox, March 10, 2018. https://www.youtube.com/watch?v=QtR5STY1HuE.
10 "Pharaoh – In Game Videos Compilation HD." YouTube. roguehwz, December 30, 2009. https://www.youtube.com/watch?v=suWiJQ4L7IQ.
11 "Pac–Intermissions." YouTube. VelvetRolo, May 25, 2010. https://www.youtube.com/watch?v=v8BT43ZWSTY.
12 "Oddworld: Abe's Oddysee – 28 *Good Ending*." YouTube. Sophie's Gaming Archive (Annse84), February 21, 2009. https://www.youtube.com/watch?v=hE9zCUFNKq8; "Oddworld Abe's Oddysee Bad Ending." YouTube. OmegaBlades, May 14, 2007. https://www.youtube.com/watch?v=UL-2NQVifDE.
13 "Saints Row 1 intro and main menu." YouTube. JJKLA, July 16, 2021. https://youtu.be/CAR3qDYT8tE?t=21.
14 "Galactic Civilizations II – Main Theme." YouTube. Blarghalt, October 30, 2009. https://www.youtube.com/watch?v=GyhSShv3noA.
15 "Caesar IV OST (2006)." YouTube. Keith Zizza, July 10, 2019. https://youtu.be/Bn3NaNB3Juo?t=178.
16 "Lords of Magic – Battle Music." YouTube. Kong, June 27, 2016. https://www.youtube.com/watch?v=SIvbjVjYcg8.
17 "Metal Gear Solid – Getting Caught." YouTube. k0shyX, December 7, 2010. https://youtu.be/NnLBZFgFhZY?t=29.
18 "Chrono Trigger Walkthrough Part 2 No Commentary (DS)." YouTube. Noire Blue, November 4, 2018. https://youtu.be/KprsR3TIQpw?t=856.
19 GamesRadarTylerWilde. "Portal 2's Dynamic Music – An Interview with Composer Mike Morasky, and Five Tracks to Listen to Now!". GamesRadar+, April 14, 2011. https://www.gamesradar.com/portal-2s-dynamic-music-an-interview-with-composer-mike-morasky-and-five-tracks-to-listen-to-now/.
20 Audiokinetic, Inc. "Building Drama with Data: Joe Thwaites." Interactive Music Symposium, October 8, 2021. https://www.audiokinetic.com/learn/videos/UX4UElZUMIg/.
21 Webster, Andrew. "How 65 Days of Static Made the Sci-Fi Sounds of No Man's Sky." The Verge, April 21, 2016. https://www.theverge.com/2016/4/21/11474292/65daysofstatic-no-mans-sky-soundtrack-interview.
22 "Crypt of the NecroDancer Walkthrough 1 No commentary." YouTube. Diego Aravena, June 1, 2020. https://www.youtube.com/watch?v=EmSaHllA6PU.
23 Phillips, Winifred. Essay. In *A Composer's Guide to Game Music*, 115. Cambridge, MA: The MIT Press, 2017.
24 MuseScore website: https://www.musescore.org.
25 Leak, Brian. "The Nordic Folk of 'God of War': Composer Bear McCreary on Making the Next Iconic Video Game Score." Billboard, March 22, 2018. https://www.billboard.com/music/music-news/bear-mccreary-god-of-war-video-game-score-interview-8256913/.

Bibliography

Marks, Aaron, and Jeannie Novak. *Game Audio Development Essentials: Game Audio Development*. Clifton Park, NY: Delmar Learning, 2009.

Phillips, Winifred. *A Composer's Guide to Game Music*. Cambridge, MA: The MIT Press, 2017.

Stevens, Richard, and Dave Raybould. *The Game Audio Tutorial: A Practical Guide to Sound and Music for Interactive Games*. Routledge, 2017.

Sweet, Michael. *Writing Interactive Music for Video Games a Composer's Guide*. Upper Saddle River, NJ: Addison-Wesley, 2015.
Thomas, Chance. *Composing Music for Games: The Art, Technology and Business of Video Game Scoring*. Boca Raton, FL: CRC Press, 2017.
Zdanowicz, Gina, and Spencer Bambrick. *The Game Audio Strategy Guide a Practical Course*. New York, NY: Routledge, Taylor & Francis Group, 2020.

Chapter 10

Dialogue

Our Connection to the Spoken Word

Since the dawn of humankind, communication has been essential to our survival. We use speech as part of our everyday lives to inform one another, share memories, stories, and events, and convey our thoughts and emotions.

It's no surprise, then, that the spoken word can have a transformative effect on us through games. Early games didn't have the processing power or storage space for dialogue, unless prerecorded media was used (such as with Don Bluth Production's *Dragon's Lair* in 1983[1]). But in more recent years, technological advancements have been made such that we can now add narration and character voices to a game, which can heighten immersion and create more dramatic moments, among other benefits.

In the world of dialogue production, we rely mainly on outside talent known as **voiceover (VO) artists** (or simply, **voice artists**) to play the roles of game characters. There are also opportunities for voice artists to provide narration, background chatter and voice-based sound effects, all of which we shall cover later in this chapter.

The term "dialogue," "voiceover," and "VO" are often interchanged among game studios. One company might employ a "dialogue editor," while another has a "voice editor" – where both may mean the same thing. In the world of broadcast radio, television, and film, *voiceover* usually refers to speech heard off-camera, and *dialogue* for any on-camera speech.

To keep things simple, I will use the term **voiceover** when referring to voice artists and the voice recording process, and **dialogue** everywhere else.

Now, let's explore the key roles voiceover and dialogue has in games, and how it brings tremendous value to the gaming experience.

The Roles of Dialogue

Create an Immersive Setting and Mood

Whether it's the drawl of cowboy speech in the *Red Dead Redemption* series, *Portal*'s autotuned AI antagonist GLaDOS, *Tomb Raider*'s famous Lara Croft, or the film-noir narration in the *Max Payne* series, dialogue can help set the game's overall mood and tone. Great narrative deserves great acting, and the two (along with proper recording!) create a synergistic effect within the game, making a lasting impression on the player.

DOI: 10.4324/9781003218821-13

Provide Critical Feedback

Dialogue can provide a wealth of feedback to the player. It can let the player know what they can and can't do, provide story clues, inform them of their progress, or receive critical information through conversations (or when eavesdropping on a secret conversation nearby).

Enhance Player-Character Engagement

Hearing the voice of an in-game character helps to augment their personality, but it also enhances the player's engagement with them. Interacting with NPCs in the *Sam and Max* series of games is a great example: the player (taking the roles of Sam *and* Max) must ask specific characters questions throughout the game, in order to obtain clues to solve a mystery.[2] Speaking with a diverse set of NPCs makes the conversations more intriguing and compelling to listen to.

Support the Narrative

Dialogue helps to propel the game's narrative forward and to reinforce its overall sense of purpose. It can also be especially helpful if the story branches in multiple directions or requires a number of mission briefings to be narrated, for example. A common theme in many RPGs and adventure games is to have a main quest to follow, along with several side quests and free-roaming NPC encounters. Here, each conversation has the potential for rich character interaction that can yield many outcomes.

Deliver an Entertaining Experience

Dialogue with interesting and diverse character types can also heighten the overall entertainment level of gameplay. In addition to the dialogue **lines** themselves (portions of the script to be read, like the lines in a play), often it's the method in which players hear them that adds excitement. The multiplayer first-person shooter (FPS) *Overwatch* offers a wealth of characters for players to choose from, be it humans, robots, or even hamsters! Each has a fun and unique set of lines to hear that are triggered by certain in-game events,[3] although there is an option to select and play a line if the player can't wait. In addition, all players can hear each other's lines as well, making the experience that much more fun.

Special Cases

Sometimes dialogue is utilized for a narrator who is also an in-game character. *Black and White 2* uses two advisors in the form of a small angel and devil character, one on either side of the screen. They serve initially as on-screen tutorial guides, but later they attempt to persuade the player to be good or evil (or a mixture of both).[4]

In *Pit People*, the narrator of the story – a gigantic space bear – is also an in-game character and the player's adversary in real time. The bear wishes to get rid of Horatio, a simple blueberry farmer, and utters all manner of threats toward him (Figure 10.1). In one particularly funny segment, the bear narrates "*Horatio dies* …,"

Figure 10.1 The narrator (a gigantic space bear) is also the player's arch enemy in Pit People. Here, they are destroying the player's home while delivering the line "Oops! Butterfingers!" (The Behemoth/Screenshot by Keith Zizza).

and then a few moments later, he calls out once more: "*I said HORATIO DIES*!!". A giant bear paw then crashes down on the player's house, immediately followed up with, "*Oops! Butterfingers!*"[5]

Dialogue Components – A Closer Look

Cutscenes

The opening cutscene is a great place to provide dialogue as the backstory, setting, or other preparatory information. It tends to be the longest cutscene in a game and makes a first impression to new gamers – so great care should go into polishing the dialogue here. On the production side, any transitional or ending cutscenes need to be considered for voice recording and editing, which can be a significant undertaking if multiple characters are involved.

Narration

The concept of the "traditional" narrator is often a central figure who is usually offscreen and omnipotent. They can help establish the game's narrative, preparing the player through cutscenes or at the beginning of each new level. Some traditional examples include the opening cutscenes for *Fable*,[6] *Caesar IV*,[7] and *Diablo Immortal*,[8] which are narrated in a serious tone. However, infusing humor into narration can also pique the gamer's interest, while keeping the levity high. Games such as *Battleblock Theater*[9] and *There Is No Game*[10] incorporate comical voice acting to match the lighthearted gameplay.

Narration from a first-person perspective has also gained popularity in recent years. Games like *What Remains of Edith Finch*,[11] *Gone Home*,[12] and *Alan Wake*[13] all employ a narrator heard when the player is reading journals or letters. Often times, the narration continues while the player resumes gameplay, rather than their waiting on a fixed screen to read the accompanying text.

But more than just for exposition, a narrator can contribute to the entertainment experience with shout-outs during gameplay as well. There are many precedents, including commentators for sports-based games, but one alternative example is in *Darkest Dungeon*, where the narrator enhances key moments during hundreds of actions, such as when exploring ruins ("Pace out the halls of your lineage, once familiar, now foreign"),[14] or when a party succeeds or suffers losses in combat ("a trifling victory … but a victory nonetheless" for winning a minor battle).[15]

Characters

Main Characters

The game's principal character(s) require the very best voice acting, as they will require the most lines of dialogue and are showcased prominently. The player usually assumes the role of one character, though there are games where the player moves between several people, such as in the adventure game *Day of the Tentacle*. Here, puzzles are solved using *three* characters, all with complimentary personalities and played throughout different time eras.[16]

Superb voice acting and excellent scriptwriting can make game characters come alive. Just like the banter between Sam and Max mentioned earlier, principal characters from games like *Psychonauts*, *Elden Ring, Ratchet and Clank*, and *The Last of Us* are also both captivating and entertaining (Figure 10.2).

Offering the player a choice of character dialogue can provide an even greater sense of attachment. The *Starcraft* series lets the player choose to be Human, Protoss, or Zerg, and each race has their own dialogue for relatively similar actions and events, be they human or alien in form. In the *Mass Effect* series, the player can choose the gender of Commander Shepard (their character), where they can both see and hear them throughout gameplay.[17]

Figure 10.2 Superb voice acting and excellent scriptwriting help make game characters like Raz from the Psychonauts series come to life (Double Fine Productions, Inc./Screenshot by Keith Zizza).

NPCs, Barks, and Grunts

Ah, the life of the NPC. They never seem to get the credit they deserve! But without their voices, the game would have significantly less personality and depth. A **bark** is a term referred to the short bursts of dialogue heard from NPCs: "Halt!," "Greetings!," "How can I help you, traveler?" are a few examples. Sometimes the bark is more of a **grunt** – often a single, non-language-specific emote to convey a mood – while proper text may be displayed to the player describing the actual event.

Words or No Words?

In the days of 16-bit games, dialogue was mostly a series of beeps or other creative sound effects that served as a replacement. This was due to the memory and storage constraints at the time making digitized speech prohibitive. Today, games like *Animal Crossing* and *Professor Layton* still use this technique, but it is more for keeping with style than substance.

To conserve on memory and storage, some games incorporate a hybrid approach by using grunts for most interactions and proper speech only for key dialogue. For example, when encountering leading NPCs in *The Legend of Zelda: Breath of the Wild*, a few characters will have spoken dialogue for the most important events (including those within cutscenes) and grunts for all other conversations.[18]

Tutorials

Years ago, complex games like RPGs and City Builders shipped with paper manuals, some of the largest being over 200 pages long (as with the earliest versions of the *Civilization* series). But as technology made it more feasible to add in-game dialogue,[19] tutorial narration became a viable option over printed books and ordinary text prompts.

The game mechanics and story in *Bastion* are gradually learned through the narrator as the game progresses.[20] For example, the Kid (the player) appears lying motionless in the center of the screen, until the player nudges him up. The narrator immediately follows with "He gets up …," "finds his lifelong friend [a hammer weapon] just lying in the road … it's a touchin' reunion." Should the Kid fall off a platform, the narrator replies, "and he falls to his death"; the Kid then rematerializes on the platform along with the narrator prompt, "nah, I'm just foolin'." However, if a particular event happens again, the dialogue will not often repeat, so as to avoid over-repetition (Figure 10.3).[21]

User Interface

Dialogue also has an important place in the UI. It is normally presented as 2D audio, either in stereo or positioned somewhere in the 2D space (mono). The UI character is often the role of an advisor, and in some cases is heard through a fictional intercom with a High-Pass EQ filter; this simulates output from a small speaker in the interface panel. Some games with this type of "intercom UI" allow any game characters to chime in, as with those in the *Starcraft* series.[22]

Figure 10.3 The narration in Bastion is based on current events and actions that unfold during gameplay (Supergiant Games/Screenshot by Keith Zizza).

Dialogue-as-Sound Effects

Just because audio is recorded by a voice artist doesn't mean it's always used as a dialogue. Depending on its length and application, it's sometimes preferable to think of speech as a sound effect, and implementing it with that purpose.

Does Treating Dialogue as a Sound Effect Matter?

Making the decision to treat dialogue as a sound effect matters significantly. For instance, it may now be grouped in with the "sound effects" volume slider on a menu screen, or it could be processed with different effects or dynamics control during gameplay, or further still, it may reside in a different location in memory or other storage media, and with a different playback priority than ordinary dialogue.

In the *Where's Waldo*?-inspired game *Hidden Folks*, tapping on almost any of the in-game objects (buildings, trees, etc.) triggers a human voice, performing either a short sound effect or a single word ("Hello!").[23] While technically the files are recorded voices, the audio files are treated as sounds (over 2,000 in all!), whether they are language-based or not.[24] This is also the case in the beat 'em up *The Simpsons*: when the player switches between playing the various family members, they respond with a short callout ("My turn!", "Flame on!", "OK!", and the like). Similarly, any time they perform a quick jump or exertion, a vocalization is heard along the lines of "oof!", "hup!", and "ungh!".[25] These can all be considered dialogue-as-sound-effects.

Walla

Yet another type of dialogue is a kind of nondescript, group background chatter called **walla**. Voice artists are recorded individually or together uttering random words, syllables, or just murmuring. After recording, the files can then be stacked up on multiple tracks in a DAW to adjust the crowd size. Walla is often used in a scene where background conversations are needed, such as in a café, marketplace, or at a sporting event. It is usually considered environmental ambience.

Nonlinear Approaches

Branching Dialogue

There are multiple ways a conversation can unfold with an NPC; the structure of the dialogue paths is more like a tree than a straight line, with multiple outcomes, or **branches**. The branches in a **dialogue tree** can be considered "trunks" which contain speech pivotal to the narrative, along with "twigs" that are small branches for more incidental outcomes. This means all characters may require additional lines for all possibilities, which will add to the recording, editing, and implementation time. Figure 10.4 shows a sample tree, with the NPC dialogue in the dotted boxes, and the player responses in the solid boxes.

Some games use branching dialogue as the primary game mechanic, and with skilled writing it can produce compelling gameplay. Choosing what to say in the adventure game *Oxenfree* ultimately leads to one of many possible outcomes, either large or small. But the developers also allowed for the interruption of speech, as well as simultaneous conversations to take place. This adds an impressive dimension of realism, even beyond the voice artists' work itself.[26] But whenever new mechanics such as these are part of the game design, the voice director must know in advance (if possible), as it could impact the way certain lines are delivered, or even increase the number of lines required per character (Figure 10.5).

Segmented Dialogue

Narration or character dialogue can be stitched together to form a sentence. This is often the case with sports games that require a commentator. Making a field goal kick in a football game, for example, would require a sportscaster to call out a specific player's last name, the event, and a yard line, something like: "[Zizza] is going to attempt a [31] yard field goal!" which must sound seamless, otherwise the illusion of live commentary is broken.

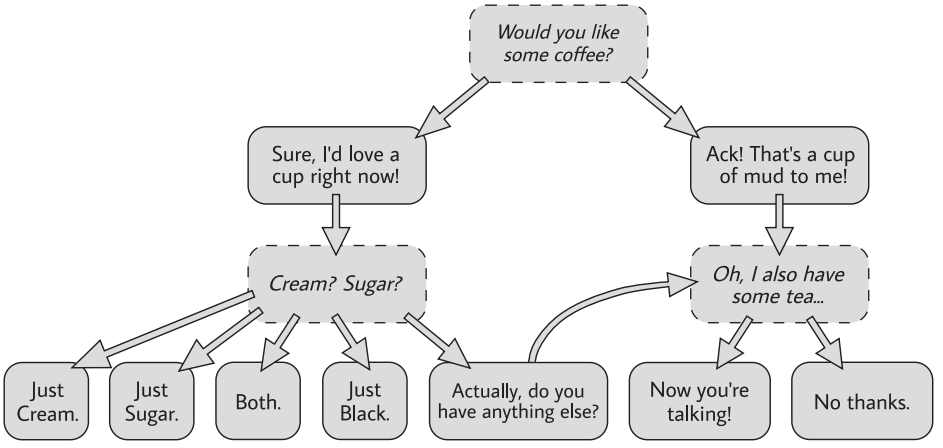


Figure 10.4 An example of a dialogue tree. Note that tree traversal can occur in both directions.

Figure 10.5 Conversations in games like Oxenfree allow various narratives to unfold, and unique dialogue paths to be heard (Night School Studio/Screenshot by Keith Zizza).

Dialogue Personnel

From the germination of character concepts to their voices heard in the final output, game dialogue is a carefully coordinated process between multiple specialists. They must all work together in order to make the audible experience a seamless one, which is no easy task! Below is a description of the five most important people to make it happen:

The Script Writer/Narrative Designer

Game writers often have a background in history, mythology, sociology, literature, or media studies, in addition to creative writing. They collaborate with game designers to work out what the characters will say, including any narration. They must also determine whether or not the game's backstory is included, either in a cutscene or presented during gameplay. Once all of this is finalized, the writer creates **scripts** for each character and/ or narrator, which are sent off to the **dialogue coordinator**.

The Dialogue Coordinator

Also known as a voice coordinator, talent coordinator, or VO supervisor, among other titles. They determine which **voice artists** will be recording for which characters or the narrator. Next, **voice casting** begins, which is a searching and auditioning process to find the best voice talent for each role. They also schedule time with a recording studio, capturing the voice artist's performances that are processed later by the dialogue editor (below).

The Voice Artist

The chosen voice artists read from their scripts and record their performances, preferably with a **voice director** and **recording engineer** participating in the session (below). Voice artists read the script in different ways as the voice director requires: either all

in one **take** (one attempt at reading a specific line of the script such as, "Howdy!") or as multiple takes for each line of dialogue, which could include reading in different styles for each ("Howdy", "How-dee!", "HOWDY!").

The Voice Director

Working with the voice artist, the **voice director** is part acting coach and part diplomat. They must first ensure that the voice artist is performing their character in the style requested – but they must also try to get the very best performance from that artist, especially when they're reading for a narrator or leading character in the game. This means keeping any tension down to a minimum, at least for the artist (there will be pressure to get the job done on time), and to be the main channel of communication for them. The worst thing that can happen is for the voice artist to receive multiple, conflicting opinions from everyone in the room.

The Recording Engineer

The recording engineer is responsible for ensuring the voice session is recorded clearly and audibly. Voiceover is typically recorded "dry" with a minimum of dynamics and EQ processing and with no effects (they can be added later but not taken out of the original recording). They may also mark which takes are preferred by the voice director, or note if there was any noise during a take such as bumping the script stand or microphone.

The Dialogue Editor

The job of the dialogue editor is to break up all of the "raw" material from the recording engineer, editing, mixing, and outputting the audio into the required files. This may also include any special processing such as additional dynamics control, EQ, or spatial effects. It is not uncommon for the editor to notice something amiss with a take (off-axis on the mic, script stand noise, etc.), or that a take is missing. In this case, they can request a **pickup** (redo) from the voice talent, who is usually happy to oblige – provided it's only an additional take or two. Once files are complete, the asset list can be updated and the files added to the game repository.

It is also not uncommon for an additional word or line to be needed after the recording session. Again, if it's a matter of a few words or a sentence, it's the norm to have the pickup read at no charge. However, if a studio needs to be commissioned again, or if there is much more to record, the normal rates will apply (Figure 10.6).

Preproduction

Auditioning Talent

The dialogue coordinator (and any other creative team leads) may audition dozens of voice artists via their VO **demo reels** (a moniker which hearkens back to the days of voice demos-on-tape). These are 2- to 3-minute audio clips demonstrating the variety of an artist's work. Many have experience in the world of broadcast, having

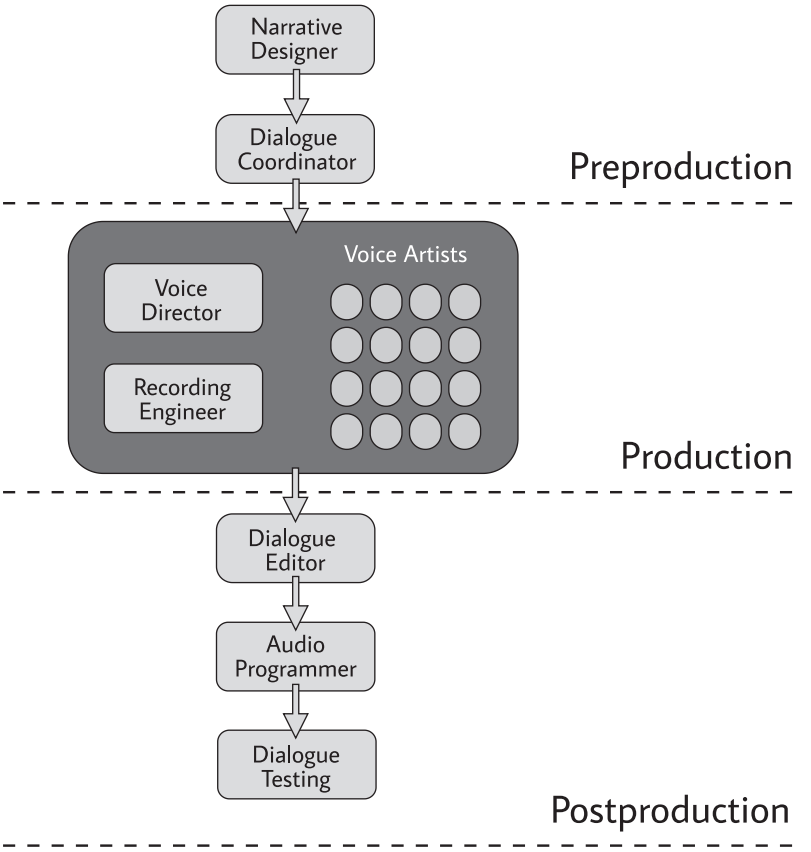


Figure 10.6 The general flow of the dialogue production process.

commercial and industrial-themed reels, but with more games produced each year, there are an ever-increasing number of character- and videogame-based reels emerging. Additionally, if the artist's vocal qualities sound promising but no demo exists, it's not uncommon for a dialogue coordinator to request a custom demo, providing some sample lines for the talent to read.

Voice Marketplaces

When searching for voice talent, online marketplaces are a great place to go. Websites such as Voices.com and Voice123.com contain thousands of demo reels and other information about each voice artist posted. The dialogue coordinator can listen to these demos or request a custom one from an artist. They can also post a free ad on the website, stating the role(s) and even some sample dialogue; this helps the team narrow in on the right voice for the job. VoiceBunny is another option, where the demo reels are posted and custom samples can be requested, but the rest of the process is a "black box": the artists are anonymous, and the session is recorded and sent back to the coordinator as both a large WAV and MP3 file containing all takes.

Booking Talent

When deciding upon voice artists, the eternal question will arise: union or non-union? A project that requires voice artists must choose one of these groups exclusively.

Union Talent

Choosing **union** talent offers the best voice artistry in the world: movie and television stars, Shakespearean-level actors, and other highly trained and experienced professionals. Of course, this level of talent comes at a premium, and so artists from the union have much higher costs – and there are specific union guidelines to follow. The Screen Actors Guild (SAG) and the American Federation of Television and Radio Artists (AFTRA) were once two separate unions, but in 2013 they combined to form SAG-AFTRA.

A game studio or their publisher may need to become a **union signatory**, which is a company that has a legal agreement between themselves and the union to *only* hire union talent, unless a strong case for a particular non-union artist can be made. More information on hiring union talent can be found from the Game Audio Network Guild: https://www.audiogang.org/unions-and-game-audio/.

Non-Union Talent

Smaller publishers and indie titles may find union costs and regulations prohibitive, and so may search for **non-union** talent instead, which is comprised of everyone else. Right now, *you* are non-union talent if you want to be! Agreements are made either directly with a non-union artist and the game studio, or through a talent agency or other third party.

Non-union talent is almost always cheaper and more flexible on scheduling, but audition your voice artists carefully – the old adage about "getting what you pay for" rings true here. Also, resist the temptation to hire your friends and family members. You may have no objectivity in hearing their true voice qualities, and you may also feel compelled to include them on the project, due to your close relationships with them. Don't give in! Find the *best* voice for the job instead.

Contingency Plans

I've been on tight dialogue production schedules where a voice artist was sick, or some artists were late to the recording session, or had to reschedule at the last minute. It happens, and all that can be done is to have a Plan B. One backup plan is to set an alternate recording date, if you can; if not, consider calling in another artist you've already worked with that may match the other's vocal qualities.

Artist Integrity

"Hey! This new voice artist for the principal character is fantastic, but the guy is an absolute jerk to work with." Should you do it anyway?

I wouldn't. There are many thousands of voice artists out there, and there is surely another with similar vocal qualities who is also a good person in and out of the voice booth. Why make everyone's life more difficult, including your own?

Beyond their demo reel, a voice artist's qualifications will be an unknown unless references are obtained. This is why personal referrals and online reviews from other clients are so helpful. There are several what-if questions regarding the qualities of a voice artist, namely: Are they **reliable? Courteous? Punctual? Professional? Able to take direction? Patient? Flexible? Versatile? Consistent in their pacing and delivery of reading lines?**

Choosing a Studio

External Studios

There are many independent recording studios that offer voice recording sessions. The dialogue coordinator can book the session with local talent (say, in a large city) and the team can either join the session in-person or be channeled in remotely via phone patch or by audio/video conferencing tools such as Zoom, Skype, or Google Meet.

Your Project Studio

Some game companies are equipped with their own voice recording studio. If the company is in a location where voice talent can visit easily, this option may be a more cost-effective option than the overhead of external studios.

Artist-Owned Setups

As the cost of audio-related hardware and software has decreased in recent years, it is quite common to find a voice artist that has their own home studio setup. They normally have an acoustically treated voice booth, a high-quality mic, and any necessary signal processing. Personally, I like to ask for a "dry" recording without too much dynamics control – I often receive audio files with too much compression on them, and I suspect this is due to normal practice in broadcast media (radio, tv, and podcasts). It's also worth asking what type of mic and other gear they are using, just for reference. This is the most cost-effective option and makes rescheduling a breeze.

Scripts

The most important thing to do before recording a single word is to make sure the scripts are **finalized**. Voice recording often comes at the end of a development project, when design is being polished and crunch time begins. There are instances when a voice artist is not available to record for one or two weeks, and somehow the script changes and isn't posted anywhere. The voice artist records the wrong material, and that means a costly redo or even their absence from the initial release of the game.

The text for all scripts should exist in one location, preferably in the **audio asset list** introduced in Chapter 8. Then, it can be split up into the requisite narrator and/or character roles and placed into the final **scripts**.

By the way, the script writer for *Cosmic Penguins* has completed the game's first character script, for Captain Beakley! Head on over to the **Companion Website** to see how this script should look prior to the recording studio date (including a spiffy illustration of Beakley).

Format It!

Each script must be **formatted** in an easy-to-read font with any dramatic cues needed. Take a look at the sample VO script on the **Companion Website**, which includes formatted lines, dramatic cues, character artwork, voice direction, a pronunciation guide, and, most important, the word **APPROVED**. The team needs to sign off on the script before recording can begin.

The Voice or the Animation?

Regarding on-screen dialogue, one question that arises is which comes first: the voice or the animation? The answer is: it depends. Most of the time, the voice is first, and animation comes later. Mouth movements are then either painstakingly animated or automated with proprietary software the game studio has created.

If the voice is last, the voice artist may not be giving their best performance as they're trying to sync up with the on-screen animation. This is necessary, however, when having to record the voice in a different language (**localization**), or when having to rerecord if the original voice in hindsight wasn't usable for some reason – due to a creative shift or syncing to a motion capture that was made without a voice recording, for example.

Production

When a voice session takes place, especially a high-profile one, more people than needed might show up – which does happen frequently because let's face it, it's fun to hear the character voices coming to life, and it's also a really cool thing to witness firsthand. However, the voice artist, voice director (with possibly the narrative designer), and the audio engineer should be there, and *that's it*. Recording studios are small and can get claustrophobic and hot really quick.

By the way, no one should be giving direction to the talent other than the voice director. Remember, their job is to ensure the narration and characters follow the intended style originally set, including the correct tone, pacing, delivery, and motivations. That is hard work enough, and doesn't need to be sidelined by anyone else.

First, the engineer usually sets up a mic with flat EQ and will make adjustments for certain voice artists' character readings. They may require a bass boost, midrange correction, high-end sibilance removal, and perhaps even a different microphone that better complements the artist's voice texture.

While the recording takes place, the voice director guides the talent in terms of getting the best performance from the artist but not so much as to micro-manage them. It's common practice to have lines read in two or three takes each, such as one read normally and one read over-the-top with more excitement. A decision can be made later as to which fits best in-game. Meanwhile, the engineer marks the timing of each take or line, and notes which take the voice director liked best, for later reference.

Once the sessions are completed, the recordings are sent to the dialogue editor as contiguous, "raw" (unedited) WAV or AIFF files. Compressed formats like MP3 aren't usually provided, as the editor should have the highest-resolution files available to work with.

Postproduction

Mixing and Editing

The first thing the dialogue editor must do is locate the session files and the accompanying approved scripts. From there, they can create a new DAW project, adding at least one track and importing the session files. The files are spot-checked to make sure they have received the correct files and also for recording quality, then dynamics and EQ plug-ins are added to the track and applied to the dialogue as needed. Plug-in settings will need to be adjusted for each artist, as no two voices or microphones will sound completely alike. Next, the sessions are listened to from start to finish, marking the specific takes within each file (using the [M] key in REAPER, for example). The takes marked as best might not hold true upon a second listen, or with another set of ears. However, preference may still be given to the marked takes.

Sometimes there may be something wrong with *every* take of a particular line – none can be used on their own. So, a piece of one take and a piece of another can be joined together as a fix. For example: if take 1 of the line "sounds good!" had a strange-sounding "s," and take 2 of the line had a very weak-sounding "d," the editor can borrow the better-sounding consonants from either take and make a new, usable one.

Regions can be created for the final edits (using the [R] key in REAPER), so they can be easily selected and output as files later.

Superhumans, Robots, and Creatures

Augmenting voice recordings to sound otherworldly is quite fun! Over the years, some techniques have emerged that represent certain types of non-human dialogue. Superhuman dialogue may benefit from chorus, downward pitch shifting, and a bit of reverb. A common practice is to add "reverse reverb": first, reverse an audio clip in your DAW, process it with reverb, then reverse the clip again. Add some chorus, and *voila*! This effect is famously heard with Tyrael in *Diablo II* and with the Protoss in *Starcraft*.

Another technique is to apply a harmonizing effect by duplicating a voice track, then pitch-shifting the duplicate up or down to an interval such as a minor third. One example is with the robot Freddie Fazbear in *Five Nights at Freddy's: Security Breach*.[27]

Auto-tuning a voice immediately says "robot," like Ellen McLain's GLaDOS character in *Portal*. Applying a bit of amplitude modulation works as well, which can be heard in the robot Legion from *Mass Effect*, some of the droids from the *Star Wars* series of films, and *IG-11* from *The Mandalorian* television series. Sometimes tank-like reverb can be added to suggest the voice is resonating inside the robot, like from the box-shaped Claptrap in *Borderlands*.

Greatly increasing the pitch of a voice recording works very well for diminutive creatures like pixies, cute Koroks (from *The Legend of Zelda* series), or Yoshi from the Nintendo universe. Pitching the voice down is very effective for large creatures as well.

Combining animal sounds with human emotes can yield interesting (and downright terrifying!) results. Elephant trumpets, bear growls, and dog barks are some of the elements blended to craft a creature- or alien-like voice. Plug-ins such as *Dehumanizer* from Krotos can achieve dramatic results with altering a once-human voiceover.

Mastering and Final Output

Because of the unique timbre of each artist's voice, each recording will have its own dynamic and frequency characteristics, as noted earlier. One final pass on all dialogue near the end of the project may be needed, in getting just the right EQ and dynamics under control. One must also consider that dialogue needs to cut through an often-busy gameplay mix, so one technique is to boost the midrange and treble frequencies slightly.

Each character may have dozens, hundreds, or even thousands of lines; manually outputting 10,000 files is not a fun task. Fortunately, some DAWs including REAPER have a renderer which allows rendering all regions in a timeline (the ones created in the editing phase) automatically, and outputting them in the format of the user's choosing. There is also a batch-processing tool that can take existing files and add a user-selected to process to them (like adding reverb or lowering volume, for example), and/or re-output each as a new file type. One popular batch process is converting WAVs to MP3s or OGGs. Most dialogue is output as mono unless there is a good reason not to (such as having a special effect in stereo).

Localization

EFIGS and Beyond

Once the dialogue is mixed, edited, and mastered, both the text and dialogue may need to be **localized**, meaning it is translated into other specific languages. The most common Western localization set is EFIGS (English, French, Italian, German, and European Spanish); beyond EFIGS, other popular language choices include Brazilian Portuguese, Polish, Russian, Simplified Chinese, Japanese, and Korean.[28] Of course, dialogue can be expanded into dozens of languages, in which case an external **localization group** is absolutely essential. They normally have teams of people ready to record, edit, render, and verify text and dialogue in many languages. You can find several companies on a Google search, but it also helps to ask colleagues about any positive experiences they may have had with a particular group.

Star Wars: The Old Republic had over 200,000 lines of dialogue, which set the world record for the most voiced dialogue in a game in 2012.[29] Though future entries such as *The Witcher 3* and Red *Dead Redemption 2* totaled 450,000 and 500,000 lines, respectively, these were not localized; in the meantime, *SWTOR* went on to localize in French and German, which was surely a tremendous undertaking in production.[30]

Special Cases ("Simlish," etc.)

Some games skirt around localization altogether, keeping the language experience common among all players, while also saving significantly on production time and cost. In *Dragon's Lair* (mentioned earlier), the protagonist Dirk the Daring emotes and grunts his way through each segment of the game but never utters a comprehensible word. This technique has been taken to the *n*th degree in EA's *SimCity* franchise, where all of the Sims speak *Simlish*, a made-up language all their own. One former voice artist revealed that the language is improvised and made up on the spot.[31] Emotions are still conveyed in the dialogue, however, be it joy, anger, or hilarity, and many of the improvised lines have been preserved in successive releases of the game (for example "sul sul" means goodbye in every version of the Sims[32]).

Testing Considerations

Consistent volume levels and proper EQ are a given when testing the final dialogue, but also ensuring that the correct lines are being played in all the right in-game locations and times. As a final check, the dialogue should be compared to the original scripts. Were the right takes rendered? Or perhaps a few lines need to be rerecorded due to a script mismatch, recording glitch, or design change. Testing offers a second chance to catch some of the errors that require a pickup from the voice artist. This is also why (1) the scripts-in-hand during recording sessions *must be the final, approved versions*, and (2) having both the voice director and recording engineer present during the sessions greatly reduce aesthetic and technical voiceover issues.

Sonic Spotlight: DB Cooper, Voice Talent, Casting, and Director for Games

KZ: Help! what should I do if I'm given the task to cast voices for my game?

DB: First up, ALWAYS ASK FOR CHARACTER NAMES AND ART. This will help your actor create a strong character especially since context is often missing early on in the casting process. Even concept art is valuable.

What kind of cast does your game have? For a small-cast, dramatic game, you will likely want to audition most, if not all, the roles.

For an MMO or the like, you'll want to create auditions for your major characters, and outside of those roles you need to have a solid idea of the "types" of characters you need in your game. Are there warriors? Nobles? Are there other-than-human races? You will want to cast for all the types you have but you don't need to cast for every single smaller role.

Your casting document should include:

Character name and type or Identity. Like "Bob Blackington-Current-day London Cat-Burglar" or "Healer for the Village of Wayfair."

Accent and/or vocal type and age range.

Character art.

Character back story/quick bio, relationships with other characters and player. Super quick stuff. Deep lore isn't called for with an initial casting round.

Highly Evocative Descriptions: "The guy who rubs your shoulders without asking."

Lines that will help you cast. Show range. Does the character speak softly but Hulk Out when threatened? Make sure the audition lines cover the extremes of the character's arc in the game.

Get exerts if you're going to need them. You'd be surprised how many of your favorite actors can't authentically take a hit.

KZ: How do I direct a voice session if I meet the talent in a recording studio (on site)?

DB: Spend a moment chatting with your actor (unless you know they specifically do not want to faff about before a session). Human stuff: How's your week been? Are you playing a game now? It's good to settle in.

Be ready to describe the scenes they are in and what has just happened. Also be ready to answer lore questions.

Tell your actor that is it OK to ask you what you mean. Don't be afraid to explain your direction. Everybody's frame of reference is different.

Don't interrupt your actor. Unless they are reading a paragraph and are going totally sideways, just wait a couple of moments for them to complete the line. Interrupting the actor may seem like a time-saver, but what it really does is alarm the actor and undermine their confidence which can lose you a strong performance, or it will irritate the actor and make them less likely to be receptive.

Keep track of the time and offer breaks after 45–50 minutes of work.

Say a dignified "Thank you" for a job well done.

KZ: How do I direct a voice session if the talent has their own studio (remotely)?

DB: Same all 'round except you'll need to describe the gestures you are making if they don't have a video feed.

Notes

1 "Dragon's Lair on Steam." Accessed August 18, 2022. https://store.steampowered.com/app/227380/Dragons_Lair/.
2 "Sam & Max Beyond Time and Space Remastered – All Episodes – English Longplay – No Commentary." YouTube. Andrea Pannocchia, December 22, 2021. https://youtu.be/uvZ1zuZuWJA?t=8949.
3 Maher, Cian, Anna Washenko, Daron Scott, and Lucas Sullivan. "Overwatch Quotes: All the Voice Lines for Every Character." Overwatch characters: check out all the heroes and decide which one is best for you. GamesRadar+, March 1, 2017. https://www.pcgamesn.com/overwatch/overwatch-quotes.
4 Ocampo, Jason. "Black & White 2 Hands-on – Discovering a Whole New Black & White." GameSpot, October 4, 2005. https://www.gamespot.com/articles/black-and-white-2-hands-on-discovering-a-whole-new-black-and-white/1100–6132585/.
5 "Pit People PC Gameplay No Commentary." YouTube. ZAG, July 7, 2019. https://youtu.be/Bl-vm9vEiek?t=256.
6 "Fable Anniversary Full Game Walkthrough – No Commentary (#FableAnniversary Full Game) 2014." YouTube. RabidRetrospectGames, February 12, 2014. https://youtu.be/woT6uFiOZmQ?t=93.
7 "Caesar IV – Hard – Empire, Mission 1: Argos." YouTube. We Are Geth, October 12, 2015. https://www.youtube.com/watch?v=1lroONxLZxM.
8 "Diablo Immortal – Cinematic Trailer | Blizzcon 2018." YouTube. GameSpot Mobile, November 2, 2018. https://www.youtube.com/watch?v=fyy7NEwm1jU.
9 "Battleblock Theater Walkthrough – Chapter 1 [No Commentary] [HD]." YouTube. Noctapus, April 28, 2013. https://youtu.be/ZuaLCw2C1AY?t=8.

10 "There Is No Game (No Commentary)." YouTube. Rainerius, July 23, 2015. https://youtu.be/6qHqzAtGFu0?t=18.
11 "What Remains of Edith Finch – Full Game Walkthrough Gameplay & Ending (No Commentary Longplay)." YouTube. Father, April 26, 2017. https://www.youtube.com/watch?v=q-fj8PhdWeI.
12 "Gone Home (Full Playthrough, No Commentary)." YouTube. lolRenee, August 21, 2013. https://youtu.be/SS5eQmRgBlY?t=1340.
13 "Alan Wake Gameplay Walkthrough Part 1 Full Game [4K 60fps PC Ultra] – No Commentary." YouTube. MKIceAndFire, March 16, 2021. https://youtu.be/0RpTykaCVNQ?t=3928.
14 "Darkest Dungeon – Weeks 1–3 [No Commentary]." YouTube. IT-Psycho, February 7, 2015. https://youtu.be/J-X5-KRJ2JQ?t=989.
15 "Darkest Dungeon – Weeks 1–3 [No Commentary]." YouTube. IT-Psycho, February 7, 2015. https://youtu.be/J-X5-KRJ2JQ?t=1645.
16 "Day of the Tentacle Remastered Full Game Walkthrough – No Commentary (#DayoftheTentacle Full) 2016." YouTube. RabidRetrospectGames, March 22, 2016. https://youtu.be/eW283uY3sLI?t=595.
17 "Commander Shepard." Mass Effect Wiki. Accessed August 18, 2022. https://masseffect.fandom.com/wiki/Commander_Shepard.
18 "The Legend of Zelda: Breath of the Wild: Part 3 – IMPA (Playthrough/No Commentary)." YouTube. Avenging Hero, March 6, 2017. https://youtu.be/S0VpGBlj70I?t=2238.
19 When making a major videogame, the development costs incurred is also a factor in not printing a manual.
20 "Bastion Gameplay Walkthrough Full Game No Commentary [1080p 60fps]." YouTube. OdebGaming, August 4, 2018. https://www.youtube.com/watch?v=0eY-xX8OmL4.
21 Nouch, James. "Why Bastion's Narrator Is a Silver-Tongued Storyteller and an Understated Narrative Lynchpin." gamesradar. GamesRadar+, May 9, 2017. https://www.gamesradar.com/bastions-narrator-is-a-silver-tongued-storyteller/.
22 "StarCraft 2 – Terran vs Harder AI. No Commentary Win, Win." YouTube. theclemgos, October 9, 2017. https://www.youtube.com/watch?v=vrKx7kixk3U.
23 "Hidden Folks Gameplay HD (Pc) | No Commentary." YouTube. FullThrough, July 22, 2020. https://www.youtube.com/watch?v=Yoz5kTn5aNY.
24 "Hidden Folks." Hidden Folks. Accessed August 18, 2022. https://hiddenfolks.com/.
25 "The Simpsons Game – Full Game Walkthrough Gameplay No Commentary." YouTube. Gamer Max Channel, July 5, 2018. https://youtu.be/L-ByJp7ooWY?t=1112.
26 Webber, Jordan Erica. "Why Choosing to Interrupt Dialogue in Oxenfree Makes the Conversations Feel Natural." GamesRadar+, May 30, 2017. https://www.gamesradar.com/interrupt-dialogue-in-oxenfree-makes-conversations-feel-natural/.
27 "Five Nights at Freddy's Security Breach Full Game Walkthrough – No Deaths – No Commentary 1440p/60fps." YouTube. Game_track, December 18, 2021. https://youtu.be/8lJTA4RT5Qc?t=154.
28 Shvetsova, Margarita. "Top 10 Languages for Game Localization in 2021." Alconost Blog, April 9, 2021. https://blog.alconost.com/en/game-localization-languages.
29 "Star Wars: The Old Republic Recognised Guinness World Records 2012 Gamer's Edition." Guinness World Records, January 2012. https://web.archive.org/web/20120217175943/http://www.guinnessworldrecords.com/news/2012/1/star-wars-the-old-republic-recognised-guinness-world-records-2012-gamers-edition.
30 Egan, James. "Star Wars: The Old Republic Being Localized in French and German." Engadget, February 9, 2020. https://www.engadget.com/2009-08-18-star-wars-the-old-republic-being-localized-in-french-and-german.html.
31 Mercante, Alyssa. "The Sims Voice Actor Confirms They 'Just Make up Gibberish' When Recording." gamesradar. GamesRadar+, June 28, 2021. https://www.gamesradar.com/the-sims-voice-actor-confirms-they-just-make-up-gibberish-when-recording/.
32 "Simlish." The Sims Wiki. Fandom. Accessed August 23, 2022. https://sims.fandom.com/wiki/Simlish#:~:text=The%20Sims%20featured%20a%20lot,when%20they%20go%20to%20work.

Part III

Practice


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 11

Production and Development

Welcome to the team! Perhaps this is your first audio gig with a game company, or maybe you've just been hired for a permanent position. In any case, we'll take a look at the various audio roles within a development studio, along with the other departments that audio personnel work with. We'll also cover some production essentials and best practices, as well as some pointers for making the most out of your game development experience.

Studio Departments

Here is a short list of the various departments in a typical game studio. Some folks may be multitaskers, perhaps operating as programmer and artist, or writer and sound designer – whatever works best for the team! Each can interface with audio in a particular way.

Designers

Create the heart and soul of the game itself, from initial design concepts all the way to last-minute decisions on gameplay. More complex games may have several designers: one for levels, one for various game systems, and another for UI, to name a few categories. In this case, there is usually a lead designer to manage their tasks.

Audio intersection: Designers provide insight into any part (or all) of the game, whether it be characters, environments, or specific gameplay elements. They maintain the overall continuity of the game, and their conversations with you can deeply inform your decision-making on audio content, as well as how you go about implementing the soundscape in general.

Artists and Animators

Craft imagery for everything including characters, UIs, maps, logos, environments, effects, icons, animations, cinematics, and any other 2D- and 3D-based visuals.

Audio intersection: The art department supply a vast array of visual content that requires all sorts of audio, for any visuals that require it.

DOI: 10.4324/9781003218821-15

Writers

Pen the game's backstory, narrator, characters, quests, UI, tutorials, author web content, branching dialogue, and more.

Audio intersection: Writers provide scripts for narrators and characters, all which need to be recorded for cinematics, gameplay, and tutorials. They also provide backgrounds for each character, which may inform your choices for appropriate sound design, musical themes, and hiring the right voice artists for the roles.

Programmers

Responsible for the game under the hood. There is an immense amount of work happening in game code, for instance, the core engine, AI, networking, multiplayer capabilities, combat, NPC logic, physics, animation, audio, and a myriad of other important systems. Sometimes there is a dedicated audio programmer on the team, which is a huge plus for you! You can collaborate with them on making an even greater sonic presentation.

Audio intersection: Programmers help with creating the audio implementation, for every part of the soundscape.

Quality Assurance (QA)

Accountable for squashing bugs in all areas of the game, releasing new versions and patches, and generally improving the game experience for all players.

Audio intersection: You can work together with QA members to detect issues with the soundscape, e.g., audio not playing correctly or in the right locations, playback glitches, or any other audible inconsistencies. They are also a great resource for objective opinions about the overall soundscape, as they are playing and scrutinizing every conceivable corner of the game.

Producers

Masters of project and task management, they ensure the game ships on time and within budget. They also make sure that each department's schedule works efficiently with the others and are meeting their deadlines at all the right stages in development.

Audio intersection: Once you establish your audio goals for the game, a producer will ensure you have enough time to meet those goals, which are often resource-dependent: for example, a character animation requires an art concept, followed by a 3D model, then textures, rigging, and sometimes visual effects. Most (if not all) of this has to happen before sound is added, which means you'll need those team members to finish their work on time. Otherwise, with each late person in the pipeline, you will lose increasing amounts of time to complete your own part of the job. A producer can help hold the line throughout each of these tasks.

Marketing

Responsible for your company's brand and overall image, including advertising, product placement, and promotions. They understand the game's target demographics,

and with that background knowledge they can inform the team of your customers' desires for your upcoming product.

Audio intersection: You can supply the marketing department with audio for trade shows, demos, trailers, commercials, and online promotions.

Web/Social Media

Listening and responding to fans and patrons of your games, moderating boards, maintaining content across all social media channels (Facebook, LinkedIn, Instagram, Twitter, YouTube, and more), sending out product updates and development news, and working with marketing on online promotions, among other responsibilities.

Audio intersection: Audio content may be needed for the company website, as well as any social media channels or online promotions.

Specific Roles in Game Audio

Now let's look at the specific roles that currently exist for those involved in game audio. Please keep in mind this is not a definitive list and serves only as a base template; many roles are often combined into one, or broken out into many more specific ones. But if you are the only audio person on board – the chief cook and bottle washer, as they say – then most if not all of the functions described below will belong to you! Some of these roles are presented here as a synopsis from previous chapters.

Any one of these roles may work with the studio departments listed in the previous section; for example, working with technical personnel to help with implementation requirements, or even helping to devise audio-specific test plans for the QA department.

Audio Director

The audio director is responsible for all audio across all game titles, and across all platforms and services. Upholds the sonic vision and integrity of each game, allowing them to blossom into their destined stylistic and technical forms. Ensures that all audio-based deliverables are fulfilled on time and within project budgets. Works with project leads and individual audio personnel, diving in to assist with any audio discipline when needed.

Audio Lead

Each audio lead serves as the point person one for a particular game, when more than one game at a time is in production. Each lead reports to the audio director, who oversees their work.

Composer

A composer is responsible for all music in the game, including gameplay and cinematics. May perform crossover work in sound design for musically based sound effects. Mixes, edits, and masters their own compositions or other music-based audio.

Sound Designer

The sound designer creates sound effects for the game, using a variety of disciplines including original field recording, Foley, custom recording techniques, and/or the use of existing source material.

Dialogue Coordinator / Voice Director, and Dialogue Designer

A dialogue coordinator casts and schedules voice artists, and the voice director records dialogue sessions. The dialogue designer Mixes, edits, and masters dialogue; perhaps creates creature dialogue or otherworldly voices, making creative use of DAW plug-ins.

Audio Designer

An audio designer may be involved in music composition or dialogue production, but almost always sound design of some sort.

Technical Sound Designer

A technical sound designer ventures beyond the normal duties of sound design into deeper levels of implementation, but perhaps not as deep as the role of an audio programmer. Uses scripting languages such as C# and Python. Responsible for engine-specific audio implementation using game engines such as Unity or Unreal, and audio middleware tools (covered in the next chapter) may help with optimizing performance of audio in the game through streamlining audio implementation; this may include proper format conversion and file compression, prioritizing channels, and optimizing the use of real-time effects processing during gameplay, to name a few examples. Often serves as a liaison between audio creatives and programming staff, bridging any gaps in communication, and helping to ensure the proper aesthetic and technical balance of the soundscape.

Audio Programmer

The audio programmer works with any member of the audio team to implement new systems at the most detailed level, perhaps coding in C++ or Assembly language. Collaborates with the technical sound designer to ensure systems are functioning properly, and help to set up objects or systems in audio middleware to communicate with the game code. They can also fix audio implementation bugs or sort out more technical issues or gray areas in audio implementation that arise (performance or space limitations, edge cases, etc.).

Outsourced Audio Talent

Sometimes a studio is short on in-house audio talent, or needs some expertise for a short period of time. Enter the audio mercenary! This work-for-hire role is usually known as an audio contractor or freelancer. They may collaborate with the team in-house for a fixed period of time, remotely, or in a hybrid arrangement, depending upon the requirements of both parties.

Because contractors make their living moving from on completed project to the next, and sometimes handle multiple projects at different companies, they must be proficient in a variety of audio editing tools. They may also be required to know their way around different audio engines, at least with a general familiarity. Some contractors learn on-the-job with advanced techniques like scripting or implementation using the studio's preferred audio engine, whether it be off-the-shelf, or proprietary. Over time, their skill set builds, and as such there are many tiers of expertise among them. Also, as each company has different workflows and productivity choices, the contractor must be able to adapt quickly to these environments. For example, one studio may use Trello for task tracking, while another uses Jira, ClickUp, or Taskworld, to name a few choices.

Extolling the Benefits of Game Audio

Sharing Your Passion

There's nothing wrong with showing your true self at work, at least for how you feel about your craft. If someone starts talking to me about game audio, well, forget it! I will wax on about it for hours if nobody stops me. I absolutely love what I do, and I can't think of anything else I'd rather be than an audio designer. So when it comes to creating a new soundscape, I have no problem voicing my thoughts and feelings about it, passionately and honestly.

Attitude

Be Team Centered

Sometimes we're so focused on making great audio that we forget we are but one cog in the development wheel. Other departments share equal importance, and we need to respect and support their efforts as well.

Be Candid and Collegial

Everyone on your team is interconnected. You are all responsible for the content, quality, and ultimately the success of your game. It's extremely important to speak your mind about your peers' work, and in turn, they should inform you honestly about their opinions. It's a good bet everyone on the team has played lots of video and tabletop games, read lots of books, and watched a lot of movies and TV; their thoughts regarding your own work comes from a wealth of acquired knowledge – whether they possess audio design skills or not – and should be highly valuable to you.

Don't be afraid to speak your mind about something that seems amiss to you: does the composer's taiko drum solo seem a bit too indulgent? Is there too much high-frequency content in your fellow sound designer's spellcasting effects? Let them know; offer your criticism or concerns, but in a positive and meaningful way. Remember to always keep the *game*'s best interests in mind:

> I'm hearing a lot of high-end distortion on those spellcaster sounds. Was that intentional? Maybe we could listen to them together … I know there's a way we can make them sound more balanced and really make those moments pop!

On Kindness

Nobody wants to collaborate with a jerk. You know that person: the one who interrupts people all the time in meetings, takes credit for someone else's accomplishments, puts you down in front of others, and makes you feel generally worthless and depressed. We spend most of our waking hours at work – why should you tolerate bad behavior? There are obvious fireable offenses when it comes to hateful speech, sexual misconduct, and so on. But there are also countless personal zingers, snubs, and passive aggressive situations that go unnoticed or never discussed, in even a "mildly" toxic work environment.

A no-jerk policy is a must. But what can *you* do to make things better? Treating others the way you want to be treated is an excellent place to start. It goes a long way with the ones you work with, but also benefits you throughout your entire career. The game industry is a small place; you may work with your current peers again at another company years down the road. Or, the boss you have now may be working for you someday. Working with kind people makes development welcoming, fun, and I also believe more productive.

Accept Constructive Criticism

"What do you think of the sounds for this monster?" you ask.
"It's not working for me; it doesn't resonate with its art style, or its animations," says the art director.

If you're new to working with a team, or this is your first gig, your instinct may be either to flip out on the art director, or go hide under your desk for the rest of the day. But instead, you should continue:

"OK, is there anything you could suggest? What do you imagine it sounding like?" is one answer; *"OK, let me try another idea, I'll have something to demo for the team later today,"* could be another.

The point is, no matter how much it stings, you need to face the team's constructive criticism of your work, then collect and filter it out, in order to make a more informed decision. You may think you're 100% right being the only audio designer on a team, but your peers' comments are vital to making decisions that are best for the game. And of course, the opinions of your beta testers down the road should matter as well.

However, if someone hears your work and says, *"it stinks,"* or *"I don't know. I just don't like it,"* then they're not being constructive. You don't have to accept responses like these, and you should ask them to either be more specific, or explain the motivations behind their response.

Do It for the Game, Not Yourself

Part of being a professional is going beyond your own needs and creating work solely for the game itself. Think of the game as a living, breathing entity; we create it, tend to it, and watch it "grow" over time, then when it's ready, we send it out the proverbial door. The game is an ecosystem all its own, with various input and output modalities, event handling, states, and behaviors. Our contributions to it should serve it completely; the thought of building something just for your portfolio or to satisfy your

ego in some way should be discarded. Do great, focused work; the professional and personal satisfaction from it will always follow.

Sense of Humor Is Mandatory!

Please take your craft seriously, but yourself … never. I cannot tell you how many times I've seen my peers (and yes, occasionally, myself) get all tied up into a knot about a problem with a file format, something that won't save correctly, or a build that keeps crashing. Losing your cool solves zero problems, and it's often when we walk away from it all for a moment and put it in perspective that a solution arises. A few deep breaths, a walk (or run.) around the office, venting with your team about it, the list goes on.

With regard to socializing with your peers, there are going to be extroverts and introverts among you. Not everyone needs to be the next Tina Fey or Dave Chappelle in the office. It's enough to know you are all in it together, and a little humor can go a long way. Granted we all having different ideas about what is funny or acceptable in the workplace, but I'm just suggesting that you should make a point of having a bit of levity with your peers each day.

Planning

How much audio is required for your game? This is a conversation for you and the lead designer, initially. Then, once a rough outline is in place, it's time to focus on a content plan and a production schedule.

Audio Project Scope

Sure, we'd love to create a million audio assets for the project. But how much is enough? Look at the preliminary game design to get a sense of the broad categories you'll need. In my experience, it tends to be about 20% fewer assets than I initially plan for. You should also be prepared for some pushback on the audio footprint, which sometimes involves shrewd negotiations with the team leads.

Agile for Game Studios

Game production can be fast-paced and have lots of moving parts, with many individuals working together on complex tasks. A widely used production method in game development is **agile**. It involves simultaneous **iteration** on tasks, with **sprints** (milestones) every two to four weeks. This means there is a new working build of the game at the end of each sprint, with the team slowly increasing the amount of detail to the game over the course of the entire schedule.

Task Tracking, Priorities, and Estimates

With regard to audio, your production plan should involve the following:

- Work with the **game design team** to establish what a potential soundscape might be.
- Begin your **audio asset list**. Don't try to fill everything all at once! Just start with what you know you'll need and build out from there.

- **Get started as early as possible**. Not necessarily to generate finished audio content, but to let ideas start percolating early in the process, and to research any new software/hardware required to create your audio. Also, determine what kind of recording methods will be needed, if any (e.g., Foley or field recordings).
- Talk to your producer to see the team's **master schedule** and to integrate your work with theirs.
- **Prioritize your audio** into levels of importance to create, e.g., "1" = *must have*, "2" = *should have*, and "3" = *nice to have*.

From Chapter 8, let's return to our fictitious in-development game called *Cosmic Penguins*. The plan this week is to come up with some penguin weapon and emote sounds. Do you know how long it might take to create them?

Looking through your audio asset list, you'll need to make your best guess at **time estimates**. You can even incorporate these times into the list itself, with the additional columns below.

Antarctic Ice Studios

Cosmic Penguins – Audio Asset List

				Est.	*Act.*	
	Item	*Filename*	*Priority*	*Time*	*Time*	*Delta*
SOUND FX						
Weapons	Penguin laser blast 01	cp_sfx_penguin_laser_01.wav	1	1.5	1	−0.5
	Penguin laser blast 02	cp_sfx_penguin_laser_02.wav	1	0.5	1	**0.5**
	Penguin laser blast 03	cp_sfx_penguin_laser_03.wav	2	0.5	0.5	0
	Penguin laser blast 04	cp_sfx_penguin_laser_04.wav	2	0.5	0.5	0
Penguins	Penguin growl 01	cp_sfx_penguin_growl_01.wav	1	1	1	0
	Penguin growl 02	cp_sfx_penguin_growl_02.wav	1	1	0.5	−0.5
	Penguin growl 03	cp_sfx_penguin_growl_03.wav	2	1	1.5	**0.5**
	Penguin growl 04	cp_sfx_penguin_growl_04.wav	2	1	1	0

The first column, **Priority**, is important not just for creation priority but also for (1) audio assets that need to be removed from the final release (due to memory or digital footprint constraints later in development), and (2) seeing what can be removed for the minimum system requirements, or porting to a lower-tier system (a low-end iOS or Android device, for example).

The second column, **Estimated Time**, is your educated guess on completion. The third column, **Actual Time**, is after the fact – how long it *really* took to do the task.

The last column is the **Delta**, the difference between the two. Sharing your tasks and time data with your producer will allow them to serve you (and everyone else) better. They can take this information and figure it in with the other departments' time throughout each day, week, sprint, or milestone. The Delta is especially helpful for you and your producer, in estimating and managing your day-to-day tasks in general.

(As mentioned in Chapter 8, please visit the **Companion Website** where you can find a sample audio asset list.)

By the way, you're probably thinking, "oh, I'll just triple the estimated times. That way I'll always have lots of room to complete each task." This is a terrible idea! It's OK and even necessary to add a little padding for unexpected situations, but be honest about your guesswork, and factor some time for the unknown. For example, if you think a sound may take an hour to create, that's the time to research, locate the source material or record your own, edit, mix, master, and deliver the final audio. It's a lot to do. But while setting up for the first of 20 laser blast sounds may take a long time, the others may go very quickly. You can certainly choose to add production and implementation time as separate line items per group, but again it's up to you and your producer – you may become saddled writing down task times all week, so strive to record the most important data, and if you happen to have some bonus time for more granularity in your task tracking, all the better.

Preventing "Zero Time Syndrome"

Everyone on the team must strive to meet their sprint goals, to the best of their ability. However, audio is often the last element in the production chain, and you must be extra careful to budget your time accordingly. You must also be aware of each cog in the wheel. There are a lot of chances for each person to take longer than estimated, so the chances of lost time for you can pile up quickly. What if each person is taking a bit longer than planned on their tasks? You may end up with little or no time for audio!

Imagine a plan to create a character in the game during a two-week sprint, and it is an immovable deadline. This means everyone needs to finish their tasks on time, otherwise you won't have any time left to finish your work.

	Task: Penguin firing giant plasma weapon – 14 days		
	Production Plan:	***How long it really took:***	***Sprint Days remaining:***
Concept Art	2 days	3 days	11
3D Model	3 days	3.5 days	7.5
Rigging	2 days	2.5 days	5
Animation	3 days	3 days	2
Visual Effects	2 days	2 days	**ZERO DAYS!**
Audio	2 days		

Sometimes events like these happen, which I've come to call **Zero Time Syndrome** (or occasionally, *Negative* Time Syndrome). It may have been due to a design change, different art direction, or someone got sick. There are lots of reasons.

Instead of waiting around for your time slot to arrive, you might want to take these steps:

- Start researching sounds for this penguin character! Compile some existing source material to work with, or start on some Foley or custom recording. You know there is a plasma- or laser-like weapon involved, so you might want to grab a variety of off-the-shelf futuristic weapon first, and experiment with them in your DAW.
- Get in touch with people in the production queue, and see how they're doing with time. Also, take a look at the approved concept art, 3D models, etc., and reference those materials against the preliminary sounds you're assembling.
- Share a few initial sonic ideas with the team, and see what they think. You can collect their comments to gain some valuable insights. If everyone thinks the laser sounds aren't cutting it, consider choosing some alternate material.

Being proactive will save you time and headaches. Not every task should end up with zero time for audio, but situations like this one can occur due to your position in the chain. This is an important conversation to have with your producer, and you should also let the team know that compressed time slots have been a big problem for you.

Crunch Time Concerns

It's going to happen. With every project, and especially at its mighty finish, there is **crunch time**. This is an all-hands-on-deck situation where most or all of the team is working an abnormally high number of hours per day.

There comes a time when the game has to get out the proverbial door. Once the known showstopping bugs are wiped out (well *ideally*, anyway), and a good deal of others are under control and the game *feels* right, both in terms of playability and overall cohesion, it's time to button it up and **ship it**. But this can mean under some circumstances, everyone has to work in an intense, coordinated fashion to successfully pull it off. It will mean multiple workdays in a row: some nights and weekends are almost certain.

At the end of each earlier sprint or milestone there may be a small amount of crunch, at first – a day or two, maybe – and toward the end of the project that may increase. But it all depends on the scope of the game as well; a casual game may see little or no crunch in its schedule, but a multi-year AAA blockbuster will almost certainly have some intense moments of crunch.

However, I must say this: please do not wear your crunch time as a **badge of honor**. I once contracted for a company whose employees were chatting with me one day about their grueling schedules: "I worked 75 hours this week," said one person. "Oh yeah? I worked 82," said another, and someone else opening their *n*th can of Red Bull said, "I did 90. Top *that*!". It was cringeworthy to witness.

This is not healthy for anyone. It may seem impressive and cool to work yourself to death on a project, with the perceived added benefit of becoming closer to your peers. Heck, you can even have some great war stories to reminisce about later. But you'll also destroy your physical and mental health, family ties, relationships with friends, and make a complete disarray of everything else in your life. I guarantee you will look

back with mixed feelings about the lost time. Take it from someone who's clocked in some insane hours. Driving home at 4:00 am from your office, only to return at 9:00 every day is not sustainable. Everyone has their limits, and you should not be afraid to set boundaries on your time. And if you find that you're crunching all the time, then it's a sign of bad departmental planning, or bad management overall.

Contingency Plans

It could be the case that your audio content isn't working well with the visuals or gameplay. Or the game changes direction in some way, leaving the aesthetic of your sounds and music sounding out of place. Perhaps the current implementation needs to be done over. You may have to resort to a Plan B (or C, or D!) to get your audio on point again. If this happens, don't hold resentment toward anyone; it is often just par for the course that some aspect of the game is going to evolve in some way, even if that means a major change is required in design or visuals. Don't cast blame in anger; stay focused on development, and keep supporting your fellow teammates as best as possible.

Looking Back: The Postmortem

Once the game is released to the public, it's time to start evaluating the development and gameplay experience, otherwise known as the **postmortem**. This is a summary of what went right and what went wrong with each group in the development process: did deadlines get extended? Did gameplay take an unusual shift in direction, or was there something odd or overlooked in the design? How do the players feel about the overall experience?

For audio, this is the time to re-evaluate everything we've covered in audio design and production up to this point, including the following chapters on implementation and listening. Music, sound, and dialogue choices are looked over as objectively as possible, and implementation is studied for its effectiveness in the game. Chances are there are portions that are less utilized than expected, and some more than others (e.g., your space station ambience is used a lot more than anticipated, and you didn't supply enough variation to keep it from sounding monotonous).

Then comes the game reviews and general public reception. It never goes the way you expect! With audio, here is another moment where you may have absolutely no feedback whatsoever, unless something is wrong with the soundscape. When receiving a compliment, the best kind will be from someone who understands *exactly* what you strived to achieve, where the soundscape wonderfully enhanced the game due to x, y, and z. If a negative review appears, try to evaluate why this happened (along with the source of the review) and carefully examine your audio design and implementation once more. The best feedback is often from the players' perspective, and rightly so; they are the end recipient of your labors!

Chapter 12

Implementation

Achieving Your Sonic Vision

Obtaining the best soundscape possible for your game is not only about crafting its individual audio components but also finding the closest aesthetic and technical equilibrium possible. For example, focusing on a beautiful soundtrack is worthwhile on its own but not if it's implemented poorly: the songs might be cued up at the wrong times, some repeated too much, or not enough; the horizontal resequencing or vertical reorchestration could be awry, or, perhaps, a better file format and sample rate could have been chosen to save on media size or to increase playback performance. Any of these issues can break the immersion and entertainment value of gameplay if not carefully considered. The integration of music, sound, and dialogue should have the team's attention throughout development, and unquestionably not as a last-minute task – as it often becomes in so many projects!

Bridging the Aesthetic and Technical

As mentioned in Chapter 7, a balance must be reached between the aesthetic choices made and the technical design of a soundscape, in order for it to be successful. That also means it must serve the game well, supporting and enhancing the overall gaming experience.

Integrating Your Audio Elements

Now more than ever, videogame consumers are enjoying high-quality audio experiences across many genres, and they are steadily becoming more appreciative of great soundscapes. They may not know *why* they like a particular soundscape (nor should they have to), but of course, *we* need to know what we're doing! Their gaming experience is being enhanced by our handiwork – we are helping to transport them to that imagined world they temporarily inhabit. And while we've got their attention, we can employ all manner of sonic smoke-and-mirrors to simulate laser weapons firing, cranky pterodactyls circling above them, or alien overlords speaking in otherworldly voices.

The complexity of soundscapes from one game to next runs the gamut: a casual card game might contain a meager handful of hard-coded sounds, but the latest **AAA title** (industry vernacular for "blockbuster" or "high-budget" game) likely has grand

DOI: 10.4324/9781003218821-16

audio implementation that employs many thousands of audio assets. Let's assume your next game will require a "moderate" amount of audio material. What will you need to consider? In all likelihood, a mountain of things!

A "Mountain of Things"

In addition to creating audio assets, you will likely be working with a programmer on integrating them into the game. Odds are that you'll need a better plan than just hard-coding your audio files; you may also want specific sounds triggered under certain conditions, have music vary based on specific maps, or perhaps limit the number of dialogue events to play on certain platforms. And of course, there's much more to consider, which we'll explore shortly.

After meeting with the creative leads, you'll have a general sense of the audio required for each game state, characters, events, etc. – at least for the time being. Most likely, however, you've left with more questions than answers as to how you'll make everything play correctly in the soundscape.

At this point, it's a good idea to break items out into a spreadsheet or table, as shown in Figure 12.1. This example is a short list, mind you! Yours will likely have more items in it. Try to think of as many tasks and questions as you can initially, to help determine your production schedule. Working with your producer (and getting the answers to your remaining questions), break these items into daily or weekly tasks. If the producer is you, send the tasks to yourself!

Once you've got a rough plan in place for your sonic vision, it's time to get your audio integrated into the game and functioning properly: a process known as **implementation**. It can be a very complex business to assemble, and you shouldn't go it alone.

Implementation normally utilizes a game and/or audio engine, playing audio based on the game's various states, and the behaviors and conditions of each element in the mix. There will be production and testing required, as well as careful scheduling and possibly localization of your dialogue assets.

Implementation Topics

Below are some of the more pertinent topics in game audio implementation, including the systems that support it, along with some of the more common examples.

Middleware

Back in the earliest days of game development, a game designer had to be the programmer, artist, writer, producer, and yes, the audio designer. And if they were releasing their product for the Atari 2600, it had to ship on a cartridge with as little as 2 kilobytes! Fast forward to the early 1990s, and games had evolved by many orders of magnitude in comparison. If a game required more than a few audio assets, the audio designer usually worked with a programmer to create a custom audio engine (or at least *something* audio-related in code) for the game. Over time, a good system might be established, but it still could take months or even years to get just the right one in place, requiring constant revisions and testing as new games were created. And as games continued to become increasingly complex, their audio demands were scaling up as well.

	AESTHETIC	TECHNICAL
MUSIC	Style / Mood: Upbeat? Sedate? In the background? In your face? Tunes: Lots of short pieces? Epic soundtrack? Orchestral, electronic, or hybrid? Perfomance: Real players? Virtual instruments (VI)? A tasteful blend of both?	Linear playback? Loop-based? MIDI or audio files? Stem-based? File size? Total time? Custom programming? Middleware? Horizontal re-sequencing? Vertical re-orchestration? Software choice for composition? VI compatible with music software? File Format? Available RAM? Sample Rate? Bit Depth? Bit Rate? Number of Channels?
SOUND	Style / Mood: Edgy? Smooth? Earthy? Industrial? Palette: Tactile? Visceral? Dramatic? Comical?	Number of layers? Number of variations? Equipment for field recording? Foley recording? Custom recording? Special plug-ins required? Custom programming? Middleware?
	Environment: Serene? Busy? Dynamic? Static? Changes with day / night? Seasonal?	Use of live FX in game / audio engine? File format? Available RAM? Sample rate? Bit depth? Bit Rate? Number of Channels?
DIALOGUE	Style / Mood: Over the top? Matter-of-fact? Palette: Wide variety of characters? Adults? Children? Accents? Creatures? Animals? Robots? Voice Artists: Number of artists? Narrator? Mission / level briefings? Tutorial dialogue? Pacing and delivery? Number of characters per artist?	Interactive / adaptive dialogue? Both? Dynamics & EQ balancing per artist? Script lengths? External studios? Remote / asynchronous VO sessions? Special plug-ins required? Custom programming? Middleware? Localization? How many languages? File format? Available RAM? Sample rate? Bit depth? Bit Rate? Number of channels?

Figure 12.1 For most videogame- projects, there are lots of audio-related tasks and items to consider. Writing out the most important ones in a grid (like the one above) helps in prioritizing them.

By the late 1990s, it was becoming increasingly popular for audio designers and programmers to use **middleware**, also known as an **audio engine**. Rather than a studio creating a system from scratch, middleware provides the most popular and essential components needed to create a customized audio system. It lives in the middle of the audio designer/programmer and the game code, hence its name.

Audio middleware is a powerful software tool and set of protocols, allowing both programmers and audio designers to work together on the execution of the soundscape. Also, if a game needs to ship on multiple platforms like PC, mobile, and console

formats, programmers don't need to keep rewriting code each time; middleware is designed to handle a variety of audio hardware setups.

For the audio designer, this work is normally done through the middleware's graphical user interface (GUI). Programmers can help hook up audio assets in middleware through the use of an application programming interface (API) – a software library that serves as a bridge between the audio engine and the game code (and game engine, if used).

Today, studios may still create their own custom middleware, but a larger percentage will choose an existing audio engine to start with. This can save an immense amount of time and money, and it also frees up the audio designer to focus on creating content. The GUI also allows the audio designer to better realize their sonic vision, greatly reducing programming overhead. Some popular middleware tools used today are Audiokinetic's **Wwise**, Firelight Technologies' **FMOD**, Tazman-Audio's **Fabric** and **SoundMaker**, and the **Miles Sound System**, among others.

Middleware is also great at facilitating **rapid iteration**, meaning you can audition a lot of audio playback options right in the application (even before your changes are in the game), and perform many more trial-and-error sessions versus that of manual coding alone.

The GUI-based setup is designed for easy viewing and editing of an audio system. For example, sets of sound effects (such as for a character) can be placed in a folder or graphical container; within that folder, the attributes of the sounds can be seen, with their respective volumes, propagation settings (2D or 3D playback), effects, and more.

Figure 12.2 is an abstract example of how middleware functions (not specific to any brand of audio engine). An audio-related **event** of some kind occurs in the game code: it could be that the player walks on a particular surface, an NPC speaks in a tavern, or vertical reorchestration is updated in some way. Events are more specifically known as states, triggers, parameter controls, syncs, switches, and many more types across different audio engines.

In the top half of Figure 12.2, when the player opens a wooden door in the game, a message from the game code (set up by a programmer) tells the middleware, "Hey, play a wood door opening sound, please!" A list of a few, similar wooden door opening sounds resides in the middleware project file, and variation #3 is randomly chosen to play, at –23 dB and with a little reverb (as seen in the bottom half of Figure 12.2). Some information is also generated about how much CPU time and memory is used to play the sound, and some data for game testers to know the sound did indeed play.

There are dozens or even hundreds of audio-related events that can take place simultaneously. However, all events are to *play* audio files; some events may carry numeric data representing a change in music volume, or a request to check the health status of the player to increase/decrease the interval of a single heartbeat sound. And again, there is data sent about what audio is playing and its performance impact on the system, and other diagnostic information.

Audio Tools within Game Engines

As mentioned earlier, development teams will often create their project in an existing game engine, as it contains most of the fundamental building blocks required to make a new 2D or 3D project. Many game engines also have their own integrated audio

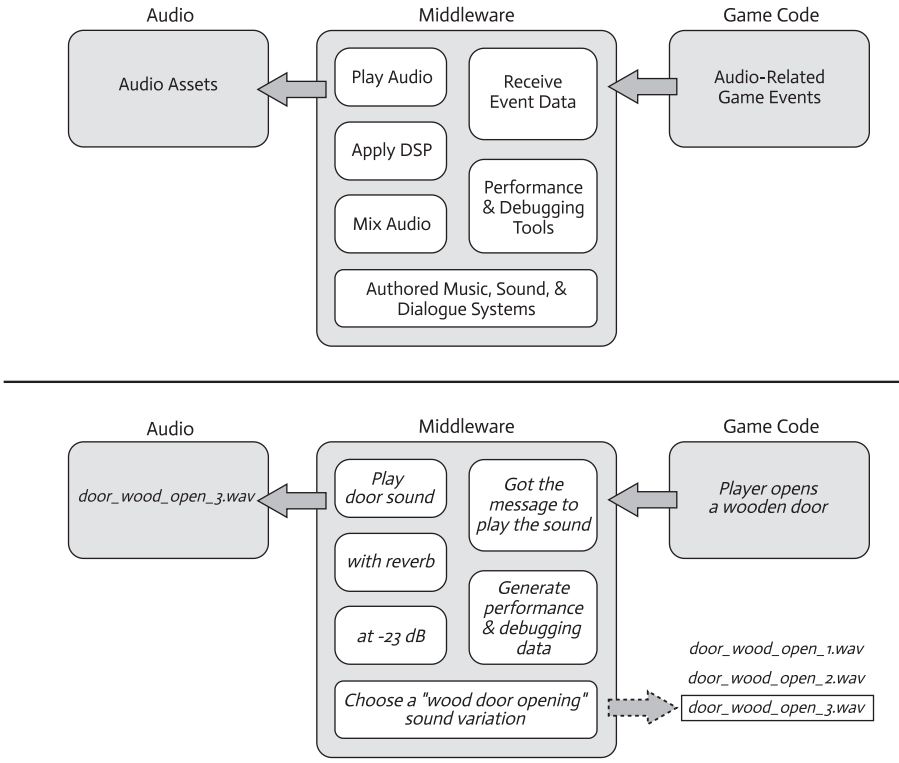


Figure 12.2 Top: Middleware rests between audio assets and the game code, providing a multitude of services. Bottom: As the player opens a wooden door in the game, that message is sent from the game code to the middleware which processes the request, and plays door_wood_open_3.wav, chosen as a random variation.

systems, the two most versatile being in Epic Game's **Unreal Engine** and in Unity Technologies' **Unity**.

Hybrid Implementation Options

It is not uncommon for an audio designer to work with tools from a game engine *and* an audio engine, leveraging both systems to the game's advantage. For example, FMOD has an event editor, where multiple sounds can be choreographed over a timeline; the output of this event could be routed to a mix group within Unity's built-in Audio Mixer system (Figure 12.3).

Determining System Requirements

Your game may need to play audio on, say, a PlayStation 5, but then also perform well on an Android flip phone. The processing power between the PS5 and a low-end mobile device is clearly and profoundly different. How can we ensure the soundscape is presented well on both hardware platforms?

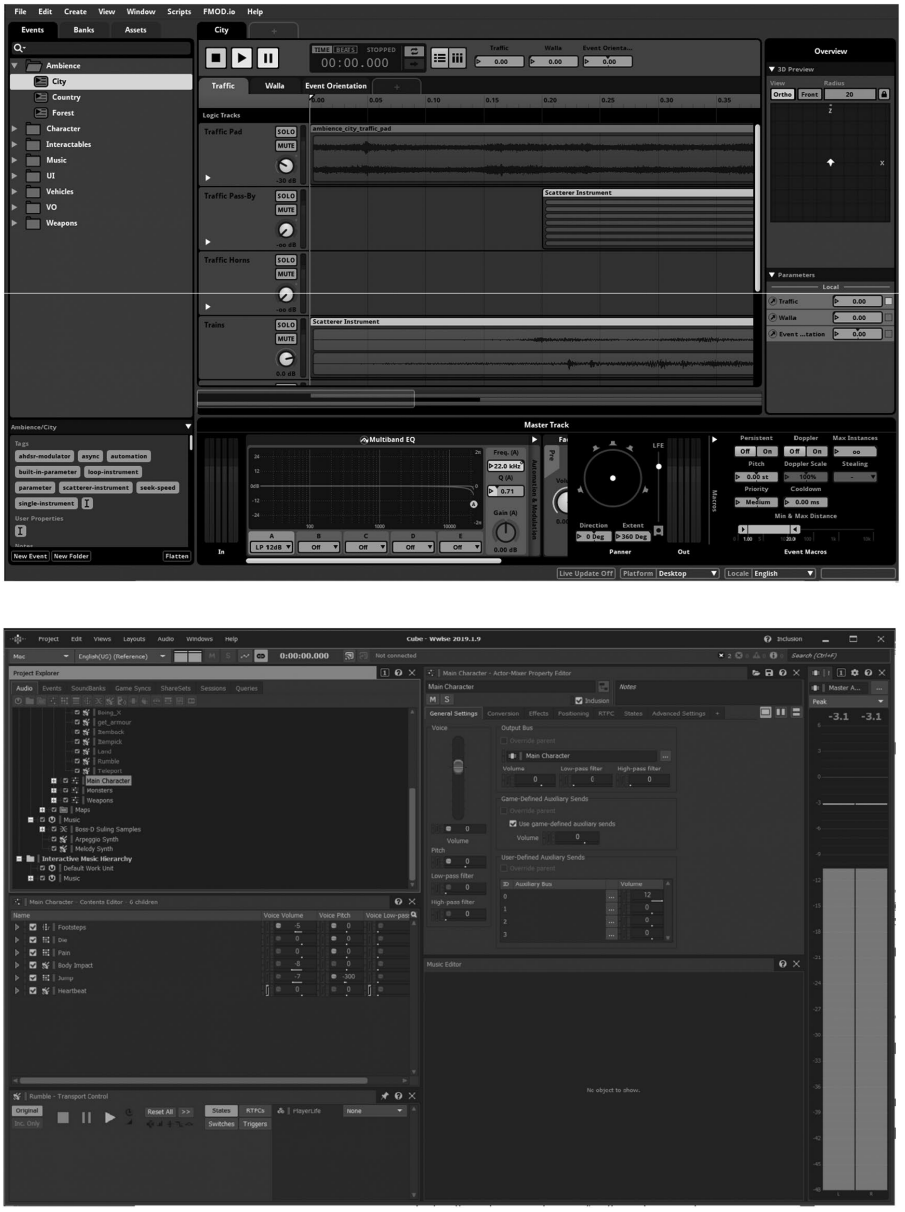


Figure 12.3 Creating audio events in FMOD Studio (top) and Wwise (bottom) (Firelight Technologies, Audiokinetic/Screenshot Excerpts by Keith Zizza).

Fortunately, audio engines can help with this. They offer multiple, scalable setups for many popular operating systems and platforms. You could set up your PlayStation 5 to utilize 50 audio channels, while a low-end Windows device might be allowed no more than eight channels at any time.

You can also declare memory limitations for each platform, and which sounds should load into RAM, or stream from a drive. Which is better? It all depends on your resource requirements, and that all begins with understanding **optimization**.

Optimization

The Triangle of Compromise

In the world of business (and especially in game development), can you have your product delivered fast, cheap, *and* of the highest quality? Well, no. In the real world, one must choose any *two* of these areas to put their energy into, at the cost of the other. This is known as the **triangle of compromise**. If you are a game audio freelancer, for instance, you might offer your work quickly and of high quality, but that should come at a financial premium to the client. If you're cheap and fast, they shouldn't expect the highest quality. And if you're cheap and do high-quality work, it might take you longer to create it.

Similar to the triad mentioned above, the **optimization** of a soundscape is crucial to its success. There is a compromise that must be reached between three audio resources: memory requirements, variations of each audio type, and quality. Music, sound, and dialogue will all have specific needs in your game, and through iteration and working with your team, you must find a way to optimize the audio implementation.

Leeds Beckett University Professors Richard Stevens and Dave Raybould refer to this compromise triangle as the **triangle of pain**, and rightly so! Any two resources come at a cost to the remaining one; negotiating the right balance between them depends on the needs of the game and the bandwidth allowed for each (Figure 12.4).[1] For instance, in the case of sound effects:

1 Loading a lot of variations into memory may not allow for high-quality audio.
2 Having a lot of high-quality variations may not all fit in memory at once.
3 High-quality audio loaded into memory may not allow for too many variations.

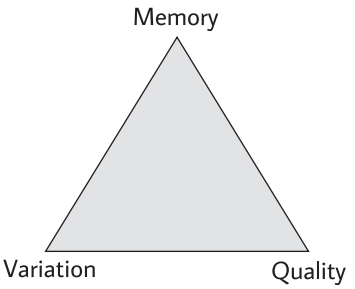


Figure 12.4 The triangle of compromise, otherwise known as the "triangle of pain."

Memory = RAM, the random access memory in the gaming device, which is also the fastest media storage. Loading your sounds into RAM ensures lightning-fast access for playback, but these resources are a hot commodity and are shared with graphics and any other game-related tasks (computing enemy AI, data processing, etc.).

Variations = the number of varieties of a particular sound (sword1.wav, sword2.wav, etc.); too many of these may be redundant and take up valuable space in memory.

Quality = the number of channels, bit depth, and sample rate. Compressed/streaming files may also include bit rate type (constant or variable), compression size, and/or quality level.

Audio Footprint

As game audio designers, we strive to make the best sonic experience possible. However, we are constantly looking for ways to keep file sizes low, and lower our memory and CPU usage during gameplay. This is because our stuff is not the only piece of the game development pie! We are sharing digital space and CPU cycles with art, animation, game engine states, and more. All of the audio files (and any supporting data like MIDI files, audio engine-specific files, etc.) should be located in the game's main repository – a master directory where all the game files live. The total size of all files shipping with the game (art, sound, level data, code, etc.) is known as its **footprint**. It's not uncommon for the **audio footprint** to be a tiny 5% of the total game assets. In some mobile games, it may only be 2%, or even lower.

Preloading versus Streaming

Chapter 5 introduced some of the more popular audio file formats for games. But which one is right for the task at hand? It comes down to what purpose the audio-in-question will have, using the triangle of compromise: memory versus variation versus quality. But you also need to know whether certain files should be **preloaded** or **streamed**, as we will see below.

Preloading: WAV and AIFF

Choosing WAV or AIFF files to load into RAM (preloading) makes sense when:

- the file sizes are relatively small;
- You need to access the files frequently …;
- … and as fast as possible.

You can certainly store it on another "slower" media type (hard drive, SD card, etc.) and load it into RAM when needed, but there will be a delay in accessing it, and the file should still be relatively small in size.

A medieval battle game is sure to have swords, shields, and arrows, which are short duration sounds that can be preloaded into RAM. Once in memory, their access time is light-years faster than any disk media, even solid-state drives. RAM is a precious

resource that needs to be shared among your fellow teammates, so stake your claim early!

Recall that WAV and AIFF files are uncompressed formats that at CD quality (in the 20 Hz–20 kHz range) take up about 10 MB per stereo minute. Loading a single bird tweet into RAM? No problem. A photon torpedo? Fire away in RAM once more! How about music? Well, not unless it's really short. As to what "short" means there is no formal definition, but generally it's nothing more than a few seconds long. Stereo uncompressed files like music or ambience aren't usually a good RAM candidate; they are generally on the longer side and are already doubled in size by having two channels. If absolutely necessary, the sample rate and/or bit depth can be reduced in this case.

Be mindful that as you reduce the sample rate, the highest sample values that can be stored are decreased. Since higher frequencies represent higher numeric values in the file, these are the first ones to be lopped off. Lowering the bit rate decreases the **fidelity** (the faithful representation of the original material) even more dramatically and reduces the **dynamic range** of the audio. This may be OK for speech and lower-frequency heavy sounds like explosions, but use caution here.

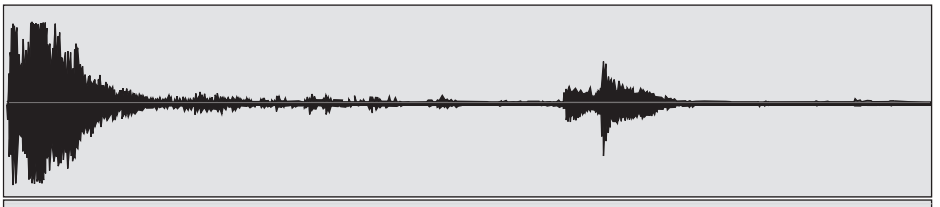
Determining Sample Rate and Bit Depth

Here is a WAV file of a small ceramic tile breaking (Figure 12.5).

Next are some frequency graphs of the same tile sound (near the start of the file), rendered at different sample rates and bit depths (Figure 12.6).

If you've just converted some files down to a lower resolution, and you're ever in doubt about the quality, simply import your newly rendered audio files back into your DAW to see the difference. But in the end, you should make your final judgment by carefully listening to it (without looking at the graph).

As seen in Chapter 2, figuring out the dynamic range can be approximated by the number of bits × 6, in decibels. Table 12.1 shows audio quality for uncompressed files; you can also refer back to Table 2.2 for file sizes versus sampling rates and bit depths (compare the values you've just seen in Table 2.2 with their counterparts in Table 12.1 shown in **bold**).



File	Peak	Clip	RMS-I	LRA	LUFS-I
ceramic tile breaking.wav	-1.5	0	-20.1	--	-17.8

Figure 12.5 Render of a ceramic tile-breaking sound effect. It is a mono 16-bit, 44.1 kHz WAV file.

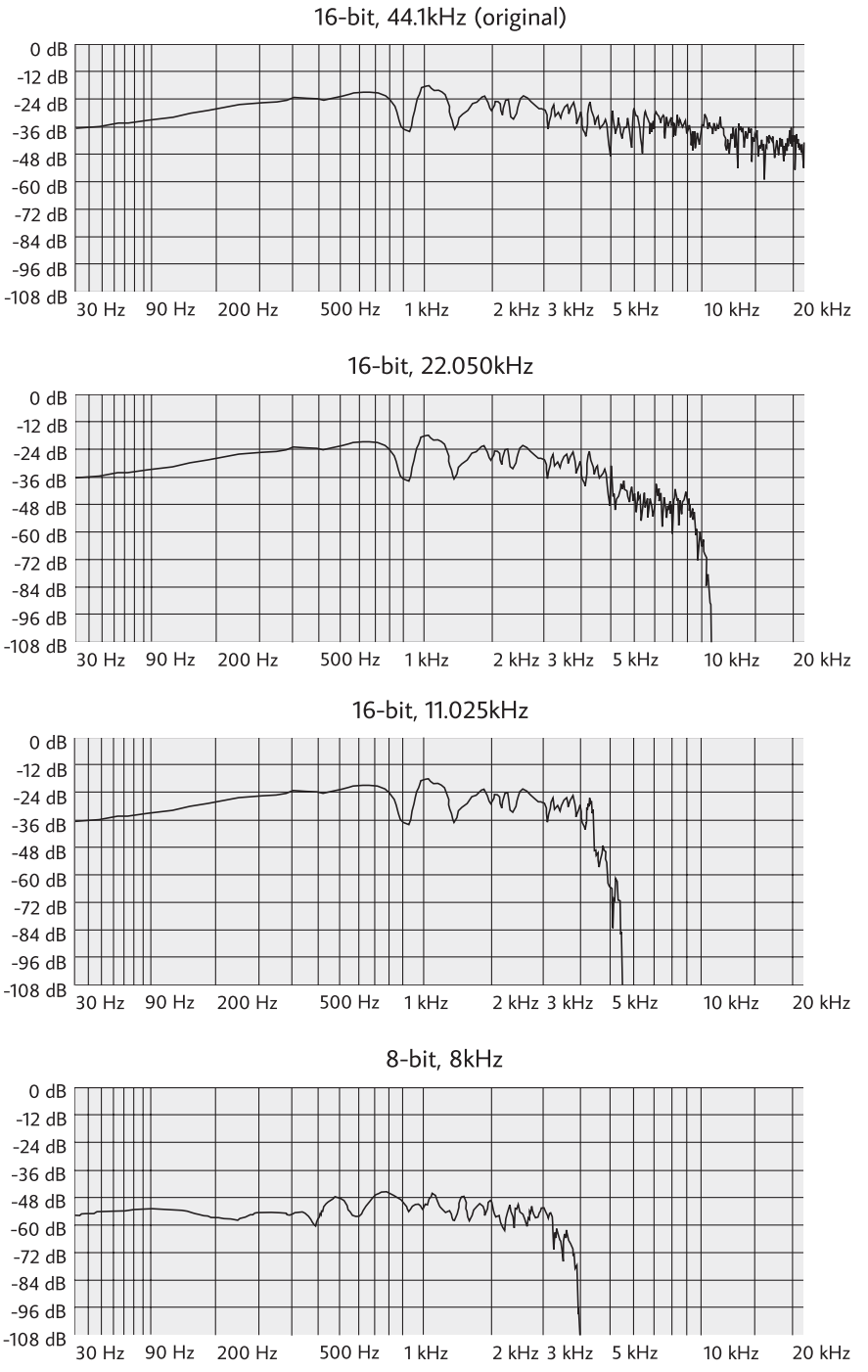


Figure 12.6 The frequency range of a tile-breaking sound effect (mono WAV file), at (a) 16 bits, 44.1 kHz; (b) 16 bits, 22 kHz; (c) 16 bits, 11.025 kHz; and (d) 8 bits, 8 kHz. As the quality deprecates, you can clearly see the sound's high frequencies being wiped away, along with its dynamic range in (d).

Table 12.1 Bandwidth and Dynamic Range

Bit Depth	*Sample Rate*	*Channels*	*Data Rate in kilobits per second (kbps)*	*Bandwidth (Hz)*	*Dynamic Range (Approx. dB)*
8	**8,000**	**1**	**64**	**20–4,000**	**48**
8	11,025	1	88.2	20–5,512	48
16	**11,025**	**1**	**176.4**	**20–5,512**	**96**
16	**22,050**	**1**	**352.8**	**20–11,025**	**96**
16	22,050	2	705.6	20–11,025	96
16	**44,100**	**2**	**1,411.2**	**20–22,050**	**96**
16	48,000	2	1,536	20–24,000	96
24	48,000	2	2,304	20–24,000	144[a]
24	96,000	2	4,608	20–480,00	144[a]
32	128,000	2	8,192	20–64,000	192[a]
32	192,000	2	12,288	20–96,000	192[a]

[a] Theoretical value (ADCs not commercially available beyond ~125 dB).

Streaming: MP3 and OGG

Choosing MP3 or OGG files to stream into memory makes sense when the file sizes are relatively large, as with music or long-playing environmental ambience. Recall that MP3s do not seamlessly repeat themselves, so OGG is the natural choice for looping audio. MP3 and OGG both have an adjustable compression parameter.

Compressed File Options

MP3 and OGG files also stream via their **bit rate**, in kilobits per second (kbps). One of the following encoding choices may be set before rendering the file:

- A **Constant Bit Rate** (**CBR**) is used when you are looking for a specific bit rate to ***maintain***.
- An **Average Bit Rate** (**ABR**) is used to attain a specific ***target*** bit rate whenever possible.
- A **Variable Bit Rate** (**VBR**) is used when a target ***quality*** is preferred; in REAPER, the VBR range for MP3 is from 10 (worst) to 100 (best); for OGG, it is 0.0 (worst) to 1.0 (best).

Implementation – A Closer Look

The Importance of Layers

Having multiple layers in a soundscape allows for a rich and interesting listening experience. It doesn't necessarily have to be a *busy* one, just enough to paint a meaningful sonic picture. Even without complex implementation, setting up audio layers gives you an idea of how many channels you'll need, along with their dimensionality (2D versus 3D).

Remember, we are going for *the least amount of audio required that makes our soundscape effective*. For example, if you were planning the soundscape for

Table 12.2 Sketching Out the Audio Layers in a City Building Game

Modern City Builder – Preliminary 2D Audio Layers

Layer	*Category*	*Layer Type*	*Contains …*	*Max. # of Channels*
1	**Music**	Music	Intro, menu screens, gameplay, win/lose	2
2	**Sound**	Ambience: stereo bed	Varies: light wind, city din, light to heavy traffic	2
3		Ambience: mono sources	Varies: animals, insects, occasional cars, trucks, and construction vehicles passing by	3
4		Objects	Building first powers up	2
5			Buildings active	3
6			Road or building cleared	1
7		Characters	Bulldozer to clear roads, buildings	1
8			Vehicles honking, revving engines in traffic	3
9			Character footsteps, other actions	4
10		UI	Alert pop-ups, messages, character text	1
11			Menu buttons and sliders	1
12	**Dialogue**	Tutorial	Tutorial VO (gameplay)	1
13		Characters	Specific VO lines per character, random emotes	1
			TOTAL # Channels required:	**25**

a modern-day 3D city building game, you might have a "laundry list" of audio to integrate.

My short audio list: *"Music, city ambience (cars, trucks, crowds, construction vehicles), UI sounds, and character dialogue. Hmm … how many channels will I need (as a rough guess)?"*

Referring to the layers described in Chapter 7, we can form a more detailed, but still preliminary, list, in Table 12.2.

You may look at this table and think, "OK, but do I really need all of these channels simultaneously?" and if the answer is "no," then you can consolidate the list some more. For example, we may never hear tutorial VO playing simultaneously over character VO, so those channels can be reduced to a single one, bringing our total down to 24. But the game design will likely evolve over time, and audio layers and channels will need adjusting along with it.

Layering Strategies

Recall from Chapters 4 that a **buss** is an audio track that other sources are routed to. They are essentially groups of tracks – central hubs that make mixing easier to manage. When using busses for game audio implementation, you also can think of them as **layers** in your soundscape mix. Recall the analog mixer from Chapter 4; the parent layers ("Music", "Sound FX", "Dialogue", etc.) represent the mixer's **busses**, and child layers ("Monsters", UI", "Player", etc.) represent its **channel strips**. Audio being triggered from the game (known as **channels** or **voices**) are like the audio **signals** being sent through those analog channel strips.

If you'd rather think about it from a REAPER perspective (Chapter 5), the parent layer is like a **Folder**, and the child layer a **track**; also similar to the example above, audio being triggered from the game are like the audio **signals** being sent through those REAPER tracks.

If you have different sound effects for trolls, kobolds, ogres, and bugbears, you could route those four sound sources into a layer called *Monsters*. You might also create an ambience layer for environmental sounds, followed by layers for the UI, player, music, and dialogue.

Now, you can control your in-game mix using just these layers, and a master control much like the master track in REAPER. Everything is controlled by the master, so go easy with it!

Layers can be created in either the audio engine or game engine, and also like REAPER, each can contain their own plug-in effects that can have a direct impact on the soundscape. Using layers not only makes your mixing experience more successful but also gives your mix more "glue" with the ability to harness all volume and effects more easily.

Using our Modern City Builder layers from Table 12.3 as a reference, here is what the layering strategy could look like in the **Unity** game engine. Below is Unity's Audio Mixer set up, using only our main layers (called **groups** in Unity): **music**, **sound**, and **dialogue**; note that the very first layer is called **master**, which is its mandatory main output (Figure 12.7):

… and below, with all of our child layers (subgroups) added to our parent layers (groups) (Figure 12.8):

Music: music

Sound: ambience, objects, reactions, characters, UI

Dialogue: characters. tutorial

Procedural Sound Design

Audio can be looped, mixed, delayed, concatenated, and more, all through an audio engine's event system, the game code, or even the game engine itself in some cases. This process may be in the form of a short script, a graphical timeline with audio files placed on it, or any other system that communicates the flow of time or sequence of events. Unreal engine, for example, uses a Sound Cue editor, as shown in Figure 12.9. Sound Cues can be as simple or as complex as you wish, drawing from a list of several

Table 12.3 Example of Audio Prioritization and Channel Allotments from Children of the Nile

Priority Level (Highest = 1)	*Layer Type*	*Max # of Channels*
1	Music	2
2	Ambience (two needed for crossfading)	4
3	UI – Messages/events	1
4	UI – Other	1
5	Dialogue	1
6	Building sounds	3
7	Figure action sounds	4
	TOTAL # Channels required:	16

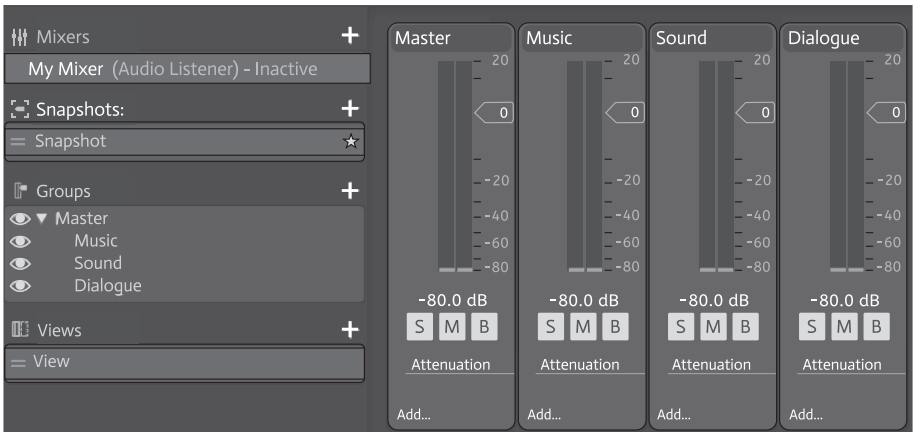


Figure 12.7 The Unity Audio Mixer with the main busses (groups) of our soundscape added. Note the channel hierarchy on the left, under "Groups," starting with the master buss (Unity/Screenshot Excerpt by Keith Zizza).

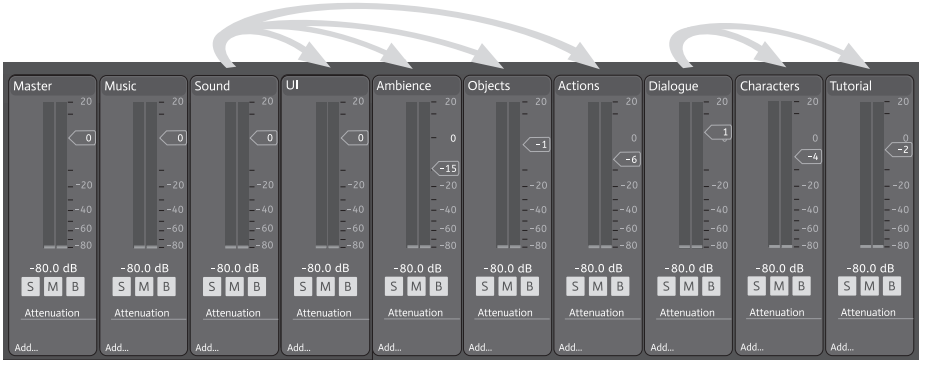


Figure 12.8 Unity Mixer with all layers of our soundscape (groups and subgroups) In this example, the layer Music is on its own with no child layer; Sound is the parent layer of UI, Ambience, Objects, and Actions; and Dialogue is the parent layer of Characters and Tutorial. (Unity/Screenshot Excerpt by Keith Zizza).

components (called nodes) such as random, delay, attenuation, and many more. This particular cue chooses two random animal sounds to play from a pool of six, then randomizes the delay of their playback. A Sound Cue is also great for creating stand-alone, "living" dynamic systems, like the sounds from a tree on a windy day (leaves rustling, birds tweeting, etc.). The logic flows from left to right.

Methods such as these can be thought of as **procedural sound design** (not to be confused with procedural audio from Chapter 8), as it is a "living" system that never unfolds quite the same during gameplay. Procedural sound design is a great technique for more complex audio events such as for the sounds of machinery in a factory;

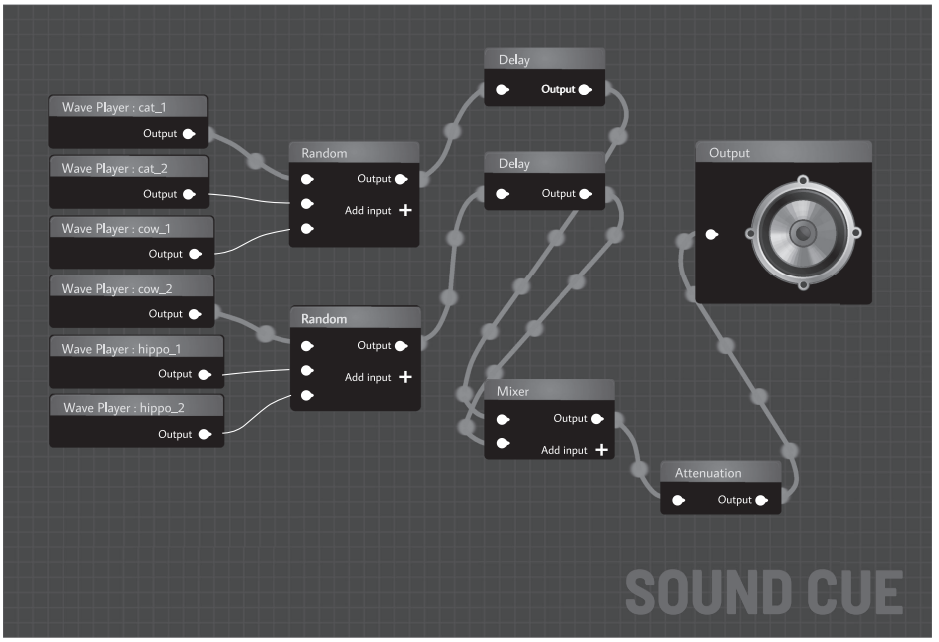


Figure 12.9 A Sound Cue in Unreal Engine. The highlighted paths (lines with spheres) show the flow of events to the speakers in real time. In this case, the flow is from left to right (Epic Games/Screenshot Excerpt by Keith Zizza).

exploring an ancient cavern complete with water drops, wind, and scuttling creatures; or being aboard the bridge of a space freighter with control panel sounds and alerts emanating from all directions.

Priority Handling

Suppose we have 24 audio channels to utilize in a game, to play across a few layers. You really wish you had 32, but you're making do with the resources you have; memory constraints and the current CPU load prohibit any more than that. Also at any time, the game is queuing up 40+ audio files to play. In this scenario, your soundscape may be "starving out" (not allowing to play) lots of necessary sonic material. So, which sound gets to go first?

Setting priority values to your audio guarantees certain elements *will* play, regardless of any traffic jams. Fortunately, audio middleware allows you to set priority values for each layer of audio – including single audio events – which makes soundscape management a lot easier. But whether or not you choose middleware, you should always make a plan for prioritizing your layers or the individual sounds themselves. This can be achieved either by hard-coding the priority values, or by reducing the number of audio requests at any given time (or both).

Here's another layering example for you: in the ancient Egyptian city builder *Children of the Nile*, I had 16 channels to work with (yup, just 16), across 7 layers. I had to create the sounds of a bustling civilization; the soundscape could represent a busy

metropolis center, or that of a lush, animal-filled grassland in the more secluded areas. And upon starting out, there was nothing more than just you and the vast swirling sands of the Egyptian desert.

I knew the music should never just get cut off midstream, so that was priority one (would you want the music to be ripped out from under you during gameplay?). I also wanted an environmental ambience to be priority two, and when you moved around the map, an ambience would have to change from one location to the next (windy reeds in one area and rushing water in another); this meant that sometimes, there would need to be a crossfade between *two* ambiences, and both were in stereo.

Next came the user interface sounds, followed by dialogue, building sounds, and figure actions. The latter three weren't crucial to hear at every moment but played whenever possible. Each provided aesthetic flavor: a sculptor's shop (if active, the sounds of people were heard working inside); outside the building, a sculptor chipping away at their latest limestone goddess creation (looping their action sound); and some sculptor dialogue for entertainment and information ("*You there! Watch out for flying chips … this is a worksite!*" or "*It is really sad, when the people who make our best luxuries are unhappy!*"). In all, there were roughly 60 characters, 40 buildings, 750 sound effects, an hour of music, and 1,600 VO files × 5 languages. Table 12.3 shows the final audio layers and their priorities.

Instance Limits

Determining the number of channels per layer is very important, but you must also consider how those channels are constrained. Should **instances** (repeated sounds) play over each other or interrupt each other?

If there are 30 penguins walking around onscreen in *Cosmic Penguins*, do you really need to play all of their footstep sounds at once? No! You might have two or three instances, and those would be of low playback priority. If laser blasts are going off everywhere, you wouldn't even hear (or care about hearing) tiny footsteps. So those audio resources can be better used elsewhere.

Envision a game where scoring a point generates a "chomp" sound, and it's common to be racking up multiple points very quickly. You'd like two or three instances of that chomp to be heard from start to finish, without being cut off in playback. Now, look at your options in Figure 12.10.

Which version should you hear? Is one better than the other? That's a conversation between you and the design team. Whichever is most important to gameplay and maintaining good feedback is probably the right choice. But this is a very common issue with many games that you'll need to solve, and middleware can help with this.

Challenges in Avoiding Repetition

I am cursed with sensitive ears. Everything I hear, whether it's in or outdoors, I latch on to its pitch, dynamics, and rhythmic patterns, like it or not! I was once given an electronic sleep machine as a gift that boasted having "the sounds of relaxation," like ocean waves breaking on the shore, birds in spring, and the whole bit. Well, whoever edited these files made bad looping points, not to mention each clip was only a few seconds long. The arrhythmic timing of each "scene" in this gadget revealed itself almost immediately and

on infinite repeat. I developed insomnia just from using it! I simply couldn't let my ears pull away from the awkward repetitions. Today, I don't have the heart to even donate this machine, lest another poor soul be tortured by these dreadful "soundscapes".

The same affliction can happen to gamers, especially with character actions.

A detective is investigating an abandoned, haunted mansion. As she walks along the creaky floorboards, how many variations of footstep sounds are needed? When you are walking in real life, no two sounds made from your shoes hitting the ground are ever quite similar; if our detective plays the same, singular footstep sound ad nauseum, when would it become annoying? One very short, *very* unobtrusive footstep sound might be okay to scrape bywith – otherwise, it may require frequent and subtle sonic changes. If you ignore it, you may risk contributing to listener fatigue.

One easy method is to randomize playback of a few footstep sound files, and be done with it. But, if in addition, you subtly randomize the pitch and volume (and even a smidge of delayed start time) of those sound files at runtime, it will make a tremendous difference with sound reusability and listener enjoyment. Now, imagine these enhancements applied to your detective character's walking animation. But why stop there? Schedule and space permitting, different footstep sounds could be employed when walking on dirt, mud, stone, and water. That also goes for running, sneaking, or landing from a jump.

All of these techniques can be implemented using a game engine, either through scripting or visual programming. Or better still, audio middleware such as FMOD or Wwise can do the heavy lifting for you, with ready-made features in place that can mitigate repetition.

[n]th Instances

As in Figure 12.10, how many chomps will you need at once? It could be 1 or 100, it's up to you and your Triangle of Compromise.

Removing Quietest Instances

Remember those 30 cosmic penguins walking around on screen? These are quieter sounds, so they can be selected to disappear if more channels are requested by the game code, for things like laser blasts, explosions, or dialogue.

Removing Oldest Instances

Your battlefield was empty and suddenly, 12 cosmic penguins arrived one by one in pengo-saucers, each with an engine whirring sound. But you only budgeted for ten vehicle sounds at once, so the oldest two are the ones that must stop playing first.

Real-Time Audio Propagation

Harnessing the Soundscape in Real Time

OK, so you've got all of your audio in place, complete with the various layers sorted out. How do you maintain control over this mix during gameplay?

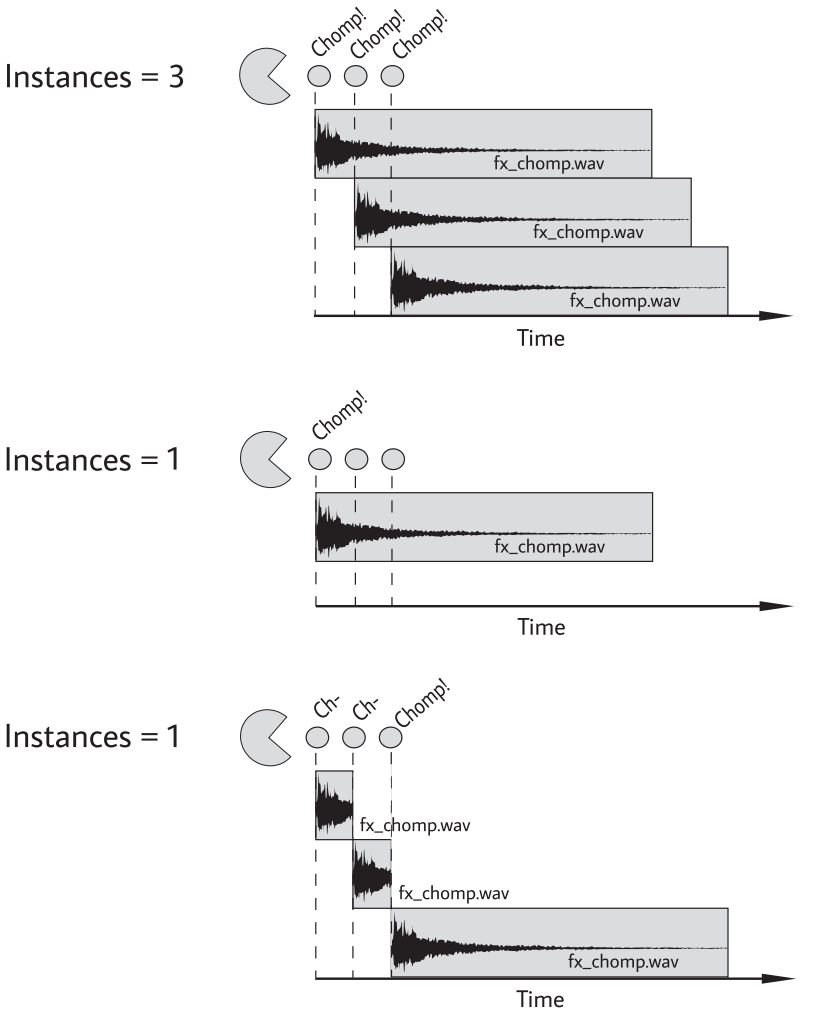


Figure 12.10 The audible results of a "Chomp!" sound effect over time, allowing for three multiple instances (top); one instance favoring the sound's duration (center); and one instance favoring every in-game occurrence (bottom).

What I will continue to call *layers* also means mixing groups, busses, or channels in other middleware and game engines. Every brand will have their own label for it, but they're still layers to us! With that, here are a few tips for mixing the soundscape in real time:

- Recheck the importance of your layers throughout the game development cycle. What may work now for a soundscape may not matter later in the schedule, when design features are added or removed.
- Use as few layers as possible. It will optimize resources but also simplify your mixing efforts.

- Figure out exactly *where* you'll be managing your soundscape mix. Suppose you are using FMOD for middleware, and Unity for the game itself. You can set up all your layers as mixer channels in FMOD, or as mixer groups in Unity. It may make a difference in your programming team's implementation strategy. Fortunately, as of this writing Unity, Unreal, FMOD, and Wwise all have the ability to mix audio layers locally and to add effects to them.
- Consider setting up compressor plug-ins on each layer, but definitely employ one for the Master buss. One common strategy is to have relatively light compression on all layers (2:1 or 3:1 ratio, perhaps, with a minimal threshold), and then rein it in with sturdy compression on the master. But again, every game has different audio needs.
- Set up EQ on as-needed basis per layer. However, nearly every plug-in takes up CPU time during gameplay, and you should save as many CPU cycles as possible for other signal processors like compression and reverb, if need be.

Real-Time Effects Processing

Adding simulated acoustics to your soundscape adds value to the gaming experience. But computing the physics of sound in real time may be too costly to justify, in terms of game performance, processing power, and memory usage.

Reverb Zones

Rather than try to calculate the propagation of reflections off every in-game surface, a room can have a "baked" reverb, essentially an effects plug-in used in real time for a specific area. Normally you can choose which sound(s) are affected by the reverb (e.g., you might only want reverb for character sounds, and not for UI sounds or music). Both audio and game engines offer **reverb zones** that you can adjust to any physical size you need, with multiple zones in your game level. So when you leave a narrow dirt passage and arrive into a large hewn-out cavern, the quality of reverb between spaces will change to fit the scene accordingly.

Other Effects

Effects, EQ, and dynamics processors can all be applied to any layer in your game or audio engine, allowing that processor to work in real time. Perhaps your alien dialogue can be enhanced with an EQ and chorus, as seen Figure 12.11.

Just like in REAPER, an effects chain can be created for this audio layer. The benefit here is that you can adjust all of the alien dialogue effects on-the-fly within the game engine, without having to re-render everything with the new settings. The trade-off, however, is that these effects will take up processing power during gameplay.

Occlusion and Obstruction

You are busy at work exploring a labyrinth, when all of a sudden you hear the shouts of a goblin through the wall on your left; you know it's behind the wall, as it sounds

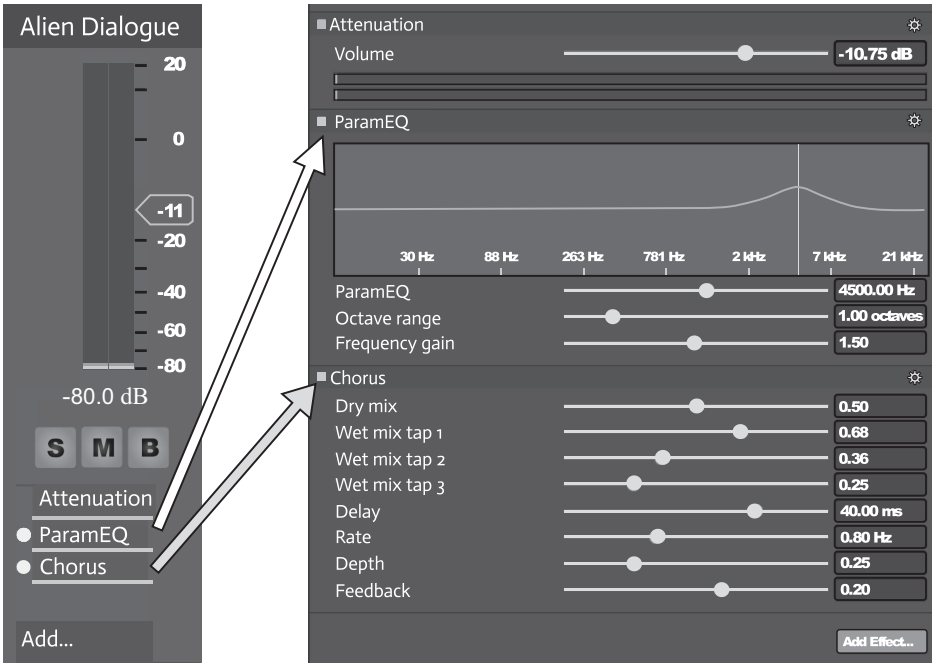


Figure 12.11 Inline effects on are added to an alien dialogue layer, in the Unity Mixer (Unity/Screenshot Excerpt by Keith Zizza).

muffled to your ears. This is **occlusion**, where the source sound and its reflections all pass through a completely enclosed space.

An audio engine can be employed to simulate occlusion on the goblin sound, using an adjustable Low-Pass filter (LPF) based on the player's distance to the wall. The LPF is gradually reduced as the player gets closer to the wall, and they should hear increasingly higher frequencies from the muffled source. This is a popular technique called **Real-Time Parameter Control** (**RTPC**), used for many effects and applications in gameplay, with Low-Pass filtering among the most popular choices.

But wait, what's that? To your right, an evil cyclopean beholder is grumbling! It sounds *somewhat* muffled to you. This is **obstruction**, where the source sound passes through an obstacle, but its reflections are able to bend around it and continue on unaffected. Through implementation via an audio engine, the source sound is again muffled using a Low-Pass filter/volume adjustment, but the reflections that diffract around the obstacle are simulated by increasing their delay values and decay times (and these are not muffled). This type of implementation can get quite complex, depending on the needs of the game, the middleware tool(s) chosen, and the team's available resources. Usually, playing one source sound and two reflections can be enough to make it sound convincing.

Another specific technique often used is called **linecasting**, where if a monster growls, for example, that monster projects one or more invisible lines between itself and the player; the fewer lines that connect between the two characters, the more muffled the audio becomes via changes to an LPF – and, of course, volume control for

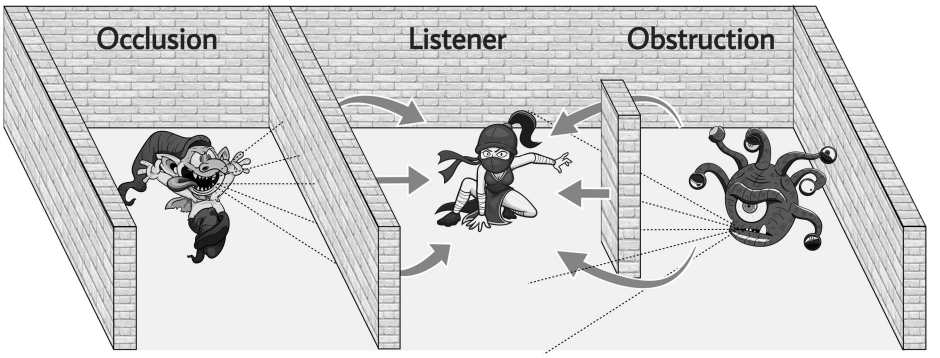


Figure 12.12 To give the illusion of obstruction and occlusion (represented with arrows), linecasting can be used; the more of the enemy's lines cast (shown as dotted lines) that do not make it to the target listener, the more obstructed their sounds become. If no lines make it to the listener, that audio is 100% occluded and completely muffled.

distance measurements to the player. If no lines connect, the sound is occluded and completely muffled, as shown in Figure 12.12.

Additional Implementation Strategies

Smooth Fading and Transitioning

Chances are, you probably don't want sounds just being lopped off when they have to stop playing. Sometimes a graceful sonic entry or exit is in order! Through middleware, fade in/out times can be applied to individual sounds, or to entire layers. This is especially good for dynamic music systems, where instruments may come and go at specific times such as those seen in Chapter 9.

Master Buss Processing

Every mixer or audio routing system should have a master or main output buss, where all other busses arrive. If possible, it's a good idea to add a limiter or compressor here to avoid clipping and distortion (not to exceed 0 dB). It's also a good location to monitor loudness levels (aim for –23 LUFS, as discussed in Chapter 6). Sometimes, a simple volume attenuation is all that's needed.

A final EQ can be utilized on the master buss as well, but I would recommend only very slight corrections here. The most common issues are:

- There is too much low-frequency accumulation (e.g., you have more explosions in your game than anticipated).
- The mix needs "mud" removal (at the 250–600 Hz range, seen in Table 6.1).
- The mix needs brightening (a slight boost beyond 6 kHz, usually as a shallow bell curve, or high shelf).

Frequency Slotting Your Layers

Just as we saw in Chapter 8 with DAW mixing, applying specific frequency ranges to your busses can vastly improve a soundscape mix. In this way, everything has its own place in the frequency spectrum, adding clarity and allowing individual sonic elements to shine through. A common place for frequencies to accumulate is in the lower-midrange/midrange from 250 to 1 kHz, as well as sub-bass/bass frequencies between the 30 and 100 Hz range.

Ducking

Some engines come with **ducking** capability, which allows one audio element or layer to be more prominently heard than others; one or more other layers are temporarily reduced in volume, according to an adjustable threshold. This allows for signature audio moments (an important message to hear during frenetic combat, or the sound of a spell being cast in a busy soundscape).

An attack and release volume envelope are often used, which makes the transition before and after ducking smoother as the effect fades in, then out.

Dynamic Soundscape

The entire soundscape, not just the music, can be intensified as the game progresses forward, or you can have it "breathe" a bit more by adding/removing audio elements from the mix when desired.

The Sound of Silence: Less Is More

As noted earlier, the least amount of audio that comprises the ideal soundscape is all we need; any less and it will feel incomplete; any more than that, and we may have a cacophony. When in doubt, always cut back slightly on the quantity of in-game audio you *think* you need, and it's probably the right amount.

Rules Are Made to Be Broken

Of course, these aren't all the strategies that can be employed. Games are evolving all the time, and so are the integration methods. But hopefully this has inspired you to think up some custom implementation of your own!

Note

1 Stevens, Richard, and Dave Raybould. *The Game Audio Tutorial: A Practical Guide to Sound and Music for Interactive Games*. Routledge, 2017.

Chapter 13

Listening

What Soundscape *Feels* Right?

In the previous chapter, we looked at the various ways that game audio can be implemented. Now, think of a game you've played where you thought the soundscape was spot-on. Why was that so? Did it enhance gameplay, provide you with good feedback, or draw you into its world (or perhaps yes to all three)? Rather than just "knowing" you like a soundscape, it's extremely beneficial to listen to audio in similar games, and reverse-engineer them with your ears. In this way, you can compare many soundscapes against one another, and determine why they are (or aren't) serving the game well. It also helps in evaluating what the competition is doing and provides a means of inspiration if that work is well-crafted.

Critical Listening

Let's say you're going to critically listen to someone else's soundscape. Where should you begin? A good start is to:

- **Break it down** into the three main categories of music, sound, and/or dialogue. Then for each category, write down what you hear using the subcategories listed in Chapter 7.
- Listen to **how each audio element behaves** in gameplay, taking an educated guess at its implementation?
- Roughly determine **how many channels** of audio are in use at any one time. 8? 16? 24? More? At some point your ears will say "that's enough!" if there is a **cacophony** in progress – a blurry, busy mix of audio that's impossible to analyze (but sometimes, this level of auditory madness is done intentionally for effect).
- Listen to the overall **volume** of the soundscape. Does it have a wide or narrow dynamic range that works with the gameplay?
- Listen to the **frequencies** in the mix. Are they distributed well among the audio elements? Is there use of intentional frequency slotting or interesting EQ filters?
- Listen to the **spatial qualities** in the mix. Is there imaging such as 2D panning, or movement of 3D sound elements? Or perhaps an appropriate blend of both?
- Any interesting **effects** within the soundscape, such as reverb, delay, chorus, or other modulation?
- Are the audio elements **utilized properly**? That is, are they distributed in such a way as to compliment the gameplay, or detract from it?

DOI: 10.4324/9781003218821-17

From here you can listen for a whole host of events, including: How many layers of ambience might there be? Does the music incorporate horizontal resequencing and/or vertical reorchestration? Are there multiple characters, and can they talk over one another? And so on. Sometimes it helps to capture the gameplay and evaluate the audio later, or if you're already familiar with the game, try analyzing the soundscape from a preexisting video. You will soon be able to hear the individual components: character actions, environmental ambience, objects, the user interface, etc. The more soundscapes you listen to, the more easily you'll be able to evaluate them.

Of course, not every successful soundscape has to have music, sound, *and* dialogue – and it doesn't need to have balanced frequencies across the hearing spectrum, or have multiple audio elements in play. It just needs to work well for the game and for the player experience.

A soundscape often has varying levels of feedback, immersion, and entertainment at different stages of gameplay, which lends itself to a more dynamic implementation – peaks and valleys of a soundscape's complexity and overall volume level over time. Having a more dynamic soundscape can reduce listener fatigue, and keep things sonically interesting. But it is also helpful in supporting the gameplay's **flow state**, where the player is in a "sweet spot" of their skill level versus the challenges presented to them. They become completely engaged with gameplay and are "in the zone"; time and outside events pass by unnoticed, and the result is a quite satisfying game experience.[1] Audio helps to maintain this engagement level through all of its roles described in Chapters 8–10, working in concert together.

Metaphors in Soundscapes

On the developer side, audio designers don't always have to be so literal with the soundscape construction. It's often what is heard but *unseen* that supplies depth to a game space, and gets the listener's imagination going. Much like reading a book, the reader internalizes the details; written events can feel visceral, even though they cannot be seen or heard. Similarly for audio, it's the listener's mind that fills in the gaps. They may hear bats in a cave, but never see them; or the sounds of beeps and alerts from the inside of a space station, never actually viewing the electronic devices. Or further still, they may hear *metaphorical* sounds that present the listener with a temporary disconnect. Film sound designer Walter Murch once said: "When sounds don't mirror what is actually on screen, [viewers] instinctively know to search for meaning."[2] For example, imagine the player is setting up their character for a combat game. The menu buttons make the sound of punches and kicks; even though a button that sounds like a punch is an abstract concept, the metaphor fits the game and serves as a "warm-up" for the soundscape to come.

Audio in Game Genres

What does the **genre** of a game have to do with its soundscape?

Everything!

Through its design, a **game genre** presents a particular series of **challenges** to the player. A platformer is a vastly different design from a card game, and so their soundscapes will differ greatly as well. But how?

The game's audio content is tailored to the **experience** of that genre: the card game will need sounds for shuffling, dealing, turning over cards, and the like; and the platformer will require jumping, walking/running, and pickup sounds. The pacing and feel of both soundscapes are different, as well as the player's **expectations** about what they might sound like.

Below is a list of games that you can give a serious listen to. Whether or not a soundscape is "good" is in the ear of the beholder, so to speak, but I feel the audio for these games are each representative of their genre. Most of these games cross over multiple genres, but I only allowed one mention per game, leaving room to add other titles. Some genres aren't listed here (like hyper-casual games, which use only a tiny number of sounds, if any), and new ones are popping up all the time. But there's certainly more than enough here to get a head start on your listening chops!

Survey of Genres: Critical Listening

4X (eXplore, eXpand, eXploit, eXterminate)

Age of Wonders, Civilization, Distant Worlds: Universe, Endless Legend, Galactic Civilizations, Masters of Orion, Shadow Empire, Sid Meier's Alpha Centauri, Starborne: Sovereign Space, Stellaris

Action

God of War, Control, Pit People, Ratchet & Clank, Uncharted, Hades, Horizon Zero Dawn, Red Dead Redemption, Spider-Man, Tomb Raider

Adventure

Night in the Woods, What Remains of Edith Finch, Grim Fandango, Day of the Tentacle, Life Is Strange, The Legend of Zelda: Breath of the Wild, Runaway, A Road Adventure, Full Throttle, Stray

Battle Royale/Auto Battler

Apex Legends, Fortnite, Spellbreak, Call of Duty: Warzone, PUBG, Garena Free Fire, Auto Chess, Auto Brawl Chess, The Dungeon Beneath, Teamfight Tactics

Card Game/DCCG (Digital Collectible Card Game)/Deck-Building

Hearthstone, Inscryption, Animation Throwdown: The Quest for Cards, Monster Train, Fable Fortune, SolForge, Magic: The Gathering Arena, Faeria, Kards, Poker Night at the Inventory

Casual

Hay Day, Slime Rancher, Fall Guys, Township, Hidden Through Time, My Talking Tom, A Short Hike, Coin Master, Battleblock Theater, Angry Birds

City Builder

SimCity, Pharaoh, Caesar, Anno, Tropico, Frostpunk, Surviving Mars, Cities, Rollercoaster Tycoon, Zeus: Master of Olympus

Educational

PBS Kids Games, codeSpark, The Oregon Trail, The Magic School Bus, Escape from Adventure Island, Crazy Machines, Super Why, Endless Ocean (too much breathing?), Chrome Music Lab, Big Brain Academy

Fantasy

The Elder Scrolls, Fable, Dragon Age, Final Fantasy, The Witcher, Dark Souls, Monster Hunter, Elden Ring, Path of Exile, Baldur's Gate

Fighting/Beat 'em Up

Super Smash Bros., Castle Crashers, Streets of Rage, The Simpsons, The Warriors, River City Girls, Dragon's Crown, Teenage Mutant Ninja Turtles, Double Dragon, Devil May Cry

Flight/Vehicle Simulation (Flight Sim, Kerbal Space Program, etc.)

Microsoft Flight Simulator, Grand Theft Auto (GTA 5 for vehicles AND planes), War Thunder, SnowRunner, Farming Simulator, BattleTech, MechWarrior, Bus Simulator, American Truck Simulator, Grand Turisimo

FPS (First-Person Shooter)

Doom, Call of Duty: Modern Warfare, Bioshock, Half-Life, Far Cry, Borderlands, Halo, Titanfall, Destiny, Overwatch

Gacha

Genshin Impact, Epic Seven, Fire Emblem Heroes, Marvel Strike Force, Summoners War, Another Eden, Arknights, Raid: Shadow Legends, Monster Strike, Alchemy Stars

Horror

Amnesia: The Dark Descent, Dead Space, Five Nights at Freddy's, Left 4 Dead, Phasmophobia, Prey, Resident Evil (the more recent versions), Soma, The Evil Within, The Last of Us

Life Sim

Animal Crossing, Children of the Nile, The Sims, Stardew Valley, Unpacking, Two Point Hospital, Animal Shelter Simulator, Spiritfarer, Littlewood, Wylde Flowers

Matching (Match-3, etc.)

Marvel Puzzle Quest, Bejeweled, Zuma, Candy Crush, Gummy Drop, Homescapes, Chuzzle, Puzzle and Dragons, Pokémon Shuffle, Royal Match

Metroidvania

Metroid, Castlevania, Hollow Knight, Guacamelee!, Ori and the Blind Forest, Owlboy, Mega Man ZX, Dead Cells, Steamworld Dig, Blasphemous

MOBA (Multiplayer Online Battle Arena)

Arena of Valor, Defense of the Ancients (DotA), Heroes of the Storm, League of Legends, Smite, Strife, Vainglory, Paladins, Awesomenauts, Monday Night Combat

MMO (Massively Multiplayer Online Game)

World of Warcraft, The Elder Scrolls Online, EVE Online, Star Wars: The Old Republic, Guild Wars, Lost Ark, Star Trek Online, Final Fantasy XIV, Blade & Soul, Runescape

Mystery/Detective

Professor Layton, Sam & Max Hit the Road, The Vanishing of Ethan Carter, L.A. Noire, Ace Attorney, Alone in the Dark, Sherlock Holmes: Crime & Punishments, Disco Elysium, Return of the Obra Dinn

Music/Rhythm-Based

Audiosurf, Beat Saber, Beatstar, Sayonara Wild Hearts, Crypt of the Necrodancer, Guitar Hero, Rez, Rock Band, Thumper, Fuser

Party

Mario Party, Among Us, Overcooked, Rocket League, Party Golf, That's You, Gang Beasts, Moving Out, Switch Sports, The Jackbox Party Pack

Platformer

Super Mario Bros., It Takes Two, Rayman Legends, LittleBigPlanet, Oddworld, The Artful Escape, Limbo, Shovel Knight, Cuphead, Psychonauts

Puzzle

Tetris, Portal, Braid, Machinarium, Hidden Folks, World of Goo, Superliminal, Scribblenauts, Gorogoa, Monument Valley

Racing/Driving

Forza, iRacing, DirT Rally, Mario Kart, Crazy Taxi, Skylanders, Rocket League, NASCAR, Gran Turisimo, Hotwheels Unleashed

Retro (Arcade)

Pac-Man, Donkey Kong, Frogger, Tempest, Centipede, Asteroids, Space Invaders, Galaxian, Robotron: 2084, Q-Bert

Retro (Inspired)

Spelunky, Undertale, Super Meat Boy, Towerfall, Hotline Miami, Downwell, Papers Please, VA-11 Hall-A, Timespinner, Loop Hero

RPG (Role-Playing Game)

Diablo, The Outer Worlds, Darkest Dungeon, Deus Ex, Titan Quest, Disco Elysium, Dungeons & Dragons, Yakuza, Chrono Trigger, System Shock

RTS/RTT (Real-Time Strategy/Tactics)

Warcraft, Starcraft, Age of Empires, Homeworld, Warhammer, Command & Conquer, Total War, Company of Heroes, Northgard, Iron Harvest

Science Fiction

Mass Effect, Sid Meier's Alpha Centauri, Endless Space, Star Wars, Death Stranding, FTL: Faster Than Light, Fallout, Marvel's Guardians of the Galaxy, XCOM, Elite Dangerous

Sports

Madden Football, NBA 2K, Super Mega Baseball, NHL, Tony Hawk's Pro Skater, Wii Sports, Nintendo Switch Sports, Windjammers, FIFA, Grand Mountain Adventure

Stealth

Thief: The Dark Project, Metal Gear Solid, Hitman, Dishonored, Assassin's Creed, Ghost of Tsushima, Deathloop, Desperados, Sly Cooper, Echo

Survival

Don't Starve, Valheim, No Man's Sky, This War of Mine, ARK: Survival Evolved, Subnautica, The Flame in the Flood, Surviving Mars, Stranded Deep, Oxygen Not Included

TBS/TBT (Turn-Based Strategy/Tactics)

Lords of Magic, Songs of Conquest, Into the Breach, Age of Wonders, Heroes of Might & Magic, Fire Emblem, The Banner Saga, Battlestar Galactica Deadlock, Endless Legend, Battletech

Tower Defense

Plants vs. Zombies, Dungeon Defenders, Bloons, Kingdom Rush, Orcs Must Die!, Iron Brigade, The Riftbreaker, Sanctum, Defense Grid, Minion Masters

VR-Specific Games

Moss, Vader Immortal, Eleven Table Tennis, Lone Echo, Doom VFR, The Climb, Lucky's Tale, Robo Recall, Half-Life: Alyx, Superhot VR

Soundscape Testing: Common Issues

Depending upon the genre(s), audio footprint, and complexity of your game, the amount of audio that must be tested is highly variable. For example, you could probably count the number of sounds on your fingers used in a hyper-casual game, whereas an epic RPG will have thousands of assets, possibly spanning hours in length.

It's OK to Walk Away

Having a critical ear for soundscape details is important, but after listening to a mix dozens of times, your ears will gloss over certain details, and favor others. In fact, you may find that over a period of several hours without a break, you will become hypersensitive to audio. Everything sounds increasingly terrible, across the board – and that's when you should walk away from your computer for a while.

The key to listening success is to take breaks, perhaps 20–30-minute intervals with 5–10 minutes of doing *something else* (get up and stretch! Take a short walk! Grab a healthy snack!). This will give you at least a tiny bit of objectivity upon return, allowing your ears to "shake off" any specific sonic characteristics it was previously attuned to.[3]

Seek Out an Objective Listener

Gaps in your critical listening checklist may go unnoticed over time, so it's essential to get as many ears on the testing as possible. For example, other people will play the game differently than you anticipated, which could reveal certain audio flaws: sounds not playing properly, dialogue miscued, or the emotional pull of the soundtrack not playing out as expected.

Upon waking the next day, your objectivity will be better but of course, now you have one less day to meet your deadline.

Personally, I find evenings the worst time to do critical listening, because if I'm analyzing the same audio over and over, I end up not being able to sleep, hearing the loops in my head all night (what we call having an "earworm" in the audio world). But to each their own!

The singular use of the word "engine" below means any game or audio engine, unless specified otherwise.

Missing Audio

No Audio at All

If there's no audio at all, check file and folder paths, your computer's main volume level, or your audio interface/mixer output level; lots of times, something just needs to be unmuted in the audio chain.

Some Audio Is Missing

Check that filenames match those in the asset list. Work with your programmer to see if this audio has been implemented yet, and you can both walk through steps, seeing if you can trigger it in-game.

Variations of an Element Not Playing

Same as above, but also looking to see if an array element has been properly defined in code (e.g., that orc_attack_[n] has two elements reserved, for *orc_attack_1.wav* and *orc_attack_2.wav*.).

Channel Dropouts

Your soundscape may reach the maximum number of channels allowed, with certain sounds being "starved out," never able to get into the mix. For example, if your soundscape allows 16 channels to play simultaneously, and a crucial 17th sound never gets a chance to play, consider prioritizing that specific sound over others (as in Chapter 12), or capping the number of instances of other audio types (footsteps, for example).

Clipping/Distortion

A cursory listen to the mix may suggest just lowering an audio layer, or attenuating the master volume.

Volume/EQ Inconsistencies

Go back to each set of audio files and either re-render them at the approximate loudness levels and proper EQ, or make the adjustments within your engine. If using EQ in the engine, be aware this impacts CPU performance.

Performance Lag

Downsize the number of plug-ins used in your engine. Consider baking in (rendering) the processing or effects from your DAW. You can also reduce the number of files or channels, lower the sample rate/bit depth, or experiment with loading them in

memory versus other media storage. Lastly, if they are compressed files (e.g., MP3 or OGG), convert them to an uncompressed format (WAV or AIFF) which does not need to perform decoding of the compressed data, saving some CPU time.

Channel Overlap (Flanging/Doubling)

When multiple, identical instances of a sound play at the same time, they are still often a fraction of a second apart – anywhere from say, 1–50 milliseconds. This causes an unwanted flanging or doubling effect, which should be immediately apparent. Reduce the number of instances via the explanation in Chapter 12 (instance limits).

Excessive Instances of One Element Playing

These instances are far enough apart but are monotonous (for instance, *ui_console_beep.wav* plays five times a second, or more). Add variations (alternate files), vary the pitch and volume of the offending instances in your engine, or reduce the number of instances via the explanation in Chapter 12 (instance limits).

In-Game Effects Not Triggering, or Seem Insignificant

Check the Obvious

Verify that the effect is not bypassed or unchecked in some way; also verify that its wet mix is not 0%.

Compression/Limiting

Try lowering the threshold or input level to a higher negative dB. Increase the ratio of compression. Also check that the wet mix is at least 50% or more (usually 100%).

Ducking

If ducking levels seem off, a common problem is that the attack time isn't fast enough, or that the release time takes too long. Experiment with an ultra-short attack (like 1–5 ms) and a shorter release time of around 100 ms to start with. Also check that the ducking level is low enough that your target audio (like dialogue) comes through the mix clearly, and without making the ducked audio too quiet.

Reverb

Increase the wet mix of the reverb. If there is a Low-Pass and High-Pass filter in the plug-in, make sure the frequency range affected is ideal. Also try lowering the dampening level to let more high frequencies shine through.

General Soundscape Issues

Listening to the soundscape as a whole, here are some general issues that are good to listen out for. **Is the soundscape:**

Cacophonous

Stop the noise! There are too many elements in the mix. In deciding what stays and what goes, evaluate each category of audio, but also the instances of specific sounds. There may be an issue with repetition; do you really need a dozen power-up sounds in a row, or 75 dagger swishes? Consider removing the oldest instances, or the quietest instances in your audio engine. It may also be the case that only a few sounds are playing, but it feels noisy. Rather than remove anything, often times just lowering the volume of one or two layers does the trick (attenuating environmental ambience or music, for example).

Sparse

Your forest is awfully quiet! There are too few elements in the mix. Try adding a looping, long-playing stereo ambience that has subtle presence (A gentle wind, for example). You could also add the sound of a bird or other animal: create three or four variations on that animal, and trigger it to play between every x and y seconds. Better yet, create a few different animals or insect sounds, and vary their volume and distance each time they are triggered. Take it further and change up the sounds for day and night!

Arrhythmic (Unusual Cadence, or Jarring Mix)

Your soundscape should have its own sort of rhythm, pulse, or cadence, call it what you will. It may have sequences that sound full of life but lack a certain, varied pacing. Or, one audio element may cause a jarring mix now and again (a crow that won't stop cawing, for example). Consider altering the timings of any randomly generated sounds, or having the music take some of the heavy lifting in the more silent passages of the game.

Discordant (Lack of Aesthetic Uniformity)

Something sounds "off" about the mix. If it's not a technical issue, it may be that your audio elements don't work well together stylistically. Perhaps the musical palette is misaligned with the visuals, or story. The sound effects might feel like they're aesthetically all over the place. Or, the quality of voice talent varies from Hollywood A-listers to amateur acting. This is worth testing out a lot during development, to avoid costly delays in audio production time.

Repetitive (Not Enough Uniqueness)

Here, you can increase the number of variations of the offending sound(s), making them less predictable. You can also decrease the percent chance to play a particular sound, or an entire sound category. Try also lowering the sounds in the overall mix, or increase the time interval they play in (e.g., a bird call that played every 3–5 seconds could be increased to every 10–20 seconds).

Dull (Low Variety of Textures/Sonic Palette Too Narrow)

Spice it up! Add audio layers that have complementary frequencies, timing, and placement within the soundscape.

Brash (Too Many Textures/Sonic Palette Too Wide)

Conversely, pull back on the simultaneous number of audio layers, specifically overlapping areas of frequency, timing, and placement of soundscape elements.

Too Bright or Dark

This may be a matter of boosting/cutting EQ for a particular audio layer in the audio engine, or in the case of everything being too bright or dark, boosting/cutting EQ in the audio engine's master layer.

Lacking/Overdoing Dynamic Range

Increase or decrease specific layer and/or master layer compression levels, specifically adjusting the threshold values, and carefully adding make-up gain as needed, if that is an option on your particular compressor plug-in.

Spatially Imbalanced

Reconsider the number of 2D versus 3D layers in the soundscape; increase the radius of fade in/out when moving between ambient zones (area loops), or moving around the ambiences of localized objects (source loops); also check the panning of spatialized elements – are they in the right locations (UI buttons, for example)?

Lacking Context

Sound as a metaphor is one thing, but a soundscape that's frequently mismatched with visuals can be distracting and draw attention to itself in the worst way. You may need to reevaluate your soundscape, and have some conversations with the game designer on how to make a new audio assessment.

Alternate Mixes

There are a multitude of ways to listen to game audio: through earbuds, headsets, headphones, surround speakers, and everything in between. How should a soundscape present itself among all of these options?

If you're preparing your own soundscape, consider at a minimum a stereo headphone mix. Gamers enjoy immersive sound and nuanced detail through headphones, thanks to high-quality speakers at a close proximity to the ear, and with recent advances in spatial audio technology, players can also enjoy more accurate 3D positioning of audio elements, through formats such as Dolby Atmos, DTS, and Windows Sonic.

Stereo speakers would be next in line, followed by surround speaker options. At the present time, most games do not offer multiple listening options – yet. But I believe there will soon be more choices, as gamers' ears are becoming increasingly appreciative of such great soundscapes, with each new game release.

When mixing for game audio, it's good practice to listen in headphones first, followed by external speakers. However, you should strive to use the most accurate

headphones possible (the flattest response from 20 Hz to 20 kHz). Later, you should also evaluate the mix on one or two "middle-of-the-road" headphones, to hear what the average consumer may be hearing. Many off-the-shelf headphones have a low-frequency bias that kicks up the bass. They may sound great for music, and some games, but they're unacceptable for professional mixing. How can you mix accurately if your reference speakers have a tweaked frequency response?

After that, an accurate pair of reference monitors is crucial for mix evaluation, and as with headphones, you should also qualify your mix with at least one or more pairs of consumer-grade speakers.

Now, go forth and listen!

Notes

1 Carr, Diane. Essay. *In Computer Games: Text, Narrative and Play*, 56–57. Cambridge, MA: Polity Press, 2014.
2 O'Falt, Chris. "Walter Murch: The Godfather of Modern Sound Reflects on a Career That Changed Filmmaking." IndieWire. IndieWire, April 30, 2019. https://www.indiewire.com/2019/04/walter-murch-godfather-modern-sound-making-waves-apocalypse-now-tribeca-1202129800/.
3 Similar to the **Pomodoro Technique** by Francesco Cirillo: https://francescocirillo.com/pages/pomodoro-technique.

Bibliography

Stevens, Richard, and Dave Raybould. *Game Audio Implementation: A Practical Guide to Using the Unreal Engine*. New York, NY: Focal Press, 2017.

Chapter 14

Finding Game Audio Work

About once a week or so, I get an email from someone asking how to get a job in the videogame industry as an audio designer of some sort. Well, at last I am writing an official chapter about it! Of course, some of the things I'm listing here are subject to change, but most of it is bedrock-solid (being a kind and generous person, for example). So without further ado, here we go!

Look Out, Studios, Here I Come!

The videogame industry is such a dynamic, exciting world to be a part of. Now more than ever, there is an explosion in the demand for game developers of all disciplines, audio designers included. That also means there is fierce competition for these jobs, among aspiring and seasoned professionals alike.

One must have not only a great portfolio but also maintain a professional online presence and good networking skills. Whether you're a freelancer or potential employee in game audio, you need to be fully prepared when the right opportunity presents itself. In fact, this is the **professional definition of luck**:

Luck = Your Preparedness + The Right Opportunity

I sometimes hear, "oh, so-and-so was lucky to get that audio gig!" *Everyday* luck had nothing to do with it. *Professional* luck had everything to do with it. They were totally prepared when a well-matched position became available.

Building a Portfolio

The first thing to do is gather your materials! Before you begin a search for work, you absolutely need a home for all of your content. That means building an **audio portfolio** of some kind, consisting of the discipline(s) you wish to showcase.

Previous Works

Listen to your past efforts and determine which of those are ready to share. Your portfolio should ideally consist of **curated work** – that is, you're presenting your very best work and holding back on those not-so-great selections. You can always re-curate the portfolio as time goes on.

DOI: 10.4324/9781003218821-18

If you have the time and resources available to improve a previous work, now is the time to do so. Review the work and listen to what should be modified. However, don't get hung up on small details – the employer may not realize the difference between the original and new content, and you should instead spend your time rounding out the portfolio itself.

Creating New Materials

If you're just starting out, and you have no previous experience professionally, you'll need to create a few demos for your portfolio. Suppose you're a soon-to-be college graduate. What material do you have? Any projects or assignments you're proud of?

If game audio is truly your passion, you should be creating work beyond what is assigned to you. You should get up every morning and feel like creating (well, most of the time, anyway – we all have our bad days). Below are some ideas to consider adding to your portfolio:

1 *Sound Design/Music Demos*
   Either one of these is a montage of the work you've done either personally or professionally. I recommend not having a demo exceeding three minutes. We're all busy people, and if a prospective employer is checking out your demo, you need to sell it in the first few moments – so have your very best material at the beginning. Many audio jobs have hundreds of applicants, and an employer's time is limited!
2 *Soundscape Reskins*
   Record or find a vidcap of a game you'd like to recreate the soundscape for. As we covered in Chapters 7–10, think of each object, character, actions and reactions, etc., as its own component in the soundscape that may repeat multiple times. Simulate how an audio engine would handle a character's footsteps or attacks with a small set of sounds, varying pitch and volume for effect. Add enhancements such as compression, reverb, delay, or modulation. Follow an object's path on (or outside of) the screen with panning and volume. The possibilities are up to you! About one minute is suitable, focusing on some solid gameplay, or at least a part of the soundscape that will be an impressive reskin. A title at the front of the video and a fade-out at the end is also a nice touch.
3 *Demo Using an Audio Engine*
   Here is a chance to show a prospective employer your expertise in an audio engine such as FMOD or Wwise. Demonstrate techniques like horizontal resequencing and vertical orchestration, or an adaptive ambient soundscape, even without integrating it with an actual game. For instance, Wwise has certifications for sound design, music, and optimization; you can go through the tutorials for free (there is a fee if you want official certification), and set up your own systems on their own. Don't forget to make a video walkthrough of your creation!
4 *Soundscapes within Game Engines*
   If you have experience in Unity, Unreal, Godot, GameMaker, or other game engine, a video demonstration of your audio systems is another impressive-looking and sounding option. For example, setting up an ambient soundscape in Unreal using Ambient Sound actors and Sound Cues (engine-specific features).

5 *Rescoring the Soundtrack*
Without an accompanying video, you could take on your favorite game and redo the soundtrack from the ground up. Just an in-game piece or two will suffice, along with a win/lose tune, if any.
6 *Dialogue for Characters*
If dialogue creation or editing is your go-to skill, show it off with a rerecord of characters in gameplay, or offline in a video demonstrating interactive/adaptive dialogue in an audio engine. There are also several available programs that allow you to construct dialogue trees and demonstrate a walkthrough. Free open-source software that supports trees and voice files includes Yarn Spinner, Twine, and Ren'Py.

Website

Your professional website should immediately show who you are, what you do, your portfolio, and how to contact you. This can take on many forms, and should also reflect some of your personality. I have a site that uses large square buttons for any particular demo or project a potential client would like to listen to, and to get more information about. But that's just my take on it – yours will look entirely different.

There are plenty of free website builders out there, including Wix, SquareSpace, and Weebly. Some require a monthly or annual fee for full access, just check to see which one works best for your current needs (you can always upgrade later! Don't pay for extra space or features you don't need right now).

Building Your Image

Social Presence

It's crucial to have an online, social connection to the larger world, as well as an in-person one. Having a profile on Facebook is fine, but you should really use a professional site such as LinkedIn or Indeed when you connect to folks from the videogame industry.

Clean Up Your Act

You might want to take down those social media photos of you partying it up. Most employers are going to do some due diligence (i.e., do some online research about you) before they interview or hire you. If there are any questionable images or writings you've posted, they may decide to pass on your candidacy. Again, they probably don't know anything about you, other than what exists online and what you've sent them.

Streamline Your Image

If you have a website, and one or more alternate sites showcasing your work or credentials, you should make the effort to keep them consistent. At the very least, have the same imagery or format across all vertical channels (e.g., all websites). This goes

a long way in showing you're organized and maintain your social presence, and also that you're conscientious about how you project yourself to the larger world.

Networking

Networking is not only about meeting new people but also about cultivating and maintaining new relationships with people in the videogame industry (and beyond). To start with, there are all sorts of online and in-person events to visit including game jams, meetups, postmortems, expos, and conventions.

On Kindness and Generosity

When you think about it, it costs nothing to be kind. In any situation personal or professional, it goes without saying that you should treat people the way you wish to be treated – with **kindness** and **respect**. **Generosity** is another underrated trait. Whether it's sharing knowledge, going above and beyond for a friend or acquaintance, or just sharing your time, your generosity helps others in countless ways. True generosity means you should expect nothing in return; even so, for you it has the potential to reduce stress and foster a sense of community, among other positive benefits.[1] In the world of game audio, your generosity can take the form of teaching, mentoring, or giving back to the game community-at-large in some other way.

Game Jams and Meetups

A **game jam** is an event where teams or individuals create a new game, held over a series of days or even just a few hours (there's nothing quite like a deadline to get you moving!). It's a great place to meet new people and to hone your game audio skills, and if all goes well, you'll have a respectable project to add to your portfolio. You should regularly check your local game community calendar for upcoming game jam events. A great place to start is on **itch.io/jams**, where anyone can host or join a game jam. As of this writing, more than 230,000 game jams have taken place there.[2]

Game meetups often include a time for social mixing, along with a talk from a game developer speaking on a specific topic. Sometimes a **postmortem** talk is given, which is a look back at a game release: its overall reception, what about it was successful/unsuccessful, and what was learned from it all.

Leveraging and Making Connections

Some folks are extroverted and seemingly make new connections with ease; others are more averse to in-person or even on-camera events. Regardless, making new connections is something you must do in order to maintain a presence in this industry. Your work will eventually speak for you, but not until you've amassed some credits first. Take the time now to make meaningful connections, both in-person and online. Having a large network of professional colleagues and acquaintances allows you to do things like share your portfolio pieces to a wide audience, inquire about work, or help another using a "signal boost", i.e., sending their message through to your

network. The community does its best work when they share knowledge and help their fellow (and prospective) members!

Expos and Conferences

Years ago, videogame industry **expos** tended to be more about showcasing games and products, and **conferences** about talks, panels, and meetups. Today, these terms are somewhat interchangeable. Gatherings such as PAX (Penny Arcade Expo) is a large festival of video and tabletop game exhibitors, as well as game merchandising. They also host panels, talks, and events such as multiplayer and tabletop gaming tournaments. The GDC (Game Developers Conference) also hosts talks and panels and provides a place for networking with developers, catching up with old friends, and even forging new partnerships.

Applying to a Game Studio

Finding Work

In addition to social and professional connections, locating new opportunities online has been getting easier in recent years, as sites such as SoundLister and DevBrada (links at end of chapter) collect and organize game audio-specific jobs found from all over the web. But while demand for game audio professionals has increased, so too has the number of capable people applying for the work.

The Résumé

If you don't have a lot of professional experience, just state what you have already done in school, your side projects, and so on. Never try to "puff up" your résumé – if someone finds out you're exaggerating or even lying about a job or credentials, you'll disgrace yourself and never find work in this industry (or any other industry, for that matter). Keep it concise at just a paragraph for a particular job, role, or relevant experience. A full one-page résumé is ideal, or two pages at the maximum. Game audio work of any kind gets first priority near the top. Check out the example on the **Companion Website** to get an idea of a preferred résumé.

The Cover Letter

Here is a chance to state your interest in the role, and briefly summarize your experience and skills. Do you know anything about DAWs (like REAPER)? Add it to the letter. Did you edit dialogue for a school project? Mention that too. I have also posted a sample cover letter on the **Companion Website** for your review.

The Materials

Verify that your portfolio is where you want it. It's often worth tailoring it to a specific job you're looking for, so long as it's truly the position you want. For example, you may be both a composer *and* a sound designer, but if the role is exclusively for

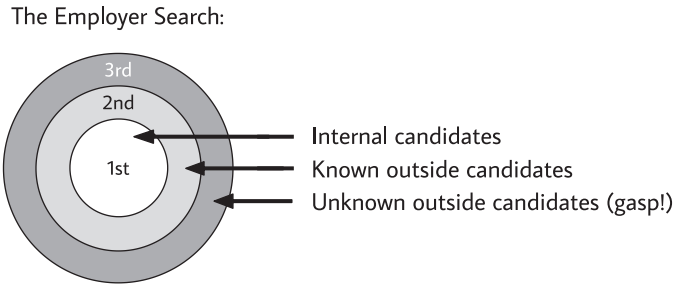


Figure 14.1 From the employer's point of view, hiring an internal candidate is the easiest and most trusted choice. Next are outside candidates the company knows of, and lastly, the most resource-intensive search is for unknown external candidates.

sound design, emphasize sound design skills in your portfolio. It's most often the case that sound design roles are exclusively just that, and not a music position – so the interviewer may dismiss your application, thinking that you'll want to be a composer as well. Remember, *they don't know anything about you other than your résumé, cover letter, and what's in your portfolio.*

From the Employer's Perspective

The employer is looking for the best candidate for their position, and doing so as efficiently as possible. Think of their search as a series of concentric rings (Figure 14.1). The innermost ring represents the first and most trusted choice – the employees who already work there: "who at our company would like this job?" If there are no takers, the next ring is chosen, reaching a bit further out: "who do we know externally who'd be great at this job?" Still no one? Lastly, the outermost ring is chosen. This expends the most time, cost, and effort for them: "who on Earth (literally) is the right candidate for this job?"

The Interview Process

First Interview

The first interview is normally a meeting with Human Resources, verifying your résumé information, and when you'll be available for the job should you be chosen. You may also be meeting with the audio director or other creative lead to get a sense of your qualifications, and also your personality.

Second Interview

The second interview will be with additional audio personnel, if any, and also with a member of the programming, art, and/or other personnel from various departments. I highly recommend (OK, I insist!) that you check out the company website and their games; make sure you are at least *familiar* with them! And get to know one of those games very well. You can watch gameplay videos, or play one of them yourself. Study its soundscape, so

you have something to share with the interviewers. They will likely ask you if you know about their games, and having some talking points here is a big advantage.

Audio Test

You must also be prepared for an **audio test**, though not all jobs are going to require one. It usually consists of a short synchronization of sound design (and/or music and dialogue) to a cutscene, character animation, or a short video capture of gameplay. In rare cases, you may actually be paid for your time in creating the work, if it is substantial, but it is almost always done for free. Sadly, there is also the occasional bad actor that profits from your free work, as they might use it on their upcoming project unbeknownst to you.

Say Thanks

Make sure you thank the interview team the same day after speaking with them; it shows that you are considerate and thoughtful. It should be a brief email thanking them for the interview and their time, and should show your interest and enthusiasm for the role. You'd be surprised at how many interviewees *don't* do this.

To Call or Not to Call ... That Is the Question

It is perfectly acceptable to email or call someone from the studio *once*, if you haven't heard back in a couple of weeks. If you don't hear back, then one more time about four to five weeks out is OK, but any more than that, and you will come off as impatient, or obsessive. After several weeks with no response, it's safe to say they have moved on. Obviously, it would be nice to get a formal rejection notice, but with busy schedules and the sheer volume of applicants, this is not always the case.

Do NOT Work for Free

This heading should be plastered over two pages: DO NOT WORK FOR FREE. It undermines everything audio designers have worked for over the years, notably our value, integrity, and importance. Working for free doesn't just lower the bar on pay, it removes it. By doing so, you enable employers to leverage you against other talented folks who require compensation. No matter who advertises it, internships and junior positions should always be paid.

Perseverance

Overcoming Imposter Syndrome

Have you ever felt like it's only a matter of time before you're "found out"? Or that the more successful you become, the harder it is to "put on an act"? This is a classic **imposter syndrome**, where you feel irrationally insecure about your own talents – and have the feeling that everyone is better than you at your particular craft. Don't worry, you're not alone! Despite their great accomplishments, famous individuals such as

Maya Angelou, Albert Einstein, Tom Hanks, and Serena Williams all have (or had) imposter syndrome as well.[3] The next time you feel this way, just remember: **your emotions are not facts**.

Those who suffer from imposter syndrome tend to overwork themselves, and never give themselves credit for their achievements. Instead, they keep moving the bar of success further away. As clinical psychologist Dr. Jessamy Hibberd mentions in her book *The Imposter Cure*: imagine you're at the end of your life, and you ask yourself what you've accomplished. Would you be happy with it, and were you being true to yourself?[4] It's okay to accept who you are, to be compassionate to yourself, and to forgive yourself for failures along the way.[5] These are learning experiences, something we call "failing forward" in game development. So keep going, you've got this!

Rejection Is a Learning Experience

J.K. Rowling, author of the blockbuster Harry Potter series of books, was rejected 12 times before a publisher took a chance on her work. Elvis was told by the owner of the Grand Ole Opry he'd be better off driving a truck instead of performing.

The same goes for rejection from a game studio. Most studios will send a form message letting you know they didn't choose you for the role. Some may "ghost" you, but this is typically due to their being frantically busy, or that there were simply too many candidates to reply to. Don't let this get you down. In fact, this is a terrific learning experience for you. Look back and review: do you have a solid portfolio? You may have to go back and curate it again, or perhaps create some new material. Also, how are your interview skills? Consider practicing with a friend or significant other. As painful as it may be to watch, sometimes a video recording of yourself being mock-interviewed can really give you some serious perspective!

Side Skills

Aside from game audio, there are also many side skills to consider picking up, including game and level design, scripting, programming, testing, writing, production, and at least a familiarity with game art and animation. Obtaining one or two of these disciplines can be very helpful in broadening your knowledge of game development as a whole, and perhaps even streamlining how you create and produce game audio. Many free tutorials also exist online, and there are relatively inexpensive online courses as well.

Resources

A list of available resources for game audio is never complete, as more opportunity and information pops up every day. That said, here is a short list to get you started on the right path!

Education/Instruction

Epic Games Learning Library: https://dev.epicgames.com/community/learning?application=unreal_engine

Game Audio Learning Portal: https://www.gameaudiolearning.com

Game Audio Institute: https://www.gameaudioinstitute.com
GameMaker Tutorials: https://gamemaker.io/en/tutorials
GDC Vault: https://www.gdcvault.com
Godot Tutorials and Resources: https://docs.godotengine.org/en/stable/community/tutorials.html
The REAPER Blog: https://reaperblog.net
School of Video Game Audio: https://school.videogameaudio.com
Unity Learn: https://learn.unity.com

Organizations

Audio Engineering Society: https://aes2.org
Game Audio Network Guild: https://www.audiogang.org
International Game Developers Association: https://igda.org

Job Listings

LinkedIn: https://www.linkedin.com
Indeed: https://www.indeed.com

Game Audio-Specific Job Listings

DevBrada: https://www.devbrada.com
Soundlister: https://www.soundlister.com

Meetups

Game Developers Conference: https://gdconf.com
GameSoundCon: https://www.gamesoundcon.com
The Develop Conference: https://www.developconference.com
PAX: https://www.paxsite.com

Websites

A Sound Effect: https://www.asoundeffect.com
Game Developer: https://www.gamedeveloper.com
Gamedev.net: https://gamedev.net
Game From Scratch: https://gamefromscratch.com

Additional Resources

A larger list of resources can be found on the **Companion Website**. Have fun and let me know if there's something new to add!

A Final Note

For those looking for work: you can do this. You have **agency**. You have **determination**. You wouldn't be reading this if you didn't! The road to becoming a professional

in the videogame industry can be quite a bumpy ride; to become a *game audio* professional within it is even more difficult than that. Many jobs never get past the internal candidate phase, so they are filled unbeknownst to you. Keep honing your portfolio, meeting new people, and learning as much as you can. Eventually, if you're determined enough, I believe you can succeed at anything you set your mind to. Take good care, and good luck!

Notes

1 Brower, Tracy. "Giving Is Good-for Others, But Also for You." Forbes. Forbes Magazine, December 5, 2021. https://www.forbes.com/sites/tracybrower/2021/11/28/giving-is-good-for-others-but-also-for-you/?sh=6957dcfd23a1.
2 https://itch.io/jams
3 Willis, Thomas. "18 Famous 'Imposters.'" The-Culture-Experts. Phoenix Performance Partners, March 1, 2021. https://www.phoenixperform.com/single-post/18-famous-imposters.
4 Hibberd, 98–99.
5 Hibberd, 208.

Bibliography

Hibberd, Jessamy. *The Imposter Cure: How to Stop Feeling Like a Fraud and Escape the Mind-Trap of Imposter Syndrome*. New York, NY: Octopus Publishing Group, 2019.


Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Index

Note: **Bold** page numbers refer to tables; *Italic* page numbers refer to figures and page numbers followed by "n" denote endnotes.