Computational Thinking through Modular Sounds Synthesis

Andrew M. Olney

2022-09-08

Contents

Welcome

This is the official website for "Computational Thinking through Modular Sound Synthesis". This book will teach you computational thinking through modular sound synthesis (hereafter *modular*). You'll learn how to trigger sounds, create sounds, and modify sounds to solve specific sound design problems and create compositions. Along the way, you'll learn computational thinking practices that transcend modular and can be applied to a variety of problem-solving domains, but which are particularly relevant to information processing domains like computing.

If you're wondering whether this is a book about computational thinking, or a book about modular, the answer is both: on the surface, most content is about modular, but computational thinking is a style of thinking reflected in the presentation of the material and gives it additional coherence. As you work through the book, you'll become more proficient in computational thinking practices like decomposition, algorithmic design, evaluation of solutions, pattern recognition, and abstraction.

This book is *interactive*, which is why it is an e-book rather than a paper book.[1] Throughout you will encounter examples, simulations, and exercises that run in your browser to demonstrate and reinforce key concepts. Don't skip the interactive activities!

[1] I've also created PDF and EPUB versions; these are best used when you don't have internet because they are **not** interactive.

Chapter 1

Introduction

1.1 Why this book?

Let's start with why I'm writing it. I got into electronic music in the 1990s when I lived in London but never transitioned from DJing to making music, though several of my friends did. A few years ago, they started talking about modular, and in talking to them and trying to find out more about it, I realized a few things:

- The best books (to me) were from the 1970s and 1980s[1]
- Modular synthesis is really well aligned with *computational thinking*

If you've never heard of computational thinking and/or modular, that last point won't make a lot of sense, so let's break it down.

Modular sound synthesis (modular) creates sound by connecting modules that each perform some function on sound. Different sounds are created by combining modules in different ways.

Computational thinking creates runnable models to solve a problem or answer a question. Models can be scientific models (e.g. meteorology), statistical models (e.g. statistics/data science), computation models (computer science), and perhaps other kinds of models.

How are they connected? Modular involves computational thinking when we:

[1] Old books that I like are Crombie (1982) and Strange (1983). Newer books of note are Bjørn and Meyer (2018), which gives a great overview of module hardware and history, Eliraz (2022), which gives a broader overview of issues related to musical equipment and production, and Dusha et al. (2020), which gives a modern but briefer introduction to modular than the older books. There are also some online courses (paid), but since I haven't taken them, I'm not listing them here.

- Simulate an instrument by reverse engineering its sound
- Create new sounds based on models of signal processing

Why should you read this book? This book is about modular, but it approaches modular in a way that highlights computational thinking. I believe this deeper approach to modular will help you do more with modular, other synthesizers, and studio production tools. Additionally, the computational thinking approach should help accelerate your learning of computational-thinking domains in the future. Since computational thinking involves problem solving, this book is full of interactive activities that will let you hone your modular skills - something you won't find in most books!

The next sections give some background on computational thinking and modular to better explain where this book is coming from.

1.2 Computational thinking

Tedre and Denning (2016) present a nice overview of the history of computational thinking. Here's a brief summary.

When the field of computing was taking off in the 1950s, there was interest and discussion about how it was different from other fields (e.g. math). One argument was that computing involved *algorithmic thinking*, which is designing algorithms to solve problems (cf. programming), and this kind of thinking was unique to computing. Some even thought that this kind of thinking could improve thinking generally.

Computational thinking appears to have been coined in the 1980s by Seymour Papert and popularized in his book *Mindstorms* (Papert, 1980). Papert was a mathematician by training, and his approach was much broader than the algorithmic thinking approach that came before. Papert's approach was empirical and embraced model building, which he implemented using simulated microworlds containing robots (LEGO Mindstorms takes its name from this work). It was revolutionary in its time and received a lot of attention from educators and policy makers of widely different backgrounds.

Unfortunately, today it's very hard to get agreement on what computational thinking is, so definitions tend to be squishy. This is likely due to the widespread use of computers and the tendency for everyone to frame computational thinking in terms of what *they* do with computers. Some want to reduce it to computer literacy, others to basic programming, and yet others to discovery learning with computers, etc.

I take a more unified view of computational thinking based on model building and problem solving. I define computational thinking as building a *runnable* model to solve a problem:

- For an algorithmic problem, this is a model of computation (the original computer science view)
- For data science/statistics, this is a statistical model
- For general scientific fields, this is a scientific model of a phenomenon or process

The model doesn't need to run on a computer, but to be a runnable model, it needs to be mechanistic. One of my favorite examples of a non-computer model is MENACE (Michie, 1963), which plays tic-tac-toe (AKA noughts and crosses). MENACE plays tic-tac-toe using matchboxes full of colored beads as shown in Figure 1.1. Each possible board position (starting with a blank board) is represented by a matchbox, and each move is represented by one of nine colored beads. To make a move, a human assistant selects the correct box for the current board position and randomly samples a colored bead, which determines where MENACE makes its move. If MENACE wins the game, the chosen bead from each box is replaced along with extra beads of the same color, and if MENACE loses, the chosen bead is removed. Over time, these bead adjustments make winning moves more likely and losing moves less likely.

Figure 1.1: Machine Educable Noughts and Crosses Engine (MENACE). Each matchbox corresponds to a possible board position and is full of colored beads corresponding to moves. The color key in the foreground shows the board location indicated by each colored bead. Image © Matthew Scroggs/CC-BY-SA-4.0.

MENACE is a nice example of computational thinking without computers because algorithmic game playing has a long history in computer science and AI. However MENACE is not "an exception to the rule" - teaching computer science without computers has been part of the model curriculum for almost 20 years (Tucker, 2003; Bell, 2021). We really don't need computers for computational thinking!

So how do we *learn* to build runnable models to solve problems (i.e., how do we learn computational thinking)? Well, models are made of interacting elements, so we need to learn those elements, and we need to learn how the elements interact.

Once we know those things, we can customize general problem solving, which has the same basic steps (Polya, 2004):

- Understand the problem
- Make a plan
- Implement the plan
- Evaluate the solution

For any new domain, the big things to learn are the "understand the problem" and the "make a plan" steps of problem solving. That's the approach of this book - for the domain of modular synthesis.

1.3 Modular synthesis

While we can pinpoint the invention of modular synthesis with some precision, it is useful to consider it in a broader context. This section briefly overviews the history of synthesis and how modular fits into it.

Humans have long been interested in musical instruments that incorporate automation or in reproducing sounds by mechanical means. Wind chimes, which play a series of notes when disturbed by wind, appeared in the historical record thousands of years ago. Even before the complete electrification of instruments (synthesizers are electric by definition), there were numerous attempts to partially automate or model sounds, such as barrel organs, player pianos, or speech synthesis using bellows (Dudley and Tarnoczy, 1950) as shown in Figure 1.2.

Consider the difference between wind chimes or a player piano and this speaking machine. Neither of the former is a model of the sound but rather uses mechanical means to trigger the sound (later we will refer to this as sequencing). In contrast, the speaking machine is a well-considered model of the human speech mechanism.

Synthesizers using electricity appeared in the late 19th century.[2] Patents were awarded just a few years apart to Elisha Gray, whose synthesizer comprised simple single note oscillators and transmitted over the telegraph, and Thaddeus Cahill, whose larger Telharmonium could sound like an organ or various wood instruments but weighed 210 tons!

[2]There is some difference of opinion on what qualifies as usage of electricity in this context. For a fuller history of synthesizers, see https://120years.net/wordpress/

Figure 1.2: YouTube video of Wolfgang von Kempelen's speaking machine circa 1780. Image © Fabian Brackhane.

The modular synthesizer was developed by Harald Bode from 1959-1960 (Bode, 1984), and this innovation quickly spread to other electronic music pioneers like Moog and Buchla. The key idea of modular is flexibility. This is achieved by refactoring aspects of synthesis (i.e. functions on sound) into a collection of modules. These modules may then be combined to create a certain sound by patching them together and adjusting module parameters (e.g. by turning knobs or adjusting sliders). An example modular synthesizer is shown in Figure 1.3.

In the 1970s, *semi-modular* synthesizers were developed that did not require patching to make a sound. Instead, semi-modulars were pre-set with an invisible default patch, meaning that the default patch wiring was internal and not visible to the user. Users could then override this default patch by plugging in patch cables. Most semi-modulars from this period also included an integrated keyboard. Arguably, theses changes made semi-modulars more approachable to typical musicians. An example semi-modular synthesizer is shown in Figure 1.4.

Digital technology began replacing the analog technology of synthesizers in the 1980s. As a result, synthesizers got smaller and cheaper. Digital synthesizers made increasing use of preset sounds so that most users never needed to create custom sounds. In comparison to digital synthesizers, modular synthesizers were more expensive and harder to use. An example digital synthesizer is shown in Figure 1.5.

By the 1990s the digital transformation was complete, such that computers could be used to create and produce music in software. Although computers were still relatively expensive at this time, they provided an all-in-one solution that included editing, mixing, and other production aspects. Over the next few

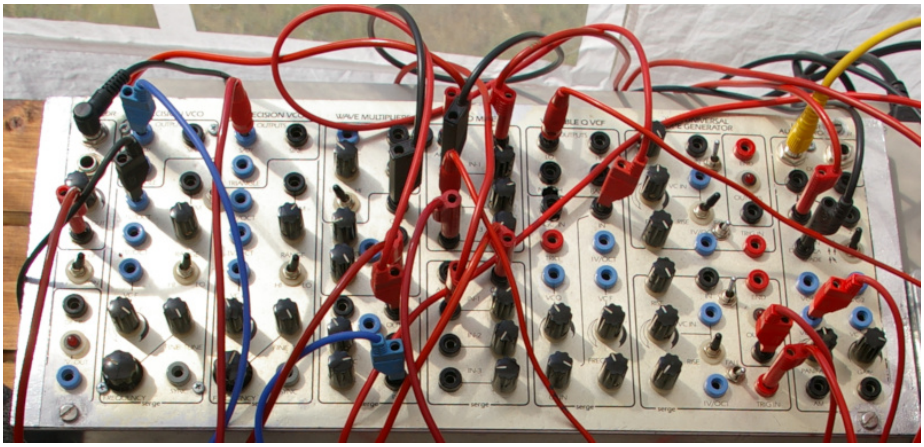

Figure 1.3: A Serge modular system based on a 1970s design. Each module is labeled at the top edge, e.g. `Wave Multiplier`, and extends down to the bottom edge in a column. Note that although the modules have the same height, they have different widths. Image © mikael altemark/CC-BY-2.0.

Figure 1.4: A Minimoog semi-modular system from the 1970s. Patch points are primarily on the top edge and hidden from view. Image public domain.

Figure 1.5: A Yamaha DX7 from the 1980s. Note the menu-based interface and relative lack of controls compared to modular and semi-modular synthesizers. Image public domain.

decades as personal computers and portable computing devices became common household items, the costs associated with computer-based music making became dominated by the cost of software and associated audio and MIDI interfaces. Figure 1.6 shows digital audio workstation (DAW) software commonly used in music production.

Figure 1.6: Logic Pro digital audio workstation software. Additional functionality is provided by 3rd-party plugins showing as additional windows on the screen. In the foreground are an audio interface and a MIDI keyboard used for recording/playing audio and entering note information respectively. Image © Musicianonamission/CC-BY-SA-4.0.

The computer-centric approach dominated synthesis for a decade or more, but by the 2010s, improved electronics manufacturing, smartphone technology, and the open-source movement led to lower cost modular synthesizers. Additionally, the Eurorack standard (Doepfer Musikelektronik, 2022a,b) was widely adopted, leading to +10,000 interoperable modules.[3] As a result, modular synthesis saw a resurgence in popularity. Figure 1.7 shows a Eurorack modular synthesizer.

Figure 1.7: A Eurorack modular synthesizer. The different modules designs and logos reflect the adoption of the Eurorack standard which makes modules from different manufacturers interoperable. Image © Paul Anthony/CC-BY-SA-4.0.

It is perhaps surprising that some 60 years after its creation, modular synthesis is more popular than ever. One possible reason is the reduction in price over time, shown in Table 1.1. However, other trends seem to be at work. While the modular synthesizer was simplified for wider adoption early in its history, first with semi-modular and later with digital synthesizers, the culmination of this trend led to large preset and sample banks that transformed the task of creating a specific sound to searching for a pre-made sound. It's plausible that as the search for sounds became more intensive, the time savings of presets diminished, making the modular approach more attractive. An intersecting trend is a commonly-expressed dissatisfaction with using computers for every aspect of music making and a corresponding return to hardware instruments, including modular.

Table 1.1: The cost of modular, semi-modular, and computer synthesizers over time. Prices are in 2022 dollars.

Decade	Synthesizer	Cost
1960s	Moog modular synthesiser	$96,000
1970s	Minimoog semi-modular	$10,000
1980s	Yamaha DX7	$6,000
1990s	Gateway computer with Cubase	$8,000
2010s	ALM System Coupe modular	$2,400

[3] https://www.modulargrid.net/

Decade	Synthesizer	Cost
...	VCVRack virtual modular	Free

Earlier in this chapter, I argued that a computational thinking approach to modular could help with other synthesizers and studio production tools. Hopefully this brief history helps explain why: modular represents the building blocks of synthesis that later approaches have appropriated and presented in their own way. A square wave oscillator in modular is fundamentally the same as that in another hardware synth or DAW software. If you understand these building blocks in modular, you should understand them everywhere.

1.4 Moving forward

Our next stop is *Sound* where the focus is to "understand the problem". Chapter **??** addresses both the physics of sound and our perception of it, which perhaps surprisingly, are not the same. From there we move into sounds commonly found in music and their properties, ranging from harmonic sounds in Chapter **??** to inharmonic sounds like percussion in Chapter **??**.

The remainder of the book alternates between learning model elements (modules), how they interact (patches), problem solving (sound design). The progressive *Modules* and *Sound Design* sections build up from basic approaches to the more complex. By the time we're done, you should have a good foundation to create patches to solve new sound design problems.

Part I

Sound

Chapter 2

Physics and Perception of Sound

From the outset, it's important to understand that the physics of sound and how we perceive it are not the same. This is a simple fact of biology. Birds can see ultraviolet, and bats can hear ultrasound; humans can't do either. Dogs have up to 40 times more olfactory receptors than humans and correspondingly have a much keener sense of smell. We can only perceive what our bodies are equipped to perceive.

In addition to the limits of our perception, our bodies also *structure* sensations in ways that don't always align with physics. A good example of this is equal loudness contours. As shown in Figure 2.1, sounds can appear equally loud to humans across frequencies even though the actual sound pressure level (a measure of sound energy) is not constant. In other words, our hearing becomes more sensitive depending on the frequency of the sound.

Why do we need to understand the physics of sound *and* perception of sound? Ultimately we hear the sounds we're going to make, but the process of making those sounds is based in physics. So we need to know how both the physics and perception of sound work, at least a bit.

2.1 Waves

Have you ever noticed a dust particle floating in the air, just randomly wandering around? That random movement is known as Brownian motion, and it was shown by Einstein to be evidence for the existence of atoms - that you can see with your own eyes! The movement is caused by air molecules[1] bombarding the

[1] In what follows, we will ignore that air is a mixture of gases because it is irrelevant to the present discussion.

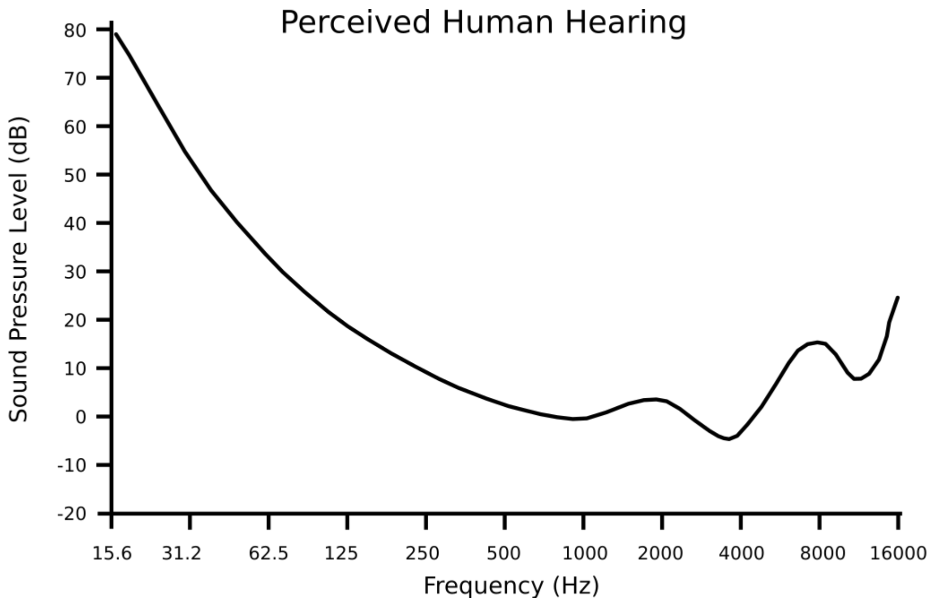

Figure 2.1: An equal loudness contour showing improved sensitivity to frequencies between 500Hz and 4KHz, which approximately matches the range of human speech frequencies. Image public domain.

much larger dust particle from random directions, as shown in Figure 2.2.

Amazingly, it is also possible to see sound waves moving through the air, using a technique called Schlieren photography. Schlieren photography captures differences in air pressure, and sound is just a difference in air pressure that travels as a wave. The animation in Figure 2.3 shows a primary wave of sound corresponding to the explosion of the firecracker in slow motion, and we can see that wave radiate outwards from the explosion.

Let's look at a more musical example, the slow motion drum hit shown in Figure 2.4. After the stick hits the drum head, the head first moves inward and then outward, before repeating the inward/outward cycle. When the drum head moves inward, it creates more room for the surrounding air molecules, so the density of the air next to the drum head decreases (i.e., it becomes less dense, because there is more space for the same amount of air molecules). The decrease in density is called rarefaction. When the drum head moves outward, it creates less room for the surrounding air molecules, so the density of the air next to the drum head increases (i.e., it becomes more dense, because there is less space for the same amount of air molecules). The increase in density is called compression.

You can see an analogous simulation of to the drum hit in Figure 2.5. If you add say 50 particles, grab the handle on the left, and move it to the right,

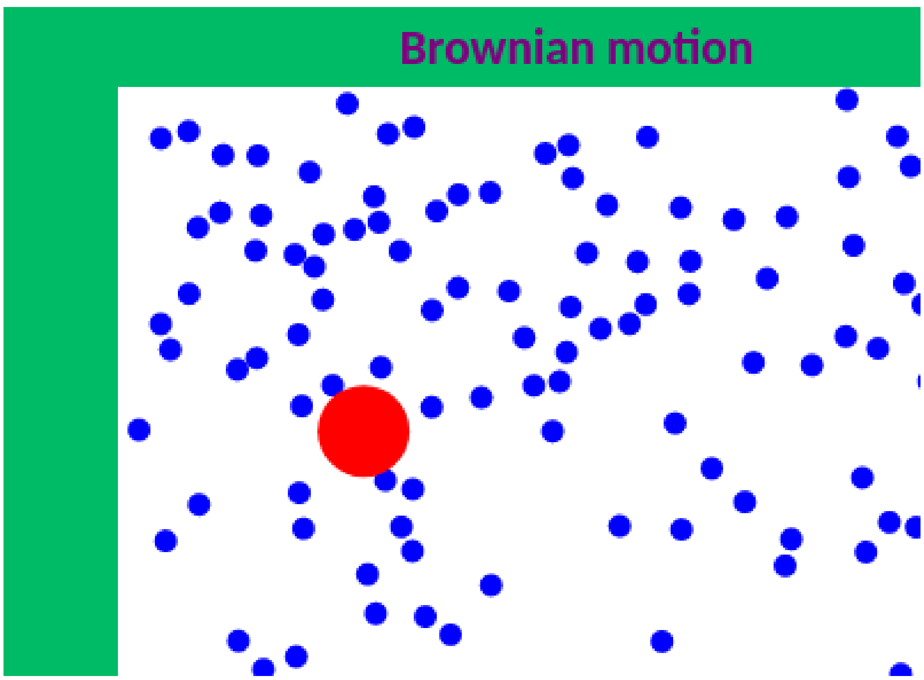

Figure 2.2: Simulation of Brownian motion. Press `Pause` to stop the simulation. © Andrew Duffy/CC-BY-NC-SA-4.0.

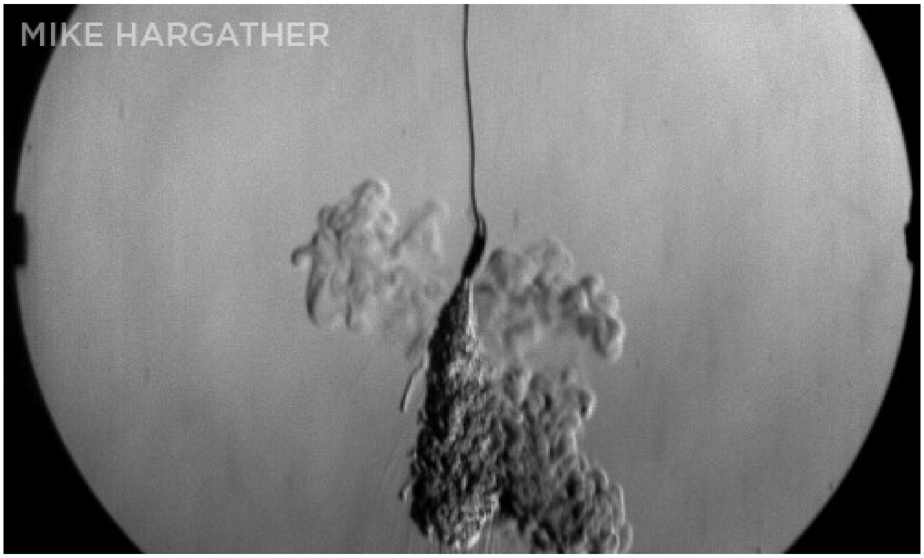

Figure 2.3: Animation of a firecracker exploding in slow motion, captured by Schlieren photography. Note the pressure wave that radiates outward. Image © Mike Hargather. Linked with permission from NPR.

Figure 2.4: Youtube video of a slow motion drum hit. Watch how the drum head continues to move inward and outward after the hit. Image © Boulder Drum Studio.

the volume of the chamber decreases, and the pressure in the chamber goes up (compression). Likewise, if you move the handle to the left, the volume of the chamber increases, and the pressure goes down (rarefaction). In the drum example, when the stick hits the head and causes it to move inward, the volume of air above the head will rush in to fill that space (rarefaction), and when the head moves outward, the volume of air above the head will shrink (compression).

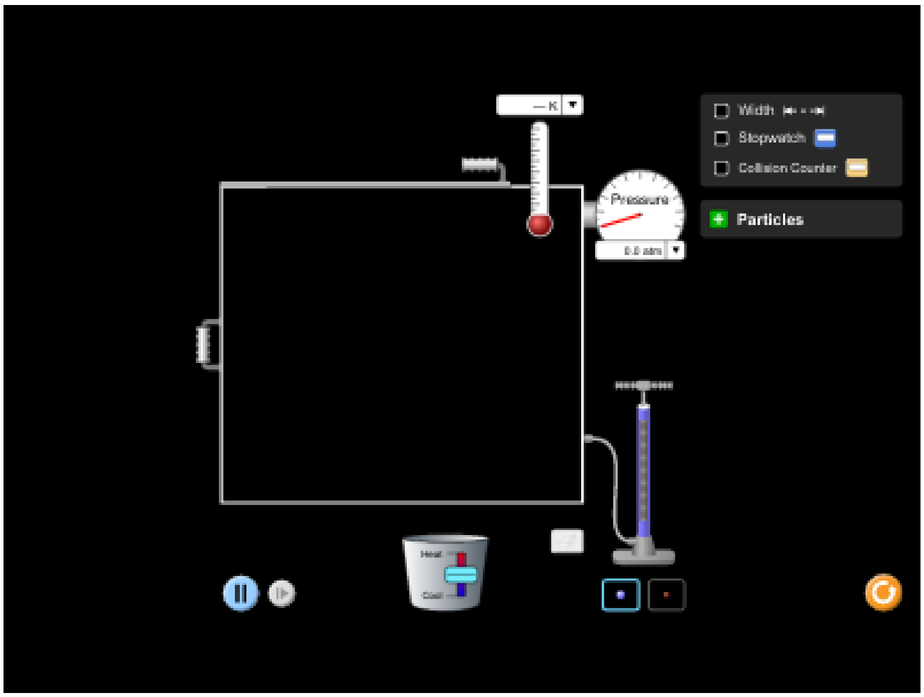

Figure 2.5: Simulation of gas in a chamber. Simulation by PhET Interactive Simulations, University of Colorado Boulder, licensed under CC-BY-4.0.

Sound is a difference in air pressure that travels as a wave through compression and rarefaction. We could see this with the firecracker example because the explosion rapidly heated and expanded the air, creating a pressure wave on the boundary between the surrounding air and the hot air. However, as we've seen with the drum and will discuss in more detail later, musical instruments are designed to create more than a single wave. The Schlieren photography animation in Figure 2.6 is more typical of a musical instrument.

The rings in Figure 2.6 represent compression (light) and rarefaction (dark) stages of the wave. It is important to understand that air molecules aren't moving from the speaker to the left side of the image. Instead, the wave is moving the entire distance, and the air molecules are only moving a little bit as

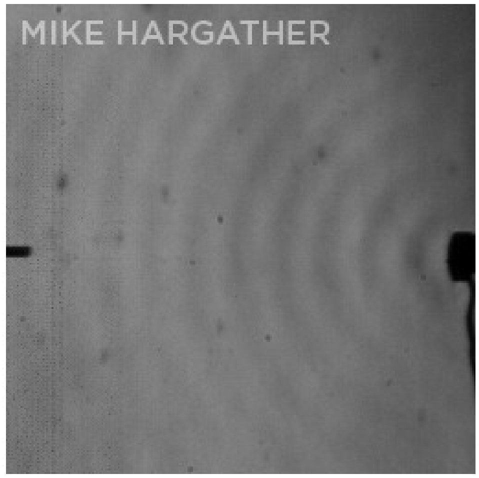

Figure 2.6: Animation of a continuous tone from a speaker in slow motion, captured by Schlieren photography. The resulting sound wave shows as lighter compression and darker rarefaction bands that radiate outward. Image © Mike Hargather. Linked with permission from NPR.

a result of the wave.[2]

To see how this works, take a look at the simulation in Figure 2.7. Hit the green button to start the sound waves and then select the `Particles` radio button. The red dots are markers to help you see how much the air is moving as a result of the wave. As you can see, every red dot is staying in their neighborhood by **moving in opposite directions** as a result of compression and rarefaction cycles. If you select the `Both` radio button, you can see the outlines of waves on top of the air molecules. Note how each red dot is moving back and forth between a white band and a dark band. If you further select the `Graph` checkbox, you will see that the white bands in this simulation correpond to increases in pressure and the black bands correspond to decreases in pressure. This type of graph is commonly used to describe waves, so make sure you feel comfortable with it before moving on.

Now that we've established what sound waves are, let's talk about some important physical properties of sound waves and how we perceive those properties. Most of these properties directly align with the shape of a sound wave.

[2]The air molecules are moving randomly in general, so the simulation shows only the movement attributable to the effect of the wave.

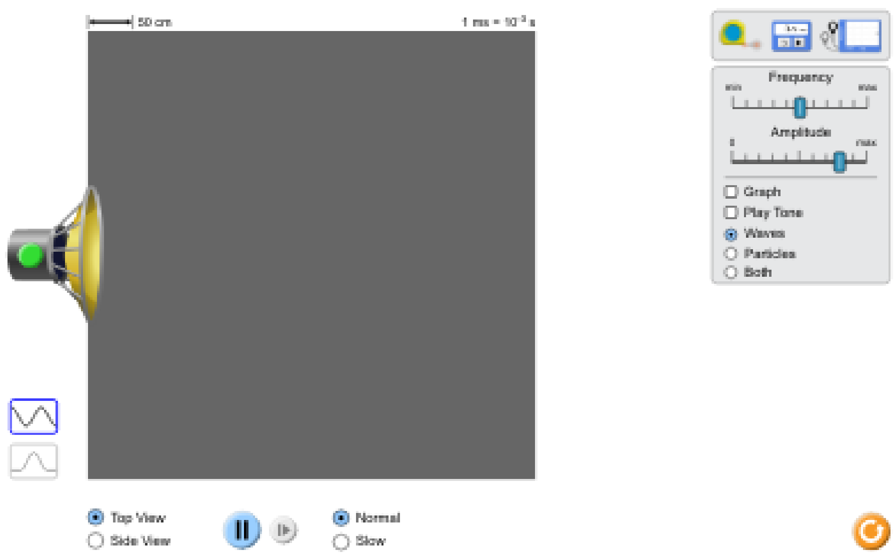

Figure 2.7: Simulation of sound waves. Simulation by PhET Interactive Simulations, University of Colorado Boulder, licensed under CC-BY-4.0.

2.2 Frequency and pitch

Almost all waves we'll talk about are periodic, meaning they repeat themselves over time. If you look at the blue wave in Figure 2.8, you'll notice that it starts at equilibrium pressure (marked as zero[3]), goes positive, hits zero again, and then goes negative before hitting zero at 2 seconds. So at 2 seconds, the blue wave has completed 1 full cycle. Now look at the yellow wave. The end of its first cycle is indicated by the circle marker at .5 seconds. By the 2 second mark, the yellow wave has repeated its cycle 4 times. Because the yellow wave has more cycles than the blue wave in the same amount of time, we say that the yellow wave has a higher frequency, i.e. it repeats its cycle more frequently than the blue wave. The standard unit of frequency is Hertz (Hz), which is the number of cycles per second. So the blue wave is .5 Hz and the yellow wave is 2 Hz.

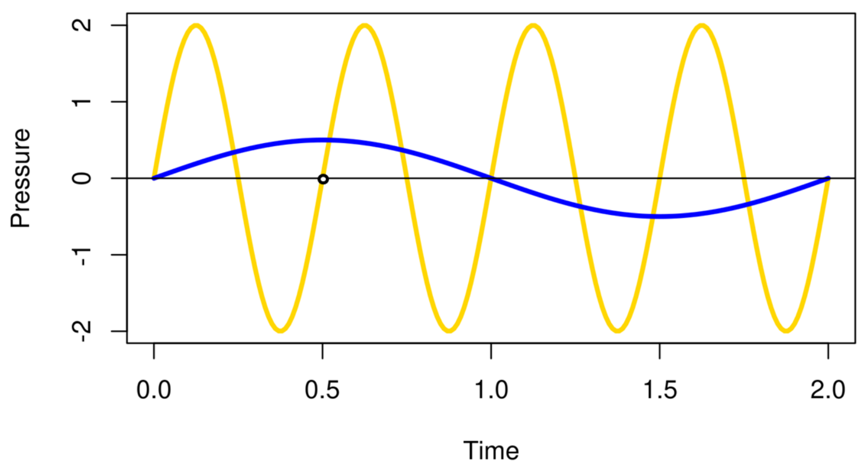

Figure 2.8: Two waves overlaid on the same graph. The yellow wave completes its cycle 4 times in 2 seconds and the blue wave completes its cycle 1 time in 2 seconds, so the frequencies of the waves are 2 Hz and .5 Hz, respectively.

Humans perceive sound wave frequency as pitch. As a sound wave cycles faster, we hear the sound's pitch increase. However, the relationship between frequency and pitch is nonlinear. For example, the pitch A above middle C is 440 Hz[4],

[3]Recall sound is a pressure wave, and it is the change in air pressure we care about. Subtracting out the equilibrium pressure to get zero here makes the positive/negative changes in air pressure easier to see.

[4]Also called A4.

but the A one octave higher is 880 Hz, and the A two octaves higher is 1760 Hz. If you wanted to write an equation for this progression, it would look something like $A_n = 440 * 2^n$, which means the relationship between frequency and pitch is exponential. Figure 2.9 shows the relationship between sound wave frequency and pitch for part of a piano keyboard, together with corresponding white keys and solfège. Notice that the difference in frequencies between the two keys on the far left is about 16 Hz but the difference in frequencies between the two keys on the far right is about 111 Hz. So for low frequencies, the pitches we perceive are closer together in frequency, and for high frequencies, the pitches we perceive are more spread out in frequency.

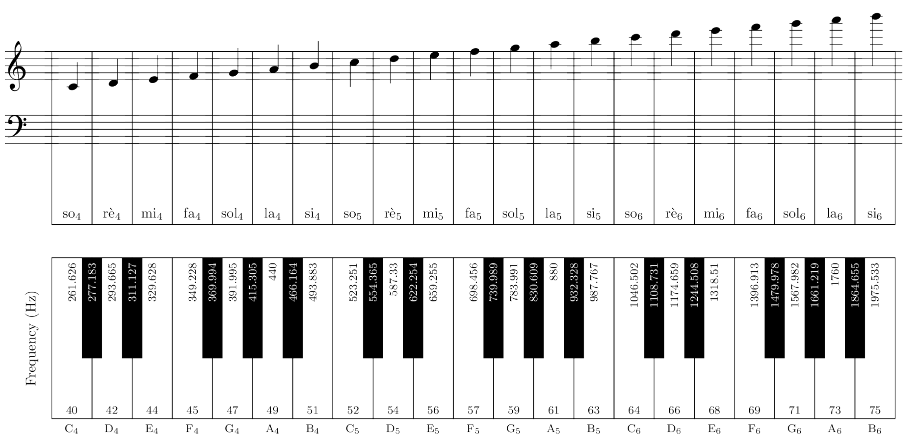

Figure 2.9: Part of an 88 key piano keyboard with frequency of keys in Hz on each key. The corresponding note in musical notation and solfège is arranged above the keys. Zoom in on the image for more detail.

Of course we experience pitch linearly, so the difference in pitch between the two keys in the far left is the same as the difference between the keys on the far right. We can make the relationship linear by taking the logarithm of the frequencies. On the left side of Figure 2.10, we see the exponential relationship between frequency and pitch: as we go higher on the piano keys and pitch increases, the frequencies increase faster, such that the differences in frequencies between keys gets wider. On the right side of Figure 2.10, we see the same piano keys, but we've taken the logarithm of the frequencies, and now the relationship is linear. It turns out that, in general, our perception is logarithmic in nature (this is sometimes called the Weber-Fechner law). Our logarithmic perception of pitch is just one example.

You might be wondering if there's point at which pitches are low enough that the notes run together! It seems the answer to this is that our ability to hear sound at all gives out before this happens. Going back to the 88-key piano keyboard, the two lowest keys (keys 1 & 2; not shown) are about 1.5 Hz appart,

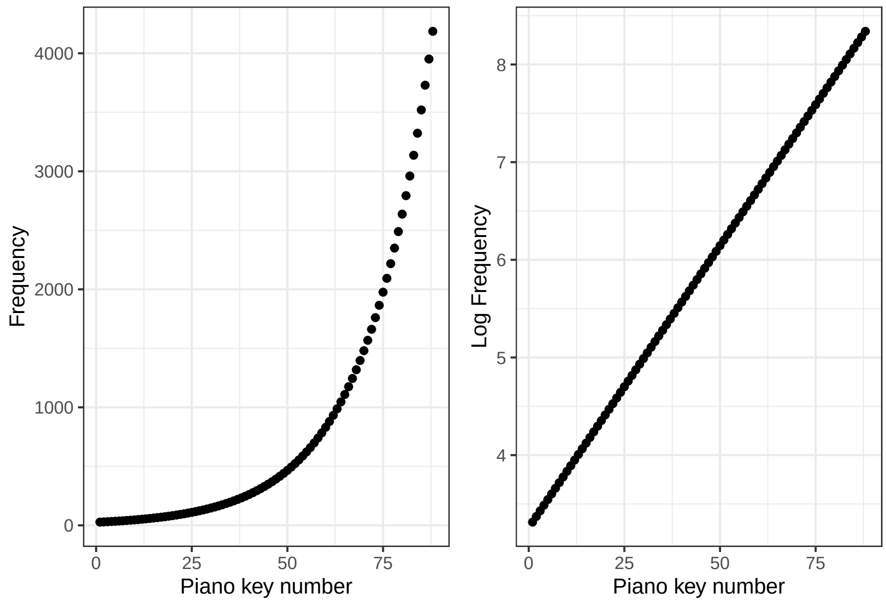

Figure 2.10: (ref:log-freq)

but the lowest key[5] is 27.5 Hz. Humans generally can only hear frequencies between 20 Hz and 20,000 Hz (20 kHz). Below 20 Hz, sounds are felt more than heard (especially if they are loud), and above 20 kHz generally can't be heard at all, though intense sounds at these frequencies can still cause hearing damage.

You may also be curious about the fractional frequencies for pitches besides A. This appears to be largely based on several historical conventions. In brief, western music divides the octave into 12 pitches called semitones based on a system called twelve-tone equal temperment. This is why on a piano, there are 12 white and black keys in an octave - each key represents a semitone. It's possible to divide an octave into more or less than 12 pitches, and some cultures do this. In fact, research suggests that our perception of octaves isn't universal either (Jacoby et al., 2019). We'll discuss why notes an octave apart feel somehow the same in a later section.

2.3 Amplitude and loudness

As discussed, sound is a pressure wave with phases of increasing and decreasing pressure. Take a look at the yellow and blue waves in Figure 2.11. The peak compression of each wave cycle has been marked with a dashed line. For the

[5] Also called A0

yellow wave, the peak positive pressure is 2, and for the blue wave, the peak positive pressure is .5. This peak deviation of a sound wave from equilibrium pressure is called amplitude.[6] It is perhaps not surprising that we perceive larger deviations (with corresponding large positive and negative pressures) as louder sounds.

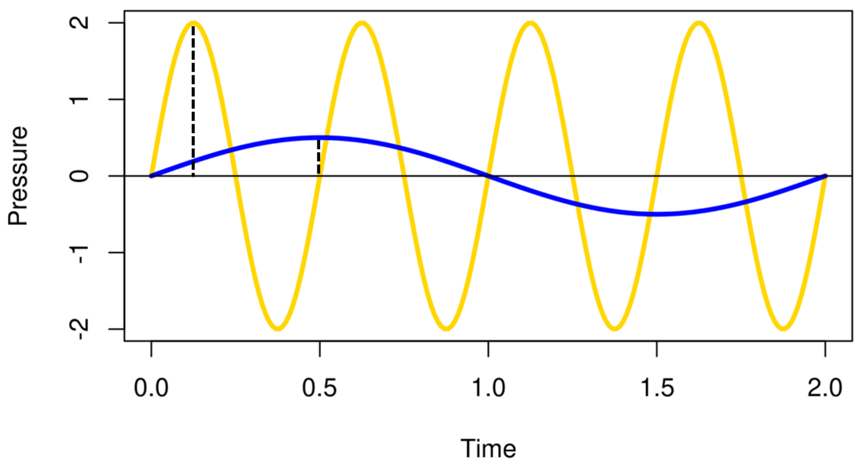

Figure 2.11: Two waves overlaid on the same graph, with a dashed line marking the amplitude of each wave as the deviation from equilibrium.

The relationship between amplitude and loudness is also nonlinear: we hear quiet sounds very well, and sounds must get a lot louder before we perceive them as being louder. In fact, the nonlinear relationship between amplitude and our perception of loudness is *even more extreme* than the relationship for frequency and pitch.

You might have heard of the unit of loudness before, the decibel (dB). Unfortunately, the decibel is a bit harder to understand than Hz, and it use used as a unit of measurement for different ways of expressing the strength of a sound, like sound pressure, sound power, sound intensity, etc. The most important thing you need to understand about decibels is that they are not an absolute measurement, but rather a relative measurement. Therefore, decibels are always based on a reference value. For hearing, that reference value is the quietest sound people can detect, which is defined as 0 dB. Some examples of 0 to 10dB sounds

[6]Note that since the positive and negative pressures are equivalent, amplitude could be measured down from equilibrium to peak negative pressure as well

are a mosquito, breathing, a pin drop, or a leaf hitting the ground.[7]

If we call the reference sound pressure S_0 and the sound pressure we are measuring S, then we calculate the sound pressure level dB of S as $20 * log_{10}(S/S_0)$ dB. Under this definition, a 6 dB increase in sound pressure level means amplitude has doubled: $20 * log_{10}(2) = 6.02$ dB.[8] Since our hearing is quickly damaged at 120 dB, you can see that our range of hearing goes from the quietest sound we can hear (0 dB) to a sound that is 1,000,000,000,000 times more intense (120 dB).

Remember from Figure 2.1 that frequency affects our perception of loudness. As a result, we can't say how loud a person will perceive a random 40 dB sound - not in general. One way of approaching this problem is to choose a standard frequency and define loudness for that frequency. The phon/sone system uses a standard frequency of 1 kHz so that a 10 dB increase in sound pressure level is perceived as twice as loud. This relationship is commonly described as needing 10 violins to sound twice as loud as a single violin. There are alternative ways of weighting dB across a range of frequencies rather than just 1 kHz, so the 10 dB figure should be viewed as an oversimplification, though a useful one. Table 2.1 summarizes the above discussion with useful dB values to remember.

Table 2.1: Useful values for working with dB. All values reflect sound pressure or sound pressure level.

Value	Meaning
0 db	Reference level, e.g. quietest audible sound.
6 db increase	Twice the amplitude
10 db increase	Twice as loud
20 db increase	Ten times the amplitude

2.4 Waveshape and timbre

When we talked about frequency and amplitude, we used the same waveshape in Figures 2.8 and 2.11. This wave shape is called the sine wave. The sine wave is defined by the trigonometric sine function and found throughout physics. There are an unlimited number of waveshapes in principle, but in electronic music you will encounter the sine wave and other waveshapes in Figure 2.12 often because they are relatively easy to produce using analogue circuitry.

[7] These are commonly given examples, but see below for how they are misleading when you take frequency into account.

[8] Note that if we'd used sound intensity level, the corresponding value would be 3 dB. Sound pressure level is more useful in our context: it connects directly to sound wave amplitude, it is measured by a microphone and reflected in microphone output voltage, and it has the same formula as dBV, a decibel measurement of voltage common in audio electronics. Sound intensity level is the square of the sound pressure level.

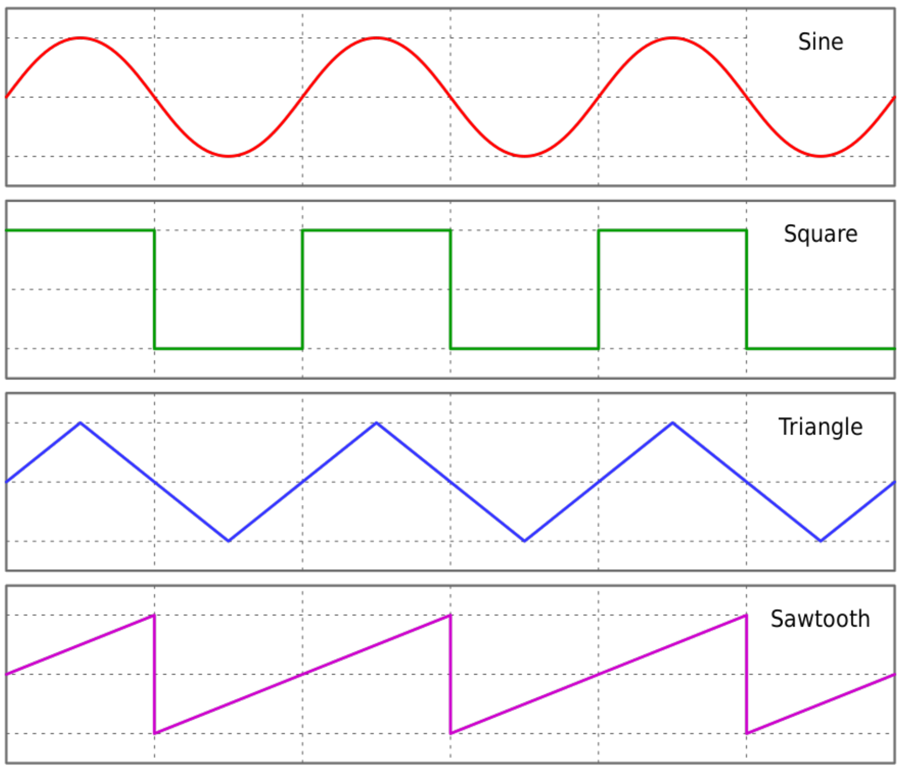

Figure 2.12: The four classic waveshapes in analogue electronic music. Image © Omegatron/CC-BY-SA-3.0.

Perhaps another reason for the widespread use of these waveshapes is that their sounds are rough approximations to real instrument sounds. As we discussed in Section 1.3, the development of electronic music has been strongly influenced by existing instruments. Figure 2.13 presents sounds for each of these waveshapes, together with real instruments that have similar sounds.

As you listen to the waveshape sounds, take a moment to consider their qualities with respect to each other, e.g. how noisy are they and how bright? The differences you're hearing are referred to as timbre or tone color. Each of the waveshape sounds is the same frequency (220 Hz; A3), and the different timbres illustrate how waveforms can have the same frequency but still have their own characteristic sound.

As you listen to the real instrument sounds, consider what about them matches the timbre of the waveshapes and what does not. You may need to listen to some real instruments longer to notice the similarities and differences. For example, when an instrument is played quietly, it may have a different timbre than when it is played loudly. We'll explore these dynamic differences in an upcoming section.

Wave	**Wave sound**		**Comparable instruments**	**Instrument s**
Sine		©	Flutes	
Square		©	Brass (no reed), e.g. trumpet	
Triangle		©	Single reed woodwind, e.g. clarinet	
Saw		©	Bowed strings, e.g. violin	

Figure 2.13: Sounds of the four classic waveshapes in electronic music, together with example instruments that make similar sounds.

There is a variation of the square waveform that you will frequently encounter called the pulse wave.[9] In a square wave, the wave is high and low 50% of the time. In a pulse wave, the amount of time the wave is high is variable: this is called the duty cycle. A pulse wave with a duty cycle of 75% is high for 75% of its cycle and low the rest of the time. By changing the duty cycle, pulse waves can morph between different real instrument sounds. At 50% they sound brassy, but at 90%, they sound more like an oboe. Figure 2.14 shows a pulse wave across a range of duty cycles, including 100% and 0%, where it is no longer a wave.

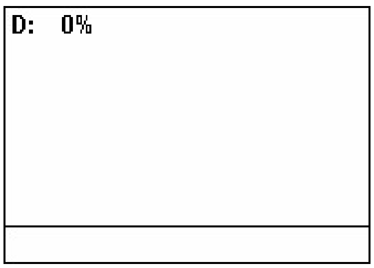

Figure 2.14: Animation of a pulse wave across a range of duty cycles. Note that 100% and 0% it ceases to be a wave. Image public domain.

There are multiple ways of creating waveshapes beyond the four discussed so far. One way is to combine waveshapes together. This is somewhat analogous to mixing paint, e.g. you can mix primary colors red and blue to make purple, and we'll get more precise about how it's done shortly. Alternatively, we can focus on what a wave looks like rather than how we can make it with analogue circuitry. Since waves are many repetitions of a single cycle, we could draw an arbitrary shape for a cycle and just keep repeating that to make a wave - this is the essence of wavetable synthesis and is only practical with digital circuitry.

2.5 Phase and interference

The last important property of sound waves that we'll discuss is not about the shape of the wave but rather the *timing* of the wave. Suppose for a moment that you played the same sine wave out of two speakers. It would be louder, right? Now suppose that you started the sine wave out of one speaker a half cycle later than the other, so that when the first sine wave was in its negative phase, the second sine wave was in its positive phase. What would happen then? These two scenarios are illustrated in Figure 2.15 and are called constructive and destructive interference, respectively. In both cases, the waves combine to create a

[9]One could say the square wave is a special case of the pulse wave. Starting from the circuit or the mathematical definition will give you a different perspective on this; both are true from a certain point of view.

new wave whose amplitude is the sum of the individual wave amplitudes. When the phase-aligned amplitudes are positive, the amplitude of the resulting wave is greater than the individual waves, and when the phase-aligned amplitudes are a mixture of positive and negative, the amplitude of the resulting wave is less than the individual waves.

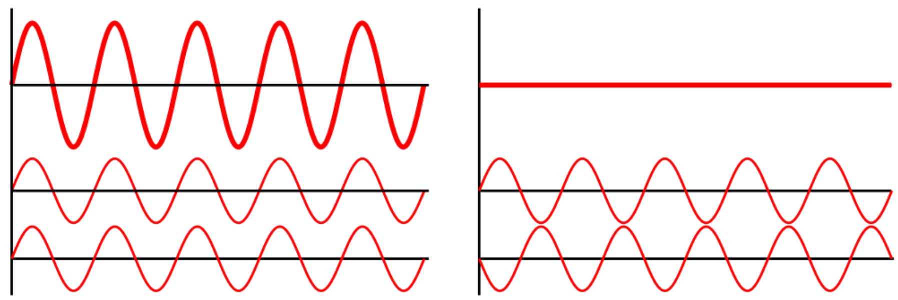

Figure 2.15: Constructive (left) and destructive interference (right). For these matched sine waves, being perfectly in phase or out of phase causes the resulting wave amplitude to be either double or zero, respectively. Image

Destructive interference is the principle behind noise cancelling headphones, which produce a sound within the headphones to cancel out the background noise outside the headphones. Figure 2.15 shows how this can be done with with a sine wave using an identical, but perfectly out of phase sine wave. However, being perfectly out of phase is not enough to guarantee cancellation in all cases. Take a look again at the waveshapes in Figure 2.12. You should find it relatively easy to imagine how the first three would be cancelled by an out of phase copy of themselves. However, the sawtooth wave just doesn't match up the same way, as shown in Figure 2.16.

Figure 2.16 shows how cancellation will only happen if the perfectly out of phase wave is identical to *inverting* the original wave. Inverting a wave means reflecting it on the x-axis so we get a mirror image of the wave.[10] When we add a wave to its mirror image, we achieve perfect cancellation, even for waves like the sawtooth. It just so happens that the first three waves in Figure 2.12 are symmetric, which makes their perfect out of phase and and inverted versions identical.

So what is phase exactly? If we consider the cycle of a wave to go from 0 to 1, then the phase of a wave is the position of the wave on that interval. You've likely seen this concept before in geometry with the sine function, where you can describe one cycle either in degrees (360°) or in radians (2π). The unit is somewhat arbitrary: the important thing to remember is that if two waves are out of sync in their cycles, they will interfere with each other, and we describe

[10]This can be accomplished by multiplying the y coordinates by -1

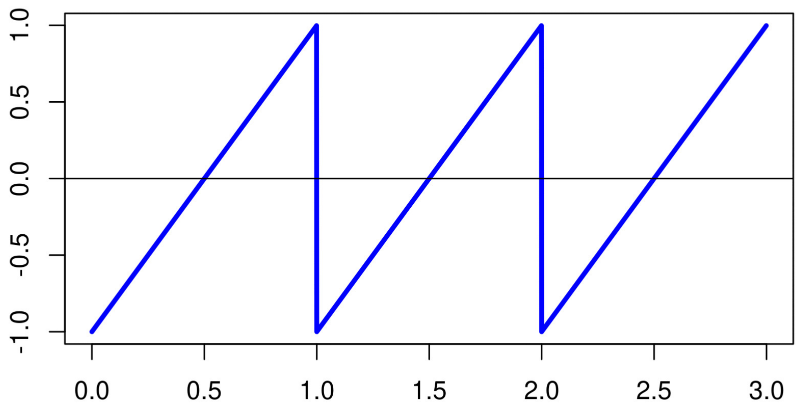

Figure 2.16: Animation of interference from a perfectly out of phase saw wave (yellow) combined with the original wave (blue). The resulting wave (green) has non-zero amplitude because the out of phase wave does not have opposite amplitude to the original wave at all locations. Dotted lines indicate the positions of the original waves.

this difference in cycle position as a difference in phase. The interference caused by phase differences can result in a wide range of effects from cancellation, to the creation of a new wave, to an exact copy of the original wave with double amplitude.

There are a few very practical contexts for phase to keep in mind. The first is that sounds waves reflect, so you don't need two speakers to get phase effects as in the examples above. We'll discuss how phase and reflection are intrinsic to the sounds of many instruments in the next chapter, but any reflective surface can affect phase, which has implications for acoustics in rooms. The second practical context is that you will often want to play multiple notes at the same time, and when you have an electronic instrument, those notes can be triggered very precisely - precisely enough that you have some control over the phase relationships and can use them to shape the sound to your liking.[11]

[11] Any wave property that lends itself to interference can be used similarly, including frequency and amplitude.

Chapter 3

Harmonic and Inharmonic Sounds

We reviewed four common waveshapes in Section 2.4, but we did not explain *why* the waveshapes have their own distinctive timbre. The short answer is that the waveshapes have different harmonics. Understanding the relationship between waveshape and harmonics will be extremely useful to you as you design your own sounds. In order to explain the harmonics behind the different waveshape timbres, we need to understand what harmonics are and how they are created. Not all sounds are harmonic, however. Most percussion sounds are inharmonic, with drums and cymbals as prominent examples.[1] These sounds have a high degree of "noise" associated with them. Finally, traditional instruments producing harmonic and inharmonic sounds have distinctly different dynamics that contribute to their timbre. Harmonic, inharmonic, and dynamic elements all play an important role in shaping sounds.

3.1 Phase reflections, standing waves, and harmonics

When we discussed wave phase and interference in Section 2.5, we mentioned the importance of phase and reflection in musical instruments. Before going any further, ask yourself what makes something a musical instrument? Is a stick hitting a sheet of paper a musical instrument? What about a stick hitting an empty glass?

[1] I prefer to think of these sounds as pitched but inharmonic, but this is not a common view.

A common property of musical instruments is that they make reliable pitches rather than noise. These pitches are created by waves reflecting in the instrument to create standing waves and, in harmonic instruments, get rid of noise. Standing waves are fairly similar and straightforward in strings and pipes, so we'll begin our discussion with strings, where the waves are easily visible.

Standing waves are created when two waves moving in opposite directions interfere with each other to create a new wave that appears to remain in place (i.e. it "stands" rather than moves). In a string, the two waves are created by an initial wave that reflects back from the ends of the string, which creates an out of phase wave. Figure 3.1 shows a simulation of a reflected wave. Select the checkbox for `Pulse` and move `Damping` to `None`, then press the green button to initiate a pulse. As you can see, when the pulse reaches a fixed end, it reflects both in direction and in phase, i.e. if it is above the line going right, it will flip below the line and go left when it hits a fixed end.

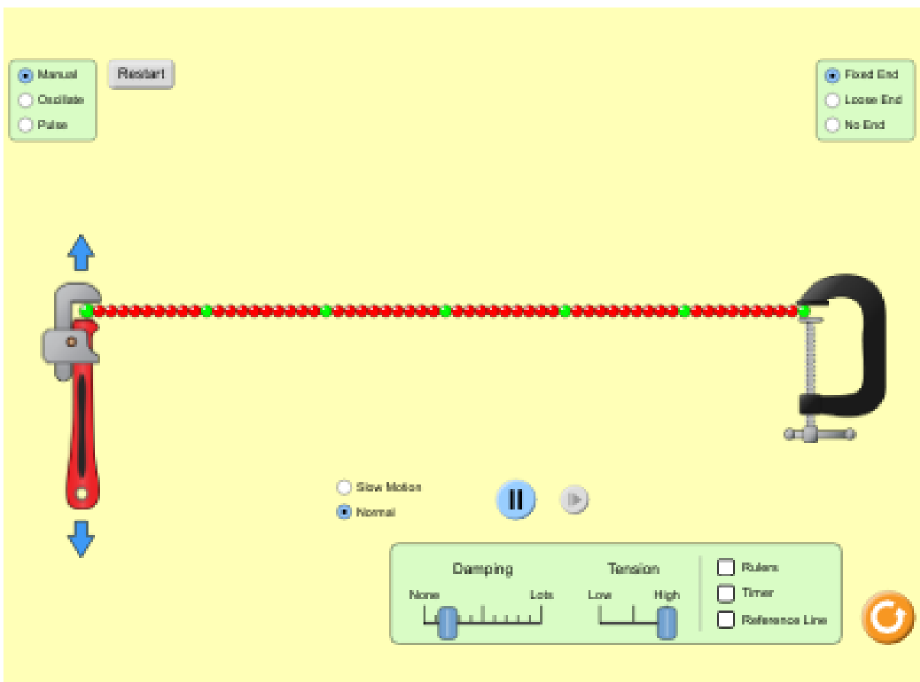

Figure 3.1: Simulation of waves on a string. Simulation by PhET Interactive Simulations, University of Colorado Boulder, licensed under CC-BY-4.0.

You can use the same simulation to make a standing wave. Set `Amplitude` to .20, `Frequency` to .17, `Damping` to `None`, `Tension` to `Low`, and select `Oscillate`. These settings will send a wave from the left to interfere with the reflected wave coming back from the right. You will very quickly see a standing wave with three nodes, indicated by the green beads. Two of the nodes are at either end of the string, and the third node is exactly in the middle. This standing wave is the 2nd harmonic. You can make the first harmonic (commonly called the fundamental frequency) by reducing `Frequency` to half, $.17/2 = .085 \approx .09$ and hitting restart. Observe that the only nodes are the two ends of the string. Finally, you can create additional harmonics by multiplying the frequency of the fundamental by integers greater than 1. Try $.085 * 3 = .255 \approx .26$, $.085 * 4 = 34$, $.085 * 5 = .425 \approx .43$, etc. Note the green beads don't always mark nodes, which by definition don't move; instead the green beads sometimes mark antinodes, or places of greatest motion.

For the odd harmonics, the simulation requires us to do some rounding that results in an imperfect standing wave. Eventually these imperfect standing waves will cancel out because they are not true standing waves. True standing waves on our string model have frequencies that are perfectly aligned with the length of our string. When this happens, the two waves that create the standing wave constructively interfere with each other, i.e. they are self-reinforcing. We can see this in the simulation when we don't round the frequencies because the amplitude of the waves keeps increasing over time as we add more energy into the system through the oscillator on the left hand side. Any wave whose frequency does not create a standing wave will eventually cancel itself out even if damping is zero.

Standing waves on a string explain why harmonics are integer multiples of the fundamental frequency. In the simulation for the first harmonic, the oscillator started its second pulse at the moment the first reflected pulse returned to the oscillator, so the first pulse had to travel twice the length of the string. For the second harmonic, the oscillator started its second pulse at the moment the first pulse reached the fixed end, or the full length of the string. In each case, the combined speed of the waves must evenly divide the length of the string (giving an integer) in order for the out of phase passing waves to align and create a standing wave. The integer relationship of harmonics is very important to how we perceive musical instruments and may even be the reason for our perception of fundamental musical relationships like octaves and fifths. For example, recall from Section 2.2 than one octave above a pitch is double the frequency of that pitch. Since the second harmonic is double the frequency of the fundamental, the second harmonic is one octave above the fundamental frequency. Similarly, the third harmonic is one fifth above the second harmonic.[2]

Before moving on, it's important to note that the oscillator simulation in Figure 3.1 does not accurately represent what happens in a plucked string because the simulation doesn't fix both sides of the string. Instead, the simulation uses an oscillator on one side to generate sine waves. We can, however, observe a very similar behavior to the simulation when a string is plucked, as shown the slow

[2]One might speculate that we created instruments in order to consistently create pitches based on the fundamental, but the instruments instilled in us a music theory as a side effect of their harmonics.

motion video in Figure 3.2. In the video, the tap on the string (a reverse of a pluck) creates two pulses that move in opposite directions, reflect off their respective ends of the string and switch phase, and then constructively interfere in the center to create an apparent standing wave.

Figure 3.2: Youtube video of a slow motion tap on a long string. Watch how the tap creates two pulses that reflect off their respective ends of the string, switching phase, and then constructively interfere to create an apparent standing wave. Image © Kemp Strings.

Although Figure 3.2 looks straightforward and might lead us to believe that the differences between the simulation and plucking a string are superficial, the true story is more complicated. A real string pluck does not create a single standing wave but rather a stack of standing waves happening all at once. This is because, unlike the simulation in 3.1, the wave created by plucking a string is not a sine wave but has an initial shape more like a triangle. In addition, a string pluck is highly likely not to occur in the middle of the string. These differences mean that when a string is plucked, waves of all different frequencies are created and begin racing back and forth on the string. Those that correspond to harmonic frequencies are sustained longer and create a tone. The remaining frequencies are known as transients and quickly cancel out.[3]

[3]Transients are commonly modeled using noise in electronic music in order to make a simulated sound more realistic.

3.2 Resonators, formants, and frequency spectrum

In a real string instrument, the vibrations of the strings are transmitted to the rest of the instrument, which includes a resonator that is typically shaped like a box with an opening. The resonator vibrates at the same frequencies as the string (primarily the harmonic frequencies as discussed) but pushes against a larger volume of air than the string, which has a small surface area by comparison. Therefore the resonator creates a larger change in air pressure for a higher amplitude (and louder) sound wave. The resonator does not amplify the string frequencies perfectly, however. Some frequencies are better amplified than others, and this means that the resonator affects the timbre of the instrument. This is why two guitars with different resonators will sound different even if they have identical strings. The effect a resonator has on a frequency's amplitude is called Q, and the relative strengths of frequencies emitted by a resonator are called formants.[4]

You can probably imagine how differences in the construction and operation of other instruments might lead to the differences in their characteristic sounds. Their mode of operation (string, wind, etc.) and their resonators (wood, metal, etc.) affect both what harmonics are produced and the relative strengths of these frequencies (the formants). For example, closed end pipes produce only odd harmonics! In each case, the instrument is producing multiple harmonic frequencies at once, with some instruments producing more harmonics than others. These differences in characteristic sounds are reflected in the four waveshapes presented in Section 2.4.

Unfortunately, when we look at a waveshape, we see the sum of all the harmonic frequencies - we can't see the individual harmonics just by looking at a waveshape. Wouldn't it be nice if we could somehow see all the harmonics that make up a waveshape? It turns out we can decompose any complex waveshape into components using a technique called Fourier analysis. Each component extracted by Fourier analysis is a sine wave called a partial, and we can reconstruct a complex waveshape by adding the sine wave partials together (potentially an infinite number of them). When the waveshape is from a harmonic instrument, the partials are harmonics, so we can use Fourier analysis to see all the harmonics in a waveshape.

Figure 3.3 is a simulation showing how Fourier analysis can use sine waves to approximate more complex waveshapes. The first waveshape is a sine, which is quite trivially approximated by a single sine wave. Take a moment to look at the amplitude of the red line (the first harmonic) and how it corresponds to the red bar in the upper bar plot. You can grab the top of that bar and move it up and down to change the amplitude of the first harmonic. As you move the

[4]You will often see the word "formants" applied to human speech, where the resonator is the vocal tract, but it also applies to instruments with resonators.

bar, note how the bottom graph showing the sum of harmonics changes exactly the same way - this is because it is the sum of harmonics, and we only have one harmonic, so the sum is equal to that harmonic.

Use the dropdown on the right to select a triangle waveshape. Note that all the harmonics are odd for the triangle, and that the sum is clearly different from the sine wave of the first harmonic. It turns out you can't get any closer to a triangle waveshape with only 11 harmonics, but feel free to try by moving the sliders around. What you will find is that as you try to change the shape at one part of the waveshape, you end up making changes everywhere. This is because the sine waves that are being summed up keep going up and down everywhere. The only way to get closer to a triangle shape is to use more and smaller sine waves. You can see what this looks like by using the `Infinite Harmonics` checkbox on the bottom. This illustrates a general principle of Fourier analysis: sharp edges in a waveshape mean high frequency partials.

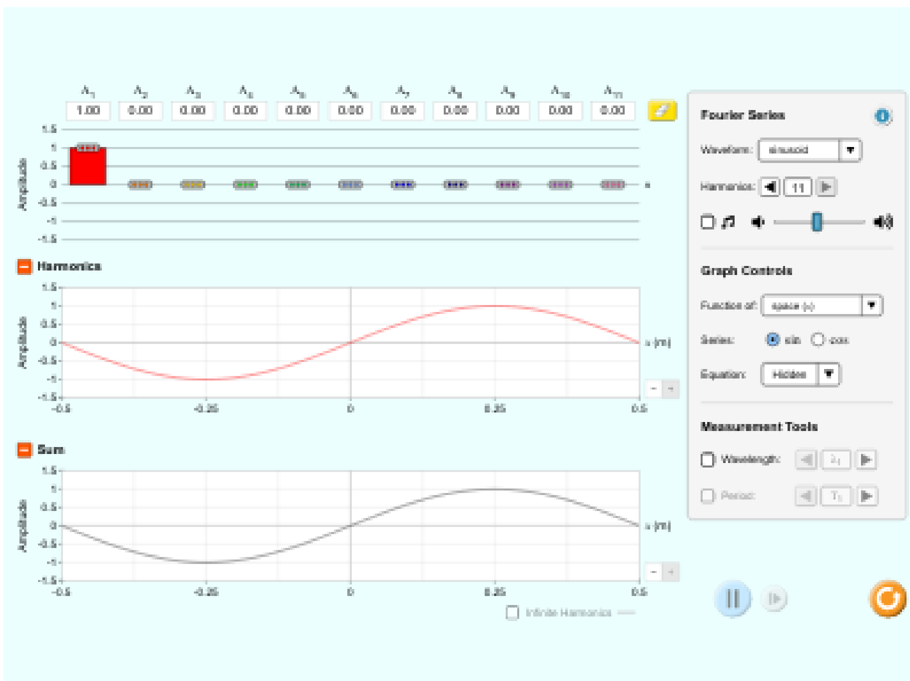

Figure 3.3: Simulation showing how Fourier analysis can approximate a complex wave using sine waves. Simulation by PhET Interactive Simulations, University of Colorado Boulder, licensed under CC-BY-4.0.

The square and sawtooth waveshapes give more impressive examples of this. If you select square, you'll see that again only odd harmonics are used, but you'll also see that the sum of harmonics is pretty far from the square waveshape we looked at before. The sawtooth waveshape, which uses both even and odd harmonics (with opposite signs), also looks pretty different from the sawtooth waveshape we saw before. Both of these waveshapes have sharp edges that need more harmonics to approximate. They also have straight regions that don't line up well with the straight-ish part of the first harmonic, and these straight regions also need more high frequency components to straighten out. In both cases, you can check infinite harmonics to see how additional harmonics would help.

It may have already occurred to you that you could create any sound by adding together the sine waves in the right combinations. This is exactly what additive synthesis does! Running a Fourier analysis to get sine wave partials of a sound and then recombine them to reproduce the sound is very appealing. However, even though the idea of additive synthesis has been around a long time, it was not practical with analogue technology because of the many oscillators and precise timings involved. Conceptually, the alternative to additive synthesis is subtractive synthesis, which has been a very popular approach in analogue synthesis to the present day. Subtractive synthesis starts with complex waveforms and then removes harmonics to create the desired sound. Harmonics can be removed with relatively simple analogue electronics as we'll discuss in a later chapter.

While the simulation in Figure 3.3 is useful for understanding how Fourier analysis works, it's difficult to see all of the frequenecy components because they are stacked on top of each other. An alternative way of visualizing a Fourier analysis is a frequency spectrogram. A frequency spectrogram shows each sine wave based on its frequency and amplitude. Figure 3.4 shows a frequency spectrogram of the same four waveshapes at 1 Hz with harmonics side by side and amplitudes normalized so all harmonics sum to 1. The order of harmonics at 1 Hz shows how much energy each waveshape has in the first harmonic, the order of harmonics at 2 Hz shows the energy at the second harmonic, etc.[5]

Figure 3.4 clearly shows what we noted before: only sawtooth has even harmonics, and only sawtooth and square have easily visible harmonics above the third harmonic. It's quite amazing that these waves have such small visible differences in their frequency spectrums and yet have such distinctive sounds. For all waveshapes, the fundamental has the most energy (highest amplitude). This is crucial to our perception of the overall pitch of the sound. If you go back to Figure 3.3 and increase the amplitude of a harmonic above the amplitude of the harmonic (and click the checkbox to hear the sound), your perception of the pitch will shift to the louder harmonic.

Are all sounds actually made out of sine waves, or is Fourier analysis only an approximation of sounds? When we talk about sounds produced using standing waves, harmonic motion tells us the sound waves are sine waves. Therefore

[5] 1 Hz is a convenient fundamental frequency to get 10 harmonics on a compact plot, but we'd see the same pattern regardless of the fundamental frequency.

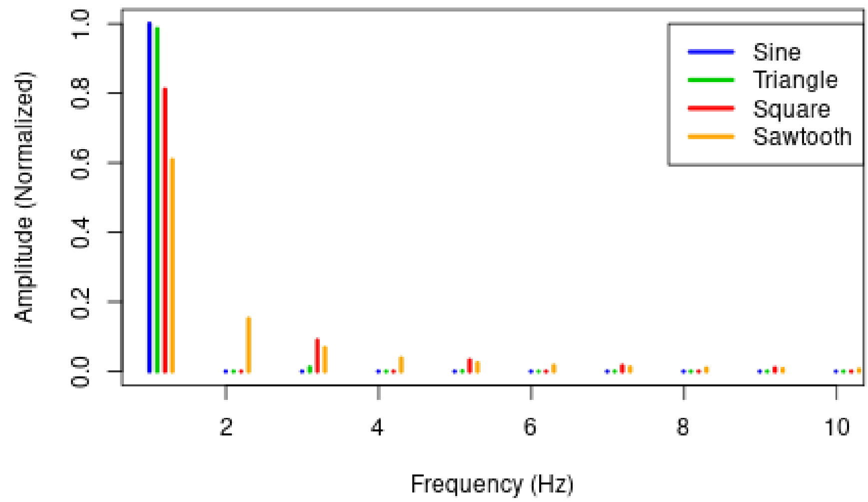

Figure 3.4: Frequency spectrum of four basic waveshapes at 1 Hz. Amplitude is normalized so all harmonics sum to 1. Harmonics are offset for comparison.

when Fourier analysis is applied to these types of sounds, its solution closely corresponds to how the sounds were generated. However, when the sounds are produced by other means, Fourier analysis no longer corresponds to how the sounds were generated, even though it may approximate them arbitrarily. These distinctions perhaps matter when we are concerned with listening to sound, since our auditory system decomposes complex sounds into frequency bands analogous to Fourier analysis, regardless of how the sound was produced.

3.3 Inharmonic standing waves and noise

While standing waves in one dimension, like a string, are necessarily harmonic, standing waves in two dimensions are typically not harmonic. Inharmonic means that the frequencies of the standing waves are not integer multiples of the fundamental. Two-dimensional standing waves are common in drums, cymbals, and related percussion instruments, where they are called modes, and the physics behind the acoustics of these instruments can get very complex. Recall that standing waves in a string had nodes which were points of no movement. In a circular membrane (a drum) or a circular plate (a cymbal) the nodes are lines and circles, because the waves can now travel in two dimensions and reflect off the edges of the vibrating surface. Some modes of a vibrating drum are shown in Figure 3.5. Each of these modes has 0, 1, or 2 nodal lines across the diameter of the drum head and a nodal circle around the perimeter of the drum head. Modes are usually denoted in pairs (d, c) where d is the number of nodal lines across the diameter and c is the number of nodal circles.

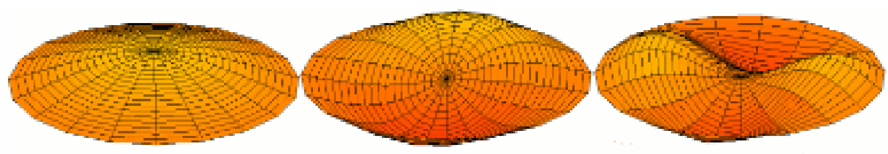

Figure 3.5: Animations of drum head vibration modes with a single nodal circle and 0, 1, or 2 nodal lines. Images public domain.

As with standing waves on a string, when a drum or cymbal is struck, infinitely many modes are excited all at once. Some of these modes are very efficient in transferring energy to the air while others are less efficient, which causes the frequencies of the modes to be relatively short, or longer lived, respectively. The fundamental mode, in particular, is so efficient at transferring energy to the air that it quickly disappates energy and dies out, which means that in these instruments, the fundamental frequency is very weak compared to harmonic instruments.

In general, the modes have inharmonic frequency relationships to each other as shown in Table 3.1. The timpani is a notable counterexample where the kettle and the style of playing enhance the $(d, 1)$ modes, a few of which in the timpani have harmonic relationships to each other. Cymbals similarly have inharmonic frequency relationships but differ in modes and spread of ratios across modes.

Table 3.1: Modes of a vibrating membrane like a drum head and their relative frequencies with respect to (0,1). Note the ratio as not integers and so the series is inharmonic.

Mode	Frequency ratio
(0,1)	1
(1,1)	1.594
(2,1)	2.136
(0,2)	2.296
(3,1)	2.653
(1,2)	2.918
(4,1)	3.156
(2,2)	3.501
(0,3)	3.600
(5,1)	3.652

Harmonics are defined as integer multiples of the fundamental frequency, and the frequency relationships in Table 3.1 are clearly not integers. Note also that they are much more closely packed together than integers: there are 10 modes with a frequency ratio below 4, whereas in a harmonic series we'd only expect four standing waves in that space. Because the frequency relationships are inharmonic, we can only call the standing wave frequencies of such instruments partials - they are not harmonics.[6]

Because the partials of percussion instruments are dense, their relationships inharmonic, and the fundamental weak, percussion instruments are commonly approximated in electronic music using some form of noise. Noise is a random mixture of frequencies, so by definition it does not have harmonic relationships or a fundamental, and the respective frequencies are very close together. For percussion instruments that have more pitch to them, one can add a simple wave, like a sine wave, to convey the sense of a fundamental.

Various types of noise have been defined with different acoustic properties. The noise types have color names that may be helpful for remembering which is which, though in some cases the color names seem a bit arbitrary. The first and most commonly though of noise is white noise. You can see this kind of noise on an old television set (the "snow" on channel with no broadcast). White noise has equal energy across its frequency spectrum. Since we perceive each doubling of frequency as an octave increase in pitch, this means that high frequency sounds are more prominent in white noise. All other colors of noise are based on white noise but change the distribution of energy in the frequency spectrum in some way, i.e. increase the amplitude of some frequencies and decrease the amplitude of others, as shown in Figure 3.6.

[6]They are not overtones either. An overtone is a harmonic above the fundamental.

Noise type	Frequency spectrum	Sound	Noise type	
White	Frequency spectrum of white noise	©	Gray	sp nc
Pink	Frequency spectrum of pink noise	©	Blue	sp nc

Figure 3.6: Frequency spectrum and sound of various "colors" of noise.

Pink noise balances energy across each octave to *roughly* approximate human perception by reducing energy (i.e., amplitude) as frequency increases. Brown noise (also called red noise) is like pink noise but reduces energy more quickly as frequency increases, which puts more energy in lower octaves. Blue noise is the opposite of pink noise: it increases energy as frequency increases by the same amount that pink noise reduced energy. Similarly, violet noise (also called purple noise) is the opposite of brown noise, and increases energy with frequency by the same amount. Finally, grey noise is based on psychoacoustics and places energy on a curve that might seem familiar - compare it to the equal loudness contour in Figure 2.1. Grey noise thus takes into account that humans hear some frequencies better than others, and allocates energy so that all frequencies sound equally loud. One might consider grey noise the the evolution of pink noise. Both take human perception into account, but grey noise does so in a more nuanced way than pink noise.

3.4 Dynamics and Envelopes

Up to this point, the discussion has focused on frequencies of sound and their relative strengths, with only occasional reference to how sound unfolds over time. However, the way a sound unfolds over time plays a critical role in its timbre. For example, think back to a time when you heard a sound played backwards. That reversed sound had the exact same frequencies and distribution of energy as the original, but the reversed version probably sounded pretty bizarre. How sounds unfold over time is sometimes referred to as dynamics in music and envelopes in physics. Our focus is on the sound of a single instrument over time, so is aligned with timbral dynamics and envelopes.

The basic concept of an envelope is that the amplitude of a sound changes over time. Ideally we'd model this change in sound with a complex curve that goes from zero, up for a time, and then back down to zero for each instrument. Such curves would be fairly complicated for different instruments and would clearly vary with how hard the instrument was played, e.g. how hard a string was plucked or a drum hit. To simplify matters, early developers of electronic music settled on envelopes with discrete stages: attack, decay, sustain, release (ADSR). All of the stages except sustain are based on time, as shown in Figure 3.7. Attack is the time it takes for an instrument's sound to reach peak amplitude, decay is the time it takes to decrease from peak amplitude to the next stage or zero, and release is the time it takes to decrease from sustain to zero. None of these set amplitude levels - they only determine how fast amplitude levels are reached (max, sustain, and zero, respectively). Sustain, on the other hand, is an amplitude level rather than a time. The duration of sustain is as long as the stage is held, e.g. a finger on a key. In this way, and ADSR envelope is a simple, yet fairly flexible model of a physical envelope.

The full ADSR envelope makes the most sense on a keyboard instrument where

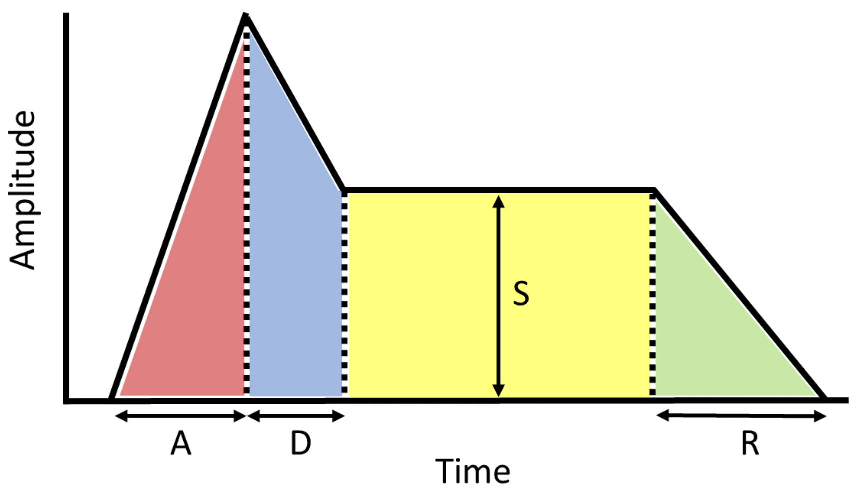

Figure 3.7: An example Attack-Decay-Sustain-Release (ADSR) envelope. Sustain ends with manual control and is the only parameter that sets amplitude level. All other stage lengths are controlled by time parameters as indicated. Note that

pressing a key begins the attack for its duration, which then gives way to decay for its duration. Sustain is then held as long as the key is held, and release begins as soon as the finger leaves the key and lasts for its duration before the amplitude returns to zero. Clearly the full ADSR envelope does not make sense for other instruments. For example, a drum only needs AD, as does a plucked string. Examples of sounds shaped by envelopes are given in Figure 3.8. While the examples in Figure 3.8 only use envelopes for amplitude, envelopes are commonly used to control other properties that change over time. One example is the brightness of a string when it is first plucked, followed by a mellowing of the tone as the higher frequencies disappear. This effect can be created by using an envelope on a filter, a technique we will cover in a future chapter.

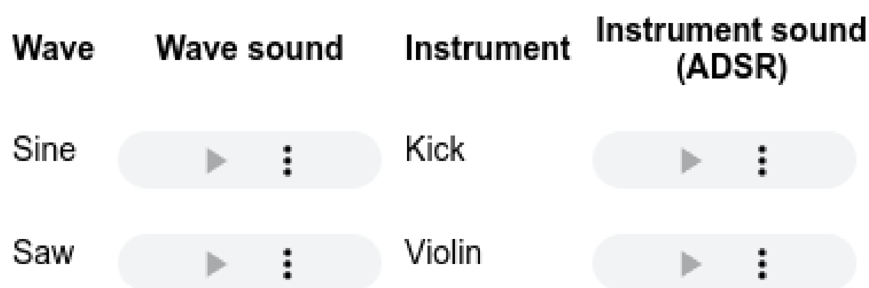

Figure 3.8: Sounds of basic sound waves shaped by envelopes. The kick has fast attack and decay, and the violin has relatively slow attack, decay, and release.

Part II

Fundamental Modules

Chapter 4

Basic Concepts

Chapter 5

Trigger

Chapter 6

Create

Chapter 7

Modify

Part III

Sound Design 1

Chapter 8

Kick & Cymbal

Chapter 9

Lead & Bass

Part IV

Complex Modules

Chapter 10

Trigger

Chapter 11

Create

Chapter 12

Modify

Part V

Sound Design 2

Chapter 13

Minimoog & 303

Bibliography

Bell, T. (2021). CS unplugged or coding classes? *Communications of the ACM*, 64(5):25–27.

Bjørn, K. and Meyer, C. (2018). *Patch & Tweak: Exploring Modular Synthesis*. Bjooks. Google-Books-ID: xmrVvQEACAAJ.

Bode, H. (1984). History of electronic sound modification. 32(10):730–739.

Crombie, D. (1982). *The Complete Synthesizer: A Comprehensive Guide*. Omnipress Books.

Doepfer Musikelektronik (2022a). Construction Details A-100.

Doepfer Musikelektronik (2022b). Technical details A-100.

Dudley, H. and Tarnoczy, T. H. (1950). The speaking machine of Wolfgang von Kempelen. 22(2):151–166. Publisher: Acoustical Society of America.

Dusha, T., Martonova, A., and Johnston, P. (2020). *Modular Sound Synthesis On the Moon*. Modular Moon. Google-Books-ID: llA4zgEACAAJ.

Eliraz, Z. (2022). Loopop's In-Complete Book of Electronic Music.

Jacoby, N., Undurraga, E. A., McPherson, M. J., Valdés, J., Ossandón, T., and McDermott, J. H. (2019). Universal and Non-universal Features of Musical Pitch Perception Revealed by Singing. 29(19):3229–3243.e12. Publisher: Elsevier.

Michie, D. (1963). Experiments on the Mechanization of Game-Learning Part I. Characterization of the Model and its parameters. 6(3):232–236.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York.

Polya, G. (2004). *How to solve it: A new aspect of mathematical method*. Princeton University Press.

Strange, A. (1983). *Electronic Music: Systems, Techniques, and Controls*. William C Brown Publishers, 2 edition.

Tedre, M. and Denning, P. J. (2016). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, page 120–129, New York, NY, USA. Association for Computing Machinery.

Tucker, A. (2003). A model curriculum for K–12 computer science: Final report of the ACM K–12 task force curriculum committee. Technical report, New York, NY, USA.