DIGITAL SOUND & MUSIC
CONCEPTS, APPLICATIONS, AND SCIENCE

Chapter 3 - Musical Sound

Section 3.1 - Concepts

3.1.1 Context

Even if you're not a musician, if you plan to work in the realm of digital sound you'll benefit from an understanding of the basic concepts and vocabulary of music. The purpose of this chapter is to give you this foundation.

This chapter describes the vocabulary and musical notation of the Western music tradition – the music tradition that began with classical composers like Bach, Mozart, and Beethoven and that continues as the historical and theoretic foundation of music in the United States, Europe, and Western culture. The major and minor scales and chords are taken from this context, which we refer to as Western music. Many other types of note progressions and intervals have been used in other cultures and time periods, leading to quite different characteristic sounds: the modes of ancient Greece, the Gregorian chants of the Middle Ages, the pentatonic scale of ancient Oriental music, the Hindu 22 note octave, or the whole tone scale of Debussy, for example. While we won't cover these, we encourage the reader to explore these other musical traditions.

To give us a common language for understanding music, we focus our discussion on the musical notation used for keyboards like the piano. Keyboard music expressed and notated in the Western tradition provides a good basic knowledge of music and gives us a common vocabulary when we start working with MIDI in Chapter 6.

3.1.2 Tones and Notes

Musicians learn to sing, play instruments, and compose music using a symbolic language of music notation. Before we can approach this symbolic notation, we need to establish a basic vocabulary.

In the vocabulary of music, a sound with a single fundamental frequency is called a **tone**. The **fundamental frequency** of a tone is the frequency that gives the tone its essential pitch. A piccolo plays tones with higher fundamental frequencies than the frequencies of a flute, and thus it is higher pitched.

A tone that has an onset and a duration is called a **note**. The **onset** of the note is the moment when it begins. The **duration** is the length of time that the note remains audible. Notes can be represented symbolically in musical notation, as we'll see in the next section. We will also use the word "note" interchangeably with "key" when referring to a key on a keyboard and the sound it makes when struck.

As described in Chapter 2, tones created by musical instruments, including the human voice, are not single-frequency. These tones have **overtones** at frequencies higher than the fundamental. The overtones create a timbre, which distinguishes the quality of the tone of one instrument or singer from another. Overtones add a special quality to the sound, but they don't change our overall perception of the pitch. When the frequency of an overtone is an integer multiple of the fundamental frequency, it is a **harmonic overtone**. Stated mathematically for frequencies f_1 and f_2, if $f_2 = nf_1$ and n is a positive integer, then f_2 is a **harmonic frequency** relative to fundamental frequency f_1. Notice that every frequency is a harmonic frequency relative to itself. It is called the first harmonic, since $n = 1$. The second harmonic is the frequency where $n = 2$. For example, the second harmonic of 440 Hz is 880 Hz; the third harmonic of 440 Hz is 3*440 Hz = 1320 Hz; the fourth harmonic of 440 Hz is 4*440 Hz = 1760 Hz; and so forth. Musical instruments like pianos and violins have harmonic overtones. Drums beats and other non-pitched sounds have overtones that are not harmonic.

Another special relationship among frequencies is the **octave**. For frequencies f_1 and f_2, if $f_2 = 2^n f_1$ where n is a positive integer, then f_1 and f_2 "sound the same," except that f_2 is higher pitched than f_1. Frequencies f_1 and f_2 and are separated by n **octaves**. Another way to describe the octave relationship is to say that each time a frequency is moved up an octave, it is multiplied by 2. A frequency of 880 Hz is one octave above 440 Hz; 1760 Hz is two octaves above 440 Hz; 3520 Hz is three octaves above 440 Hz; and so forth. Two notes separated by one or more octaves are considered equivalent in that one can replace the other in a musical composition without disturbing the harmony of the composition.

In Western music, an octave is separated into 12 frequencies corresponding to notes on a piano keyboard, named as shown in Figure 3.1. From C to B we have 12 notes, and then the next octave starts with another C, after which the sequence of letters repeats. An octave can start on any letter, as long as it ends on the same letter. (The sequence of notes is called an octave because there are eight notes in a diatonic scale, as is explained below.) The white keys are labeled with the letters. Each of the black keys can be called by one of two names. If it is named relative to the white key to its left, a **sharp** symbol is added to the name, denoted C#, for example. If it is named relative to the white key to its right, a **flat** symbol is added to the name, denoted D ♭, for example.

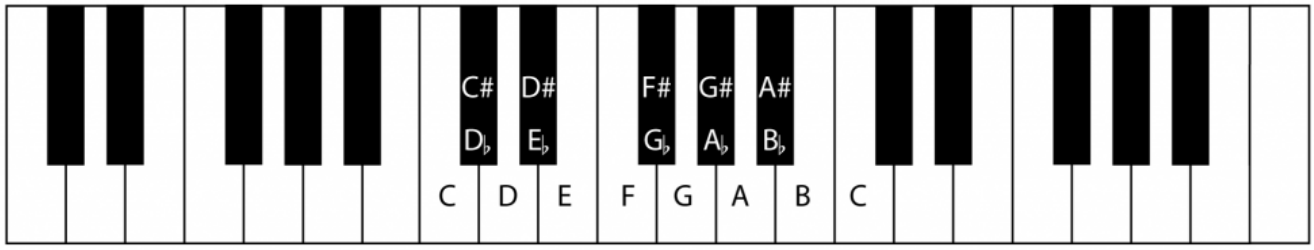


Figure 3.1 Keyboard showing octave and key labels

Each note on a piano keyboard corresponds to a physical key that can be played. There are 88 keys on a standard piano keyboard. MIDI keyboards are usually smaller. Since the notes from A through G are repeated on the keyboard, they are sometimes named by the number of the octave that they're in, as shown in Figure 3.2.

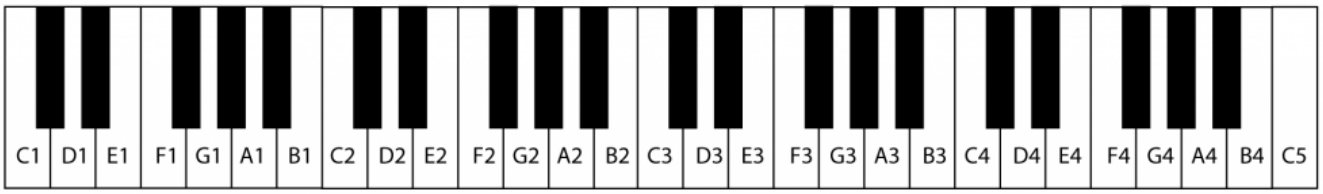


Figure 3.2 MIDI keyboard

Middle C on a standard piano has a frequency of approximately 262 Hz. On a piano with 88 keys, middle C is the fourth C, so it is called C4. On the smaller MIDI keyboard shown above, it is C3. Middle C is the central position for playing the piano, with regard to where the right and left hands of the pianist are placed. The standard reference point for tuning a piano is the A above middle C, which has a frequency of 440 Hz. This means that the next A going up the keys to the right has a frequency of 880 Hz. A note of 880 Hz is one octave away from 440 Hz, and both are called A on a piano keyboard.

The interval between two consecutive keys (also called notes) on a keyboard, whether the keys are black or white, is called a **semitone**. A semitone is the smallest frequency distance between any two notes. Neighboring notes on a piano keyboard (and equivalently, two neighboring notes on a chromatic scale) are separated by a frequency factor of approximately 1.05946. This relationship is described more precisely in the equation below.

Let f be the frequency of a note k. Then the note one octave above f has a frequency of $2f$. Given this octave relationship and the fact that there are 12 notes in an octave, the frequency of the note after k on a chromatic scale is $\sqrt[12]{2}f \approx 1.05946f$.

Equation 3.1

Thus, the factor 1.05946 defines a semitone. If two notes are divided by a semitone, then the frequency of the second is 1.05946 times the frequency of the first. The other frequencies between semitones are not used in Western music

(except in pitch bending).

Two semitones constitute a **whole tone**, as illustrated in Figure 3.3. Semitones and whole tones can also be called **half steps** and **whole steps** (or just steps), respectively. They are illustrated in Figure 3.3.

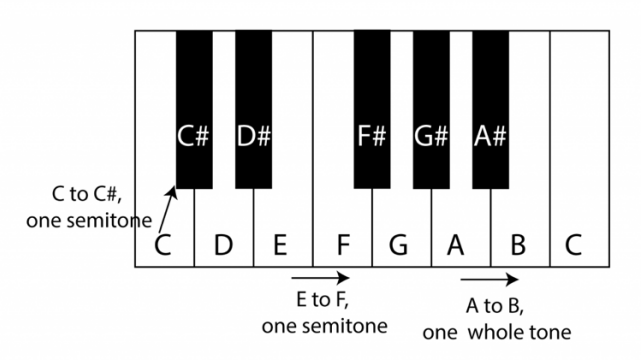


Figure 3.3 Semitones and whole tones

The symbol #, called a **sharp**, denotes that a note is to be raised by a semitone. When you look at the keyboard in Figure 3.3, you can see that moving up by a semitone takes you to the F key. Thus E# denotes and sounds the same note as F. When two notes have different names but are the same pitch, they said to be **enharmonically equivalent**.

The symbol ♭ , called a **flat**, denotes that a note is to be lowered by a semitone. C ♭ is enharmonically equivalent to B. A natural symbol ♮ removes a sharp or flat from a note when it follows the same note in a measure. Sharps, flats, and naturals are examples of **accidentals**, symbols that raise or lower a note by a semitone.

3.1.3 Music vs. Noise

It's fascinating to consider the way humans perceive sound, experiencing some sounds as musical and some as noise. Understanding frequency components and harmonics gives us some insight into why music is pleasing to our senses.

Consider the waveforms in Figure 3.4 and Figure 3.5. The first shows the notes C, E, and G played simultaneously. The second is a recording of scratching sounds. The first waveform has a regular pattern because the frequency components are pure sine waves with a harmonic relationship. Patterns such as this are common in music and the sounds produced by acoustic instruments. The second sound has no pattern; the relationship of the frequency components is random at any moment in time and varies randomly over time. Randomness is characteristic of noise.

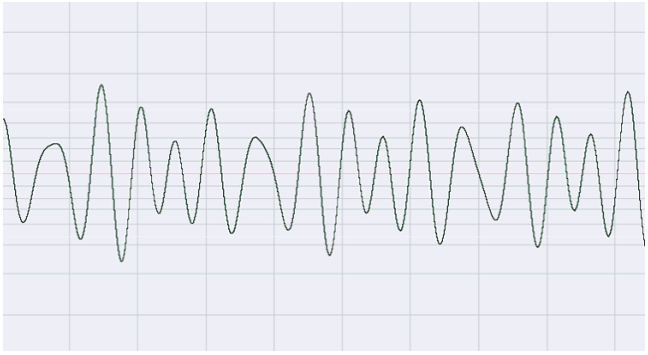


Figure 3.4 Waveform of musical sound

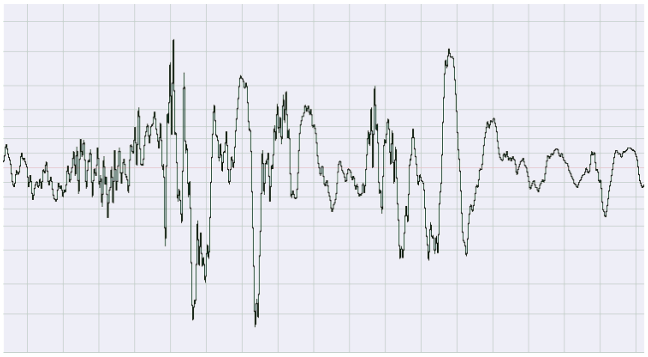


Figure 3.5 Waveform of noise

Musical instruments like violins, pianos, clarinets, and flutes naturally emit sounds with unique harmonic components. Each instrument has an identifiable **timbre** governed by its overtone series. This timbre is sometimes called the instrument's **tone color**, which results from its shape, the material of which it is made, its resonant structure, and the way it is played. These physical properties result in the instrument having a characteristic range of frequencies and harmonics. (See Figure 3.6.)

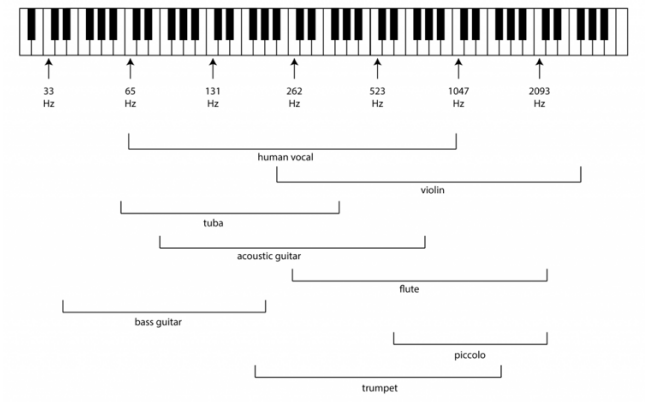


Figure 3.6 Approximate frequencies of various instruments

Instruments also are distinguished by the **amplitude envelope** for individual sounds created by the instrument. The amplitude envelope gives a sense of how the loudness of a single note changes over the short period of time when it is played. When you play a certain instrument, do you burst into the note or slide into it gently? Does the note linger or end abruptly? Imagine a single note played by a flute compared to the same note played by a piano.

Although you don't always play a piano note the same way – for example, you can strike the key gently or briskly – it's still possible to get a picture of a typical amplitude envelope for each instrument and see how they differ. The amplitude envelope consists of four components: **attack**, **decay**, **sustain**, and **release**, abbreviated **ADSR**, as illustrated in Figure 3.7. The attack is the time between when the sound is first audible and when it reaches its maximum loudness. The decay is the period of time when the amplitude decreases. Then the amplitude can level to a plateau in the sustain period. The release is when the sound dies away. The attack of a trumpet is relatively sudden, rising steeply to its maximum, because you have to blow pretty hard into a trumpet before the sound starts to come out. With a violin, on the other hand, you can stroke the bow across a string gently, creating a longer, less steep attack. The sustain of the violin note might be longer than that of the trumpet, also, as the bow continues to stroke across the string. Of course, these envelopes vary in individual performances depending on the nature of the music being played. Being aware of the amplitude envelope that is natural to an instrument helps in the synthesis of music. Tools exist for manipulating the envelopes of MIDI samples so that they sound more realistic or convey the spirit of the music better, as we'll see in Chapter 6.

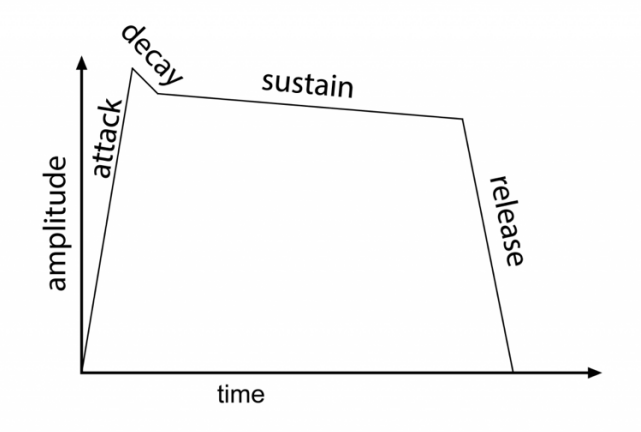


Figure 3.7 Amplitude envelope

The combination of harmonic instruments in an orchestra gives rise to an amazingly complex pattern of frequencies that taken together express the aesthetic intent of the composer. Non-harmonic instruments – e.g., percussion instruments – can contribute beauty or harshness to the aesthetic effect. Drums, gongs, cymbals, and maracas are not musical in the same sense that flutes and violins are. The partials (frequency components) emitted by percussion instruments are not integer multiples of a fundamental, and thus these instruments don't have a distinct pitch with harmonic overtones. However, percussion instruments contribute accents to music, called **transients**. Transients are high-frequency sounds that come in short bursts. Their attacks are quite sharp, their sustains are short, and their releases are steep. These percussive sounds are called "transient" because they come and go quickly. Because of this, we have to be careful not to edit them out with noise gates and other processors that react to sudden changes of amplitude. Even with their non-harmonic nature, transients add flavor to an overall musical performance, and we wouldn't want to do without them.

3.1.4 Scales

Let's return now to a bit more music theory as we establish a working vocabulary and an understanding of musical notation.

A **scale** is an ordered set of musical notes that are defined relative to a fundamental frequency, also known as the **tonal center** or **tonic**. There are different types of scales defined in music, which vary in the number of notes played and the intervals between the notes.

Table 3.1 lists seven types of scales divided into three general categories – chromatic, diatonic, and pentatonic. A list of 1s and 2s is a convenient way to represent the semitone intervals between each note in a scale. Moving by a semitone is represented by the number 1, meaning "move over one note from the previous one." Moving by a whole tone is represented by the number 2, meaning "move over two notes from the previous one."

Max Demo:
Scale Generator

A **chromatic scale** consists of 12 notes. (However, when the scale is played, usually the 13th note is played, which is one octave above the first note.) A chromatic scale starts on any pitch, and then each note is separated from the previous one by a semitone. Thus the pattern of a chromatic scale (where the 13th note is played) is represented simply by the list [1 1 1 1 1 1 1 1 1 1 1 1]. Note that while 13 notes are played, there are only 12 numbers on the list. The first note is played, and then the list represents how many semitones to move over to play the following notes.

A **diatonic scale** has seven notes. In a **major diatonic scale**, the pattern is [2 2 1 2 2 2 1] (assuming again that one more note is added as the scale is played, ending the scale one octave above where it began). This means, "start on the first note, move over by a tone, a tone, a semitone, a tone, a tone, a tone, and a semitone." A major diatonic scale beginning on D3 (and adding an extra note) is depicted in Figure 3.8. Those who grow up in the tradition of Western music become accustomed to the sequence of sounds in a diatonic scale as the familiar "do re mi fa so la ti do." A diatonic scale can start on any note, but all diatonic scales have the same pattern of sound in them; one scale is just higher or lower than another. Diatonic scales sound the same because the differences in the frequencies between consecutive notes on the scale follow the same pattern, regardless of the note you start on.

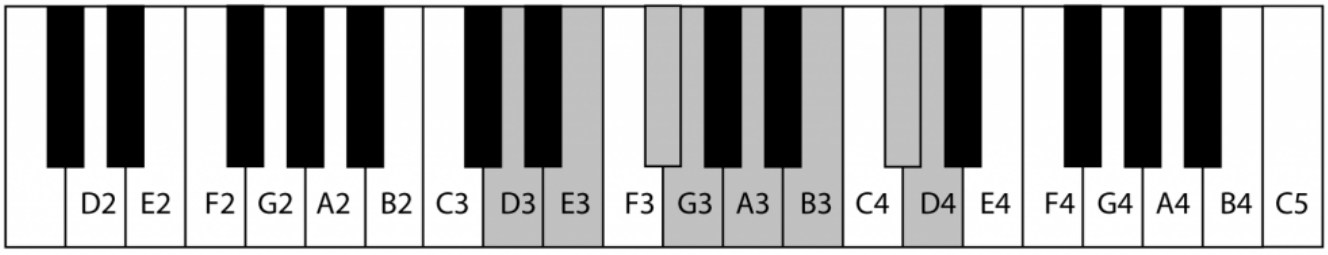


Figure 3.8 The key of D

A **minor diatonic scale** is played in the pattern [2 1 2 2 1 2 2]. This pattern is also referred to as the **natural minor**. There are variations of minor scales as well. The **harmonic minor** scale follows the pattern [2 1 2 2 1 3 1], where 3 indicates a step and a half. The **melodic minor** follows the pattern [2 1 2 2 2 2 1] in the ascending scale, and [2 2 1 2 2 1 2] in the descending scale (played from highest to lowest note). The pattern of whole and half steps for major and minor scales are given in Table 3.1.

Type of scale	Pattern of whole steps (2) and half steps (1)

chromatic	1 1 1 1 1 1 1 1 1 1 1 1
major diatonic	2 2 1 2 2 2 1
natural minor diatonic	2 1 2 2 1 2 2
harmonic minor diatonic	2 1 2 2 1 3 1
melodic minor diatonic	2 1 2 2 2 2 1 ascending 2 2 1 2 2 1 2 descending
pentatonic major	2 3 2 2 3 or 2 2 3 2 3
pentatonic minor	3 2 3 2 2 or 3 2 2 3 2

Table 3.1 Pattern of whole and half steps for various scales

Two final scale types ought to be mentioned because of their prevalence in music of many different cultures. These are the pentatonic scales, created from just five notes. An example of a pentatonic major scale results from playing only black notes beginning with F#. This is the interval pattern 2 2 3 2 3, which yields the scale F#, G#, A#, C#, D#, F#. If you play these notes, you might recognize them as the opening strain from “My Girl” by the Temptations, a pop song from the 1960s. Another variant of the pentatonic scale has the interval pattern 2 3 2 2 3, as in the notes C D F G A C. These notes are the ones used in the song “Ol’ Man River” from the musical *Showboat* (Figure 3.9).

Ol' man river,
C C D F
Dat ol' man river
D C C D F
He mus'know sumpin'
G A A G F
But don't say nuthin',
G A C D C
He jes'keeps rollin'
D C C A G
He keeps on rollin' along.
A C C A G A F

Figure 3.9 Pentatonic scale used in "Ol Man River"

To create a minor pentatonic scale from a major one, begin three semitones lower than the major scale and play the same notes. Thus, the pentatonic minor scale relative to F#, G#, A#, C#, D#, F# would be D#, F#, G#, A#, C#, D#, which is a D# pentatonic minor scale. The minor relative to C D F G A C would be A C D F G A, which is an A pentatonic minor scale. As with diatonic scales, pentatonic scales can be set in different keys.

It's significant that a chromatic scale really has no variation in the sequence of notes played. We simply move up by semitone steps of [1 1 1 1 1 1 1 1 1 1 1 1]. The distance between neighboring notes is always the same. In a diatonic scale, on the other hand, there *is* a pattern of varying half steps and whole steps between notes. The pattern gives notes different importance to the ear relative to the **key**, also called the **tonic**. This difference in the roles that notes play in a diatonic scale is captured in the names given to these notes, listed in Table 3.2. Roman numerals are conventionally used for the positions of these notes. These names will become more significant when we discuss chords later in this chapter.

Position	Name for note
I	tonic
II	supertonic
III	mediant
IV	subdominant
V	dominant
VI	submediant
VII	leading note

Table 3.2 Technical names for notes on a diatonic scale

3.1.5 Musical Notation

3.1.5.1 Score

In the tradition of Western music, a system of symbols has been devised to communicate and preserve musical

compositions. This system includes symbols for notes, timing, amplitude, and keys. A musical composition is notated on a **score**. There are scores of different types, depending on the instrument being played. The score we describe is a standard piano or keyboard score.

A piano score consists of two **staves**, each of which consists of five horizontal lines. The staves are drawn one above the other. The top staff is called the **treble clef**, representing the notes that are played by the right hand. The symbol for the treble clef is placed at the far left of the top staff. (The word "clef" is French for "key.") The bottom staff is the **bass clef**, representing what is played with the left hand. The symbol for the bass clef is placed at the far left of the bottom staff. A blank score with no notes on it is pictured in Figure 3.10.

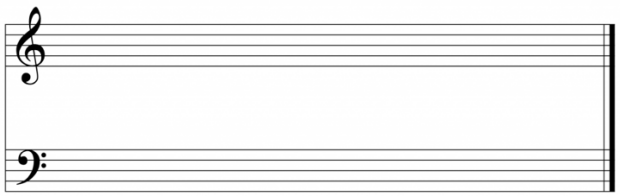


Figure 3.10 Treble and bass clef staves

Each line and space on the treble and bass clef staves corresponds to a note on the keyboard to be played, as shown in Figure 3.11 and Figure 3.12. A whole note (defined below) is indicated by an oval like those shown. The letter for the corresponding note is given in the figures. The letters are ordinarily not shown on a musical score. We have them here for information only. The treble clef is sometimes called the G clef because its bottom portion curls around the line on the staff corresponding to the note G. The bass clef is sometimes called the F clef because it curls around the line on the staff corresponding to the note F.

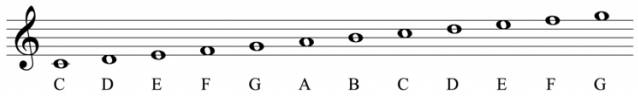


Figure 3.11 Notes on the treble clef staff

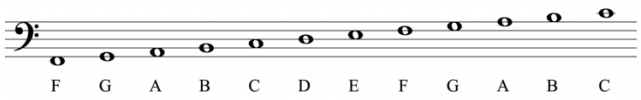


Figure 3.12 Notes on the bass clef staff

It's possible to place a note below or above one of the staves. The note's distance from the staff indicates what note it is. If it's a note that would fall on a line, a small line is placed through it, and if the note would fall on a space, a line is placed under or over it. The lines between the staff and the upper or lower note are displayed by a short line, also. The short lines that are used to extend the staff are called **ledger lines**. Examples of notes with ledger lines are shown in Figure 3.13 and Figure 3.14.

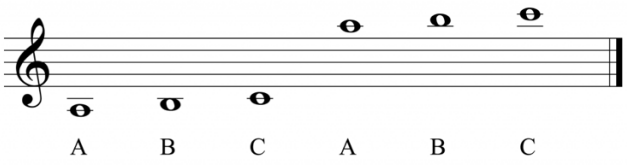


Figure 3.13 Placing notes above or below the treble clef staff

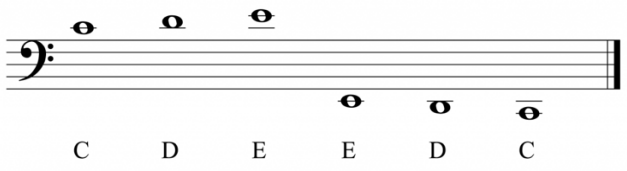


Figure 3.14 Placing notes above or below the bass clef staff

Beginners who are learning to read music often use mnemonics to remember the notes corresponding to lines and spaces on a staff. For example, the lines on the treble clef staff, in ascending order, correspond to the notes E, G, B, D, and F, which can be memorized with the mnemonic "every good boy does fine." The spaces on the treble clef staff, in ascending order, spell the word FACE. On the bass clef, the notes G, B, D, F, and A can be remembered as "good boys deserve favor always," and the spaces A, C, E, G can be remembered as "a cow eats grass" (or whatever mnemonics you like).

Max Demo:
Recognizing Notes

3.1.5.2 Notes and their Durations

There are types of notes – whole notes, half notes, quarter notes, and so forth, as shown in Table 3.3. On the score, you can tell what type a note is by its shape and whether or not the note is filled in (black or white). The durations of notes are defined relative to each other, as you can see in the table. We could continue to create smaller and smaller notes, beyond those listed in the table, by adding more flags to a note. The part of the note called the **flag** is shown in Figure 3.15. Each time we add a flag, we divide the duration of the previously defined note by 2.

Note	Name	Duration
	thirty-second note	
	sixteenth note	
	eighth note	
	quarter note	
	half note	
	whole note	

Table 3.3 Notes and their durations

Figure 3.15 Flag on note

3.1.5.3 Rhythm, Tempo, and Meter

The timing of a musical composition and its performance is a matter of meter, tempo, and rhythm.

Meter is the regular grouping of beats and placement of accents in notes. Since the durations of different notes are defined relative to each other, we have to have a base line. This is given in a score's **time signature** (a synonym of meter), which indicates which note gets one **beat**. (There are two types of meter – simple and

compound. We are considering only simple meter here.)

Flash Tutorial:
Beats in a Measure

Beats and notes are grouped into **measures**. Measures are sometimes called **bars** because they are separated from each other on the staff by a vertical line called a **bar line**.

A time signature of $\frac{x}{y}$, where *x* and *y* are integers, indicates that there are *x* beats to a measure, and a note of size $\frac{1}{y}$ gets one beat. (A half note is size ½, a quarter note is size ¼, and so forth.) A time signature of $\frac{4}{4}$ time is shown in Figure 3.16, indicating four beats to a measure with a quarter note getting one beat. A time signature of $\frac{3}{4}$ indicates three beats to a measure with a quarter note getting one beat. $\frac{3}{4}$ is the meter for a waltz.

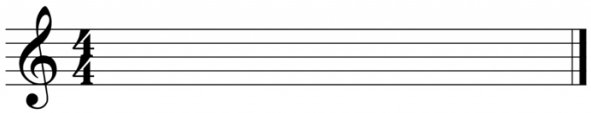


Figure 3.16 Time signature showing 4/4 time

If the time signature of a piece is $\frac{4}{4}$, then there are four beats to a measure and each quarter note gets a beat. Consider the score in Figure 3.17, which shows you how to play "Twinkle, Twinkle Little Star" in $\frac{4}{4}$ time in the key of C. Each of the six syllables in "Twin-kle Twin-kle Lit-tle" corresponds to a quarter note, and each is given equal time – one beat – in the first measure. Then the note corresponding to the word "star" is held for two beats in the second measure. Measures three and four are similar to measures one and two.

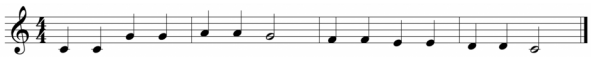


Figure 3.17 Score for right hand of "Twinkle, Twinkle Little Star"

Tempo is the pace at which music is performed. We can sing "Yankee Doodle" in a fast, snappy pace or slowly, like a dirge. Obviously, most songs have tempos that seem more appropriate for them.

The **tempo** of a musical piece depends on which type of note is given one beat and how many beats there are per second. Tempo can be expressed at the beginning of the score explicitly, in terms of **beats per minute** (**BPM**). For example, a marking such as bpm = 72, ♩ = *72*, or mm = 72 (for Maezel's metronome) placed above a measure indicates that a tempo of 72 beats per minute is recommend from this point on, with a quarter note getting one beat. Alternatively, an Italian key word like *allegro* or *andante* can be used to indicate the beats per minute in a way that can be interpreted more subjectively. A list of some of these tempo markings is given in Table 3.4.

Tempo Marking	Meaning
prestissimo	extremely fast (more than 200 BPM)
presto	very fast (168 – 200 BPM)
allegro	fast (120 – 168 BPM)

moderato	moderately (108 – 120 BPM)
andante	“at a walking pace” (76 – 108 BPM)
adagio	slowly (66 – 76 BPM)
lento or largo	very slowly (40 – 60 BPM)

Table 3.4

Students learning to play the piano sometimes use a device called a **metronome** to tick out the beats per minute while they practice. Metronomes are also usually built into MIDI sequencers and keyboards.

Rhythm is a pattern in the length and accents of sounds and pauses between sounds. Rhythm is a complicated feature of music and is difficult to define in a general sense because of its many variations, so we won't go into detail about it here.

3.1.5.4 Rests and their Durations

The duration of silence between notes is indicated by a **rest**. The symbols for rests are given in Table 3.5. The length of a rest corresponds to the length of a note of the same name.

Video Tutorial:
Music with Rests

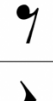
Rest	Name
	thirty-second rest
	sixteenth rest
	eighth rest
	quarter rest
	half rest
	whole rest

Table 3.5 Rests and their durations

A dot placed after a note or rest increases its duration by one-half of its original value. This is shown in Table 3.6. Figure 3.18 shows a fragment of Mozart's "A Little Night Music," which contains eighth rests in the second, third, and fourth measures. Since this piece is in $\frac{6}{8}$ time, the eighth rest gets one beat.

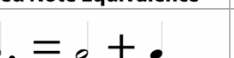
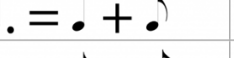
Dotted Note Equivalence	Dotted Rest Equivalence
	
	

Table 3.6 Dotted notes and rests

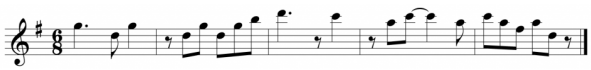


Figure 3.18 Example of music with rests (in compound meter 6/8)

3.1.5.5 Key Signature

Many words in the field of music and sound are overloaded – that is, they have different meanings in different contexts. The word "key" is one of these overloaded terms. Thus far, we have been using the word "key" to denote a physical key on the piano keyboard. (For the discussion that follows, we'll call this kind of key a "piano key.") There's another denotation for key that relates to diatonic scales, both major and minor. In this usage of the word, a **key** is a group of notes that constitute a diatonic scale, whether major or minor.

Each key is named by the note on which it begins. The beginning note is called the **key note** or **tonic** (**tonal center**)

note. If we start a major diatonic scale on the piano key of C, then following the pattern 2 2 1 2 2 2 1, only white keys are played. This group of notes defines the key of C major. Now consider what happens if we start on the note D. If you look at the keyboard and consider playing only white keys starting with D, you can see that the pattern would be 2 1 2 2 2 1 2, not the 2 2 1 2 2 2 1 pattern we want for a major scale. Raising F and C each by a semitone – that is, making them F# and C# – changes the pattern to 2 2 1 2 2 2 1. Thus the notes D, E, F#, G, A, B, C# and D define the key of D. By this analysis, we see that the key of D requires two sharps – F and C. Similarly, if we start on D and follow the pattern for a minor diatonic scale – 2 1 2 2 1 2 2 – we play the notes D, E, F, G, A, B ♭ , and D. This is the key of D minor.

Each beginning note (tonic note) determines the number of sharps or flats that are played in the 2 2 1 2 2 2 1 scale for a major key or in the 2 1 2 2 1 2 2 sequence for a minor key. Beginning an octave on the piano key C implies you're playing in the key of C and play no sharps or flats for C major. Beginning a scale on the piano key D implies that you're playing in the key of D and play two sharps for D major.

Let's try a similar analysis on the key of F. If you play a major scale starting on F using all white keys, you don't get the pattern 2 2 1 2 2 2 1. The only way to get that pattern is to lower the fourth note, B, to B ♭ . Thus, the key of F major has one flat. The key of F minor has four flats – A ♭ , B ♭ , D ♭ , and E ♭ , as you can verify by following the sequence 2 1 2 2 1 2 2 starting on F.

Like meter, the key of a musical composition is indicated at the beginning of each staff, in the key signature. The key signature indicates which notes are to be played as sharps or flats. The key signatures for all the major keys with sharps are given in Table 3.7. The key signatures for all the major keys with flats are given in Table 3.8.

You may have noticed that the keys of F major and D minor have the same key signature, each having one flat, B ♭ . So when you see the key signature for a musical composition and it has one flat in it, how do you know if the composition is written in F major or D minor, and what difference does it make? One difference lies in which note feels like the "home" note, the note to which the music wants to return to be at rest. A composition in the key of F tends to begin and end in F. The use of specific chords reinforces the key, also. A second difference is a subjective response to major and minor keys. Minor keys generally sound more somber, sad, or serious while major keys are bright and happy to most listeners.

You can see in the tables below that each major key has a relative minor key, as indicated by the keys being in the same row in the table and sharing a key signature. Given a major key *k*, the relative minor is named by the note that is three semitones below *k*. For example, A is three semitones below C, and thus A is the relative minor key with respect to C major. When keys are described, if the words "major" and "minor" are not given, then the key is assumed to be a major key.

You've seen how, given the key note, you can determine the key signature for both major and minor keys. So how do you work in reverse? If you see the key signature, how can you tell what key this represents? A trick for major keys is to name the note that is one semitone above the last sharp in the key signature. For example, the last sharp in the second key signature in Table 3.7 is F#. One semitone above F# is G, and thus this is the key of G major. For minor keys, you name the note that is two semitones below the last sharp in the key signature. For the first key signature in Table 3.7, the last sharp is F#. Two semitones below this is the note E. Thus, this is also the key signature for E minor.

To determine a major key based on a key signature with flats, you name the note that is five semitones below the last

flat. In the key that has three flats, the last flat is A ♭ . Five semitones below that is E ♭ , so this is the key of E ♭ major. (This turns out to be the next to last flat in each key with at least two flats.) To determine a minor key based on a key signature with flats, you name the note that is four semitones above the last flat. Thus, the key with three flats is the key of C minor. (You could also have gotten this by going three semitones down from the relative major key.)

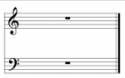
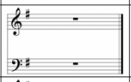
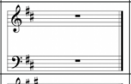
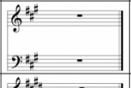
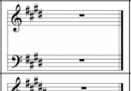
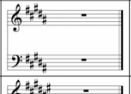
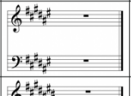
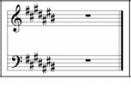
Major key	Sharps	Minor key
C		A
G		E
D		B
A		F♯
E		C♯
B		G♯
F♯		D♯
C♯		A♯

Table 3.7 Relative major and minor keys with sharps

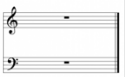
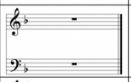
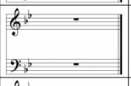
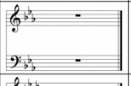
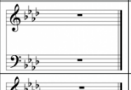
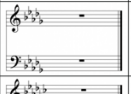
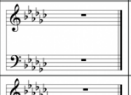
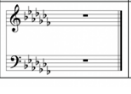
Major key	Sharps	Minor key
C		A
F		D
B♭		G
E♭		C
A♭		F
D♭		B♭
G♭		E♭
C♭		A♭

Table 3.8 Relative major and minor keys with flats

The key signature for a harmonic minor key is the same as the natural minor. To create the pattern [2 1 2 2 1 3 1] (harmonic minor) from [2 1 2 2 1 2 2] (natural minor), it suffices to raise the seventh note a semitone. Since the natural minor's key signature is being used, sharps or naturals are needed on individual notes to create the desired pattern of whole tones and semitones for the harmonic minor. The scale for a example harmonic minor is notated as

shown in Figure 3.19. (In this section and following ones, we won't include measures and time signatures when they are not important to the point being explained.)

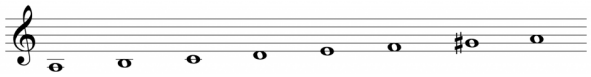


Figure 3.19 A minor harmonic

Similarly, the melodic minor uses the key signature of the natural minor. Then adjustments have to be made in both the ascending and descending scales to create the melodic minor. To create the ascending pattern [2 1 2 2 2 2 1] (melodic minor) from [2 1 2 2 1 2 2] (natural minor), it suffices to raise both the sixth and seventh note by a semitone. This is done by adding sharps, flats, or naturals as necessary, depending on the key signature. The descending pattern for a melodic minor is the same as for the natural minor – [2 2 1 2 2 1 2]. Thus, the notes that are raised in the ascending scale have to be returned to their natural minor position for the key by adding sharps, flats, or naturals, as necessary. This is because once an accidental is added in a measure, it applies to the note to which it was added for the remainder of the measure.

F minor melodic is notated as shown in Figure 3.20. In F minor melodic, since D and E are normally flat for that key, a natural is added to each to remove the flat and thus move them up one semitone. Then in the descending scale, the D and E must explicitly be made flat again with the flat sign.

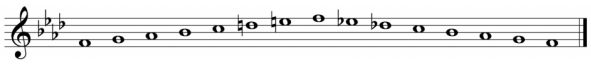


Figure 3.20 F minor melodic

In E minor melodic, a sharp is added to the sixth and seventh notes – C and D – to raise them each as semitone. Then in the descending scale, the D and E must explicitly be made natural again with the sign, as shown in Figure 3.21.

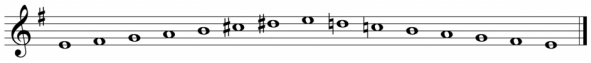


Figure 3.21 E minor melodic

3.1.5.6 The Circle of Fifths

In Section 3.1.5, we showed how keys are distinguished by the sharps and flats they contain. One way to remember the keys is by means of the **circle of fifths**, a visual representation of the relationship among keys. The circle is shown in Figure 3.22. The outside of the circle indicates a certain key. The inner circle tells you how many sharps or flats are in the key.

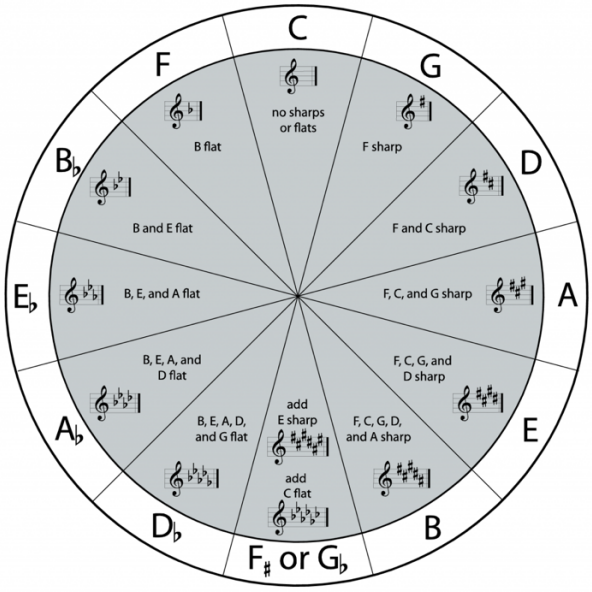


Figure 3.22 Circle of fifths

A **fifth** is a type of interval. We'll discuss intervals in more detail in Section 3.1.6.2, but for now all you need to know that a fifth is a span of notes that includes five lettered notes. For example, the notes A, B, C, D, and E comprise a fifth, as do C, D, E, F, and G.

If you start with the key of C and move clockwise by intervals of fifths (which we call "counting up"), you reach successive keys each of which has one more sharp than the previous one – the keys of G, D, A, and so forth. For example, starting on C and count up a fifth, you move through C, D, E, F, and G, five lettered notes comprising a fifth. Counting up a fifth consists of moving up seven semitones – C to C#, C# to D, D to D#, D# to E, E to F, F to F#, and F# to G. Notice that on the circle, moving clockwise, you're shown G as the next key up from the key of C. The key of G has one sharp. The next key up, moving by fifths, is the key of D, which has two sharps. You can continue counting up through the circle in this manner to find successive keys, each of which has one more sharp than the previous one.

Flash Tutorial:
Key Signatures

Starting with the key of C and moving *counterclockwise* by intervals of fifths ("counting down"), you reach successive keys each of which has one more flat than the previous one – the keys of F, B ♭ , E ♭ , and so forth. Counting down a fifth from C takes you through C, B, A, G, and F – five lettered notes taking you through seven semitones, Counting down a fifth from F takes you through F, E, D, C, and B. However, since we again want to move seven semitones, the B is actually B ♭ . The seven semitones are F to E, E to E ♭ , E ♭ to D, D to D ♭ , D ♭ to C, C to B, and B to B ♭ . Thus, the key with two flats is the key of B ♭ .

The keys shown in the circle of fifths are the same keys shown in Table 3.7 and Table 3.8. Theoretically, Table 3.7 could continue with keys that have seven sharps, eight sharps, and on infinitely. Moving the other direction, you could have keys with seven flats, eight flats, and on infinitely. These keys with an increasing number of sharps would continue to have equivalent keys with flats. We've shown in Figure

3.22 that the key of F#, with six sharps, is equivalent to the key of G ♭, with six flats. We could have continued in this manner, showing that the key of C#, with seven sharps, is equivalent to the key of D ♭, with five flats. Because the equivalent keys go on infinitely, the circle of fifths is sometimes represented as a spiral. Practically speaking, however, there's little point in going beyond a key with six sharps or flats because such keys are harmonically the same as keys that could be represented with fewer accidentals. We leave it as an exercise for you to demonstrate to yourself the continued key equivalences.

Flash Tutorial:
Circle Of Fifths

3.1.5.7 Key Transposition

All major keys have a similar sound in that the distance between neighboring notes on the scale follows the same pattern. One key is simply higher or lower than another. The same can be said for the similarity of minor keys. Thus, a musical piece can be **transposed** from one major (or minor) key to another without changing its composition. A singer might prefer one key over another because it is in the range of his or her voice.

Figure 3.23 shows the first seven notes of "Twinkle, Twinkle Little Star," right-hand part only, transposed from the key of C major to the keys of E major and C minor. If you play these on a keyboard, you'll hear that C major and E major sound the same, except that the second is higher than the first. The one in C minor sounds different because the fifth and sixth notes (A) are lowered by a semitone in the minor key, giving a more somber sound to the sequence of notes. It doesn't "twinkle" as much.

Flash Tutorial:
Transposing Keys

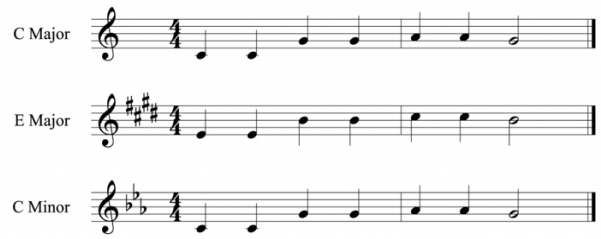


Figure 3.23 Transposition from key of C major to E major and C minor

3.1.6 Musical Composition

3.1.6.1 Historical Context

For as far back in history as we can see, we find evidence that human beings have made music. We know this from ancient drawings of lutes and harps, Biblical references to David playing the lyre for King Saul, and the Greek philosopher Pythagoras's calculation of harmonic intervals. The history of music and musical styles is a fascinating evolution of classical forms, prescriptive styles, breaks from tradition, and individual creativity. An important thread

woven through this evolution is the mathematical basis of music and its relationship with beauty.

In the 6th century BC, the Greek philosopher Pythagoras recognized the mathematical basis of harmony, noting how the sound of a plucked string changes in proportion to the length of the string. However, harmony was not the most important element of music composition as it developed through the Middle Ages. The chanting of medieval monks evolved in a variety of styles and modalities. The early chants, called **plainchant**, were essentially syllabically-accented speech. Later styles involved two to eight monks chanting multiple melodic lines, traditionally with no instrumental accompaniment and in different musical modes. Madrigals of the Middle Ages were composed of single solo lines with simple instrumental accompaniment. In the Baroque period, multiple strands became interwoven in a style called **polyphony**, which reached its zenith in the fugues, canons, and contrapuntal compositions of Johann Sebastian Bach. Although we have come to think of harmony as an essential element of music, its music-theoretic development and its central position in music composition didn't begin in earnest until the 18th century, spurred by Jean-Philippe Rameau's *Treatise on Harmony*.

In this chapter, we emphasize the importance of harmony in musical composition because it is an essential feature of modern Western music. However, we acknowledge that perceptions of what is musical vary from era to era and culture to culture, and we can give only a small part of the picture here. We encourage the reader to explore other references on the subject of music history and theory for a more complete picture.

3.1.6.2 Intervals

An understanding of melody and harmony in Western music composition begins with intervals. An **interval** is the distance between two notes, measured by the number of lettered notes it contains counting both the starting and ending notes. If the notes are played at the same time, the interval is a **harmonic interval**. If the notes are played one after the other, it is a **melodic interval**. If there are at least three notes played simultaneously, they constitute a **chord**. When notes are played at the same time, they are stacked up vertically on the staff. Figure 3.24 shows a harmonic interval, melodic interval, and chord.

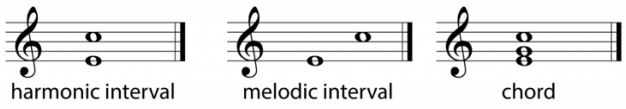


Figure 3.24 Intervals and chords

Intervals are named by the number of lettered notes from the lowest to highest, inclusive. The name of the interval can be expressed as an ordinal number. For example, if the interval begins at F and ends in D, there are six lettered notes in the interval, F, G, A, B, C, and D. Thus, this interval is called a sixth. If the interval begins with C and ends in E, it is called a third, as shown in Figure 3.25 Note that in this regard the presences of sharps or flats isn't significant. The interval between F and D is a sixth, as is the interval between F and D#.

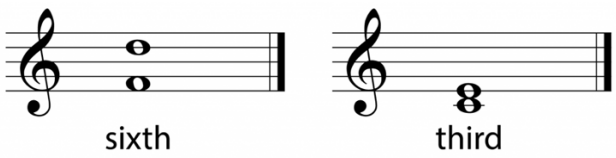


Figure 3.25 Intervals of different sizes: a sixth and a third

Because there are eight notes in an octave, with the last letter repeated, there are eight intervals in an octave. These are shown in Figure 3.26, in the key of C. Each interval is constructed in turn by starting on the key note and moving up by zero lettered notes, one lettered note, two lettered notes, and on up to eight lettered notes. Moving up one lettered note means moving up to the next line or space on the staff.

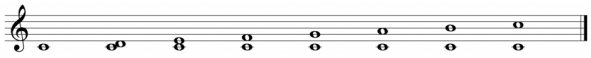


Figure 3.26 Intervals in an octave, key of C major

The same intervals can be created in any key. For example, to create them in the key of D, all we have to do is move the bottom note of each interval to the key note D and move up zero notes from D for perfect unison, one note for a major second, two notes for a major third, and so forth, as shown in Figure 3.27. Note, however, that F and C are implicitly sharp in this key.

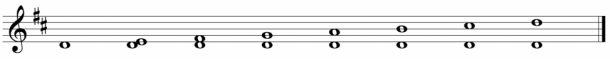


Figure 3.27 Intervals in key of D major

A major interval can be converted to a minor interval by lowering the higher note one semitone. For example, in the key of C, the minor intervals are C to D ♭ (minor second), C to E ♭ (minor third), C to A ♭ (minor sixth), and C to B ♭ (minor seventh). It isn't possible to turn a fourth or fifth into a minor interval. This is because if you lower the higher note by one semitone, you've change the interval to a smaller one because you've lost one of the lettered notes in the interval. For example, in the key of C if you shorten a perfect fourth by one semitone, the top note is an E rather than F, so the interval is no longer a fourth. There's no black key to use between E and F. A third, on the other hand, can be shortened so that instead of being from C to E, it's from C to E ♭ . There are still three lettered notes in the interval – C, D, and E – but now we have a minor third instead of a major third.

The intervals that cannot be made minor are called **perfect**. These are the perfect unison, perfect fourth, perfect fifth, and perfect eighth.

Two other types of intervals exist: diminished and augmented. A **diminished interval** is one semitone smaller than a perfect interval or two semitones smaller than a major interval. An **augmented interval** is one semitone larger than a perfect or major interval. All the intervals for the key of C are shown in Figure 3.28. The intervals for other keys could be illustrated similarly.

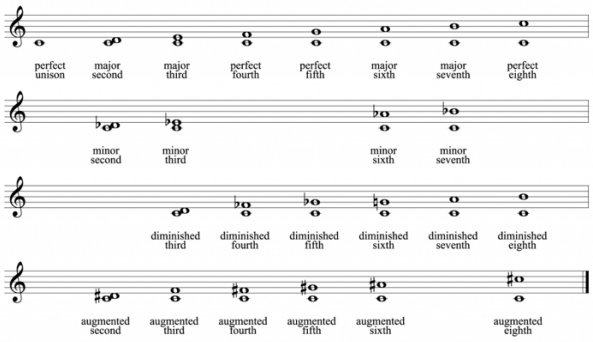


Figure 3.28 All intervals

You can create a **compound interval** by taking any of the interval types defined above and moving the upper note up by one or more octaves. If you're trying to identify intervals in a score, you can count the number of lettered notes from the beginning to the end of the interval and "mod" that number by 7. The result gives you the type of interval. For example, if the remainder is 3, then the interval is a third. The *mod* operation divides by an integer and gives the remainder. This is the same as repeatedly subtracting 7 (the number of notes in an octave, not counting the repeat of the first note an octave higher) until you reach a number that is less than 7. Figure 3.29 shows examples of two compound intervals. The first goes from C4 to G5, a span of 12 notes. 12 mod 7 = 5, so this is a compound perfect fifth. The second interval goes from E4 to G5#, a span of 10 notes. 10 mod 7 = 3, so this is a compound third. However, because the G is raised to G#, this is, more precisely, a compound *augmented* third.

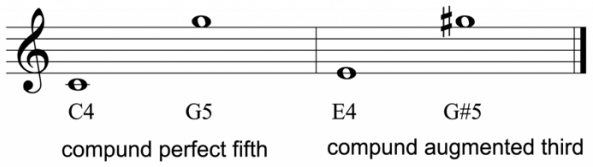


Figure 3.29 Compound intervals

Intervals can be characterized as either consonant or dissonant. Giving an oversimplified definition, we can say that sounds that are pleasing to the ear are called **consonant**, and those that are not are called **dissonant**. Of course, these qualities are subjective. Throughout the history of music, the terms consonant and dissonant have undergone much discussion. Some music theorists would define consonance as a state when things are in accord with each other, and dissonance as a state of instability or tension that calls for resolution. In Section 3.3.2, we'll look more closely at a physical and mathematical explanation for the subjective perception of consonance as it relates to harmony.

Intervals help us to characterize the patterns of notes in a musical composition. The musical conventions from different cultures vary not only in their basic scales but also in the intervals that are agreed upon as pleasing to the ear. In Western music the perfect intervals, major and minor thirds, and major and minor sixths are generally considered consonant, while the seconds, sevenths, augmented, and diminished intervals are considered dissonant. The consonant intervals come to be used frequently in musical compositions of a culture, and the listener's enjoyment is enhanced by a sense of recognition. This is not to say that dissonant intervals are simply ugly and never used. It is the combination of intervals that makes a

Max Demo:
Ear Training for Music

composition. The combination of consonant and dissonant intervals gives the listener an alternating sense of tension and resolution, expectation and completion in a musical composition.

3.1.6.3 Chords

A chord is three or more notes played at the same time. We will look at the most basic chords here – **triads**, which consist of three notes stacked vertically in thirds, creating **tertian harmony** .

Triads can be major, minor, diminished, or augmented. The lowest note of a triad is its **root**. In a **major triad**, the second note is a major third above the root, and the third note is a perfect fifth above the root. In the **root position** of a triad in a given key, the tonic note of the key is the root of the triad. (Refer to Table 3.2 for the names of the notes in a diatonic scale.) The major triads for the keys of C, F, and A are shown in root position in Figure 3.30.

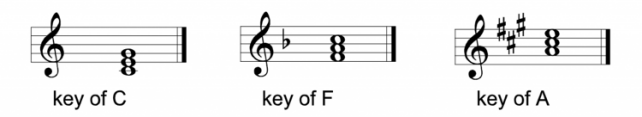


Figure 3.30 Major triads in root position

In a **minor triad**, the second note is a diatonic minor third above the root, and the third note is a diatonic perfect fifth above the root. The minor triads for the keys of C, F, and A are shown in root position in Figure 3.31.

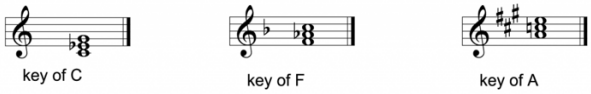


Figure 3.31 Minor triads in root position

Chords can be inverted by rotating the notes up or down – in other words, making the 3rd or the 5th the bottom or lowest note. In the first inversion, the lowest note is the 3rd, and in the second inversion, the lowest note is the 5th. In the **second inversion**, the mediant of the key becomes the root and the tonic is moved up an octave. In the **third inversion**, the dominant of the key becomes the root, and both the mediant and tonic are moved up an octave. The first and inversions are shown in Figure 3.32.

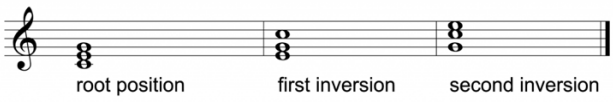


Figure 3.32 Inversions for major triad in key of C

A **diminished triad** has a root note followed by a minor third and a diminished fifth. An **augmented** triad has a root note followed by a major third and an augmented fifth. The diminished and augmented triads for the key of C are shown in Figure 3.33.

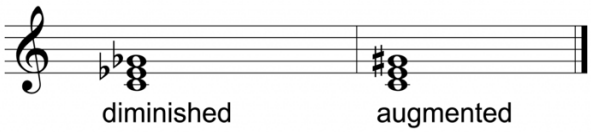


Figure 3.33 Diminished and augmented triads in key of C

We've looked at only triads so far. However, chords with four notes are also used, including the major, minor, and dominant sevenths. The **major seventh** chord consists of the root, major third, perfect fifth, and major seventh. The **minor seventh** consists of the root, minor third, perfect fifth, and minor seventh. The **dominant seventh** consists of the root, major third, perfect fifth, and minor seventh. Dominant chords have three inversions. The dominant seventh for the key of C is shown with its inversions in Figure 3.34. (Note that B is flat in all the inversions.)

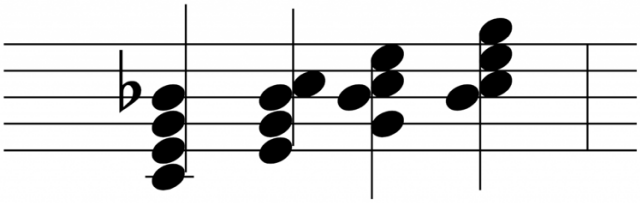


Figure 3.34 Dominant seventh chords for the key of C

Thus we have seven basic chords that are used in musical compositions. These are summarized in Table 3.9.

name	chord	notes
major triad		root, major third, perfect fifth
minor triad		root, minor third, perfect fifth
diminished triad		root, minor third, diminished fifth
augmented triad		root, major third, augmented fifth
major seventh		root, major third, perfect fifth, major seventh
minor seventh		root, minor third, perfect fifth, minor seventh
dominant seventh		root, major third, perfect fifth, minor seventh

Table 3.9 Types of chords

All major keys give rise to a sequence of triads following the same pattern. To see this, consider the sequence you get in the key of C major by playing a triad starting on each of the piano keys in the scale. These triads are shown in Figure 3.35. Each is named with the name of its root note. (Note that some sources use lower case Roman numerals for minor and diminished chords.)

Flash Tutorial:
Chords

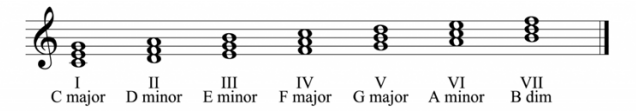


Figure 3.35 Triads in C major

Each of the chords in this sequence can be characterized based on its root note. Triad I is easy to see. The first note is C and it's a major triad, so it's C major. Triad II starts on D, so consider what kind of triad this would be in the key of D. It has the notes D, F, and A. However, in the key of D, F should be sharp. D is not sharp in the triad II shown, so this is a minor triad from the key of D. Triad III has the notes E, G, and B. In the key of E, G is sharp, but in the triad III shown, it is not. Thus, the third triad in the key of C is E minor. Let's skip to triad VII, which has the notes B, D, and F. In the key of B, the notes D and F are sharp, but they're not in triad VII. This makes triad VII a B diminished. By this analysis, we can determine that the sequence of chords in C major is C major, D minor, E minor, F major, G major, A minor, and B diminished.

Flash Tutorial:
Identifying Chords in a Key

All major keys have this same sequence of chord types – major, minor, minor, major, major, minor, and diminished. For example, the key of D has the chords D major, E minor, F# minor, G major, A major, B minor and C# diminished. If we do a similar analysis for the minor keys, we find that they follow the pattern minor, diminished, augmented, minor, major, major, and diminished, as summarized in Table 3.10.

Chord number	Chord type if the triad sequence comes from a major key	Chord type if the triad sequence comes from a minor key
I	major	minor
II	minor	diminished
III	minor	major
IV	major	minor
V	major	major
VI	minor	major
VII	diminished	diminished

Table 3.10 Types of triad chords in major and minor keys

3.1.6.4 Chord Progressions

So what is the point of identifying and giving names to intervals and chords? The point is that this gives us a way to analyze music, communicate about it, and compose it in a way that yields a logical order and an aesthetically pleasing form within a style. When chords follow one another in certain progressions, they provoke feelings of tension and release, expectation and satisfaction, conflict and resolution. This emotional rhythm is basic to music composition.

Let's look again at the chords playable in the key of C, as shown in Figure 3.35. Chord progressions can be represented by a sequence of Roman numerals that correspond to scale degrees. For example, the right-hand part of "Twinkle, Twinkle Little Star" in the key of C major, shown in Figure 3.23, could be accompanied by the chords I I IV I IV I V I in the left hand, as shown in Figure 3.36. This chord progression is in fact a very common pattern.

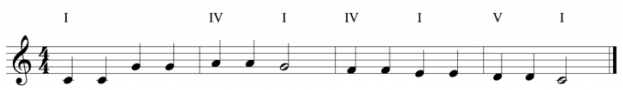


Figure 3.36 "Twinkle, Twinkle Little Star" with chords as Roman numerals

You can play this yourself with reference to the triads from the key of C major, shown in Figure 3.35. However, to make the chords sound more interesting, you can invert them so that the lowest note leads more smoothly to the next chord's lowest note. A beginning piano student may invert chords to make them easier to play. For example, chord I can be played as C, E, G; chord IV as C, F, A (2^{nd} inversion); and chord V as B, D, G (1^{st} inversion), as shown in Figure 3.37.

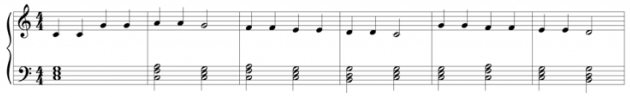


Figure 3.37 "Twinkle, Twinkle Little Star" with chords in bass clef

A simple way of looking at chord progressions is based on **tonality** – a system for defining the relationships of chords based on the tonic note for a given key. Recall that the tonic note is the note that begins the scale for a key. In fact, the key's name is derived from the tonic note. In the key of C, the tonic note is C. Triad I is the tonic triad because it begins on the tonic note. You can think of this as the home chord, the chord where a song begins and to which it wants to return. Triad V is called the dominant chord. It has this name because chord progressions in a sense pull toward the dominant chord, but then tend to return to the home chord, chord I. Returning to chord I gives a sense of completion in a chord progression, called a **cadence**. The progression from I to V and back to I again is called a **perfect cadence** because it is the clearest example of tension and release in a chord progression. It is the most commonly used chord progression in modern popular music, and probably in Western music as a whole.

Another frequently used progression moves from I to IV and back to I. IV is the subdominant chord, named because it serves as a counterbalance to the dominant. The I IV I progression is called the **plagal cadence**. Churchgoers may

recognize it as the sound of the closing "amen" to a hymn. Together, the tonic, subdominant, and dominant chords and their cadences serve as foundational chord progressions. You can see this in "Twinkle, Twinkle Little Star," a tune for which Wolfgang Amadeus Mozart wrote twelve variations.

The perfect and plagal cadences are commonly used chord progressions, but they're only the tip of the iceberg of possibilities. "Over the Rainbow" from *The Wizard of Oz*, with music written by Harold Arlen, is an example of how a small variation from the chord progression of "Twinkle, Twinkle Little Star" can yield beautiful results. If you speed up the tempo of "Over the Rainbow", you can play it with the same chords as are used for "Twinkle, Twinkle Little Star," as shown in Figure 3.38. The only one that sounds a bit "off" is the first IV.

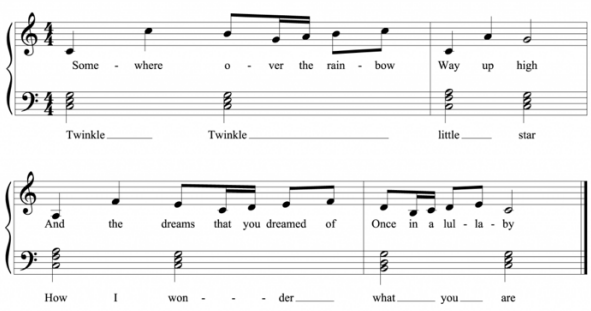


Figure 3.38 "Over the Rainbow" played in C with chords of "Twinkle, Twinkle Little Star"

It's remarkable how the color of a melody can shift when the accompanying chords are changed only slightly. Figure 3.39 shows "Over the Rainbow" played in the key of E ♭ with the chord sequence I III IV I IV I III V I For the key of E ♭, these chords are E ♭, G minor, A ♭, E ♭, A ♭, E ♭, G minor, B ♭, E ♭. The use of G minor adds a melancholy tone which fits the song's theme. We could also change the fourth chord, currently an E ♭ major, to a G minor and add to this effect.

Video Tutorial:
Over the Rainbow

Video Tutorial:
Over the Rainbow Eflat

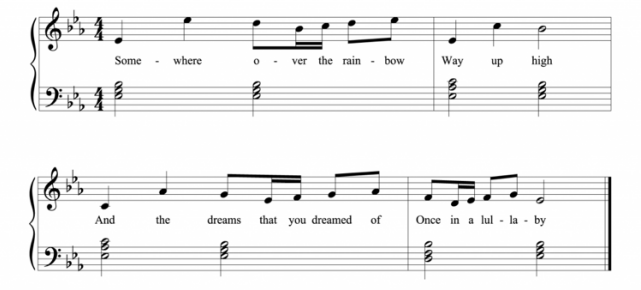


Figure 3.39 "Over the Rainbow" in the key of E ♭

3.1.7 Dynamics and Articulation

Dynamics symbols tell how loud or quietly music should be played. These terms were brought into usage by Italian composers, and we still use the Italian terms and symbols. The terms are summarized in Table 3.11 along with the signs used to denote them on the staff. Additional *p*s and *f*s can be added to emphasize the loudness or quietness, but four *p*s or *f*s is usually the limit. The symbols are displayed below the staff in the area to which they are applied.

Term	Sign	Meaning
piano	***p***	quiet
mezzo piano	***mp***	moderately quiet
pianissimo	***pp***	very quiet
forte	***f***	loud
mezzo forte	***mf***	moderately loud
fortissimo	***ff***	very loud
subito forte	***sf***	suddenly loud
subito forte piano	***sfp***	suddenly loud and soft
sforzando	***sfz***	forceful sudden accent

Table 3.11 Dynamics terms and symbols

Gradual changes in dynamics are indicated by **crescendo** and **decrescendo** symbols (Figure 3.40). The **decrescendo** can also be called a **diminuendo**. The crescendo indicates that the music should gradually grow louder. The decrescendo is the opposite. These symbols can be made longer or shorter depending on the time over which the change should occur. They are usually placed below the staff, but occasionally above.

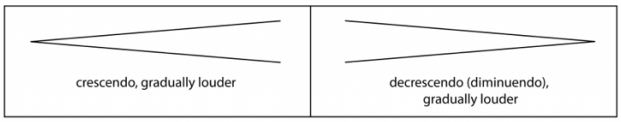


Figure 3.40 Symbols for gradual changes in dynamics

Articulation marks (Table 3.12) indicate the manner in which a note is to be performed. A little dot above or below a note (depending on the direction of the note's stem) indicates that it is to be played **staccato** – that is, played as a short note not held down and thus not linked directly in sound to the next note. A "greater than" sign above or below a note, called an **accent**, indicates that the note should be played louder, with special emphasis. A straight line above or below a note, called **tenuto**, emphasizes that a note is to be held for its full value. A dot with a semicircle above it, called a **fermata**, means that a note is to be held longer than its normal value. The articulation marks and dynamics symbols help a composer to guide the style and technique applied to the performance of a composition.

Term	Symbol	Meaning
staccato		Play the note with a quick key strike, not holding the note to blend with the next
accent		Accent the note
tenuto		Hold the note for its full length
fermata		Hold the note longer than its full length

Table 3.12 Articulation marks

Section 3.2 - Applications

3.2.1 Piano Roll and Event List Views

Learning to read music on a staff is important for anyone working with music. However, for non-musicians working on computer-based music, there are other, perhaps more intuitive ways to picture musical notes -- e.g., by means of MIDI.

MIDI (**Musical Instrument Digital Interface**) is a protocol designed for encoding and exchange of musical information. (MIDI is introduced in Chapter 1, and we'll talk more about MIDI in Chapters 5 and 6, but what follows will give you a preview graphical user interfaces between MIDI and the user.) MIDI uses computer-based **synthesizers** and **samplers**

to turn the encoding into digital audio that can be played through the computer's sound card. A **MIDI sequencer** is the software interface between the user and the MIDI samplers and synthesizers. MIDI sequencers enable the computer to receive, store, modify, and play MIDI data. Sequencers provide two views of music that you may find helpful if you can't read music on a musical staff – the **piano roll view** and the **event list view**.

A **piano roll view** is essentially a graph with the vertical axis representing the notes on a piano and the horizontal axis showing time. In Figure 3.41 you can see a graphical representation of a piano keyboard flipped vertically and placed on the left side of the window. On the horizontal row at the top of the window you can see a number representing each new measure. This graph forms a grid where notes can be placed at the intersection of the note and beat where the note should occur. Notes can also be adjusted to any duration. In this case, you can see each note starting on a quarter beat. Some notes are longer than others. The quarter notes are shown using bars that start on a quarter beat but don't extend beyond the neighboring quarter beat. The half notes start on a quarter beat and extend beyond the neighboring quarter beat. In most cases you can grab note events using your mouse pointer and move them to different notes on the vertical scale as well as move them to a different beat on the grid. You can even extend or shorten the notes after they've been entered. You can also usually draw new notes directly into the piano roll.

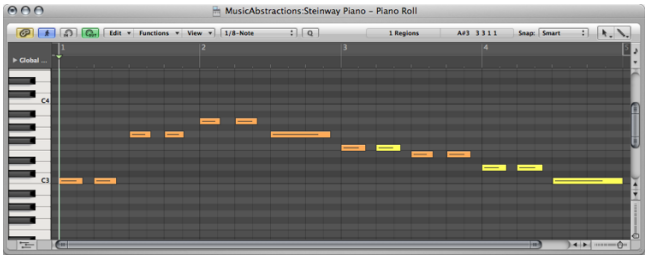


Figure 3.41 Piano roll view of melody of "Twinkle, Twinkle Little Star"

Another way you might see the musical data is in an **event list view**, as shown in Figure 3.42. This is a list showing information about every musical event that occurs for a given song. Each event usually includes the position in time that the event occurs, the type of event, and any pertinent details about the event such as note number, duration, etc. The event position value is divided into four columns. These are Bars:Beats:Divisions:Ticks. You should already be familiar with bars and beats. Divisions are some division of a beat. This is usually an eighth or asixteenth and can be user-defined. Ticks are a miniscule value. You can think of these like frames or milliseconds. The manual for your specific software will give you its specific for time value of ticks.

You can edit events in an event list by changing the properties, and you can add and delete events in this view. The event list is useful for fixing problems that you can't see in the other views. Sometimes data gets captured from your MIDI keyboard that shouldn't be part of your sequence. In the event list you can select the specific events that shouldn't be there (like a MIDI volume change or program change).

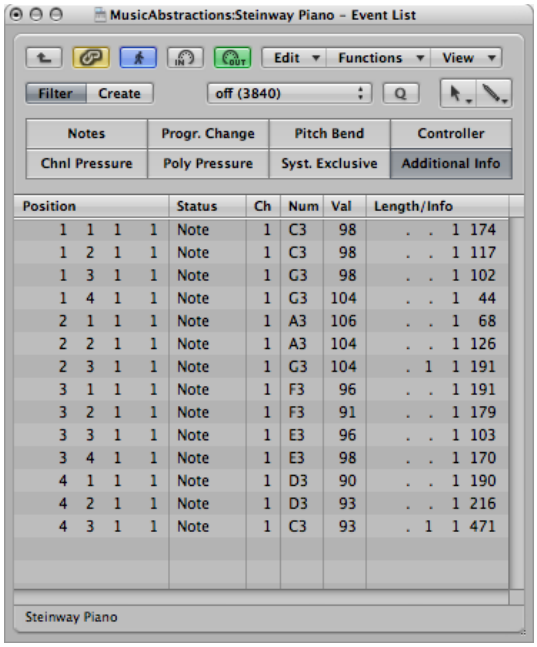


Figure 3.42 Event list view of melody of "Twinkle, Twinkle Little Star"

3.2.2 Tablature

Tablature (tab) is a common way of notating music when played on plucked string instruments. Basic guitar tablature requires that the player be somewhat familiar with the song already. As shown in Figure 3.43, guitar tabs use six lines like the regular music staff, but in this case, each line represents one of the six guitar strings. The top line represents the high E string and the bottom line represents the low E string. Numbers are then drawn on each line representing the fret number to use for that note on that string. To play the song, every time you see a number, you put your finger on that string at the fret number indicated and pluck that string. Simple tablature doesn't give any indication of how long to play each note, so you need to be familiar with the song already. In some cases, the spacing of the numbers can give some indication of rhythm and duration.

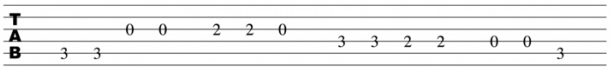


Figure 3.43 Guitar tablature for melody of "Twinkle, Twinkle Little Star"

3.2.3 Chord Progression

The most abstract method of encoding musical data is probably the chord progression. Consider the following chord progression for "Twinkle, Twinkle Little Star":

C F C

```
Twinkle, twinkle little star

 F      C       G       C
How I wonder what you are.
 C   F       C       G
Up above the world so high,
 C      F       C    G
Like a diamond in the sky,
 C               F   C
Twinkle, twinkle little star,
 F      C      G       C
How I wonder what you are.
```

A trained musician who is already somewhat familiar with the tune should be able to extract the entire song as shown in Figure 3.37 from this chord progression. You may have witnessed an example of this if you've ever seen a live band that seemed able to play anything requested on the spur of the moment. Most good "gigging bands" can do this. Certainly, their ability to do this in part comes from a good familiarity with popular music, but that doesn't mean they've memorized all those songs ahead of time. Musicians call this approach "faking it" and at most bookstores you can purchase "Fake Books" that contain hundreds of chord progressions for songs. With the musical data so highly compressed, it is not unheard of for a fake book to have well over 1000 songs in a size that can fit quite comfortably in a small backpack.

In Section 3.1.6.2 we learned about intervals and in Section 3.1.6.3 we learned about chords. Chord progressions are a repeating sequence of chords at musical intervals. Music is built with chord progressions. If you know what chord is being used at any given time in the song, you should also know which notes you could play for that part of the song. In the case of the chord progression for "Twinkle, Twinkle Little Star," we begin with the tonic C major chord. Then we move to the perfect fourth, an F major chord. Occasionally, we use the perfect fifth, a G major chord. This is called a **I-IV-V** chord progression. Many popular songs are built from this chord progression. In the simplest form, you could play these chords on the keyboard as you sing the melody and you should have something that sounds respectable. From there you could improvise bass lines, solos, and arrangements by playing notes that fit with the key signature and chord currently in use. In addition to fake books, you can train your ears to recognize chord progressions and then fake the songs by ear. See the learning supplements for this section to start training your ears and try experimenting with improvising along with a chord progression.

Max Demo:
Music Improvisation Demo

3.2.4 Guitar Chord Grid

A guitar chord grid representation is of a chord sequence shown in Figure 3.44. The chord grid corresponds to the guitar fret board. The vertical lines represent the six strings and the horizontal lines represent the frets. A black dot represents a place on the fret board where you should put one of your fingers. When all your fingers are in the indicated locations, you can strum the strings to play the chord. Keep in mind that the chord grid shows you one way to play the chord, but there are always other fingering combinations that result in the same chord. If the grid you're looking at looks too hard to play, you might be able to find a grid showing an alternate fingering that is easier.

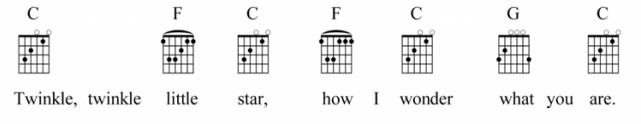


Figure 3.44 - Guitar chord grids for "Twinkle, Twinkle Little Star"

Section 3.3 - Science, Mathematics, and Algorithms

3.3.1 The Mathematics of Music

3.3.2 Equal Tempered vs. Just Tempered Intervals

In Section 3.1.4 we described the diatonic scales that are commonly used in Western music. These scales are built by making the frequency of each successive note $\sqrt[12]{2}$ times the frequency of the previous one. This is just one example of a **temperament**, a system for selecting the intervals between tones used in musical composition. In particular, it is called **equal temperament** or **equal tempered intonation**, and it produces **equal tempered** scales and intervals. While this is the intonation our ears have gotten accustomed to in Western music, it isn't the only one, nor even the earliest. In this section we'll look at an alternative method for constructing scales that dates back to Pythagoras in the sixth century B. C. – **just tempered intonation** – a tuning in which frequency intervals are chosen based upon their harmonic relationships.

First let's look more closely and the equal tempered scales as a point of comparison. The diatonic scales described in Section 3.1.4 are called equal tempered because the factor by which the frequencies get larger remains equal across the scale. Table 3.13 shows the frequencies of the notes from C4 to C5, assuming that A4 is 440 Hz. Each successive frequency is $\sqrt[12]{2}$ times the frequency of the previous one.

Note	Frequency

C4	261.63 Hz
C4# (D4 ♭)	277.18Hz
D4	293.66 Hz
D4# (E4 ♭)	311.13 Hz
E4	329.63 Hz
F4	349.23 Hz
F4# (G4 ♭)	369.99 Hz
G4	392.00 Hz
G4# (A4 ♭)	415.30 Hz
A4	440.00 Hz
A4# (B4 ♭)	466.16 Hz
B4	493.88 Hz
C5	523.25 Hz

Table 3.13 Frequencies of notes from C4 to C5

It's interesting to note how the ratio of two frequencies in an interval affects our perception of the interval's consonance or dissonance. Consider the ratios of the frequencies for each interval type, as shown in Table 3.14. Some of these ratios reduce very closely to fractions with small integers in the numerator and denominator. For example, the frequencies of G and C, a perfect fifth, reduce to approximately 3:2; and the frequencies of E and C, a major third, reduce to approximately 5:4. Pairs of frequencies that reduce to fractions such as these are the ones that sound more consonant, as indicated in the last column of the table.

Interval	Notes in Interval	Ratio of Frequencies		Common Perception of Interval
perfect unison	C	261.63/261.63	1	consonant
minor second	C C#	277.18/261.63	≈ 1.059/1	dissonant
major second	C D	293.66/261.63	≈ 1.122/1	dissonant
			≈ 1.189/1 ≈	

minor third	C D E ♭	311.13/261.63	6/5	consonant
major third	C D E	329.63/261.63	≈ 1.260/1 ≈ 5/4	consonant
perfect fourth	C D E F	349.23/261.63	≈ 1.335/1 ≈ 4/3	strongly consonant
augmented fourth	C D E F#	369.99/261.63	≈ 1.414/1	dissonant
perfect fifth	C D E F G	392.00/261.63	≈ 1.498/1 ≈ 3/2	strongly consonant
minor sixth	C D E F G A ♭	415.30/261.63	≈ 1.587/1 ≈ 8/5	consonant
major sixth	C D E F G A	440.00/261.63	≈ 1.681/1 ≈ 5/3	consonant
minor seventh	C D E F G A B ♭	466.16/261.63	≈ 1.781/1	dissonant
major seventh	C D E F G A B	493.88/261.63	≈ 1.887/1	dissonant
perfect octave	C C	523.26/261.63	1	consonant

Table 3.14 Ratio of beginning and ending frequencies in intervals

There's a physical and mathematical explanation for this consonance. The fact that the frequencies of G and C have approximately a 3/2 ratio is visible in a graph of their sine waves. Figure 3.45 shows that three cycles of G fit almost exactly into two cycles of C. The sound waves fit together in an orderly pattern, and the human ear notices this agreement. But notice that we said the waves fit together *almost* exactly. The ratio of 392/261.63 is actually closer to 1.4983, not exactly 3/2, which is 1.5. Wouldn't the intervals be even more harmonious if they were built upon the frequency relationships of natural harmonics?

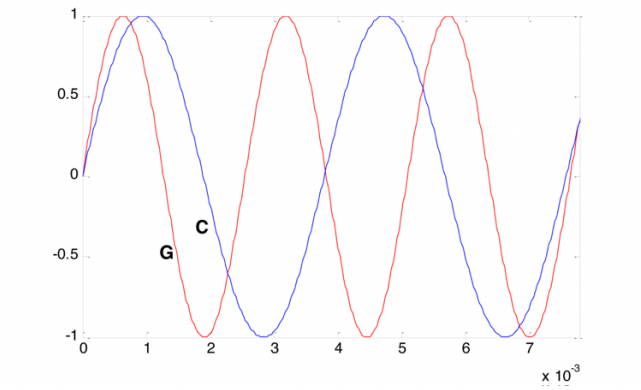


Figure 3.45 How cycles match for pairs of frequencies

To consider how and why this might make sense, recall the definition of harmonic frequencies from Chapter 2. For a fundamental frequency f, the harmonic frequencies are $1f, 2f, 3f, 4f$, and so forth. These are frequencies whose cycles *do* fit together exactly. As depicted in Figure 3.46, the ratio of the third to the second harmonic is $\frac{3f}{2f} = \frac{3}{2}$. This corresponds very closely, but not precisely, to what we have called a perfect fifth in equal tempered intonation – for example, the relationship between G and C in the key of C, or between D and G in the key of G. Similarly, the ratio of the fifth harmonic to the fourth is $\frac{5f}{4f} = \frac{5}{4}$, which corresponds to the interval we have referred to as a major third.

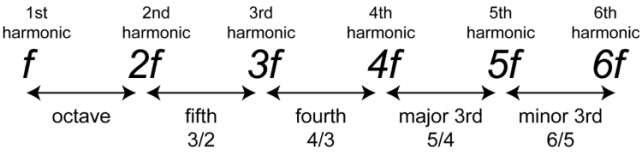


Figure 3.46 Ratio of frequencies in harmonic intervals

Just tempered intonations use frequencies that have these harmonic relationships. The Pythagoras diatonic scale is built entirely from fifths. The just tempered scale shown in Table 3.15 is a modern variant of Pythagoras's scale. By comparing columns three and four, you can see how far the equal tempered tones are from the harmonic intervals. An interesting exercise would be to play a note in equal tempered frequency and then in just tempered frequency and see if you can notice the difference.

Interval	Notes in Interval	Ratio of Frequencies in Equal Temperament	Ratio of Frequencies in Just Temperament
perfect unison	C	261.63/261.63 = 1.000	1/1 = 1.000
minor second	C C#	277.18/261.63 ≈ 1.059	16/15 ≈ 1.067
major second	C D	293.66/261.63 ≈ 1.122	9/8 = 1.125
minor third	C D E_	311.13/261.63 ≈ 1.189	6/5 = 1.200

major third	C D E	329.63/261.63 ≈ 1.260	5/4 ≈ 1.250
perfect fourth	C D E F	349.23/261.63 ≈ 1.335	4/3 ≈ 1.333
augmented fourth	C D E F#	369.99/261.63 ≈ 1.414	7/5 = 1.400
perfect fifth	C D E F G	392.00/261.63 ≈ 1.498	3/2 = 1.500
minor sixth	C D E F G A ♭	415.30/261.63 ≈ 1.587	8/5 = 1.600
major sixth	C D E F G A	440.00/261.63 ≈ 1.682	5/3 ≈ 1.667
minor seventh	C D E F G A B ♭	466.16/261.63 ≈ 1.782	7/4 = 1.750
major seventh	C D E F G A B	493.88/261.63 ≈ 1.888	15/8 = 1.875
perfect octave	C C	523.26/261.63 = 2.000	2/1 = 2.000

Table 3.15 Equal tempered vs. just tempered scales

3.3.3 Experimenting with Music in Max

Max (and its open source counterpart, PD) is a powerful environment for experimenting with sound and music. We use one of our own learning supplements, a Max patcher for generating scales (3.1.4), as an example of the kinds of programs you can write.

When you open this patcher, it is in presentation mode (Figure 3.47), as indicated by the drop-down menu in the bottom right corner. In presentation mode, the patcher is laid out in a user-friendly format, showing only the parts of the interface that you need in order to run the tutorial, which in this case allows you to generate major and minor scales in any key you choose. You can change to programming mode (Figure 3.48) by choosing this option from the drop-down menu.

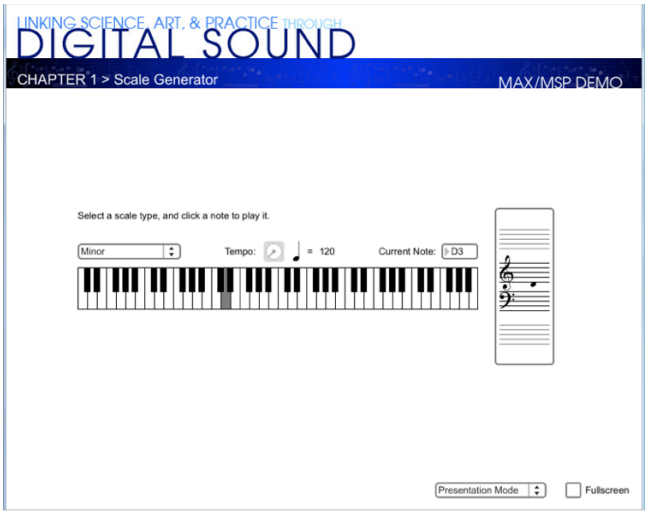


Figure 3.47 Max patcher in presentation mode

Programming mode reveals more of the Max objects used in the program. You can now see that the scale-type menu is connected to a list object. This list changes in accordance with the selection, each time giving the number of semitones between consecutive notes on the chosen type of scale. You can also see that the program is implemented with a number of send, receive, and patcher objects. Send and receive objects pass data between them, as the names imply. Objects named *s* followed by an argument, as in *s notepitch*, are the *send* objects. Objects named *r* followed by an argument, as in *r notepitch*, are the receive objects. Patcher objects are "subpatchers" within the main program, their details hidden so that the main patcher is more readable. You can open a patcher object in programming mode by double-clicking on it. The *metronome*, *handlescale*, and *handlenote* patcher objects are open in Figure 3.49, Figure 3.50, and Figure 3.51.

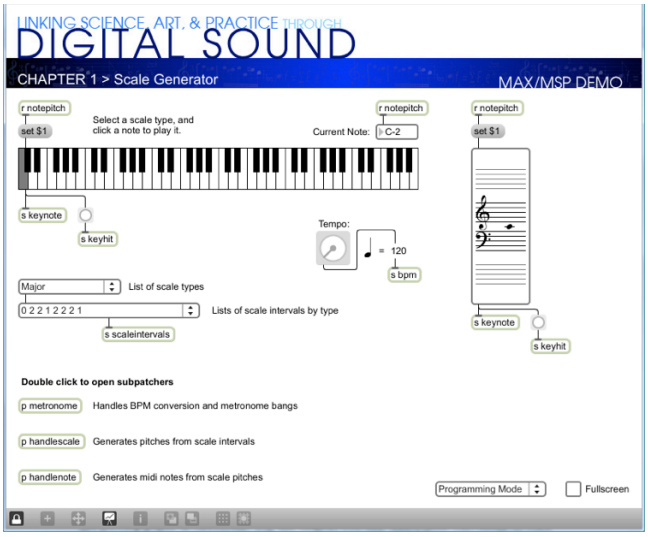


Figure 3.48 Max patcher in programming mode

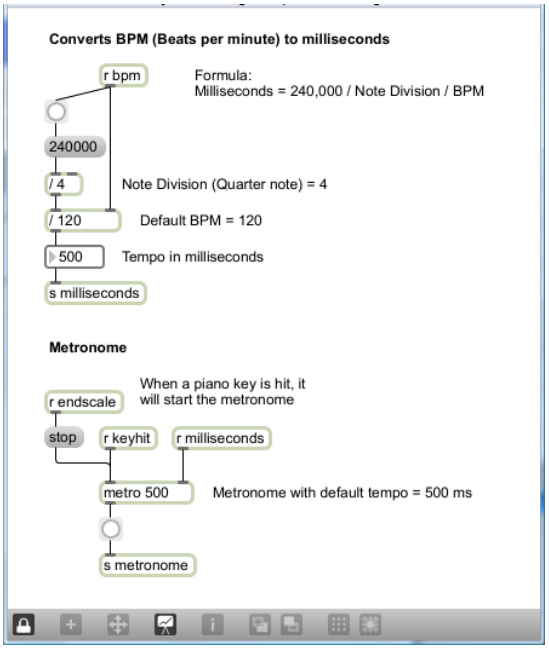


Figure 3.49 Metronome patcher object

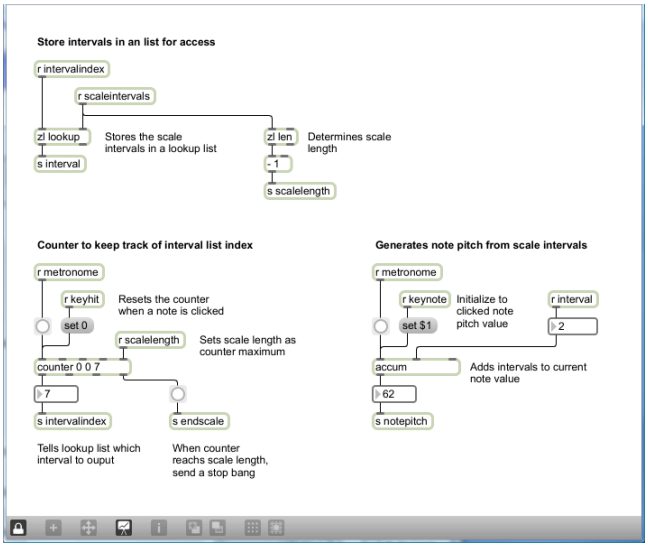


Figure 3.50 Handlescale patcher object

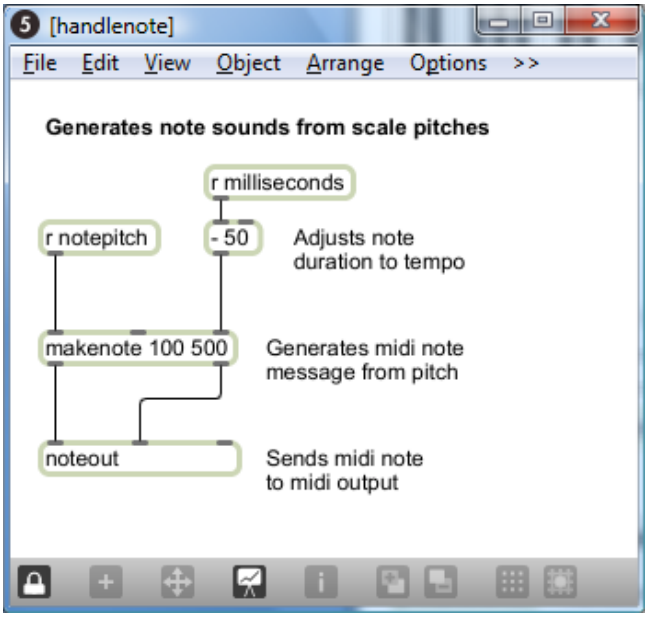


Figure 3.51 Handlenote patcher object

In this example, you still can't see the details of how messages are being received and processed. To do so, you have to unlock the main patcher window by clicking the lock icon in the bottom left corner. This puts you in a mode where you can inspect and even edit objects. The main patcher is unlocked in Figure 3.52. You must have the full Max application, not just the runtime, to go into full programming mode.

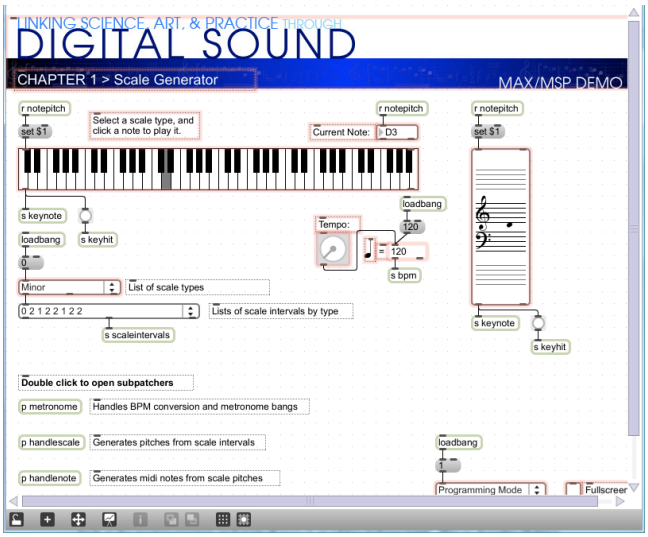


Figure 3.52 Max patcher unlocked, accessible to the programmer

With all the patcher objects exposed and the main patcher unlocked, you can now see how the program is initialized and how the user is able to trigger the sending and receiving of messages by selecting from a menu, turning the dial to set a tempo, and clicking the mouse on the keyboard display. Here's how the program runs:

- The *loadbang* object is triggered when the program begins. It is used to initialize the tempo to 120 bpm and to set the default MIDI output device. Both of these can be changed by the user.
- The user can select a type of scale from a menu. The list of intervals is set accordingly and sent in the

variable *scalesintervals*, to be received by the *handlescale* patcher object.

- The user can select a tempo through the *dial* object. The input value is sent to the *metronome* patcher object to determine the timing of the notes played in the scale.
- The user can click a key on the *keyboard* object. This initiates the playing of a scale by sending a *keyhit* message to both the *metronome* and the *handlescale* patcher.
- When the *metronome* patcher receives the *keyhit* message, it begins sending a tick every *n* milliseconds. The value of *n* is calculated based on the *bpm* value. Each tick of the metronome is what Max calls a *bang* message, which is used to trigger buttons and set actions in motion. In this case, the *bang* sets off a counter in the *handlescale* patcher, which counts out the correct number of notes as it plays a scale. The number of notes to play for a given type of scale is determined by a lookup list in the *zl* object, a multipurpose list processing object.
- The pitch of the note is determined by setting the keynote to the correct one and then accumulating offsets from there. As the correct *notepitch* is set for each successive note, it is sent to the *handlenote* patcher, which makes a MIDI message from this information and outputs the MIDI note. To understand this part of the patcher completely, you need to know a little bit about MIDI messages, which we don't cover in detail until Chapter 6. It suffices for now to understand that middle C is sent as a number message with the value 60. Thus, a C major scale starting on middle C is created from the MIDI notes 60, 62, 64, 65, 67, 69, 71, and 72.

Max has many types of built-in objects with specialized functions. We've given you a quick overview of some of them in this example. However, as a beginning Max programmer, you can't always tell from looking at an object in programming mode just what type of object it is and how it functions. To determine the types of objects and look more closely at the implementation, you need to unlock the patcher, as shown in Figure 3.52. Then if you select an object and right click on it, you get a menu that tells the type of object at the top. In the example in Figure 3.53, the object is a *kslider*. From the context menu, you can choose to see the Help or an Inspector associated with the object. To see the full array of Max built-in objects, go to Max Help from the main Help menu, and from there to Object by Function.

Max Demo:
Transposing Notes

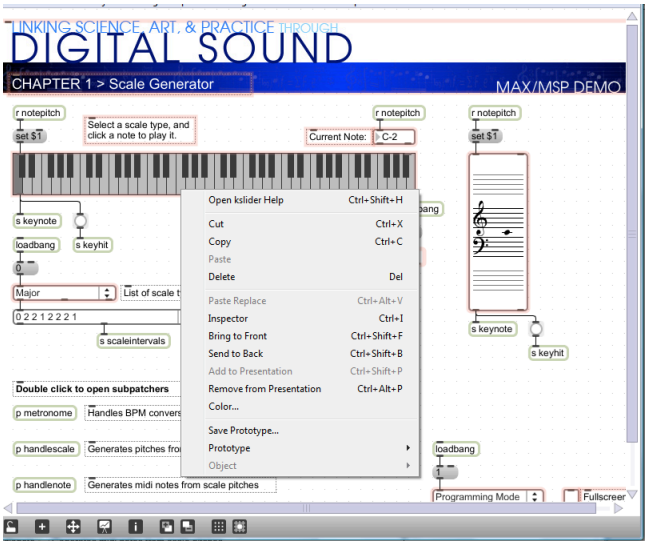


Figure 3.53 Context menu for a kslider object

3.3.4 Experimenting with Music in MATLAB

Chapter 2 introduces some command line arguments for evaluating sine waves, playing sounds, and plotting graphs in MATLAB. For this chapter, you may find it more convenient to write functions in an external program. Files that contain code in the MATLAB language are called M-files, ending in the *.m* suffix. Here's how you proceed when working with M-files:

- Create an M-file using MATLAB's built-in text editor (or any text editor)
- Write a function in the M-file.
- Name the M-file *fun1.m* where *fun1* is the name of function in the file.
- Place the M-file in your current MATLAB directory.
- Call the function from the command line, giving input arguments as appropriate and accepting the output by assigning it to a variable if necessary.

The M-file can contain internal functions that are called from the main function. We refer you to MATLAB's Help for details of syntax and program structure, but offer the program in Algorithm 3.1 to get you started. This program allows the user to create major and minor scales beginning with a start note. The start note is represented as a number of semitones offset from middle C. The function plays eight seconds of sound at a sampling rate of 44,100 samples per second and returns the raw data to the user, where it can be assigned to a variable on the command line if desired.

MATLAB Exercise:
Chords in MATLAB

```
function outarray = MakeScale(startnoteoffset, i
%outarray is an array of sound samples on the sca
%outarray contains the 8 notes of a diatonic musi
%each note is played for one second
%the sampling rate is 44100 samples/s
%startnoteoffset is the number of semitones up or
%which the scale should start.
%If isminor == 0, a major scale is played; otherw
sr = 44100;
s = 8;
outarray = zeros(1,sr*s);
majors=[0 2 4 5 7 9 11 12];
minors=[0 2 3 5 7 8 10 12];
if(isminor == 0)
    scale = majors;
else
    scale = minors;
end
scale = scale/12;
scale = 2.^scale;
%.^ is element-by-element exponentiation
t = [1:sr]/sr;
%the statement above is equivalent to
startnote = 220*(2^((startnoteoffset+3)/12))
scale = startnote * scale;
%Yes, ^ is exponentiation in MATLAB, rather than
for i = 1:8
    outarray(1+(i-1)*sr:sr*i) = sin((2*pi*scale(i
end
 sound(outarray,sr);
```

Algorithm 3.1 Generating scales

The variables *majors* and *minors* hold arrays of integers, which can be created in MATLAB by placing the integers between square brackets, with no commas separating them. This is useful for defining, for each note in a diatonic scale, the number of semitones that the note is away from the key note.

The lineThe variable *scale* is also an array, the same length as *majors* and *minors* (eight elements, because eight notes are played for the diatonic scale). Say that a major scale is to be created. Each element in scale is set to $2^{\frac{majors[i]}{12}}$, where *majors[i]* is the original value of element *i* in the array *majors*. (Note that arrays are numbered beginning at 1 in MATLAB.) This sets *scale* equal to $\left[1, 2^{\frac{2}{12}}, 2^{\frac{4}{12}}, 2^{\frac{5}{12}}, 2^{\frac{7}{12}}, 2^{\frac{9}{12}}, 2^{\frac{11}{12}}\right]$. When the start note is multiplied by each of these numbers, one at a time, the frequencies of the notes in a scale are produced.

MATLAB Exercise:
Equal vs. Just Tempered Scales in MATLAB

```
x = [1:sr]/sr;
```

creates an array of 44,100 points between 0 and 1 at which the sine function is evaluated. (This is essentially equivalent to *x = linspace(0,1, sr)*, which we used in previous examples.)

A3 with a frequency of 220 Hz is used as a reference point from which all other frequencies are built. Thus

```
startnote = 220*(2^((startnoteoffset+3)/12));
```

sets the start note to be middle C plus the user-defined offset.

In the *for* loop that repeats for eight seconds, the statement

```
outarray(1+(i-1)*sr:sr*i) = sin((2*pi*scale(i))*t);
```

writes the sound data into the appropriate section of *outarray*. It generates these samples by evaluating a sine function of the appropriate frequency across the 44,100-element array *x*. This statement is an example of how conveniently MATLAB handles array operations. A single call to a sine function can be used to evaluate the function over an entire array of values. The statement

```
scale = scale/12;
```

works similarly, dividing each element in the array *scale* by 12. The statement

```
scale = 2.^scale;
```

is also an element-by-element array operation, but in this case a dot has to be added to the exponentiation operator since ^ alone is matrix exponentiation, which can have only an integer exponent.

3.3.5 Experimenting with Music in C++

Algorithm 3.2 is a C++ program analogous to the MATLAB program for generating scales. It uses the same functions for writing to the sound device as used the last section of Chapter 2. You can try your hand at a similar program by doing one of the suggested programming exercises. The first exercise has you generate (or recognize) chords of

seven types in different keys. The second exercise has you generate equal tempered and just tempered scales and listen to the result to see if your ears can detect the small differences in frequencies. Both of these programs can be done in either MATLAB or C++.

Programming Exercise:
Equal vs. Just Tempered Scales in C++

```
//This program works under the OSS library
#include <sys/ioctl.h> //for ioctl()
#include <math.h> //sin(), floor()
#include <stdio.h> //perror
#include <fcntl.h> //open, O_WRONLY
#include <linux/soundcard.h> //SOUND_PCM*
#include <iostream>
#include <stdlib.h>
using namespace std;
#define LENGTH 1 //number of seconds
#define RATE 44100 //sampling rate
#define SIZE sizeof(short) //size of sample, in
#define CHANNELS 1 // number of stereo channels
#define PI 3.14159
#define SAMPLE_MAX 32767 // should this end in 8
#define MIDDLE_C 262
#define SEMITONE 1.05946
enum types {BACK = 0, MAJOR, MINOR};
double getFreq(int index){
   return MIDDLE_C * pow(SEMITONE, index-1);
}
int getInput(){
   char c;
   string str;
   int i;
   while ((c = getchar()) != '\n' && c != EOF)
      str += c;
   for (i = 0; i < str.length(); ++i)
      str.at(i) = tolower(str.at(i));
   if (c == EOF || str == "quit")
      exit(0);
   return atoi(str.c_str());
}
int getIndex()
{
   int input;
   cout
      << "Choose one of the following:\n"
      << "\t1) C\n"
      << "\t2) C sharp/D flat\n"
      << "\t3) D\n"
      << "\t4) D sharp/E flat\n"
      << "\t5) E\n"
      << "\t6) F\n"
      << "\t7) F sharp/G flat\n"
      << "\t8) G\n"
      << "\t9) G sharp/A flat\n"
      << "\t10) A\n"
      << "\t11) A sharp/B flat\n"
      << "\t12) B\n"
      << "\tor type quit to quit\n";
   input = getInput();
   if (! (input >= BACK && input <= 12))
      return -1;
   return input;
}
void writeToSoundDevice(short buf[], int buffS
   int status;
   status = write(deviceID, buf, buffSize);
```

Programming Exercise:
Chords in C++

Programming Exercise:
Playing With Music

```
60       if (status != buffSize)
61          perror("Wrote wrong number of bytes\n");
62       status = ioctl(deviceID, SOUND_PCM_SYNC, 0);
63       if (status == -1)
64          perror("SOUND_PCM_SYNC failed\n");
65    }
66    int getScaleType(){
67       int input;
68       cout
69          << "Choose one of the following:\n"
70          << "\t" << MAJOR << ") for major\n"
71          << "\t" << MINOR << ") for minor\n"
72          << "\t" << BACK << ") to back up\n"
73          << "\tor type quit to quit\n";
74       input = getInput();
75       return input;
76    }
77    void playScale(int deviceID){
78       int arraySize, note, steps, index, scaleType
79       int break1; // only one half step to here
80       int break2; // only one half step to here
81       int t, off;
82       double f;
83       short *buf;
84       arraySize = 8 * LENGTH * RATE * CHANNELS;
85       buf = new short[arraySize];
86       while ((index = getIndex()) < 0)
87          cout << "Input out of bounds.  Please try
88       f = getFreq(index);
89       while ((scaleType = getScaleType()) < 0)
90          cout << "Input out of bounds.  Please try
91       switch (scaleType) {
92          case MAJOR :
93             break1 = 3;
94             break2 = 7;
95             break;
96          case MINOR :
97             break1 = 2;
98             break2 = 5;
99             break;
100         case BACK :
101            return;
102         default :
103            playScale(deviceID);
104      }
105      arraySize = LENGTH * RATE * CHANNELS;
106      for (note = off = 0; note < 8; ++note, off +=
107         if (note == 0)
108            steps = 0;
109         else if (note == break1 || note == break2
110            steps = 1;
111         else steps = 2;
112         f *= pow(SEMITONE, steps);
113         for (t = 0; t < arraySize; ++t)
114            buf[t + off] = floor(SAMPLE_MAX*sin(2*
115      }
116      arraySize = 8 * LENGTH * RATE * SIZE * CHANNE
117      writeToSoundDevice(buf, arraySize, deviceID);
118      delete buf;
119      return;
120   }
121   int main(){
122      int deviceID, arg, status, index;
123      deviceID = open("/dev/dsp", O_WRONLY, 0);
124      if (deviceID < 0)
```

```
125         perror("Opening /dev/dsp failed\n");
126      arg = SIZE * 8;
127      status = ioctl(deviceID, SOUND_PCM_WRITE_BITS
128      if (status == -1)
129         perror("Unable to set sample size\n");
130      arg = CHANNELS;
131      status = ioctl(deviceID, SOUND_PCM_WRITE_CHAN
132      if (status == -1)
133         perror("Unable to set number of channels\
134      arg = RATE;
135      status = ioctl(deviceID, SOUND_PCM_WRITE_RATE
136      if (status == -1)
137         perror("Unable to set sampling rate\n");
138      while (true)
139               playScale(deviceID);
140  }
```

Algorithm 3.2

3.3.6 Experimenting with Music in Java

Let's try making chords in Java as we did in C++. We'll give you the code for this first exercise to get you started, and then you can try some exercises on your own. The "Chords in Java" program plays a chord selected from the list shown in Figure 3.54. The program contains two Java class files, one called Chord and the other called ChordApp. The top level class (i.e., the one that includes the *main* function) is the ChordApp class. The ChordApp class is a subclass of the JPanel, which is a generic window container. The JPanel provides functionality to display a window as a user interface. The constructor of the ChordApp class creates a list of radio buttons and displays the window with these options (Figure 3.54). When the user selects a type of chord, the method *playChord()* is called through the Chord class (line 138).

Programming Exercise:
Chords in Java

Figure 3.54 Interface for Java chord-playing program

The ChordApp class contains the method *playChord* (line 7), which is a variation of the code shown in the MATLAB section. In the *playChord* method, the ChordApp object creates a Chord object, initializing it with the array *scale*, which contains the list of notes to be played in the chord. These notes are summed in the Chord class in lines 46 to 48.

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class ChordApp extends JPanel implements ActionListener {
6
7      public static void playChord(String command)
8      {/**Method that decides which Chord to play */
9          float[] major7th = {0, 4, 7, 11};
10         float[] minor7th = {0, 3, 7, 10};
11         float[] domin7th = {0, 4, 7, 10};
12         float[] dimin7th = {0, 3, 6, 10};
13         float[] majorTri = {0, 4, 7};
14         float[] minorTri = {0, 3, 7};
15         float[] augmeTri = {0, 4, 8};
16         float[] diminTri = {0, 3, 6};
17
18         float[] scale;
19
20         if      (command == "major7th")
21             scale = major7th;
22         else if (command == "minor7th")
23             scale = minor7th;
24         else if (command == "domin7th")
25             scale = domin7th;
26         else if (command == "dimin7th")
27             scale = dimin7th;
28         else if (command == "majorTri")
29             scale = majorTri;
30         else if (command == "minorTri")
31             scale = minorTri;
32         else if (command == "augmeTri")
33             scale = augmeTri;
34         else
35             scale = diminTri;
36
37         float startnote = 220*(2^((2+3)/12));
38
39         for(int i=0;i<scale.length;i++){
40             scale[i] = scale[i]/12;
41             scale[i] = (float)Math.pow(2,scale[i]);
42             scale[i] = startnote * scale[i];
43         }
44
45         //once we know which Chord to play, we call the Chord function
46         Chord chord = new Chord(scale);
47         chord.play();
48     }
49
50     public ChordApp() {
51         super(new BorderLayout());
52
53         try {
54     UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
55             SwingUtilities.updateComponentTreeUI(this);
56         } catch (Exception e) {
57             System.err.println(e);
```

```
58          }
59
60          JRadioButton maj7thButton = new JRadioButton("Major Seventh");
61          maj7thButton.setActionCommand("major7th");
62
63          JRadioButton min7thButton = new JRadioButton("Minor Seventh");
64          min7thButton.setActionCommand("minor7th");
65
66          JRadioButton dom7thButton = new JRadioButton("Dominant Seventh
67          dom7thButton.setActionCommand("domin7th");
68
69          JRadioButton dim7thButton = new JRadioButton("Diminished Seven
70          dim7thButton.setActionCommand("dimin7th");
71
72          JRadioButton majTriButton = new JRadioButton("Major Triad");
73          majTriButton.setActionCommand("majorTri");
74
75          JRadioButton minTriButton = new JRadioButton("Minor Triad");
76          minTriButton.setActionCommand("minorTri");
77
78          JRadioButton augTriButton = new JRadioButton("Augmented Triad"
79          augTriButton.setActionCommand("augmeTri");
80
81          JRadioButton dimTriButton = new JRadioButton("Diminished Triad
82          dimTriButton.setActionCommand("diminTri");
83
84          ButtonGroup group = new ButtonGroup();
85          group.add(maj7thButton);
86          group.add(min7thButton);
87          group.add(dom7thButton);
88          group.add(dim7thButton);
89          group.add(majTriButton);
90          group.add(minTriButton);
91          group.add(augTriButton);
92          group.add(dimTriButton);
93
94          maj7thButton.addActionListener(this);
95          min7thButton.addActionListener(this);
96          dom7thButton.addActionListener(this);
97          dim7thButton.addActionListener(this);
98          majTriButton.addActionListener(this);
99          minTriButton.addActionListener(this);
100         augTriButton.addActionListener(this);
101         dimTriButton.addActionListener(this);
102
103         JPanel radioPanel = new JPanel(new GridLayout(0, 1));
104         radioPanel.add(maj7thButton);
105         radioPanel.add(min7thButton);
106         radioPanel.add(dom7thButton);
107         radioPanel.add(dim7thButton);
108         radioPanel.add(majTriButton);
109         radioPanel.add(minTriButton);
110         radioPanel.add(augTriButton);
111         radioPanel.add(dimTriButton);
112
113         add(radioPanel, BorderLayout.LINE_START);
114         setBorder(BorderFactory.createEmptyBorder(20,20,20,20));
115     }
116
117     public static void main(String[] args) {
118         javax.swing.SwingUtilities.invokeLater(new Runnable() {
119             public void run() {
120                 ShowWindow();
121             }
122         });
```

```
    }

    private static void ShowWindow() {
        JFrame frame = new JFrame("Chords App");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JComponent newContentPane = new ChordApp();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        frame.pack();
        frame.setVisible(true);
    }
    /** Listens to the radio buttons. */
    public void actionPerformed(ActionEvent e) {
        ChordApp.playChord(e.getActionCommand());
    }
}
```

Algorithm 3.3 ChordApp class

```


import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.SourceDataLine;
import javax.sound.sampled.LineUnavailableException;

public class Chord {
    float[] Scale;

    public Chord(float[] scale) {
        Scale=scale;
    }

    public void play() {
        try {
            makeChord(Scale);
        } catch (LineUnavailableException lue) {
            System.out.println(lue);
        }
    }

    private void makeChord(float[] scale)
    throws LineUnavailableException
    {
        int freq = 44100;
        float[] x=new float[(int)(freq)];

        byte[] buf;
        AudioFormat audioF;

        for(int i=0;i<x.length;i++){
            x[i]=(float)(i+1)/freq;
        }

        buf = new byte[1];
        audioF = new AudioFormat(freq,8,1,true,false);

        SourceDataLine sourceDL = AudioSystem.getSourceDataLine(audioF)
        sourceDL = AudioSystem.getSourceDataLine(audioF);
        sourceDL.open(audioF);
        sourceDL.start();

```

```
            for (int j=0;j<x.length;j++){
                buf[0]= 0;
                for(int i=0;i<scale.length;i++){
                  buf[0]=(byte)(buf[0]+(Math.sin((2*Math.PI*scale[i])*x[j]
                }
                sourceDL.write(buf,0,1);
            }

            sourceDL.drain();
            sourceDL.stop();
            sourceDL.close();
        }
    }
```

Algorithm 3.4 Chord class

As another exercise, you can try to play and compare equal vs. just tempered scales, the same program exercise linked in the C++ section above.

The last exercise associated with this section, you're asked to create a metronome-type object that creates a beat sound according to the user's specifications for tempo. You may want to borrow something similar to the code in the ChordApp class to create the graphical user interface.

Programming Exercise:
Creating a Metronome in Java

Programming Exercise:
Equal vs. Just Tempered Scales in Java

3.4 References

In addition to references listed in previous chapters:

Barzun, Jacques, ed. *Pleasures of Music: A Reader's Choice of Great Writing about Music and Musicians*. New York: Viking Press, 1951.

Hewitt, Michael. *Music Theory for Computer Musicians*. Boston, MA: Course Technology CENGAGE Learning, 2008.

Loy, Gareth. *Musimathics: The Mathematical Foundations of Music*. Cambridge, MA: The MIT Press, 2006.

Roads, Curtis. *The Computer Music Tutorial*. Cambridge, MA: The MIT Press, 1996.

Swafford, Jan. *The Vintage Guide to Classical Music*. New York: Random House Vintage Books.

Wharram, Barbara. *Elementary Rudiments of Music*. The Frederick Harris Music Company, Limited, 1969.

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.