Proceedings of Student Research Day, CSIS, Pace University, May 9th, 2003

# A System for Effective Ear Training

Brian Ordona

**Abstract**

*Every musician, from the casual hobbyist to the seasoned professional, shares the desire to become better, and to this end, long hours of practice are requisite. Practice not only involves physical proficiency over an instrument, but ear training as well. Ear training software addresses a musician's need for an ear training tool that is accurate and convenient. One such software system, named "Ultimate Pitch," is described below. The following discusses the features of Ultimate Pitch that facilitate effective ear training, as well as the inner workings of the system itself from a software engineering perspective.*

## 1. Introduction

### 1.1 *Music Theory: A Primer*

Assuming that the reader has no knowledge of music terminology or theory, this section will introduce the basic music concepts and idioms necessary to understand the remainder of this paper. Readers who already have a grasp of basic music concepts may wish to skip this section and proceed directly to the section entitled *Motivation*.

Music is both an art and a science. Although art and science are usually thought of as polar opposites, music is a rather beautiful amalgamation of the two. Music as an art offers the individual infinite modes of self expression. One may express feelings of happiness, melancholy, confusion, love (or hate for that matter) through music. Music, as a science, borrows concepts and ideas from a variety of disciplines such as mathematics, physics, psychology, even biology and sociology, amongst others, in order to produce its own body of laws and (for lack of a better term) rules. Since art is, in a sense, about "breaking the rules," it is a paradox that artful expressions generated through music exist because of these very laws and rules: they are what help us differentiate between noise and music, between unremarkable random vibrations of the air and expressions of brilliance and beauty through sound.

The discipline of music itself is broad. To start, the general concept of music can be broken down into different *systems*. The system of interest will be the Western music system. There are many other systems: Indian music systems, Asian music systems, Arabic music systems, amongst others, each of which have their own concepts, idioms, and "style."

Western music is a 12 tone system (Figure 1). Each tone has a distinct name. It is easy to identify them on a keyboard. Each of the white keys correspond to the tones C, D, E, F, G, A, and B. The black keys correspond to the tones C#/Db, D#/Eb, F#/Gb, G#/Ab, A#/Bb, where "#" is pronounced "sharp," which means "raised," and "b" is pronounced "flat," which means "lowered." You will notice that these last five tones have two names each, a "sharp" name and a "flat" name. Throughout, the "flat" names will be used for the sake of simplicity. Hence, the 12 tones in Western music are C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, and B.

This set of 12 tones is repeated over and over, each new set beginning at a higher register, or *octave*, than the previous one. In Figure 1, each enclosing, colored rectangle

represents a different octave. The octaves shown are named "Low" (red), "Middle" (blue), and "High" (green). Hence, "Middle C" refers to the note of C in the "Middle" register/octave. "High Eb" refers to the note of E Flat in the "High" register/octave, etc.

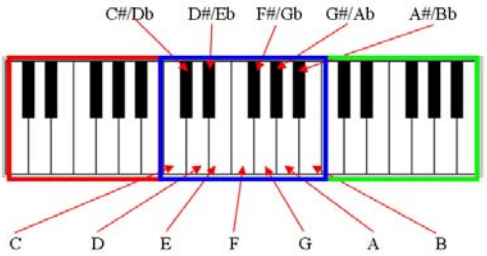



Figure 1: Western music is a 12 tone system.

Subsets of these 12 tones are called *scales* (or *modes*) (Figure 2). Different subsets produce different scales. In Figure 2, the notes in blue comprise a C Major scale. The notes in red comprise a C Minor scale. There are many, many others. For the musician, different scales produce different sounds. All scales have an anchor point called a *root*. In Figure 2, the root of both scales is the note of C.

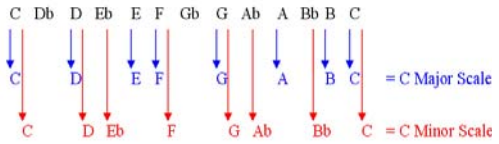



Figure 2: Scales.

Scales themselves can be subset into *chords* (Figure 3). As with scales, different subsets produce different chords. In Figure 3, the notes in blue comprise a basic C Major chord. The notes in red comprise a C Major 7$^{th}$ chord. Both are subsets of the C Major scale (which in turn is a subset of the 12 tone system). For the musician, different chords also produce different sounds. Since chords are based on a parent scale, a chord's root is the same as its parent scale's root. The notes of a chord are usually played simultaneously. But a chord may also be played one note at a time in succession (but not necessarily in sequence). This "broken up chord" is called an *arpeggio*.

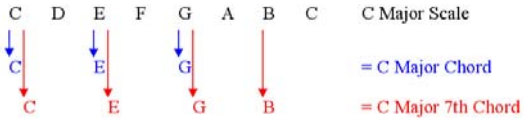
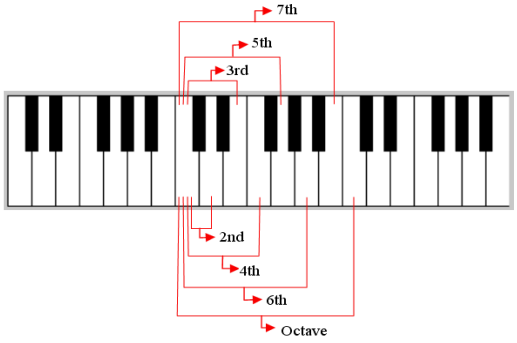



Figure 3: Chords.

A scale can be produced in any *key*. A key is just another way of referring to a scale's anchor point, similar to the root. The difference being that the "root" refers more to the *position* of the first note of the scale, whereas "key" refers more to the scale's *tonal center*, or very simply, the scale's most audibly dominant note. There are 12 tones and therefore 12 different possible keys.

The space between two notes is referred to as an *interval* (Figure 4). Within a scale, the interval between a note and the next one directly above it (one position away), is called a "second." For example, within the C Major scale, C and D are a second apart, F and G are a second apart, A and B are a second apart, etc. The interval between a note and another note that is two positions above is called a "third." For example, C and E are a third apart. G and B are a third apart, D and F are a third apart, etc. The same line of reasoning goes for "fourths," "fifths," "sixths," "sevenths," even "ninths" and "tenths." Note that there is no such thing as an "eighth." The note that is an "eighth" away from the reference note is the same as the reference note itself, just at the next highest register. Instead, this special interval is called an "octave." For example, Middle C and High C are an octave apart; Low E and Middle E are an octave apart, etc.

Figure 4: Intervals



Each of the 12 tones resonates at a specific frequency in units of Hertz (Hz). Thus, frequencies between two tones will have a mathematical ratio. For example, Middle A has a frequency of 440 Hz. High A has a frequency of 880 Hz. The ratio between these two tones is 440:880 or 1:2. (As a matter of fact, any two notes that are an octave apart will always have a frequency ratio of 1:2.) Different *tuning systems* dictate different frequency ratios between notes (Figures 5 and 6). The tuning system shown in Figure 5 is known as *Equal Temperment*. This particular system has the following properties: it maintains the 1:2 frequency ratio between octaves, and it maintains a $1:\sqrt[12]{2}$ ratio between any two adjacent tones. Equal Temperment is the most widely used tuning system in Western music. Another tuning system is called *Just* tuning (Figure 6). Like Equal Temperment, Just tuning also maintains a 1:2 frequency ratio between octaves, but enforces different ratios between adjacent or nearly adjacent tones. There are still other tuning systems, each dictating differing ratios between frequencies.

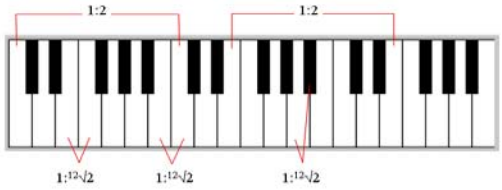



Figure 5: Equal Temperment.

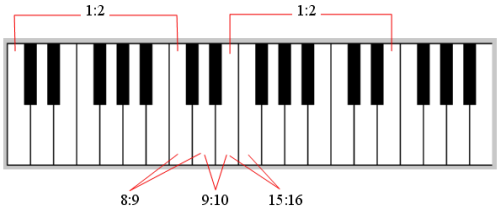



Figure 6: Just Tuning.

The reasons for using different tuning systems are purely subjective and a matter of personal taste. For example, more discerning musicians feel that Equal Temperment, although it is a mathematically perfect tuning system, can make certain musical passages sound "monotonous" or "dry" in regards to melodic or harmonic sonority, and will prefer to use something like Just tuning.

### 1.2 *Motivation*

Due to the varying design among instruments, playing different instruments requires differing sets of skills: both motor skills and conceptual skills. For example, a piano requires the striking of notes with both hands. A violin requires that one hand frets (presses down on) the strings along its neck while the other hand uses a bow. Thus, playing the same scale on a piano and on a violin requires differing physical techniques. Conceptually, pianos and violins differ as well. On a piano, notes are laid out *linearly*. On a violin, notes are laid out in a grid. That is, a violinist must visualize notes in two dimensions: *along* a string, and *across* the strings.

Another important aspect in which pianos and violins differ is the manner in which each physically produces sound. A piano is an intricate system of levers. Pressing a key activates this system in which a hammering mechanism for each key strikes a corresponding set of strings (which are secured tightly within the confines of the piano). A piano is engineered in such a way that frequency generation is *automatic*. In other words, the inner workings of a piano produce the sound. The pianist never has to touch the strings, just hit the keys. It is exactly like pressing buttons. On the other hand, a violin (or any fretless instrument for that matter) is engineered as such that frequency generation *depends* on the violinist. This is due to the fact that a string vibrates at the point where the violinist presses down on the string. Thus, frequency generation is not automatic like it is on a piano. For the violinist, hand and finger positioning is crucial. Accurate pitch is only acquired by accurate positioning.

The novice violinist is satisfied just to be playing the correct notes. While bowing the strings, the novice, upon hearing an

incorrect note, will quickly adjust their fret hand to the correct position. Even when fretting the correct note, the novice may not know, or care, that they may be fretting the correct note, but at the wrong pitch. In other words, they are "in the ballpark" of the note. However, the expert violinist is not satisfied by merely playing the correct note, but by playing the correct note, *at the correct pitch*. Like the novice, the expert player will adjust their fretting hand upon bowing an incorrect note. But unlike the novice, the expert player will further "micro-adjust" their fret hand if they notice that they are playing the correct note but not at the exact pitch. The expert will reach a point where this process of micro-adjusting becomes second nature.

Naturally, the question is, how does the novice acquire such a skill? How does the novice reach the level of the expert? The answer is obviously, effective practice. But a large part of effective practice, that unfortunately gets overlooked, is **effective ear training**. The expert has not only mastered the mechanics of the violin, but has trained themselves to master pitch. In other words, the expert (not just the expert violinist, but the expert *musician*) can actually hear the exact pitch of a note in their mind even before producing a single note. A musician who has acquired such skill is said to have what is known as *Perfect Pitch* (or *Absolute Pitch*) [4]. In addition, the expert musician can recognize the interval between any pair of notes. A musician who has acquired this particular skill is said to have *Relative Pitch* [4]. Effective ear training and the acquisition of Perfect Pitch and Relative Pitch requires the correct tools.

### 1.3 *Common Ear Training Tools and Techniques*

There are many tools and techniques that one can utilize when ear training. But most of these tools and techniques are not without shortcomings.

One of the most utilized methods of ear training is with a piano. A common exercise in pitch recognition involves turning your back to a piano while a partner plays a random note. The goal is to try to name the note that your partner is playing. While this can be an excellent technique for acquiring Perfect Pitch, using this method has its weaknesses. For instance, it is based upon the assumption that the piano you are working with is in tune. Even after having a piano tuned, it can still be thrown out of tune even by a few cents (micro-divisions between frequencies) after days of constant use. Of course, none of the preceding matters if one does not own a piano nor has access to one in the first place.

Another method of ear training is to have a friend or a music teacher with Perfect or Relative Pitch help you. However, this suffers from the same problem as above. It works upon the assumption that your friend or teacher has full acquisition of Perfect or Relative Pitch. This is on top of the fact that this method is impractical: lessons can be expensive, friends need to find free time, etc.

### 1.4 *A Proposal for a Better Solution*

A software solution addresses these problems of convenience and accuracy: The average American household has at least one computer. Plus, increased processor speeds, as well as advances in computer architecture and design, make it possible to faithfully reproduce sounds at any given frequency (with the correct, well-developed software). All of these factors make a software solution the most practical solution.

Ear training software systems already exist such as Auralia (by Rising Software) [5], Magic Ear Trainer (by www.ilovemusic.com) [3], and EarTest (by Brent Hugh) [6] that focus on pitch and interval recognition, by playing back notes, chords and intervals and asking the user to guess the note name, chord quality, or interval quality. Hence, they employ a "name that tune" approach to teaching.

*Ultimate Pitch* is also a software system designed to aid musicians in ear training and

effective practice. It has the following features:

- Allows a variety of scales to be played in all 12 keys. Scales may also be played in double stops, i.e. each scale tone is doubled up with another tone that is a specific interval away.
- Allows a variety of arpeggios to be played in all 12 keys.
- Allows a user to produce and play back small musical passages/compositions through the use of a working on-screen keyboard.
- Provides a metronome.
- Provides a tuner.

Although other ear training software packages may offer the same features above, Ultimate Pitch is different in that scales and arpeggios may be played in a variety of tuning systems (i.e. Just, Pythagorean), not just Equal Temperment. However, there are some systems, such as Cochlear Consciousness Ear Training Software (by Julius Pierce) [2], that are also sophisticated enough to offer a means for applying different tuning systems to scales or chords. But in Ultimate Pitch, the frequency of any note within a scale or chord may be controlled with an accuracy of 1/100 of a cent, for those who wish to use more "exotic," off-the-beaten-path tunings.

The teaching approach of Ultimate Pitch is also different. The focus is not so much on pitch or interval recognition as is the case with the aforementioned systems (i.e. "*What* note is this?"). Rather, the focus is on *how* a note, or scale, or chord is supposed to sound. Instead of exercises where the user has to guess the names of notes and intervals, Ultimate Pitch's exercises involve actually playing (or singing) along in tune with scales or chords (broken up into arpeggios) as they are played back by the system. In other words, Ultimate Pitch aids in developing *pitch accuracy*, and is directly concerned with the musician's "sound."

## 2. Methodology

### 2.1 *Development*

Ultimate Pitch was originally written in Turbo Pascal and was engineered to work under DOS. The goal of the current development effort is to convert Ultimate Pitch over to a GUI-based operating system such as Windows. The language of choice is Java so that the software may be used across differing platforms. Consequently, Ultimate Pitch needs to be re-engineered using an Object-Oriented design model, since earlier versions of Turbo Pascal did not support Object-Oriented Programming.

### 2.2 *Component Level Design*

The software is comprised of three main components, each with a specific duty:

- A Sound Component
- A Music Component
- A Presentation Component (the GUI)

Sound Component:

The sound component is responsible for the production of sound generation. There is a single Java class in this component called the Sound class. It contains two primary methods. The first, called calculateFrequency( ), calculates the frequencies of notes in Hertz (Hz), with floating-point precision. The other method, makeClip( ), accepts a frequency value as an argument of data type double, and returns a Clip reference (the Clip interface is found within the javax.sound.sampled package) that generates the desired frequency through the standard audio output. The returned clip(s) may then be manipulated (i.e. played, stopped or paused) by other application logic.

calculateFrequency( ) uses the following formula to calculate frequencies [7]:

$F = \sqrt[1200]{2^x} * 440$, where,
x = (number of tones from Middle A) * 100.

For example, to calculate the value of Middle C:

Middle C is 9 tones below Middle A. Hence,
x = -9 * 100, or -900
$F = \sqrt[1200]{2}^{-900} * 440$
F = 261.625 Hz.

This is the basis for all frequency calculations throughout the application.

Music Component:
The music component is a container for classes representing different music systems. For example, it has a WesternMusic class that contains all the concepts and idioms particular to Western music. These concepts are used by the application to calculate items such as interval lengths, deviations in tuning, scale and arpeggio roots, etc.

The WesternMusic class contains many "conversion" methods that convert string names of notes, intervals, octaves, etc., into integer values that can be used in calculations. For example, using the method translateNoteToConstant( ), the string value of the note of D Flat ("Db") is converted to an integer value of 1. This value is more useful in calculations than its string counterpart. Now, lets say that you want to compute the distance in tones between notes D Flat and A. Using translateNoteToConstant( ), the value of the note A is converted to an integer value of 9. D Flat, as stated above, is converted to 1. Since 9 – 1 = 8, D Flat is 8 tones away from A, which is truly the case.

The music component also contains an important class called Note that represents a generic musical note. It contains the properties of a note such as its frequency and duration (time value). Instances of the Note class are used in the generation of scales, arpeggios and compositions since each of these constructs are represented within the application code as vectors of Note objects.

Having mentioned the preceding, the basic algorithm for building a scale, arpeggio, or short composition, all of which will be referred to as a "note sequence" for simplicity, is as follows:

1. Create an instance of the Vector class which will represent the note sequence.
2. Find the first note of the sequence.
3. Use the calculateFrequency( ) method of the Sound class (discussed above) to calculate this note's frequency.
4. Create an instance of the Note class passing in the frequency calculated from the previous step as an argument to its constructor. For compositions, an extra argument for the note's time value (i.e. whole, half, quarter, dotted-quarter, etc.) is passed in, since notes within a composition will have various time values.
5. Add this newly created Note instance to the vector created in step 1.
6. Repeat steps 2 through 5 for each remaining note until the sequence is exhausted.

Presentation Component:
The presentation component is in charge of the GUI. Its purpose is to track user preferences (e.g. selected key, selected tempo, etc.) and maintain the application's playback state, i.e. play, stop, and record. The main panel is an instance of JTabbedPane which can be switched between four screens:

1. The Scale Screen: Here, the user may select a scale from a standard library to play back. The scale may be adjusted for key, tempo, starting octave, and number of ascending/descending octaves. The user may also select from different tuning systems under which the scale frequencies will be generated.
2. The Arpeggio Screen: This is similar to the Scale Screen, but used for arpeggios.
3. The Composition Screen: Here, the user may create and edit a small composition using an on-screen keyboard, and play it back in any tempo.
4. The Tools Screen: Here, the user is provided with miscellaneous tools useful to any musician such as a metronome and a tuner.

The GUI is actually an interwoven subsystem of specialized JPanel subclasses. Each of these specialized JPanels is a

reusable component that may be utilized across multiple screens. For example, three of the screens have control panels (panels containing standard play, stop, and record buttons that control playback), each of which are instances of a ControlPanel class (Figure 7).

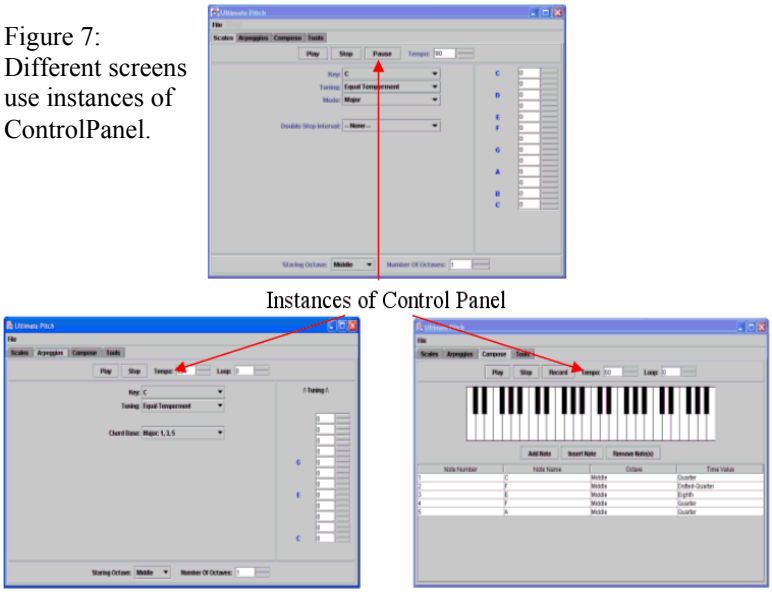



Figure 7: Different screens use instances of ControlPanel.

## 2.3 *Use of Design Patterns*

To keep this subsystem from becoming unwieldy, the *Mediator* design pattern [1] is used. The mediator class in the application is UPitchGUI2. Within its constructor, the primary JTabbedPane is created, as well as all of the instances of the specialized JPanels which are added to various parts of the primary JTabbedPane. More importantly, UPitchGUI2 controls the interaction between *all* of the classes employed by the application including those in the other two main components. It contains all of the necessary logic (methods) to carry out the application's main functionality such as refreshing the GUI, via its refresh( ) method, and pulling user preferences together in order to produce the correct scales or arpeggios via its play( ) method. The Mediator pattern is used mainly to promote looser coupling between all the classes used by the system, and for better organization of code (Figure 8).

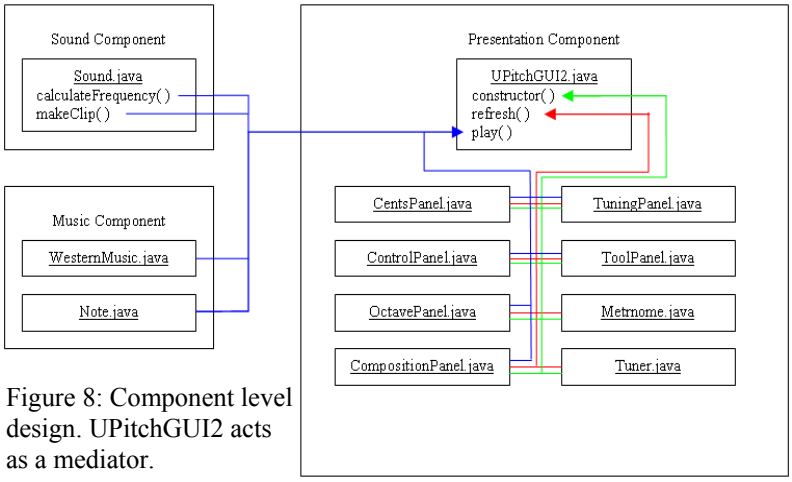



Figure 8: Component level design. UPitchGUI2 acts as a mediator.

## 2.4 *Future Development*

The following are areas of interest for future, possibly immediate development:

- The application as it has existed for the past 12 years, up to its current development, has solely concerned itself with the Western music system. It has always been the aspiration of Ultimate Pitch's creator to include Indian music systems. Development in the near future will head in this direction.
- Additional ear training tools may be added to the application in order to further facilitate of the music learning process. As discussed earlier, a classic ear training exercise involves having a partner sit at a piano and play random notes as you try to guess them. This could possibly be simulated within the application as a game. That is, the application plays 10 tones, and the user attempts to guess them, scoring a point for each correct answer.

## 3. Conclusion

Every musician from the casual hobbyist to the seasoned expert has heard the story:

*A tourist visiting New York is lost and asks a stranger for help. The tourist asks, "Sir, how do you get to Carnegie Hall?" The stranger replies, "Practice, practice, practice!" - Author Unknown*

The most important aspect of practice is that it needs to be done effectively. Effective practice not only involves training the hands for increased physical proficiency over an instrument, but also involves training the mind, along with the ear, in order to truly hear music, and more importantly, to produce music with an aesthetically pleasing quality. It is akin to learning a new language: A solid grasp of vocabulary is not enough. One must also master pronunciation in order to be truly speaking the language. Vowels must be enunciated, R's need to roll. The spoken word must ring with the "flavor" of the native tongue. The same holds true for the language of music.

While many tools and techniques for ear training are readily available, most cannot ensure the degree of accuracy necessary to develop a musically discerning ear. Plus, they come at the sacrifice of convenience. A software solution such as Ultimate Pitch addresses these shortcomings by offering an organized, accurate and practical method of ear training for both the casual hobbyist and the seasoned expert.

## References


[1] Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[2] http://cctr.umkc.edu/userx/bhugh/cochlear.html, accessed 4/1/2003.

[3] http://www.ilovemusic.com/, accessed 4/1/2003.

[4] http://www.perfectpitch.com/perfectrelative.htm, accessed 11/15/2002.

[5] http://www.rising.com.au/, accessed 3/13/2003.

[6] http://www.vocalist.org/perfectpitch.html, accessed 3/13/2003.

[7] http://www.wilsonmusic.com/global-music/building/cents.htm, accessed 9/26/2002.